Skill India
कौशल भारत-कुशल भारत

सत्यमेव जयते
GOVERNMENT OF INDIA
MINISTRY OF SKILL DEVELOPMENT
& ENTREPRENEURSHIP

N·S·D·C
National
Skill Development
Corporation
Transforming the skill landscape

IT - ITeS SSC
nasscom

प्रतिभागी पुस्तिका

सेक्टर
IT-ITeS

उप – सेक्टर
IT सेवा

व्यवसाय
टेस्ट इंजीनियर

संदर्भ ID: **SSC/Q7001, संस्करण 3.0**
NSQF स्तर 4

टेस्ट इंजीनियर

द्वारा प्रकाशित

IT – ITES सेक्टर स्किल काउंसिल nasscom

सेक्टर स्किल काउंसिल संपर्क विवरण:

पता: प्लॉट नंबर – 7, 8, 9 और 10 सेक्टर – 126, नोएडा, उत्तर प्रदेश – 201303

नई दिल्ली – 110049

वेबसाइट: www.sscnasscom.com

फोन: 0120 4990111 - 0120 4990172

तीसरा संस्करण, अगस्त, 2024

IT – ITES सेक्टर स्किल काउंसिल nasscom

सेक्टर स्किल काउंसिल संपर्क विवरण:

पता: प्लॉट नं. – 7, 8, 9 और 10 सेक्टर – 126, नोएडा, उत्तर प्रदेश – 201303 नई दिल्ली – 201301

वेबसाइट: www.sscnasscom.com

फोन: 0120 4990111 - 0120 4990172

यह पुस्तक IT द्वारा प्रायोजित है - ITES सेक्टर स्किल काउंसिल nasscom

डिस्क्लेमर

यहां निहित जानकारी IT के विश्वसनीय स्रोतों से प्राप्त की गई है - ITES सेक्टर स्किल काउंसिल nasscom nasscom ऐसी सूचना की सटीकता, पूर्णता अथवा पर्याप्तता के लिए सभी वारंटियों को अस्वीकार करता है। nasscom के पास यहां निहित जानकारी में त्रुटियों, चूक, या अपर्याप्तता के लिए, या उसकी व्याख्याओं के लिए कोई दायित्व नहीं होगा। पुस्तक में शामिल कॉपीराइट सामग्री के मालिकों का पता लगाने के लिए हर संभव प्रयास किया गया है। पुस्तक के भावी संस्करणों में पावती के लिए उनके ध्यान में लाई गई किसी भी चूक के लिए प्रकाशक आभारी होंगे। nasscom में कोई भी यूनिट किसी भी व्यक्ति द्वारा किसी भी नुकसान के लिए जिम्मेदार नहीं होगी जो इस सामग्री पर भरोसा करता है। इस प्रकाशन की सामग्री कॉपीराइट है। इस प्रकाशन के किसी भी भाग को किसी भी रूप में या किसी भी माध्यम से कागज या इलेक्ट्रॉनिक मीडिया पर पुन: प्रकाशित, संग्रहीत या वितरित नहीं किया जा सकता है, जब तक कि nasscom द्वारा अधिकृत न किया गया हो।

कौशल विकास से बेहतर भारत का निर्माण हो रहा है। अगर हमें भारत को विकास की ओर ले जाना है तो कौशल विकास हमारा मिशन होना चाहिए।

श्री नरेंद्र मोदी
मुख्य मंत्री का भारत


Skill India
कौशल भारत-कुशल भारत
IT - ITeS SSC
NASSCOM
N·S·D·C
National
Skill Development
Corporation
Certificate
COMPLIANCE TO
QUALIFICATION PACK- NATIONAL OCCUPATIONAL STANDARDS
is hereby issued by the
IT-ITeS Sector Skills Council NASSCOM
for
SKILLING CONTENT: PARTICIPANT HANDBOOK
Complying to National Occupational Standards of
Job Role/ Qualification Pack: 'Test Engineer'
QP No. 'SSC/Q7001, NSQF Level 4'
Date of Issuance : January 27th, 2022
Valid Up to* : January 27th, 2025
*Valid up to the next review date of the Qualification Pack or the 'Valid up to' date mentioned above (whichever is earlier)
Authorised Signatory
(IT-ITeS Sector Skills Council NASSCOM)

स्वीकृतियाँ

टेस्ट इंजीनियर के लिए इस प्रतिभागी की हैंडबुक इस नौकरी की भूमिका में मौजूदा और भावी नौकरी धारकों को सभी प्रासंगिक जानकारी की उपलब्धता सुनिश्चित करने का एक ईमानदार प्रयास है। हमने प्रासंगिक विषय वस्तु विशेषज्ञों (एसएमई) और उद्योग के सदस्यों के इनपुट के साथ सामग्री को संकलित किया है ताकि यह सुनिश्चित हो सके कि यह नवीनतम और प्रामाणिक है। हम उन सभी एसएमई और उद्योग के सदस्यों के प्रति अपनी ईमानदारी से आभार व्यक्त करते हैं जिन्होंने इस प्रतिभागी की हैंडबुक को पूरा करने में अमूल्य योगदान दिया है। हम उन सभी विशेषज्ञों और संगठनों को भी धन्यवाद देना चाहते हैं जिन्होंने सामग्री की समीक्षा करके और इसकी गुणवत्ता में सुधार के लिए अपनी प्रतिक्रिया प्रदान करके हमारी मदद की है।

यह पुस्तिका टेस्ट इंजीनियर के क्षेत्र में कौशल आधारित प्रशिक्षण देने में मदद करेगी। हमें उम्मीद है कि इससे प्रतिभागियों, प्रशिक्षकों और मूल्यांकनकर्ताओं जैसे सभी हितधारकों को लाभ होगा। हमने यह सुनिश्चित करने के लिए सभी प्रयास किए हैं कि प्रकाशन QP/NOS-आधारित प्रशिक्षण कार्यक्रमों के सफल वितरण के लिए वर्तमान गुणवत्ता मानकों को पूरा करता है। हम इस पुस्तिका में भविष्य के सुधार के लिए किसी भी सुझाव का स्वागत और सराहना करते हैं।

इस किताब के बारे में

इस प्रतिभागी पुस्तिका को उन प्रतिभागियों के लिए एक मार्गदर्शक के रूप में सेवा करने के लिए डिज़ाइन किया गया है जो टेस्ट इंजीनियर की भूमिका में विभिन्न गतिविधियों को करने के लिए आवश्यक ज्ञान और कौशल प्राप्त करना चाहते हैं। इसके कंटेंट को जॉब रोल के लिए तैयार किए गए लेटेस्ट क्वालिफिकेशन पैक (QP) के साथ अलाइन्ड किया गया है। एक योग्य प्रशिक्षक के मार्गदर्शन के साथ, प्रतिभागियों को नौकरी की भूमिका में कुशलता से काम करने के लिए निम्नलिखित से लैस किया जाएगा:

- **ज्ञान और समझ:** आवश्यक कार्यों को करने के लिए प्रासंगिक परिचालन ज्ञान और समझ।
- **प्रदर्शन मानदंड:** लागू गुणवत्ता मानकों के लिए आवश्यक संचालन करने के लिए हाथों पर प्रशिक्षण के माध्यम से आवश्यक कौशल।
- **व्यावसायिक कौशल:** कार्य के क्षेत्र के बारे में उचित परिचालन निर्णय लेने की क्षमता।

हैंडबुक में टेस्ट इंजीनियर द्वारा की जाने वाली संगत गतिविधियों का ब्यौरा दिया गया है। इस पुस्तिका का अध्ययन करने के बाद, नौकरी धारकों को लागू गुणवत्ता मानकों के अनुसार अपने कर्तव्यों को पूरा करने में पर्याप्त रूप से कुशल बनाया जाएगा। हैंडबुक को टेस्ट इंजीनियर QP के नवीनतम और अनुमोदित संस्करण में विस्तृत निम्नलिखित राष्ट्रीय व्यावसायिक मानकों (NOS) के साथ गठबंधन किया गया है:

1. SSC/N1301: सॉफ्टवेयर उत्पादों/ एप्लिकेशन / मॉड्यूलों के लिए डिजाइन परीक्षण
2. SSC/N1302: सॉफ़्टवेयर उत्पादों/ एप्लिकेशन /मॉड्यूल्स पर स्वचालित परीक्षण करें
3. SSC/N1303: सॉफ्टवेयर उत्पादों/एप्लिकेशन/मॉड्यूल पर मैन्युअल परीक्षण करें
4. SSC/N9014: लिंग संवेदनशीलता, पीडब्ल्यूडी (व्यक्ति/विकलांग लोग) संवेदनशीलता और हरियाली को लागू करना और सुधारना
5. DGT/VSQ/N0102: Employability Skills (60 Hours)

हैंडबुक को प्रासंगिक QP की सामग्री के आधार पर उचित संख्या में इकाइयों और उप-इकाइयों में विभाजित किया गया है। हमें उम्मीद है कि यह प्रतिभागियों के लिए आसान और संरचित सीखने की सुविधा प्रदान करेगा, जिससे उन्हें उन्नत ज्ञान और कौशल प्राप्त करने की अनुमति मिलेगी।

Symbols Used


Key Learning Outcomes


Exercise


Notes


Unit Objectives


Activitiy

विषय-सूची

S.N.	मॉड्यूल और यूनिट	पृष्ठ संख्या
1.	**IT-ITeS/IT समर्थन सेवा उद्योग - एक परिचय**	**1**
	यूनिट 1.1 - भारतीय आईटी-आईटीईएस उद्योग का अवलोकन	3
	यूनिट 1.2 - परीक्षण इंजीनियरों और नौकरी जिम्मेदारियों के लिए कैरियर के अवसर	9
2.	**गुणवत्ता परीक्षण की अवधारणा और सिद्धांत (एसएससी/एन1301)**	**21**
	यूनिट 2.1 - गुणवत्ता परीक्षण की अवधारणा और सिद्धांत	23
3.	**सॉफ्टवेयर उत्पादों/ऍप्लिकेशन्स/मॉड्यूल के लिए डिजाइन परीक्षण (एसएससी/एन1301)**	**43**
	यूनिट 3.1 - सॉफ्टवेयर उत्पादों/ऍप्लिकेशन्स/मॉड्यूल के लिए डिजाइन परीक्षण	45
4.	**सॉफ्टवेयर उत्पादों/ऍप्लिकेशन्स/मॉड्यूल (एसएससी/एन1302) पर स्वचालित परीक्षण करें**	**70**
	यूनिट 4.1 - सॉफ्टवेयर उत्पादों/ऍप्लिकेशन्स/मॉड्यूलों पर स्वचालित परीक्षण करना	72
5.	**परियोजनाओं की गुणवत्ता आश्वासन में योगदान (SSC/N1302)**	**90**
	यूनिट 5.1 - परियोजनाओं की गुणवत्ता आश्वासन में योगदान	92
6.	**सॉफ्टवेयर ऍप्लिकेशन्स के लिए मुख्य संकेतक (SSC/N1303)**	**110**
	यूनिट 6.1 - सॉफ्टवेयर ऍप्लिकेशन्स के लिए मुख्य संकेतक	112
7.	**मैनुअल टेस्ट के लिए तकनीकी कौशल (SSC/N1303)**	**128**
	यूनिट 7.1 - घटनाओं से निपटने के लिए तकनीकी कौशल	130
8.	**सॉफ्टवेयर उत्पादों/ऍप्लिकेशन्स/मॉड्यूल (SSC/N1303) पर मैनुअल परीक्षण करें**	**155**
	यूनिट 8.1 - सॉफ्टवेयर उत्पादों/ऍप्लिकेशन्स/मॉड्यूलों पर मैनुअल परीक्षण करना	157
9.	**एक समावेशी, पर्यावरणीय रूप से स्थायी कार्यस्थल बनाए रखें (SSC/N9014)**	**175**
	यूनिट 9.1 - धारणीय व्यवहार	177
	यूनिट 9.2 - विविधता का सम्मान करें और समानता को बढ़ावा देने के लिए प्रथाओं को मजबूत करें	192
10.	**Employability Skills (DGT/VSQ/N0102)**	**205**
	Employability Skills is available at the following locaon : hps://www.skillindiadigital.gov.in/content/list Scan the QR code below to access the eboo	
11.	**अनुलग्नक**	**207**

Skill India
कौशल भारत-कुशल भारत

सत्यमेव जयते
GOVERNMENT OF INDIA
MINISTRY OF SKILL DEVELOPMENT
& ENTREPRENEURSHIP

N·S·D·C
National
Skill Development
Corporation
Transforming the skill landscape

1. IT-ITeS/IT समर्थन सेवा उद्योग - एक परिचय

IT - ITeS SSC
nasscom

यूनिट 1.1 - भारतीय IT-ITES उद्योग का अवलोकन
यूनिट 1.2- परीक्षण इंजीनियरों और नौकरी की जिम्मेदारियों के लिए कैरियर के अवसर

प्रमुख सीखने के परिणाम

इस मॉड्यूल के अंत तक, प्रतिभागी निम्न में सक्षम होंगे:

1. सॉफ्टवेयर उत्पाद विकास उद्योग में उपयोग किए जाने वाले विभिन्न वितरण मॉडल की व्याख्या करें।

यूनिट 1.1: भारतीय IT-ITES/IT उद्योग का अवलोकन

यूनिट के उद्देश्य

इस यूनिट के अंत तक, प्रतिभागी निम्न में सक्षम होंगे:

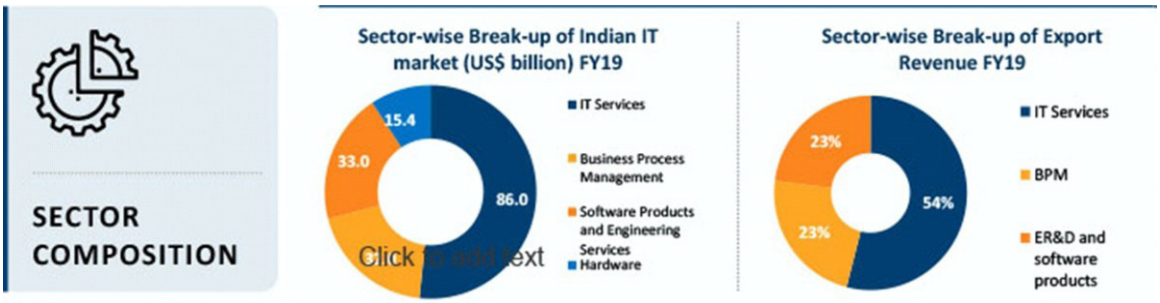
1. IT-ITES क्षेत्र की प्रासंगिकता की व्याख्या कीजिए।
2. IT-ITES उद्योग के भविष्य की रूपरेखा तैयार करें।
3. IT-ITeS/समर्थन सेवाओं से संबंधित डेटा, साक्ष्य और लेख एकत्र करने के लिए इंटरनेट खोज का संचालन करें।

1.1.1 भारत का IT-ITES/टेस्ट इंजीनियर उद्योग

- सूचना प्रौद्योगिकी (IT), सूचना प्रौद्योगिकी समथत सेवाएं (ITES)/व्यवसाय प्रक्रिया प्रबंधन (BPM) भारतीय अर्थव्यवस्था के लिए महत्वपूर्ण हैं।
- IT और BPM बाजार भारत के सकल घरेलू उत्पाद का 9.3% और वैश्विक आउटसोर्सिंग बाजार का 56% है।
- भारत का IT और व्यापार सेवा बाजार 2025 तक 19.93 बिलियन अमेरिकी डॉलर तक पहुंचने का अनुमान है।
- एक अनुमान के अनुसार, भारत में IT खर्च 2021 में 81.89 बिलियन अमेरिकी डॉलर से बढ़कर 2022 में 101.8 बिलियन अमेरिकी डॉलर होने का अनुमान है।
- भारत का IT और BPM उद्योग बैंकिंग, वित्तीय सेवाओं और बीमा (बीएफएसआई) क्षेत्र, दूरसंचार और खुदरा जैसे कार्यक्षेत्रों में अच्छी तरह से विविध है।
- FY21 में, भारत प्रौद्योगिकी सहित सभी वर्टिकल में 608,000 क्लाउड विशेषज्ञों के साथ दुनिया भर में तीसरे स्थान पर रहा।
- भारत में कंप्यूटर सॉफ्टवेयर और हार्डवेयर क्षेत्र ने अप्रैल 2000 और दिसंबर 2021 के बीच 81.31 बिलियन अमेरिकी डॉलर के संचयी प्रत्यक्ष विदेशी निवेश (FDI) प्रवाह को आकर्षित किया।
- IT कंपनियां देश के संगठित क्षेत्र में शीर्ष नियोक्ताओं में से एक हैं।

स्रोत: www.ibef.org/industry/information-technology-india

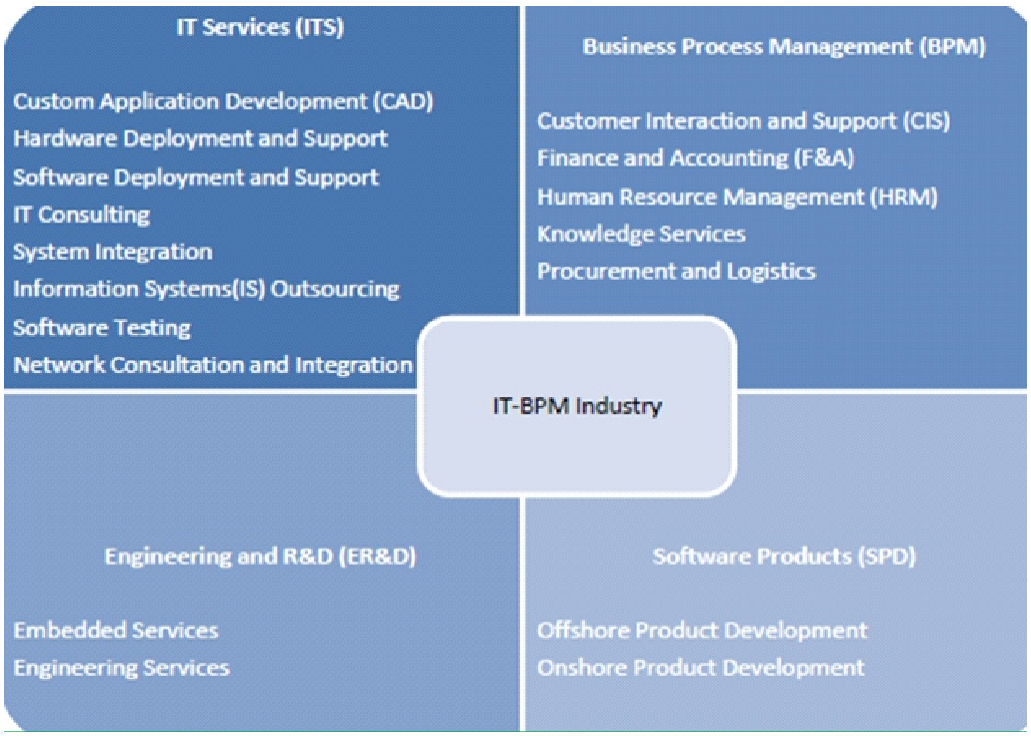
सेक्टर संरचना



स्रोत: www.ibef.org/industry/information-technology-india/infographi

चित्र 1.1.1 भारतीय IT बाजार की सेक्टर संरचना

यह देखा गया है कि IT सेवाओं और ITES-BPO उद्योगों ने भारतीय अर्थव्यवस्था के विकास को प्रभावित किया है। ITES उद्योग देश की सबसे बड़ी सफलता की कहानियों में से एक बन गया है, जिसने इसे सूचना प्रौद्योगिकी (IT) और बिजनेस प्रोसेस आउटसोर्सिंग (BPO) में अग्रणी के रूप में दुनिया भर के नक्शे पर रखा है। हर तरह से, भारतीय सूचना प्रौद्योगिकी (IT) और सूचना प्रौद्योगिकी-सक्षम सेवा (ITES) उद्योग आपस में जुड़े हुए हैं। उद्योग ने न केवल भारत की वैश्विक छवि में सुधार किया है। हालांकि, इसने आर्थिक प्रगति को भी बढ़ावा दिया है और सामाजिक परिवर्तन में महत्वपूर्ण योगदान दिया है। अपनी कम लागत, बड़े संसाधन पूल और क्षमता के साथ, भारत के पास तेजी से बढ़ते बाजार में टैप करने का अवसर है।



चित्र 1.1.2 IT-BPM उद्योग की संरचना

1.1.2 IT-ITES उद्योग

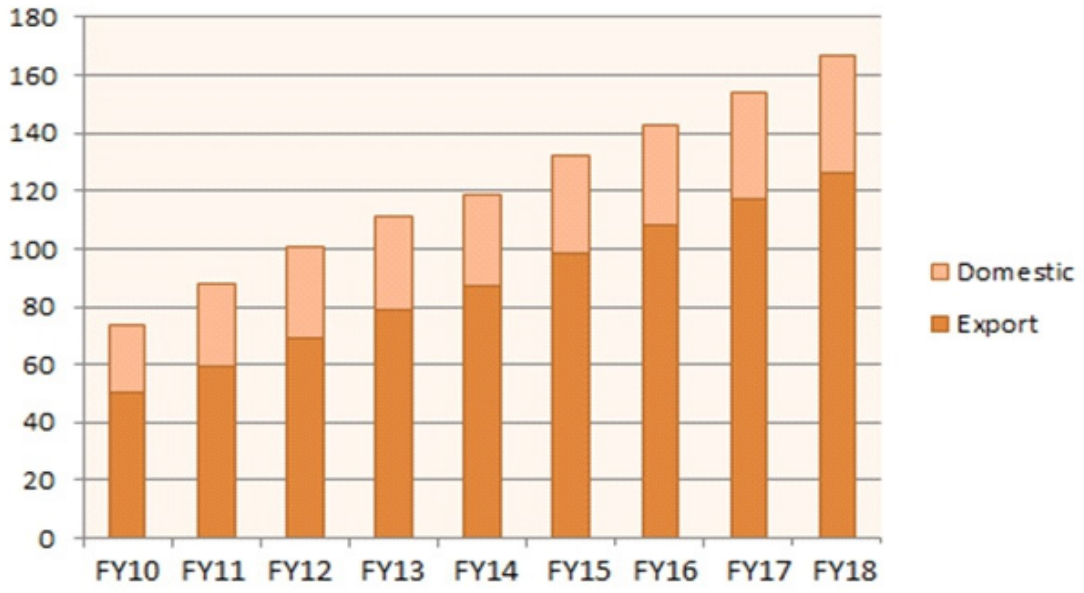
IT-ITES या सूचना प्रौद्योगिकी - सूचना प्रौद्योगिकी सक्षम सेवाओं में मुख्य रूप से संगठन की दक्षता में सुधार के लिए सूचना प्रौद्योगिकी के विभिन्न संचालन और उपयोग शामिल हैं। दो घटक IT उद्योग के लिए मजबूत स्तंभ के रूप में खड़े हैं। वो हैं:

- IT सेवाएँ
- BPO (बिजनेस प्रोसेस आउटसोर्सिंग)

भारतीय IT क्षेत्र से संबंधित महत्वपूर्ण जानकारी

- भारत के कुल प्रत्यक्ष विदेशी निवेश हिस्से के अनुसार, वैश्विक स्तर पर भारतीय IT क्षेत्र की रैंक # 3 है। (स्रोत: https://telanganatoday.com/it-sector-ranks-3rd-indias-fdi).
- सूचना प्रौद्योगिकी क्षेत्र में भारतीय उद्यम निवेशों और निजी इक्विटी का योगदान लगभग 37% है।

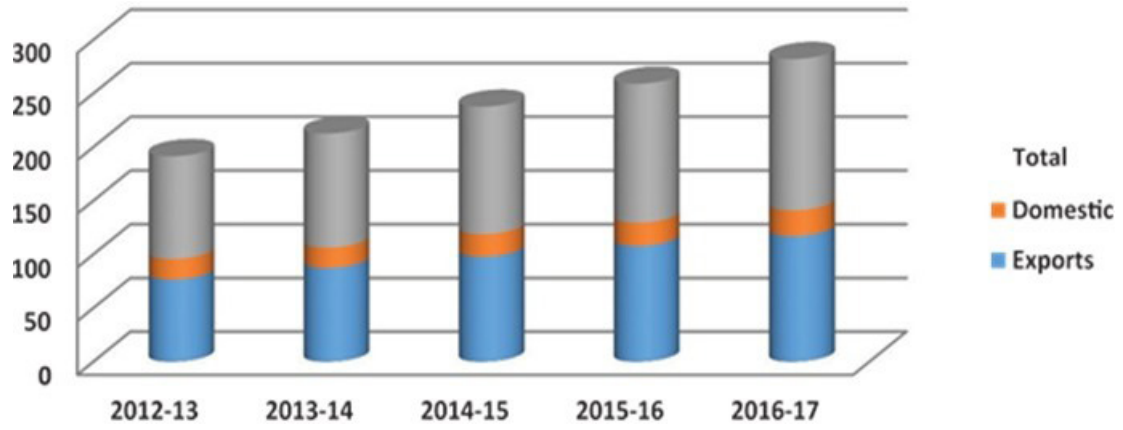
- भारतीय GDP में इस क्षेत्र का अंतिम रिकॉर्ड योगदान 7.7 प्रतिशत (2017-2018) था।
- 2017 में भारतीय IT क्षेत्र से कुल राजस्व लगभग 160 बिलियन अमेरिकी डॉलर होने का अनुमान लगाया गया था। (स्रोत: nasscom)।
- अगर अमेरिका से तुलना की जाए तो भारतीय IT और ITES सेक्टर करीब 5 से 6 गुना सस्ता है और कंपनियों को कम लागत का फायदा देता है।
- भारतीय IT फर्मों के वितरण केंद्र अच्छी तरह से विविध हैं और विश्व स्तर पर फैले हुए हैं। इन वर्टिकल में रिटेल, टेलीकॉम और BFSI (बैंकिंग, फाइनेंशियल सर्विसेज और इंश्योरेंस) शामिल हैं।



चित्र 1.1.3 भारत में भारतीय IT-ITES उद्योग का बाजार आकार

1.1.3 आउटलुक और - IT-ITES उद्योग का भविष्य

- यह खंड समृद्धि की भविष्यवाणियों पर प्रकाश डालता है कि IT उद्योग आने वाले वर्षों में खुद को देखता है।
- IT उद्योग पिछले 3 - 5 वर्षों को अपने "स्वर्णिम काल" के रूप में देखता है।
- यह IT-BPM क्षेत्र में वृद्धि के कारण है, वित्त वर्ष 17 (वित्तीय वर्ष 2017) में CAGR या चक्रवृद्धि वार्षिक वृद्धि दर बढ़कर 11.14% हो गई।
- इस प्रतिशत वृद्धि का निरूपण केवल 7 वित्तीय वर्षों (वित्तीय वर्ष 10-17) में 74 बिलियन अमेरिकी डॉलर से बढ़कर 154 बिलियन अमेरिकी डॉलर हो गया। यह मूल रूप से अन्य समय की तुलना में 3 - 4 गुना अधिक है।
- सूचना प्रौद्योगिकी क्षेत्र में निरंतर और सतत वृद्धि के अनुसार, वर्ष 2025 तक उद्योग की वृद्धि 350 मिलियन अमरीकी डालर होने का अनुमान है।
- IT उद्योग में रोजगार वृद्धि में 7-8% की वृद्धि होने का अनुमान है।



चित्र 1.1.4 पिछले 5 वर्षों में भारतीय ITES क्षेत्र की राजस्व वृद्धि (स्रोत: nasscom) (सभी राशियां अरबों अमेरिकी डॉलर में)

1.1.4 IT-ITeS/टेस्ट इंजीनियर्स उद्योग के बारे में इंटरनेट पर खोजें

1. **एंड्रॉइड/टैबलेट**

- Android फ़ोन या टैबलेट पर, Chrome ऐप्लिकेशन Chrome खोलें.
- Address Bar में, टाइप करें IT-ITES/टेस्ट इंजीनियर्स उद्योग और खोजें।
- परिणाम टैप करें, जाएं या जारी रखें जारी रखें।

सलाह: एक प्रकार के रूप में, किसी को वेब और ऐप्लिकेशन गतिविधि के आधार पर सुझाव मिल सकते हैं. उपयोगकर्ता खोज इतिहास से अलग-अलग सुझाव हटा सकते हैं या गतिविधि के आधार पर सुझावों के अनुभागों को छिपा सकते हैं, जब वे दिखाई देते हैं।

2. कंप्यूटर

- कंप्यूटर पर, Chrome खोले
- Address bar में, **IT-ITeS/BPM** उद्योग खोज दर्ज करें।
- कोई परिणाम चुनें या Enter दबाएं.

सलाह: एक प्रकार के रूप में, किसी को वेब और ऐप्लिकेशन गतिविधि के आधार पर सुझाव मिल सकते हैं. उपयोगकर्ता खोज इतिहास से अलग-अलग सुझाव हटा सकते हैं या गतिविधि के आधार पर सुझावों के अनुभागों को छिपा सकते हैं, जब वे दिखाई देते हैं।

एक्टिविटी

कॉन-वीडियो सत्र

- ट्रेनर इस सत्र के दौरान एक वीडियो चलाएगा।
- वीडियो IT उद्योग के अवलोकन के बारे में होगा।
- वीडियो के लिए YouTube लिंक है www.youtube.com/watch?v=Gk9dcfR7Oec&t=54s (वीडियो सौजन्य: स्क्रोच)।
- प्रशिक्षु पिन-ड्रॉप मौन के साथ वीडियो का निरीक्षण करेंगे।
- वे वीडियो से पॉइंटर्स को नोट कर सकते हैं जो उन्हें प्रासंगिक लग सकते हैं।
- प्रशिक्षु कक्षा में शिष्टाचार बनाए रखेंगे और कक्षा में बात नहीं करेंगे, कानाफूसी नहीं करेंगे या चर्चा नहीं करेंगे।
- किसी भी प्रश्न या भ्रम के मामले में, प्रशिक्षु अपनी नोटबुक में लिख लेंगे।

चर्चा सत्र

- प्रशिक्षक पूछताछ करेगा कि क्या प्रशिक्षुओं ने अध्याय को समझा है। इस गतिविधि में, प्रशिक्षक प्रशिक्षु से विषय से संबंधित कुछ प्रश्न पूछेगा।
- प्रशिक्षु विषय की अपनी समझ के अनुसार प्रश्नों का उत्तर देंगे।
- यदि छात्रों के कोई प्रश्न या चिंताएँ हैं, तो वे उन्हें प्रशिक्षक के सामने प्रस्तुत कर सकते हैं।
- प्रशिक्षक यह सुनिश्चित करेगा कि वह प्रशिक्षुओं द्वारा रखे गए प्रश्नों के उचित उत्तर दे सके।

नोट्स

यूनिट 1.2: परीक्षण इंजीनियरों और नौकरी की जिम्मेदारियों के लिए कैरियर के अवसर

यूनिट के उद्देश्य

इस यूनिट के अंत तक, प्रतिभागी निम्न में सक्षम होंगे:

1. एक टेस्ट इंजीनियर की भूमिका और जिम्मेदारियों पर चर्चा करें।
2. उन गुणों का वर्णन करें जो एक टेस्ट इंजीनियर के पास होने चाहिए।
3. एक टेस्ट इंजीनियर के लिए कैरियर पथ निर्धारित करें।
4. परीक्षण और गुणवत्ता आश्वासन सेवाओं के लिए प्रमुख ऍप्लिकेशन्स को वर्गीकृत करें।

1.2.1 परिचय

किसी भी सॉफ्टवेयर उत्पाद या एप्लिकेशन की गुणवत्ता का उसकी सफलता पर महत्वपूर्ण प्रभाव पड़ता है। आज, परीक्षण को किसी भी उत्पाद की गुणवत्ता की गारंटी देने का सबसे अच्छा तरीका माना जाता है। गुणवत्ता परीक्षण परियोजना के पुनर्विक्रय के व्यापक प्रभावों को काफी कम कर सकता है, जिसमें बजट बढ़ाने और शेड्यूल को पीछे धकेलने की क्षमता है। व्यवसायों पर कम समय में जटिल एप्लिकेशन बनाने का दबाव होता है, जिससे परीक्षण की आवश्यकता बढ़ जाती है। परीक्षण एक प्रकार की जांच है जिसका उपयोग यह निर्धारित करने के लिए किया जाता है कि कोई सॉफ़्टवेयर सेवा या उत्पाद कितनी अच्छी तरह काम करता है। इसके अतिरिक्त, यह निर्धारित करने की प्रक्रिया है कि कोई उत्पाद सही है या नहीं और यह कितनी अच्छी तरह कार्य करता है।

उत्पाद दोषों की पहचान करने के लिए परीक्षण प्रक्रिया में तुलना की एक विधि का उपयोग किया जाता है। किसी दिए गए उत्पाद के व्यवहार और स्थिति की तुलना मानकों के एक सेट के खिलाफ की जाती है, जिसमें उत्पाद के विनिर्देश, अनुबंध और पिछले पुनरावृत्तियां शामिल हो सकती हैं। सॉफ्टवेयर परीक्षण विसंगतियों, खामियों या त्रुटियों को खोजने के लिए एक क्रमिक और पुनरावृत्ति प्रक्रिया है। मायर्स के अनुसार, "परीक्षण त्रुटियों को खोजने के इरादे से एक कार्यक्रम चलाने की प्रक्रिया है। IEEE 83a के अनुसार, सॉफ्टवेयर परीक्षण, किसी सिस्टम या सिस्टम घटक को मैन्युअल रूप से या स्वचालित रूप से व्यायाम या मूल्यांकन करने की प्रक्रिया है ताकि यह सुनिश्चित हो सके कि यह पूर्व निर्धारित आवश्यकताओं का अनुपालन करता है।

सॉफ्टवेयर परीक्षण क्यों

इंसान हर समय गलतियाँ करता है !!

"सॉफ्टवेयर परीक्षण वास्तव में उन खामियों और गलतियों को उजागर करने के लिए आवश्यक है जो विकास चरणों के दौरान किए गए थे। हम इंसान अपने काम में अपनी त्रुटियों को पहचानने में असमर्थ हैं। हमें किसी और से हमारे काम की समीक्षा करने के लिए कहना चाहिए क्योंकि वे हमारे द्वारा की गई त्रुटियों को पकड़ सकते हैं। इसी तरह, सॉफ्टवेयर डेवलपर्स अपने द्वारा बनाए गए प्रोग्राम या एप्लिकेशन में विसंगतियों को नोटिस नहीं कर सकते हैं कि सॉफ्टवेयर टेस्ट इंजीनियर के रूप में जाना जाने वाला विभाग स्पॉट कर सकता है।

यह उच्च गुणवत्ता वाले सॉफ़्टवेयर का उपयोग करने के लिए समय और पैसा बचाता है। सॉफ्टवेयर में बग होने की संभावना कम होगी, जो परीक्षण और रखरखाव चरणों के दौरान समय खाली कर देगा। कम रखरखाव लागत और अधिक ग्राहक संतुष्टि अधिक विश्वसनीयता के दो लाभ हैं। क्योंकि सॉफ्टवेयर लागत का एक महत्वपूर्ण हिस्सा रखरखाव के लिए आवंटित किया जाता है, परियोजना की कुल लागत तुलनीय पहल से कम होने की संभावना होगी। परीक्षण आवश्यक है क्योंकि सॉफ्टवेयर बग महंगे या खतरनाक भी हो सकते हैं। इतिहास ऐसे उदाहरणों से भरा पड़ा है जहां सॉफ्टवेयर बग ने संभावित रूप से धन और जीवन की हानि की है।

यहां दो उदाहरण दिए गए हैं जो दिखाते हैं कि सॉफ्टवेयर की गुणवत्ता कितनी महत्वपूर्ण है:

- एयरबैग संवेदी डिटेक्टरों के साथ सॉफ्टवेयर मुद्दों के कारण, निसान ने सड़क से 1 मिलियन से अधिक वाहनों को वापस बुला लिया। इस सॉफ्टवेयर की खराबी के परिणामस्वरूप, दो दुर्घटनाओं की सूचना मिली है।
- 26 अप्रैल, 1994 को, एक चीन एयरलाइंस एयरबस ए 300 एक सॉफ्टवेयर त्रुटि के परिणामस्वरूप दुर्घटनाग्रस्त हो गया, जिसमें 264 अनजाने में पीड़ित मारे गए।

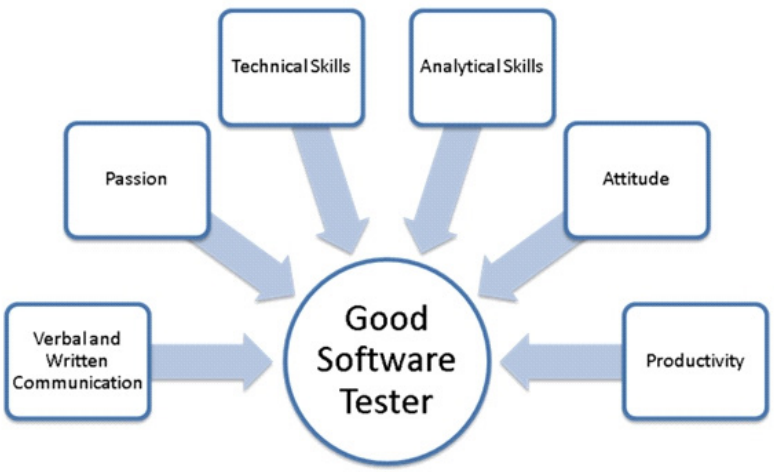
1.2.2 एक टेस्ट इंजीनियर की भूमिकाएं और जिम्मेदारियां

- **आवश्यकता विश्लेषण**. एक परीक्षक को सबसे पहले ग्राहक द्वारा प्रदान की गई आवश्यकताओं का विश्लेषण करना चाहिए। परीक्षक के कर्तव्यों में जिम्मेदारी की आवश्यकताओं और प्रासंगिक क्षेत्रों को समझना, एक क्वेरी सूची बनाना और इसे टेस्ट लीड के साथ साझा करना शामिल है।
- **परीक्षण प्रयास अनुमान।** परीक्षक से परीक्षण योजना बैठक के दौरान किए जाने वाले कार्यों की बारीकियों को समझने की उम्मीद है। इसके अतिरिक्त, उसे अनुमान लगाना चाहिए कि कार्य को प्रभावी ढंग से पूरा करने में कितना समय लगेगा।
- **परीक्षण मामलों के दस्तावेज़ीकरण।** परीक्षक से उस कार्य के लिए परीक्षण मामलों को रिकॉर्ड करने की उम्मीद की जाती है जिसे परिभाषित किया गया है और जिसके लिए मॉड्यूल सौंपे गए हैं। परीक्षण मामले या परीक्षण परिदृश्य संगठन और विकास पद्धति के आधार पर एक विशिष्ट प्रारूप में बनाए जाते हैं।
- **रिपोर्टिंग और ट्रैकिंग दोष।** परीक्षण कार्यों को करते समय पाए गए किसी भी बग की रिपोर्ट करना महत्वपूर्ण है। फिर, प्रत्येक संगठन दोष रिपोर्टिंग के लिए टेम्पलेट्स और उपकरणों का एक अनूठा सेट नियोजित करता है। परीक्षक को इस बात की जानकारी होनी चाहिए कि कोई विशेष उपकरण कैसे कार्य करता है और उसे पूरी तरह से दोष रिपोर्ट प्रदान करनी चाहिए। परीक्षक को किसी भी रिपोर्ट की गई खामियों पर नज़र रखनी चाहिए और यह सुनिश्चित करना चाहिए कि वे कितने महत्वपूर्ण हैं, इसके आधार पर वे तय हो गए हैं।
- **सुधार क्षेत्रों की सूची।** एक परीक्षक की अपेक्षा उत्पाद की उपयोगिता को बढ़ाने के लिए सुझाव देना है क्योंकि वे उत्पाद पर आंखों का एक नया सेट हैं।
- **परीक्षण लीड / QA लीड को रिपोर्टिंग।** परीक्षक को परीक्षण लीड को एक दैनिक स्थिति रिपोर्ट भेजने की आवश्यकता होती है जो परीक्षण गतिविधियों और किसी भी प्रासंगिक विकास को रेखांकित करती है। एक दैनिक स्थिति रिपोर्ट एक उपकरण है जिसका उपयोग जूनियर परीक्षक परीक्षण लीड को उनके द्वारा पूरा किए गए कार्य के बारे में सूचित करने के लिए कर सकते हैं।
- **रिपोर्ट :** सबसे अधिक बार, टेस्ट लीड (संगठनात्मक संरचना के आधार पर)

1.2.3 टेस्ट इंजीनियर के रूप में आवश्यक ज्ञान और कौशल

सॉफ्टवेयर परीक्षक बनने के लिए आवश्यक तकनीकी और गैर-तकनीकी क्षमताओं और दक्षताओं को नीचे सूचीबद्ध किया गया है।

- **गैर-तकनीकी कौशल:** एक सफल सॉफ्टवेयर गुणवत्ता परीक्षक बनने के लिए, आपके पास निम्नलिखित क्षमताएं होनी चाहिए। नीचे दी गई चेकलिस्ट के खिलाफ अपने कौशल सेट की जांच करें कि क्या सॉफ्टवेयर परीक्षण आपके लिए एक संभावना है:
- **विश्लेषणात्मक कौशल:** एक अच्छे सॉफ्टवेयर टेस्टर के लिए तेज विश्लेषणात्मक कौशल आवश्यक है। एक जटिल सॉफ्टवेयर सिस्टम के लिए परीक्षण मामलों को बेहतर ढंग से समझने और विकसित करने के लिए, विश्लेषणात्मक क्षमताओं की आवश्यकता होगी। यदि आप अनिश्चित हैं कि आपके पास मजबूत विश्लेषणात्मक क्षमताएं हैं, तो इस लिंक पर क्लिक करें। यदि आप केवल एक समस्या को हल कर सकते हैं, तो आपके पास उत्कृष्ट विश्लेषणात्मक क्षमताएं हैं।
- **संचार कौशल:** एक अच्छा सॉफ्टवेयर परीक्षक मौखिक और लिखित दोनों में संवाद करने में सक्षम होना चाहिए। सॉफ्टवेयर परीक्षकों को परीक्षण मामलों / योजनाओं, परीक्षण रणनीतियों, बग रिपोर्ट आदि जैसे परीक्षण कलाकृतियों का निर्माण करना चाहिए जो पढ़ने और समझने में आसान हों। डेवलपर्स से निपटने के लिए एक निश्चित मात्रा में चातुर्य और विवेक की आवश्यकता होगी (बग या किसी अन्य मुद्दे के मामले में)।
- **समय प्रबंधन और संगठन कौशल:** परीक्षण कभी-कभी एक चुनौतीपूर्ण कार्य हो सकता है, खासकर जब नया कोड जारी किया जा रहा हो। एक सॉफ्टवेयर परीक्षक को अपने कार्यभार को प्रभावी ढंग से प्रबंधित करने, अत्यधिक उत्पादक होने और उत्कृष्ट समय प्रबंधन और संगठन कौशल रखने की आवश्यकता होती है।
- **महान एट्यूड:** सॉफ्टवेयर का प्रभावी ढंग से परीक्षण करने के लिए आपको एक शांत रवैये की आवश्यकता है। प्रक्रियाओं को बेहतर बनाने के तरीके सीखने और सुझाव देने की उत्सुकता, साथ ही साथ "ब्रेक टू ब्रेक" मानसिकता भी। क्योंकि प्रौद्योगिकियां सॉफ्टवेयर उद्योग में इतनी तेज़ी से आगे बढ़ती हैं, एक अच्छे सॉफ्टवेयर परीक्षक को अपने तकनीकी सॉफ्टवेयर परीक्षण कौशल को अद्यतन करके बदलती प्रौद्योगिकियों के साथ रहना चाहिए। आपके कार्यों को स्वतंत्रता का एक निश्चित स्तर दिखाना चाहिए, यह दिखाते हुए कि आप कार्य का स्वामित्व लेते हैं और इसे बहुत प्रत्यक्ष पर्यवेक्षण के बिना पूरा करते हैं।
- **जुनून:** इसमें उत्कृष्टता प्राप्त करने के लिए किसी को अपने पेशे या नौकरी के लिए महत्वपूर्ण मात्रा में जुनून की आवश्यकता होती है। एक सॉफ्टवेयर परीक्षक को उनके बारे में भावुक होना चाहिए कि वे क्या करते हैं। लेकिन अगर आपने पहले कभी सॉफ़्टवेयर का परीक्षण नहीं किया है, तो आप कैसे बता सकते हैं कि आप इसके बारे में भावुक हैं या नहीं? सरल अगर सॉफ्टवेयर परीक्षण किसी की रुचि पर कब्जा नहीं करता है, तो किसी को कुछ और करना चाहिए।



चित्र 1.2.1 सॉफ्टवेयर परीक्षक के गैर-तकनीकी कौशल

तकनीकी कौशल

तकनीकी उपकरणों में बहुत अधिक संभावनाएं हैं क्योंकि गैर-कार्यात्मक परीक्षण के लिए आवश्यक है कि ऍप्लिकेशन्स को प्रदर्शन के लिए परीक्षण किया जाए।

- **डेटाबेस / SQL का बुनियादी ज्ञान:** पृष्ठभूमि में, सॉफ्टवेयर सिस्टम में बहुत अधिक डेटा होता है। बैकएंड पर, यह जानकारी विभिन्न डेटाबेस में रखी जाती है, जिसमें Oracle, My SQL, आदि शामिल हैं। इसलिए, ऐसे उदाहरण होंगे जिनमें इस डेटा को सत्यापित करने की आवश्यकता होगी। यदि यह मामला है, तो सरल या जटिल SQL प्रश्नों का उपयोग यह निर्धारित करने के लिए किया जा सकता है कि बैकएंड डेटाबेस में सही डेटा संग्रहीत है या नहीं।
- **लिनक्स कमांड का बुनियादी ज्ञान:** वेब सेवाओं, डेटाबेस और एप्लिकेशन सर्वर सहित अधिकांश सॉफ़्टवेयर प्रोग्राम लिनक्स-आधारित कंप्यूटरों पर स्थापित होते हैं। इसलिए यह आवश्यक है कि परीक्षक लिनक्स कमांड से परिचित हों।
- **एक परीक्षण प्रबंधन उपकरण का ज्ञान और व्यावहारिक अनुभव:** सॉफ्टवेयर परीक्षण का एक अनिवार्य घटक परीक्षण प्रबंधन है। सॉफ्टवेयर परीक्षण प्रक्रिया उपयुक्त परीक्षण प्रबंधन तकनीकों के बिना सफल नहीं होगी। अपने परीक्षण-संबंधी कलाकृतियों का प्रबंधन परीक्षण प्रबंधन का सार है।

उदाहरण के लिए: आपकी टीम द्वारा विकसित किए गए सभी परीक्षण मामलों पर नज़र रखना टेस्टलिंक जैसे टूल की मदद से किया जा सकता **है।**

अतिरिक्त उपकरण उपलब्ध हैं जिनका उपयोग परीक्षण प्रबंधन के लिए किया जा सकता है। नतीजतन, इन उपकरणों का ज्ञान और उपयोग करने का अनुभव होना महत्वपूर्ण है क्योंकि उनका उपयोग अधिकांश व्यवसायों द्वारा किया जाता है।

- **दोष ट्रैकिंग उपकरण ज्ञान और व्यावहारिक अनुभव** -दोष जीवन चक्र और दोष ट्रैकिंग सॉफ्टवेयर परीक्षण के दो आवश्यक तत्व हैं। दोषों का उचित प्रबंधन और उनकी व्यवस्थित ट्रैकिंग नितांत आवश्यक है। दोष ट्रैकिंग की आवश्यकता इस तथ्य से उत्पन्न होती है कि प्रबंधकों, डेवलपर्स और परीक्षकों को सभी को किसी भी दोष के बारे में पता होना चाहिए। QC, Bugzilla, Jira, आदि सहित विभिन्न उपकरणों का उपयोग करके दोषों को लॉग किया जाता है।
- **स्वचालन उपकरण का ज्ञान और व्यावहारिक अनुभव:** मैन्युअल परीक्षण के कुछ वर्षों के बाद, यदि आप तय करते हैं कि आप "स्वचालन परीक्षक" बनना चाहते हैं, तो आपको एक उपकरण में एक विशेषज्ञ बनने और स्वचालन उपकरणों का व्यापक, व्यावहारिक ज्ञान प्राप्त करने की आवश्यकता है।
- यदि आप स्वचालन में नौकरी की तलाश कर रहे हैं, तो होने के साथ **किसी भी स्क्रिप्टिंग भाषा का ज्ञान, जैसे V B स्क्रिप्ट, Javascript, या C #, हमेशा फायदेमंद होता है।** स्वचालन में काम की तलाश में परीक्षकों के लिए V B स्क्रिप्ट, Javascript, या C # जैसी स्क्रिप्टिंग भाषाओं से परिचित होना हमेशा फायदेमंद होता है। इस तथ्य के बावजूद कि केवल सीमित संख्या में कंपनियां शेल / पर्ल स्क्रिप्टिंग को नियोजित करती हैं, इस कौशल के साथ परीक्षकों की काफी आवश्यकता है। एक बार फिर, यह उस व्यवसाय और उस व्यवसाय द्वारा नियोजित उपकरणों पर निर्भर करेगा।

1.2.4 एक टेस्ट इंजीनियर के लिए कैरियर मैप

एक टेस्ट इंजीनियर से एक वरिष्ठ टेस्ट इंजीनियर, टेस्ट लीड और परीक्षण प्रबंधक के रूप में आगे बढ़ सकता है; वैकल्पिक रूप से, कोई QA लीड, या QA मैनेजर बन सकता है। परीक्षण उपकरणों के लिए कई विकल्प उपलब्ध हैं। सॉफ्टवेयर परीक्षण में करियर में रुचि रखने वाले किसी भी व्यक्ति को निम्नलिखित पदों के बारे में सोचना चाहिए:

चित्र 1.2.2 एक टेस्ट इंजीनियर के लिए कैरियर मानचित्र

एक टेस्ट इंजीनियर के रूप में वैकल्पिक कैरियर ट्रैक एक बार जब आप पर्याप्त अभ्यास कर लेते हैं, तो आप निम्नलिखित विशेषज्ञताओं का पीछा कर सकते हैं।

- एक स्वचालन टेस्ट इंजीनियर के रूप में, मैन्युअल परीक्षण केस निष्पादन प्रक्रिया को स्वचालित करना आपकी ज़िम्मेदारी है क्योंकि ऐसा करना अन्यथा समय लेने वाला होगा। उपकरण में आईबीएम रेशनल रोबोट, सिल्क परफॉर्मर और क्यू टी पी शामिल थे।
- प्रदर्शन परीक्षण: अन्य बातों के अलावा, एक प्रदर्शन टेस्ट इंजीनियर के रूप में आपका काम यह पता लगाना होगा कि कोई एप्लिकेशन कितनी जल्दी और कितना लोड का समर्थन कर सकता है।
- एक व्यापार विश्लेषक के अनुसार, डेवलपर्स पर परीक्षकों के मुख्य लाभों में से एक व्यवसाय की उनकी गहन समझ है। व्यवसाय विश्लेषण एक ऐसा करियर है जिसे परीक्षकों को आगे बढ़ाने पर विचार करना चाहिए। आप, एक व्यापार विश्लेषक, अपनी कंपनी के व्यवसाय मॉडल और वर्कफ़्लो का विश्लेषण और मूल्यांकन करेंगे। बीए के रूप में, आप इन वर्कफ़्लोज़ और मॉडल में प्रौद्योगिकी को एकीकृत करेंगे।

जैसा कि पहले ही उल्लेख किया गया था, एक टेस्ट इंजीनियर से एक वरिष्ठ टेस्ट इंजीनियर, टेस्ट लीड और परीक्षण प्रबंधक के रूप में आगे बढ़ सकता है; वैकल्पिक रूप से, कोई QA लीड, QA मैनेजर बन सकता है। पिछले दस वर्षों में एक सॉफ्टवेयर परीक्षक की प्रोफाइल काफी बदल गई है। कई एप्लिकेशन / उत्पाद कार्यान्वयन के लिए, सॉफ्टवेयर परीक्षण में करियर एक डील-ब्रेकर बन गया है, और व्यवसाय ने महसूस किया है कि रिलीज से पहले ऍप्लिकेशन्स के परीक्षण की संरचना करना कितना महत्वपूर्ण है। परीक्षण समुदाय के पास अब परीक्षण मामलों के सुस्त निष्पादक होने के बजाय चुनने के लिए कई प्रकार के कैरियर विकल्प हैं। परीक्षण उपकरण पक्ष में विकल्पों की एक विशाल सरणी है। एचपी से गुणवत्ता केंद्र, आईबीएम से सीक्यूटीएम, आदि जैसे परीक्षण प्रबंधन उपकरणों के अलावा, कई कार्यात्मक, प्रदर्शन और सुरक्षा परीक्षण उपकरण उपलब्ध हैं।

सुरक्षा और एसओए परीक्षकों जैसे विशेष ज्ञान की बढ़ती आवश्यकता है। परीक्षण स्वचालन क्षेत्रों में स्क्रिप्टिंग कौशल की कमी है, जिसमें V B, Java और अन्य टूल भाषाओं के साथ-साथ पर्ल, शेल, पायथन आदि जैसी स्क्रिप्टिंग भाषाएं शामिल हैं।

स्वचालन उपकरणों का आकलन करने, स्वचालन ढांचे को विकसित करने और पुन: प्रयोज्य घटकों का उत्पादन करने की क्षमता वाले तकनीकी संसाधनों की मांग है। अच्छे प्रदर्शन परीक्षक जो परीक्षण के परिणामों का विश्लेषण कर सकते हैं, बाधाओं का पता लगा सकते हैं, और ट्यूनिंग विधियों की सिफारिश कर सकते हैं, हमेशा मांग में होते हैं। एप्लिकेशन के डोमेन की मजबूत समझ होने से परीक्षण पेशेवरों का मूल्य बढ़ जाता है। BFSI, टेलीकॉम, हेल्थकेयर, मैन्युफैक्चरिंग और एम्बेडेड जैसे डोमेन लगातार सक्रिय हैं।

1.2.5 परीक्षण और गुणवत्ता आश्वासन सेवाओं के लिए आवेदन

तीन आम तौर पर स्वीकृत परीक्षण तकनीकें इस प्रकार हैं:

1. **ब्लैक बॉक्स:** भले ही इनपुट को आउटपुट में कैसे परिवर्तित किया जाए, ब्लैक-बॉक्स परीक्षण में अपेक्षित इनपुट और आउटपुट को ध्यान में रखते हुए AUT को उसके विनिर्देशों के विरुद्ध मान्य करना शामिल है। आंतरिक संगठन या कोड जो एप्लिकेशन के व्यावसायिक तर्क को लागू करता है, परीक्षकों के लिए कम से कम महत्वपूर्ण है।

   ब्लैक बॉक्स परीक्षण के लिए परीक्षण मामले बनाने के चार मुख्य तरीके इस प्रकार हैं:

   - B V A (सीमा मूल्य विश्लेषण)
   - EP (समतुल्यता विभाजन)
   - निर्णय तालिकाएँ
   - state Transition तालिकाएँ (और आरेख) ब्लैक बॉक्स परीक्षण आमतौर पर कार्यात्मक, गैर-कार्यात्मक और प्रतिगमन परीक्षण के लिए नियोजित किया जाता है

2. **व्हाइट बॉक्स:** इस पद्धति का मुख्य उद्देश्य यह पुष्टि करना है कि कोड या प्रोग्राम में एप्लिकेशन का व्यावसायिक तर्क कैसे लागू किया जाता है।
3. **ग्रे बॉक्स:** वास्तविकता में, यह ब्लैक बॉक्स और व्हाइट बॉक्स का एक संकर है। इस पद्धति में, परीक्षक मुख्य रूप से आवेदन का परीक्षण करने के लिए ब्लैक-बॉक्स दृष्टिकोण का उपयोग करता है। परीक्षण एक सफेद बॉक्स का उपयोग करके किया जाता है, हालांकि, किसी एप्लिकेशन के कुछ व्यवसाय-महत्वपूर्ण या कमजोर मॉड्यूल के लिए।

SQA तकनीक और एप्लीकेशन

गुणवत्ता आश्वासन (QA) परीक्षण वह प्रक्रिया है जिसका उपयोग आप यह सुनिश्चित करने के लिए करते हैं कि आपका उत्पाद आपके ग्राहकों के लिए सबसे अच्छा हो सकता है। सीधे शब्दों में कहें, गुणवत्ता आश्वासन (QA) आपके सॉफ़्टवेयर, उत्पाद या सेवा के साथ समस्याओं से बचने और आपके ग्राहकों के लिए सकारात्मक उपयोगकर्ता अनुभव की गारंटी देने के लिए उपयोग की जाने वाली विधियों को संदर्भित करता है।

SQA के लिए कई तकनीकें हैं।

- अक्सर उपयोग की जाने वाली मुख्य विधि ऑडिटिंग है। हालाँकि, कुछ अतिरिक्त तरीके भी हैं।

विभिन्न SQA तकनीकों में शामिल हैं:

- **ऑडिटिंग:** ऑडिट करने में कार्य उत्पादों और उनके साथ जाने वाली जानकारी को देखना शामिल है, यह देखने के लिए कि स्थापित मानक संचालन प्रक्रियाओं का पालन किया गया था या नहीं।
- **समीक्षा करना:** एक सभा जहां आंतरिक और बाहरी हितधारक सॉफ़्टवेयर उत्पाद की समीक्षा करते हैं और प्रतिक्रिया और अनुमोदन प्रदान करते हैं
- **कोड निरीक्षण:** बग खोजने और इसके बाद के चरणों में दोष वृद्धि को रोकने के लिए, सबसे औपचारिक प्रकार की समीक्षा स्थिर परीक्षण करती है। यह एक प्रशिक्षित मध्यस्थ या सहकर्मी द्वारा किया जाता है और दिशानिर्देशों, एक चेक लिस्ट और प्रवेश और निकास मानदंडों पर आधारित होता है। कोड का लेखक समीक्षक नहीं होना चाहिए।
- **नीचे सूचीबद्ध सॉफ्टवेयर डिजाइन के क्षेत्रों को देखने के लिए एक चेकलिस्ट का उपयोग करके डिजाइन निरीक्षण किया जाता है:**

- इंटरफ़ेस और कार्यात्मक विनिर्देशों
- मौलिक विनिर्देशों और डिजाइन
- आवश्यकताओं का पता लगाने की क्षमता
- इंटरफेस और संरचनाएं
- सम्मेलनों
- तर्कशास्त्र
- प्रदर्शन
- त्रुटि हैंडलिंग और पुनर्प्राप्ति
- परीक्षण क्षमता, एक्स्टेंसिबिलिटी
- युग्मन और सामंजस्य

- **सिमुलेशन**: एक सिमुलेशन एक उपकरण है जो वास्तविक दुनिया के परिदृश्य का अनुकरण करता है ताकि अध्ययन के तहत सिस्टम के व्यवहार की वस्तुतः जांच की जा सके।
- **कार्यात्मक परीक्षण:** यह एक गुणवत्ता आश्वासन तकनीक है जो यह जांचती है कि सिस्टम यह कैसे करता है, इस पर ध्यान केंद्रित किए बिना क्या पूरा करता है। इस तरह के ब्लैक बॉक्स परीक्षण का प्राथमिक उद्देश्य सिस्टम की विशेषताओं या विशिष्टताओं का मूल्यांकन करना है।
- **मानकीकरण:** गुणवत्ता नियंत्रण के लिए मानकीकरण आवश्यक है। यह अस्पष्टता और अटकलों को कम करके गुणवत्ता सुनिश्चित करता है।
- **स्थैतिक विश्लेषण:** इस प्रकार का सॉफ़्टवेयर विश्लेषण प्रोग्राम को चलाने की आवश्यकता के बिना एक स्वचालित उपकरण द्वारा किया जाता है। परमाणु, चिकित्सा और विमानन उद्योगों के लिए सॉफ्टवेयर गुणवत्ता आश्वासन अक्सर इस पद्धति को नियोजित करता है। स्थैतिक विश्लेषण विधियों में शामिल हैं, उदाहरण के लिए, सॉफ्टवेयर मेट्रिक्स और रिवर्स इंजीनियरिंग।
- **पूर्वाभ्यास:** एक सॉफ्टवेयर वॉकथ्रू, जिसे कोड वॉकथ्रू के रूप में भी जाना जाता है, एक प्रकार की सहकर्मी समीक्षा है जिसमें डेवलपर टीम के सदस्यों का नेतृत्व करता है क्योंकि वे उत्पाद की जांच करते हैं और प्रश्न पूछते हैं, समाधान प्रदान करते हैं, और किसी भी संभावित दोष, मानक उल्लंघन या अन्य समस्याओं के बारे में टिप्पणी करते हैं।
- **पथ परीक्षण:** इस सफेद बॉक्स परीक्षण पद्धति के साथ, पूर्ण शाखा कवरेज की गारंटी के लिए प्रत्येक स्वतंत्र पथ को कम से कम एक बार चलाया जाता है।
- **तनाव परीक्षण:** इस तरह का परीक्षण किसी सिस्टम की मजबूती को निर्धारित करने के लिए किया जाता है, इसे भारी भार, या ऐसी स्थितियों के माध्यम से रखा जाता है जो इसकी विशिष्ट सीमा से बाहर हैं।
- **सिक्स सिग्मा:** लगभग सही वस्तुओं और सेवाओं का उत्पादन सिक्स सिग्मा गुणवत्ता आश्वासन पद्धति का उद्देश्य है। सॉफ्टवेयर विकास सहित कई उद्योगों में इसका बड़े पैमाने पर उपयोग किया जाता है। सिक्स सिग्मा का प्राथमिक उद्देश्य सॉफ्टवेयर का उत्पादन करने के लिए प्रक्रिया में सुधार है जो 99.76% दोषों से मुक्त है।

कुछ आवश्यक QA उपकरण

- **परियोजना रणनीति** - माइक्रोसॉफ्ट वर्ड
- **परियोजना/स्प्रिंट योजना** -JIRA

- **संचार** - Slack, Microsoft Teams, Yammer
- **दस्तावेज़ रिपॉजिटरी** - SharePoint
- **टेस्ट रिपोजिटरी और दोष प्रबंधन** - JIRA
- **स्वचालन उपकरण** - Jenkins, Selenium, Cucumber, Java

1.2.6 सॉफ्टवेयर गुणवत्ता आश्वासन और सॉफ्टवेयर परीक्षण के बीच अंतर

निम्न तालिका सॉफ़्टवेयर गुणवत्ता आश्वासन (SQA) और सॉफ़्टवेयर परीक्षण के बीच अंतर के बारे में बताती है:

सॉफ्टवेयर गुणवत्ता आश्वासन (SQA)	सॉफ्टवेयर परीक्षण
गुणवत्ता सुनिश्चित करने वाली इंजीनियरिंग प्रक्रिया को सॉफ्टवेयर गुणवत्ता आश्वासन रूप में जाना जाता है	किसी उत्पाद का उपयोग करने वाली समस्याओं के लिए उपयोग के लिए जारी होने से पहले सॉफ्टवेयर परीक्षण किया जाता है
प्रक्रियाओं के आवेदन से संबंधित क्रियाएं शामिल हैं,	उत्पाद सत्यापन से संबंधित क्रियाएँ शामिल हैं, जैसे समीक्षा परीक्षण.
प्रक्रिया केंद्रित	उत्पाद केंद्रित
निवारक तकनीक	सुधारात्मक तकनीक
सक्रिय उपाय	प्रतिक्रियाशील उपाय
संगठन द्वारा उत्पादित भविष्य के सभी उत्पादों को SQA के दायरे में शामिल किया गया था।	जिस उत्पाद का परीक्षण किया जा रहा है वह सॉफ्टवेयर परीक्षण के दायरे में आता है।

सारांश

- IT और व्यापार सेवाओं के लिए भारतीय बाजार 2025 तक 19.93 बिलियन डॉलर तक पहुंचने की उम्मीद है।
- भारत के कुल प्रत्यक्ष विदेशी निवेश हिस्से के अनुसार, वैश्विक स्तर पर भारतीय IT क्षेत्र की रैंक # 3 है।
- सूचना प्रौद्योगिकी क्षेत्र में निरंतर और सतत वृद्धि के अनुसार, वर्ष 2025 तक उद्योग की वृद्धि 350 मिलियन अमरीकी डालर होने का अनुमान है।
- परीक्षण में त्रुटियों की पहचान करने के लक्ष्य के साथ एक कार्यक्रम चलाना शामिल है।
- एक टेस्ट इंजीनियर को यह सुनिश्चित करने के लिए स्वचालित और मैन्युअल परीक्षण दोनों चलाना चाहिए कि डेवलपर्स द्वारा विकसित सॉफ़्टवेयर इसके इच्छित उपयोग के लिए उपयुक्त है और सामान्य उपयोगकर्ताओं के लिए जारी किए जाने से पहले किसी भी बग या समस्या को ठीक किया गया है।

- एक टेस्ट इंजीनियर विभिन्न नौकरी भूमिकाओं में आगे बढ़ सकता है क्योंकि वे ज्ञान और अनुभव प्राप्त करते हैं।
- इस जॉब रोल में सफल होने के लिए प्रमुख रूप से कठिन कौशल और सॉफ्ट स्किल्स की आवश्यकता होती है।
- आम तौर पर स्वीकार किए जाने वाले तीन परीक्षण पद्धतियों में ब्लैक बॉक्स शामिल है, जिसका उपयोग अक्सर कार्यात्मक, गैर-कार्यात्मक और प्रतिगमन परीक्षण के लिए किया जाता है। ग्रे बॉक्स, जो ब्लैक बॉक्स और व्हाइट बॉक्स का एक संयोजन है, मुख्य रूप से यह मान्य करने से संबंधित है कि कोड या प्रोग्राम द्वारा एप्लिकेशन का व्यावसायिक तर्क कैसे लागू किया जाता है।
- जिन ऍप्लिकेशन्स पर टेस्ट इंजीनियर को काम करने की आवश्यकता होती है, वे जेनकिंस, सेलेनियम, cucumber, Java, माइक्रोसॉफ्ट वर्ड, स्लैक, माइक्रोसॉफ्ट टीम्स, यमर, शेयर प्वाइंट, जे आई आर ए जैसे हैं।

एक्टिविटी

कॉन-वीडियो सत्र

- Industry में।
- वीडियो एक टेस्ट इंजीनियर की नौकरी की भूमिका की एक झलक देगा।
- यू ट्यूब वीडियो के लिए लिंक है: https://www.youtube.com/watch?v=3eOd9NTRgJo
- प्रशिक्षु पिन ड्रॉप साइलेंस के साथ वीडियो का अवलोकन करेंगे।
- वे वीडियो से पॉइंटर्स को नोट कर सकते हैं जो उन्हें प्रासंगिक लग सकते हैं।
- प्रशिक्षु कक्षा में शिष्टाचार बनाए रखेंगे और कक्षा में बात नहीं करेंगे, कानाफूसी नहीं करेंगे या चर्चा नहीं करेंगे।

किसी भी प्रश्न या भ्रम के मामले में, प्रशिक्षु उन्हें अपनी नोटबुक में लिखेंगे।

चर्चा सत्र

- प्रशिक्षक पूछताछ करेगा कि क्या प्रशिक्षुओं ने अध्याय को समझा है।
- इस गतिविधि में, प्रशिक्षक प्रशिक्षु से विषय से संबंधित कुछ प्रश्न पूछेगा।
- प्रशिक्षु विषय की अपनी समझ के अनुसार प्रश्नों का उत्तर देंगे।
- यदि प्रशिक्षुओं के मन में प्रश्न और भ्रम हैं; वे ट्रेनर के सामने उन लोगों को सामने रख सकते हैं।
- प्रशिक्षक यह सुनिश्चित करेगा कि वह प्रशिक्षुओं द्वारा रखे गए प्रश्नों के उचित उत्तर दे सके।

अभ्यास

A. रिक्त स्थानों की पूर्ति कीजिए

1. IT उद्योग पिछले 3 - 5 वर्षों को अपने "" के रूप में देखता है।

   a. प्लेटिनम युग b. स्वर्णिम काल c. डायमंड बोनांजा

2. 2017 में IT क्षेत्र से कुल राजस्व लगभग होने का अनुमान लगाया गया था

   a. यूएस $ 160 बिलियन b. रु. 5 करोड़ c. 1 बिलियन अमेरिकी डॉलर

3. किसी भी अनसुलझी समस्या को उचित रूप से आगे बढ़ाने की आवश्यकता है

   a. प्रोग्रामर जन्म। b. आंतरिक टीमें c. पथरी

4. IT उद्योग में रोजगार वृद्धि में वृद्धि होने का अनुमान है:

   a. 15-50% b. 7-8% c. 55-80%

5. सॉफ्टवेयर परीक्षण एक प्रक्रिया है_____और_____
   एक विसंगति, एक दोष, या एक त्रुटि।

6. इंगित करें कि निम्न में से प्रत्येक दावा सही है या गलत:

   a. सॉफ्टवेयर परीक्षण से किसी प्रोग्राम या उत्पाद में कमजोर बिंदुओं का पता चलता है। ☐

   b. मायर्स ने अवलोकन किया कि परीक्षण त्रुटियों की पहचान करने के उद्देश्य से एक कार्यक्रम चलाने की प्रक्रिया है। ☐

7. परीक्षण का प्राथमिक उद्देश्य कसकर विनियमित स्थितियों में बार-बार उपयोग करके सॉफ्टवेयर सिस्टम की निर्भरता की पुष्टि करना है। सफ़ाई देना।

   _____

   _____

   _____

   _____

   _____

नोट्स

Skill India
कौशल भारत-कुशल भारत

सत्यमेव जयते
GOVERNMENT OF INDIA
MINISTRY OF SKILL DEVELOPMENT
& ENTREPRENEURSHIP

N·S·D·C
National
Skill Development
Corporation
Transforming the skill landscape

2. गुणवत्ता परीक्षण की अवधारणा और सिद्धांत

IT - ITeS SSC
nasscom

यूनिट 2.1 - गुणवत्ता परीक्षण की अवधारणा और सिद्धांत

SSC/N1301

प्रमुख सीखने के परिणाम

इस मॉड्यूल के अंत तक, प्रतिभागी निम्न में सक्षम होंगे:

1. किए जा रहे गुणवत्ता आश्वासन कार्य के उद्देश्यों और दायरे पर चर्चा करें।
2. डेटा के प्रकारों के आधार पर गुणवत्ता परीक्षण बनाम गुणवत्ता नियंत्रण के प्रमुख अंतरों पर चर्चा करें।

यूनिट 2.1: गुणवत्ता परीक्षण की अवधारणा और सिद्धांत

यूनिट के उद्देश्य

इस यूनिट के अंत तक, प्रतिभागी निम्न में सक्षम होंगे:

1. नियंत्रण, नौकरी प्रबंधन, पर्याप्त प्रक्रिया, प्रदर्शन और अखंडता मानदंडों से संबंधित गुणवत्ता आश्वासन के विभिन्न पहलुओं पर चर्चा करें।
2. नीतियों, मानकों, प्रक्रियाओं, प्रक्रियाओं और संस्करण नियंत्रण सहित गुणवत्ता परीक्षण के लिए प्रमुख आवश्यकताओं पर चर्चा करें।
3. विभिन्न ड्रोतों से एकत्रित व्यक्तिपरक एवं वस्तुनिष्ठ आँकड़ों के सिद्धांतों की विवेचना कीजिए।
4. अस्वीकृति सिद्धांत के आधार पर व्यक्तिपरक और उद्देश्य डेटा को वर्गीकृत करने का तरीका प्रदर्शित करें।
5. उद्देश्य और व्यक्तिपरक डेटा विश्लेषण के माध्यम से गुणवत्ता नियंत्रण करें।
6. गुणवत्ता प्रक्रिया पर कार्मिक अखंडता, आत्मविश्वास, संगठनात्मक संस्कृति, प्रेरणा, टीम भावना और गुणवत्ता संबंधों जैसे नरम तत्वों के प्रभाव का विश्लेषण करें।

2.1.1 QA परीक्षण क्या है?

गुणवत्ता आश्वासन (QA) परीक्षण वह प्रक्रिया है जिसका उपयोग यह सुनिश्चित करने के लिए किया जाता है कि आपका उत्पाद आपके ग्राहकों के लिए उच्चतम गुणवत्ता का है। सीधे शब्दों में कहें, गुणवत्ता आश्वासन (QA) आपके सॉफ़्टवेयर, उत्पाद या सेवा के साथ समस्याओं से बचने और आपके ग्राहकों के लिए सकारात्मक उपयोगकर्ता अनुभव की गारंटी देने के लिए उपयोग की जाने वाली विधियों को संदर्भित करता है।

गुणवत्ता आश्वासन का इतिहास

गुणवत्ता नियंत्रण की अवधारणा नई नहीं है। इसका एक स्पष्ट उदाहरण मध्य युग में पाया जा सकता है, जब गिल्ड ने अपने सदस्यों द्वारा पेश की जाने वाली वस्तुओं और सेवाओं के मानक को बनाए रखने के लिए सदस्यता के लिए आवश्यकताओं की स्थापना की।

रॉयल्टी को गुणवत्ता नियंत्रण की जिम्मेदारी भी दी गई थी, जैसे कि उनके युद्धपोतों को बनाए रखना। उन्होंने लंबी समुद्री यात्राओं की सुरक्षा सुनिश्चित करने के लिए नौसेनाओं के प्रशिक्षण और उपकरणों के मानकीकरण में भी योगदान दिया। औद्योगिक क्रांति के दौरान गुणवत्ता नियंत्रण में भी क्रांति आई। श्रमिक अब अपने स्वयं के काम की क्षमता के लिए पूरी तरह से जवाबदेह नहीं थे। फोरमैन और प्रबंधन को इस काम की देखरेख की जिम्मेदारी दी गई थी, और गुणवत्ता आश्वासन में विशेषज्ञता वाले सिस्टम और समूह बनाए गए थे।

सॉफ्टवेयर गुणवत्ता का विचार यह सुनिश्चित करने के लिए पेश किया गया था कि जारी किया गया सॉफ़्टवेयर सुरक्षित है और इरादा के अनुसार प्रदर्शन करता है। वाक्यांश "स्पष्ट या अंतर्निहित आवश्यकताओं और अपेक्षाओं के अनुरूप की डिग्री" का उपयोग अक्सर इसका वर्णन करने के लिए किया जाता है। ये तथाकथित स्पष्ट और अंतर्निहित अपेक्षाएं सॉफ्टवेयर गुणवत्ता के दो बुनियादी स्तरों के अनुरूप हैं:

- **कार्यात्मक** - डिजाइन दिशानिर्देशों और कार्यात्मक (स्पष्ट) आवश्यकताओं के लिए आइटम का पालन। सॉफ़्टवेयर का यह पहलू इस बात पर केंद्रित है कि उपयोगकर्ता द्वारा वास्तव में इसका उपयोग कैसे किया जाता है, जिसमें इसका प्रदर्शन, उपयोगिता और बग की कमी शामिल है।
- **गैर-कार्यात्मक** - डिजाइन दिशानिर्देशों और कार्यात्मक (स्पष्ट) आवश्यकताओं के लिए आइटम का पालन। सॉफ़्टवेयर का यह क्षेत्र इसके साथ उपयोगकर्ता की वास्तविक बातचीत पर केंद्रित है, जिसमें इसकी कार्यक्षमता, उपयोगिता और बग-मुक्त स्थिति शामिल है।

चित्र 2.1.1 सॉफ्टवेयर विकास प्रक्रिया में QA, QC और Testing
स्रोत:https://content.altexsoft.com/media/10/2016/qaqc-and-testing.png)

2.1.2 संस्करण नियंत्रण क्या है? गुणवत्ता आश्वासन में यह कैसे महत्वपूर्ण है?

एक संस्करण नियंत्रण प्रणाली सॉफ्टवेयर और संबंधित सूचना फ़ाइलों की पुनरावृत्तियों की एक श्रृंखला को संग्रहीत करने के लिए सुविधाएँ प्रदान करती है जिन्हें विकसित किया जा रहा है। संस्करण नियंत्रण उपकरणों के संदर्भ में, एक सिस्टम रिलीज़ संबंधित फ़ाइलों का एक समूह है। इन फ़ाइलों में दस्तावेज़ीकरण, स्रोत कोड, संकलित कोड, सॉफ़्टवेयर बनाने के लिए उपयोग किए जाने वाले टूल का एक संस्करण, साथ ही पर्यावरण के बारे में जानकारी शामिल हो सकती है। संस्करण नियंत्रण का मुख्य लक्ष्य एक संरचित और पता लगाने योग्य सॉफ्टवेयर विकास प्रक्रिया सुनिश्चित करके हर समय एक सॉफ्टवेयर सिस्टम के लिए एक अच्छी तरह से परिभाषित स्थिति बनाए रखना है जिसमें सभी परिवर्तनों को सावधानीपूर्वक प्रबंधित किया जाता है।

रिपॉजिटरी एक मुख्य स्थान है जिसमें अधिकांश संस्करण नियंत्रण उपकरणों के लिए सभी फाइलों की मास्टर कॉपी होती है। किसी भी व्यवसाय को डेटा बैकअप, कर्मचारी सहयोग और फ़ाइलों के कई पुनरावृत्तियों को संग्रहीत करने में समस्याएँ आ सकती हैं। एक कंपनी के सफल होने के लिए, यह जरूरी है कि इन सभी मुद्दों को हल किया जाए। इस स्थिति में एक संस्करण नियंत्रण प्रणाली तब आवश्यक है।

इन कंपनियों के सामने ये कुछ प्रमुख कठिनाइयाँ हैं:

- **सहयोग**. क्योंकि विभिन्न स्थानों में बहुत सारे लोग फैले हुए हैं, इसलिए किसी विशिष्ट कारण से संवाद करने की आवश्यकता हो सकती है, या विभिन्न क्षेत्रों के लोगों का एक समूह एक ही परियोजना पर काम कर सकता है।
- **संस्करणों को संग्रहीत करना।** परियोजना को कई संस्करणों में पूरा किया गया है, जिससे इन सभी कमिट को एक स्थान पर रखना मुश्किल हो गया है।

- **पिछले संस्करणों को पुनर्स्थापित करना।** बग के स्रोत को खोजने के लिए कभी-कभी पहले के संस्करण पर वापस जाने की आवश्यकता होती है।
- **पता लगाओ कि क्या हुआ।** यह जानना कि स्रोत कोड के पुराने संस्करणों में क्या परिवर्तन किए गए थे या वास्तव में फ़ाइल में परिवर्तन कहाँ किए गए थे, महत्वपूर्ण हैं।
- **बैकअप।** यदि उपयोगकर्ता की डिस्क या सिस्टम बैकअप के बिना खराब हो जाता है तो सभी काम बेकार हैं। विलयन।

बचाव के लिए संस्करण नियंत्रण!

बहुराष्ट्रीय निगम अपने कर्मचारियों के बीच सहयोग कर सकते हैं और संस्करण नियंत्रण प्रणाली के लिए दूरस्थ भंडार बैकअप के साथ मदद कर सकते हैं। सिस्टम डेवलपर्स को आवश्यक होने पर पहले के कमिट और स्रोत कोड के नवीनतम संस्करण पर वापस जाने में सक्षम करेगा।

एक एकल सर्वर रिपॉजिटरी वर्कस्टेशन की एक बड़ी संख्या द्वारा उपयोग किया जा सकता है। प्रत्येक वर्कस्टेशन में वर्तमान वर्किंग कोड की एक प्रति होगी, और इनमें से प्रत्येक वर्कस्टेशन अपने स्रोत कोड को एक विशिष्ट सर्वर रिपॉजिटरी में सहेज लेगा।

कोई भी डेवलपर अब इसके लिए रिपॉजिटरी के साथ किए जा रहे काम को आसानी से एक्सेस कर सकता है। काम जारी रहेगा क्योंकि केंद्रीय भंडार में स्रोत कोड की एक प्रति होगी, भले ही ऑनडेवलपर का सिस्टम विफल हो जाए।

2.1.3 संस्करण नियंत्रण के लाभ

- **स्रोत कोड का प्रबंधन और सुरक्षा।** संस्करण नियंत्रण प्रणाली सभी कोड संशोधनों का ट्रैक रखकर स्रोत कोड के प्रबंधन में सॉफ्टवेयर टीम की सहायता करती है। इसके अतिरिक्त, यह अनपेक्षित मानवीय त्रुटि और स्रोत कोड पर इसके प्रभावों से बचाता है।
- **सभी कोड संशोधनों पर नज़र रखना।** परियोजना के लिए जिम्मेदार टीम लगातार नए स्रोत कोड बनाती है और वर्तमान कोड में परिवर्तन करती है। इन संशोधनों को भविष्य में उपयोग के लिए प्रलेखित किया गया है और किसी विशिष्ट समस्या के अंतर्निहित कारण की पहचान करने के लिए आवश्यक होने पर परामर्श किया जा सकता है।
- **कोड के पुराने संस्करणों की तुलना**। डेवलपर्स किसी भी समय वापस जा सकते हैं और कोड के सभी संस्करणों को सहेजे जाने के कारण सभी टीम सदस्यों के लिए व्यवधान को कम करते हुए त्रुटि को ठीक करने में मदद करने के लिए कोड के पुराने संस्करणों की तुलना कर सकते हैं।
- **किसी भी कठोर कार्य विधियों** को नियोजित करने के बजाय डेवलपर्स के कार्य प्रवाह का समर्थन करता है। कोई भी विश्वसनीय संस्करण नियंत्रण प्रणाली चीजों को करने का एक विशेष तरीका लागू नहीं करेगी .. संस्करण नियंत्रण प्रणाली का उपयोग करते समय, जो कोड परिवर्तनों के निर्बाध और निरंतर प्रवाह को वितरित करने के लिए प्रसिद्ध हैं, डेवलपर्स के लिए इस कठिन तंत्र से परेशान महसूस करना आम बात है।

चित्र 2.1.2 कुछ संस्करण नियंत्रण प्रणालियों के उदाहरण

2.1.4 टेस्ट एंट्री & एग्जिट क्राइटेरिया

सॉफ्टवेयर परीक्षण सॉफ्टवेयर विकास जीवन चक्र का एक अनिवार्य चरण है क्योंकि यह सॉफ्टवेयर उत्पादों की प्रभावशीलता और गुणवत्ता का मूल्यांकन करने के लिए परीक्षकों से महत्वपूर्ण समय और प्रयास की मांग करता है। हालांकि बेहद मददगार, यह प्रक्रिया अक्सर थकाऊ हो जाती है क्योंकि कई मौकों पर इसे विभिन्न प्लेटफार्मों पर किया जाना चाहिए। इसके अतिरिक्त, कई प्रकार की आवश्यकताएं हैं जिन्हें ध्यान में रखा जाना चाहिए और परीक्षण किया जाना चाहिए, जो कभी-कभी परीक्षकों के लिए भ्रम पैदा कर सकते हैं, खासकर जब यह तय करना कि परीक्षण कहां से शुरू करना है और कहां रोकना है। इस गलतफहमी को रोकने के लिए, QA टीम परीक्षण शुरू होने से पहले विशिष्ट शर्तों और आवश्यकताओं को स्थापित करती है जो परीक्षण जीवन चक्र में परीक्षकों की सहायता करती हैं। इन आवश्यकताओं को प्रवेश और निकास मानदंड के रूप में जाना जाता है, और वे सॉफ्टवेयर परीक्षण के जीवन चक्र के लिए आवश्यक हैं।

प्रवेश मानदंड

परीक्षण प्रक्रिया की सटीकता की गारंटी प्रवेश मानदंडों द्वारा दी जाती है, जिन्हें सॉफ्टवेयर और व्यावसायिक आवश्यकताओं के गहन विश्लेषण के बाद अंतिम रूप दिया जाता है और चुना जाता है। उन्हें अनदेखा करने से प्रक्रिया की गुणवत्ता कम हो सकती है।

निम्नलिखित कुछ प्रवेश आवश्यकताएं हैं, जिनका उपयोग आमतौर पर परीक्षण की शुरुआत का संकेत देने के लिए किया जाता है:

- पूरी तरह या आंशिक रूप से परीक्षण योग्य कोड उपलब्ध है।
- परिभाषित और अनुमोदित आवश्यकताएं हैं।
- वांछित और आवश्यक परीक्षण डेटा की उपलब्धता।
- परीक्षण मामलों को तैयार और विकसित किया जाता है।
- परीक्षण वातावरण स्थापित किया गया है, और उपकरण और उपकरण सहित अन्य सभी संसाधन उपलब्ध हैं।

Exit मानदंड

निकास मानदंड QA टीम द्वारा स्थापित समय सीमा और बजट का पालन करने के लिए बनाया गया एक महत्वपूर्ण दस्तावेज है। यह दस्तावेज़ उन शर्तों और मांगों की रूपरेखा तैयार करता है जिन्हें सॉफ़्टवेयर परीक्षण प्रक्रिया के समापन से पहले पूरा किया जाना चाहिए। परीक्षकों की टीम निकास मानदंडों की मदद से सॉफ्टवेयर की गुणवत्ता और प्रभावकारिता का त्याग किए बिना परीक्षण को लपेटने में सक्षम हैं।

निष्पादन के माध्यम से परीक्षण योजना और विनिर्देश से प्रत्येक परीक्षण स्तर के लिए निकास मानदंड को परिभाषित किया जा सकता है, और सॉफ्टवेयर परीक्षण चरण के उप-उत्पाद पर बहुत अधिक निर्भर करता है, जैसे परीक्षण योजना, परीक्षण रणनीति, परीक्षण मामले, परीक्षण लॉग, आदि।

- परीक्षण प्रक्रिया को रोकने या लपेटने के लिए आमतौर पर स्वीकृत निकास मानदंड निम्नलिखित हैं:
- बजट की कमी या समय सीमा का पालन।
- प्रत्येक परीक्षण मामले का निष्पादन और अद्यतन।
- परीक्षण की आवश्यकताओं और कार्यात्मकताओं का कवरेज जो वांछित और पर्याप्त दोनों है।
- सभी पाए गए दोष तय और बंद हैं।
- कोई महत्वपूर्ण, उच्च प्राथमिकता या गंभीर बग की अनदेखी नहीं की गई है।
- उनकी वर्तमान स्थिति के साथ दोषों को बनाए रखना।

बाजार में प्रवेश और निकास मानदंड विचारों के लिए कुछ प्रमुख सर्वोत्तम प्रथाएं निम्नलिखित हैं:

- किसी भी प्रक्रिया को शुरू करने से पहले प्रत्येक परीक्षण प्रकार के लिए प्रवेश और निकास मानदंड को स्पष्ट रूप से परिभाषित करना महत्वपूर्ण है।
- ऐसे उपाय या शर्तें जिन्हें गुणात्मक के बजाय मात्रात्मक रूप से दर्शाया जाता है
- यदि प्रवेश या निकास मानदंड पूरे नहीं होते हैं, तो सुधारात्मक कार्रवाई आवंटित की जानी चाहिए, या आवश्यक परिवर्तनों के साथ पूरी प्रक्रिया को फिर से शुरू किया जाना चाहिए।
- प्रक्रिया के निर्माण और समीक्षा के दौरान, मॉडरेटर को निरंतर निगरानी बनाए रखनी चाहिए और अनुवर्ती प्रदान करना चाहिए।

2.1.5 डेटा संग्रह के तरीके

ऐसे समय में जानकारी प्राप्त करना महत्वपूर्ण है जब "ज्ञान शक्ति है। लेकिन सवाल यह है कि आपकी विशिष्ट आवश्यकताओं के लिए कौन सी डेटा संग्रह तकनीक सबसे अच्छी है? जैसे, कच्चे डेटा को हमेशा विशेष रूप से फायदेमंद नहीं होना चाहिए। यह उचित ढांचे और संदर्भ के बिना असंबंधित तथ्यों और आंकड़ों का एक संग्रह है। हालाँकि, यदि आप उस डेटा को व्यवस्थित, वर्गीकृत और विश्लेषण करते हैं, तो आपके पास अपने निपटान में निर्णय लेने के लिए एक मजबूत "ईंधन" होगा। यद्यपि "डेटा संग्रह" शब्द बहुत उच्च तकनीक और डिजिटल लग सकता है, इसका मतलब यह नहीं है कि बड़ा डेटा, इंटरनेट और कंप्यूटर हमेशा शामिल होते हैं। डेटा एकत्र करना एक फोन सर्वेक्षण, एक मेल-इन टिप्पणी कार्ड, या यहां तक कि एक क्लिपबोर्ड वाला व्यक्ति भी यादृच्छिक दर्शकों से कुछ प्रश्न पूछ सकता है। लेकिन आइए विभिन्न डेटा संग्रह तकनीकों को सुसंगत श्रेणियों के समान कुछ में समूहित करने का प्रयास करें।

डेटा इकट्ठा करने के दो तरीके हैं। एक तरफ: तकनीकों, विधियों और प्रकारों सहित कई शब्दों का अर्थ अलग-अलग चीजों पर निर्भर करता है कि उनका उपयोग कौन करता है। उदाहरण के लिए, एक स्रोत डेटा संग्रह विधियों को "विधियों" के रूप में संदर्भित कर सकता है। हमारे द्वारा चुने गए लेबल के बावजूद, मौलिक विचार और ब्रेकडाउन समान रहते हैं चाहे हम वैज्ञानिक अनुसंधान परियोजना या विपणन विश्लेषण पर चर्चा कर रहे हों।

दो विधियाँ हैं:

1. **प्राथमिक**

   यह प्रामाणिक, प्रथम-हाथ डेटा है जिसे शोधकर्ताओं ने एकत्र किया है, जैसा कि नाम से पता चलता है। किसी भी अतिरिक्त या जुड़े शोध का संचालन करने से पहले, यह प्रक्रिया जानकारी एकत्र करने में पहला कदम है। प्राथमिक डेटा के परिणाम बहुत सटीक होते हैं जब तक शोधकर्ता डेटा एकत्र करता है। हालांकि, एक खामी है क्योंकि व्यक्तिगत रूप से अनुसंधान महंगा और समय लेने वाला हो सकता है।

2. **द्वितीयक**

   द्वितीयक डेटा वह जानकारी है जो पहले से ही सांख्यिकीय विश्लेषण से गुजर चुकी है और अन्य स्रोतों से प्राप्त की गई है। यह जानकारी या तो वह जानकारी है जिसे शोधकर्ता ने देखा है या वह जानकारी जिसे शोधकर्ता ने दूसरों को इकट्ठा करने के लिए कहा है। यह दूसरे हाथ की जानकारी है, इसे सीधे शब्दों में कहें। यद्यपि माध्यमिक जानकारी प्राथमिक जानकारी की तुलना में अधिक सुलभ और कम खर्चीली है, इसकी प्रामाणिकता और सटीकता पर सवाल उठाया जाता है। अधिकांश माध्यमिक डेटा संख्यात्मक जानकारी से बना होता है।

   आँकड़ों के संग्रहण के तरीकों का वर्गीकरण सारणी 2-1 में दर्शाया गया है। प्रत्येक तकनीक को इस आधार पर वर्गीकृत किया जाता है कि लोगों के साथ कितना संपर्क आवश्यक है। प्रथम डिग्री संपर्क के लिए दर्शकों के साथ सीधा संपर्क आवश्यक है। दूसरी डिग्री के संपर्क के लिए प्रतिभागियों के कामकाजी वातावरण तक पहुंच की आवश्यकता होती है, लेकिन यह प्रतिभागियों तक सीधी पहुंच या प्रतिभागियों और शोधकर्ताओं के बीच बातचीत के लिए नहीं कहता है। दूसरी ओर, थर्ड-डिग्री संपर्क को केवल स्रोत कोड या प्रलेखन जैसी कलाकृतियों तक पहुंच की आवश्यकता होती है।

Category	Technique
पहली डिग्री (सॉफ्टवेयर इंजीनियरों की प्रत्यक्ष भागीदारी)	**जिज्ञासु तकनीक** • बुद्धिशीलता और फोकस समूह • साक्षात्कार और प्रश्नावली • वैचारिक मॉडलिंग
	अवलोकन तकनीक • काम डायरी • थिंक-अलाउड प्रोटोकॉल • छायांकन और अवलोकन सिंक्रनाइज़ छायांकन • प्रतिभागी अवलोकन (टीम में शामिल होना)
दूसरी डिग्री (सॉफ्टवेयर इंजीनियरों की अप्रत्यक्ष भागीदारी)	**इंस्ट्रूमेंटिंग सिस्टम** • दीवार पर उड़ना (प्रतिभागियों को उनके काम का टेपिंग) • प्रदर्शन किए गए कार्य के इलेक्ट्रॉनिक डेटाबेस का विश्लेषण • उपकरण उपयोग लॉग का विश्लेषण • दस्तऐवजीकरण विश्लेषण • एक प्रणाली का स्थिर और गतिशील विश्लेषण

तालिका 2.1.1 डेटा संग्रह तकनीक

2.1.6 वस्तुनिष्ठ और व्यक्तिपरक विश्लेषण को समझना

परिभाषाओं को जानने से आपको वस्तुनिष्ठ और व्यक्तिपरक शब्दों के बीच अंतर करने में मदद मिलेगी:

- उद्देश्य (adj) - व्यक्तिगत भावनाओं, स्वाद या राय से प्रभावित नहीं
- व्यक्तिपरक (adj) - व्यक्तिपरक विचारों, भावनाओं या विचारों से प्रभावित परिभाषाओं को जानने से उद्देश्य और व्यक्तिपरक अवधारणाओं के बीच अंतर करना आसान हो जाएगा।

इन परिभाषाओं के परिणामस्वरूप, हम कह सकते हैं कि डिजाइन मुख्य रूप से एक उद्देश्य प्रक्रिया है। एक परियोजना की शुरुआत में, यह सलाह दी जाती है कि हम अपनी भावनाओं, "स्वाद," सिद्धांतों और निराधार मान्यताओं को प्रभावित करने के बजाय एक उद्देश्य प्रक्रिया का पालन करें।

उपयोगकर्ता अनुसंधान का लक्ष्य कार्य विश्लेषण, अवलोकन विधियों और प्रतिक्रिया के अन्य तरीकों के माध्यम से उपयोगकर्ता की जरूरतों, व्यवहारों और प्रेरणाओं को समझना है। माइक कुनियास्की के अनुसार, यह "दर्शकों पर डिजाइन के प्रभाव को समझने की प्रक्रिया है। उपयोगकर्ता अनुसंधान अंतिम उपयोगकर्ता को ध्यान में रखते हुए निष्पक्ष रूप से डिजाइन करने में हमारी सहायता करने के लिए है। अनुसंधान हमें अपने लिए डिजाइन करने और डिजाइन में हमारी व्यक्तिगत प्राथमिकताओं को शामिल करने से रोकता है। इसके बजाय, उपयोगकर्ता अनुसंधान उस व्यक्ति, वातावरण की पहचान करता है जिसमें वे उत्पाद या सेवा का उपयोग करेंगे, और उन्हें हमसे आवश्यक सहायता की आवश्यकता होती है

2.1.7 तनाव परीक्षण

तनाव: तनाव परीक्षण एक ऐसी प्रक्रिया है जहां चुनौतीपूर्ण परिस्थितियों में लगातार या स्वीकार्य प्रदर्शन बनाए रखने की क्षमता निर्धारित करने के लिए सॉफ्टवेयर की जांच की जाती है। इनमें उच्च नेटवर्क ट्रैफ़िक, एक उच्च प्रक्रिया लोड, अंतर्निहित हार्डवेयर के अंडर- या ओवरक्लॉकिंग, और परिधीय या सिस्टम संसाधन उपयोग के लिए सख्त आवश्यकताओं के तहत संचालन शामिल हो सकता है। तनाव परीक्षण मजबूती और निर्भरता की डिग्री निर्धारित करने में सहायता करता है, तब भी जब सिस्टम की सामान्य परिचालन सीमा पार हो जाती है। सॉफ्टवेयर जो उच्च-दांव या वास्तविक समय की स्थितियों में कार्य करता है, तनाव परीक्षण से बहुत लाभान्वित होता है।

ब्राउज़र विंडो पर एक नज़र डालें। एक साथ कई पृष्ठों के बीच स्विच करने के लिए, उपयोगकर्ता कई ब्राउज़र विंडो खोल सकते हैं। हालांकि, उनकी अन्योन्याश्रितता के कारण, यदि इनमें से एक खिड़की दुर्घटनाग्रस्त हो जाती है, तो वे सभी दुर्घटनाग्रस्त हो जाते हैं। इस परिदृश्य में, ब्राउज़र तनाव परीक्षण के माध्यम से रखा गया है।

तनाव परीक्षण कैसे किए जाते हैं?

तनाव परीक्षण प्रक्रिया के पांच मुख्य चरण हैं:

- **तनाव परीक्षण की तैयारी:** यहां, आप सिस्टम डेटा इकट्ठा करते हैं, सिस्टम का विश्लेषण करते हैं, और तनाव परीक्षण के उद्देश्यों को निर्दिष्ट करते हैं।
- **स्वचालित तनाव परीक्षण स्क्रिप्ट निर्माण:** इस चरण के दौरान, आप दबाव परिदृश्यों के लिए परीक्षण डेटा का उत्पादन और दबाव परीक्षण के लिए स्वचालन स्क्रिप्ट बनाएँ।
- **स्क्रिप्ट निष्पादन:** इस चरण में, आप दबाव परीक्षण के लिए स्वचालन स्क्रिप्ट चलाएँ और परिणामों को संग्रहीत।
- **परिणाम विश्लेषण:** इस बिंदु पर, आप किसी भी अड़चन को खोजने के लिए तनाव परीक्षण के परिणामों की जांच करते हैं।
- **ट्वीकिंग और ऑप्टिमाइज़ेशन:** इस चरण में, आप सिस्टम की सेटिंग्स को समायोजित करते हैं और वांछित बेंचमार्क प्राप्त करने के प्रयास में कोड में सुधार करते हैं।

यह जांचने के लिए कि समायोजन से वांछित परिणाम प्राप्त हुए हैं, आप पूरे चक्र को एक बार फिर से चलाते हैं। प्रदर्शन लक्ष्यों को प्राप्त करने के लिए, तनाव परीक्षण प्रक्रिया, उदाहरण के लिए, आमतौर पर तीन से चार चक्रों की आवश्यकता होती है।

तनाव परीक्षण के लिए अनुशंसित उपकरण:

- **लोड रनर :**एक लोकप्रिय लोड परीक्षण उपकरण एचपी से लोड रनर है। बेंच मार्किंग लोड रनर द्वारा बनाए गए लोड टेस्ट परिणामों का उपयोग करके किया जाता है।
- **J मीटर:** एक ओपन सोर्स टेस्टिंग टूल J मीटर है। यह केवल एक तनाव और प्रदर्शन परीक्षण Java एप्लीकेशन है। जे मीटर को विभिन्न परीक्षण प्रकारों को संभालने के लिए डिज़ाइन किया गया है, जिसमें लोड, कार्यात्मक, तनाव आदि शामिल हैं। जे डी के 5 या बाद में काम करने के लिए आवश्यक है।
- **तनाव परीक्षक:** यह उपकरण वेब एप्लिकेशन के प्रदर्शन की गहन जांच प्रदान करता है और परिणामों को ग्राफिक रूप से प्रदर्शित करता है। यह काफी यूजर फ्रेंडली भी है। यह निवेश पर एक उत्कृष्ट रिटर्न प्रदान करता है और इसके लिए किसी परिष्कृत कोडिंग की आवश्यकता नहीं होती है।
- **नियो लोड:** यह लोकप्रिय उपकरण मोबाइल और वेब एप्लिकेशन का परीक्षण करने के लिए बाजार में आसानी से उपलब्ध है। लोड के तहत किसी एप्लिकेशन के प्रदर्शन का आकलन करने और प्रतिक्रिया समय की जांच करने के लिए, यह उपकरण हजारों उपयोगकर्ताओं का अनुकरण कर सकता है। इसके अतिरिक्त, यह क्लाउड-एकीकृत प्रदर्शन, लोड और तनाव परीक्षण का समर्थन करता है। इसका उपयोग करना आसान है, किफायती है, और अच्छी मापनीयता प्रदान करता है।

- **तनाव परीक्षण मेट्रिक्स:** मेट्रिक्स की आमतौर पर तनाव परीक्षण के समापन पर जांच की जाती है और सिस्टम के प्रदर्शन का आकलन करने में सहायता मिलती है। अक्सर उपयोग किए जाने वाले मेट्रिक्स हैं-

स्केलेबिलिटी और प्रदर्शन माप

- प्रत्येक सेकंड अनुरोधित पृष्ठों की संख्या को पेज प्रति सेकंड द्वारा मापा जाता है।
- थ्रूपुट के लिए मूल मैट्रिक: प्रतिक्रिया डेटा आकार/सेकंड
- राउंड: परीक्षण परिदृश्यों की प्रत्याशित संख्या बनाम क्लाइंट निष्पादन पृष्ठों की वास्तविक संख्या प्रति सेकंड अनुरोध किए गए पृष्ठों की संख्या को मापता है।

आवेदन प्रतिक्रिया

- हिट समय: एक छवि या पृष्ठ को लोड होने में आमतौर पर एक निश्चित समय लगता है।
- **पहली बाइट बार:** डेटा या जानकारी का पहला बाइट लौटाने के लिए आवश्यक समय की मात्रा
- **पृष्ठ समय:** किसी पृष्ठ पर प्रत्येक जानकारी को पुनः प्राप्त करने के लिए आवश्यक समय।

विफलताओं

- **विफल कनेक्शन:** असफल कनेक्शन की मात्रा जिसे क्लाइंट ने अस्वीकार कर दिया है (कमजोर सिग्नल)
- **राउंड विफल:** राउंड की कुल संख्या यह विफल रहता है
- **असफल हिट:** सिस्टम विफलताओं की कुल संख्या (जैसे, टूटे हुए लिंक या छिपी हुई छवियां)।

तनाव परीक्षण का उद्देश्य कठिन परिस्थितियों में प्रणाली की जांच करना है। यह सिस्टम संसाधनों जैसे मेमोरी, प्रोसेसर, नेटवर्क आदि पर नजर रखता है और सामान्य ऑपरेशन पर लौटने के लिए सिस्टम की क्षमता का मूल्यांकन करता है। यह जांच करता है कि दबाव में होने पर सिस्टम उपयुक्त त्रुटि संदेश जारी करता है या नहीं।

2.1.8 सॉफ्टवेयर परीक्षण सिद्धांत

सॉफ्टवेयर परीक्षण एक ऐसी प्रक्रिया है जिसमें बग या खामियों को खोजने के लिए सॉफ़्टवेयर या एप्लिकेशन का उपयोग करना शामिल है। कुछ दिशानिर्देशों का पालन करने से हमें बिना किसी दोष के सॉफ़्टवेयर या एप्लिकेशन का परीक्षण करने में मदद मिलेगी, और यह परीक्षण इंजीनियरों के समय और प्रयास को भी बचाएगा क्योंकि वे ऐसा करने में अपना समय और प्रयास लगाते हैं। हम इस खंड में सॉफ्टवेयर परीक्षण के सात मूलभूत सिद्धांतों के बारे में जानेंगे।

नीचे सात अलग-अलग परीक्षण सिद्धांत दिए गए हैं:

चित्र 2.1.3 सॉफ्टवेयर परीक्षण सिद्धांत

1. **परीक्षण दोषों की उपस्थिति को दर्शाता है**

   आवेदन को टेस्ट इंजीनियर द्वारा परीक्षण के माध्यम से रखा जाएगा ताकि यह सुनिश्चित किया जा सके कि कोई बग या खामियां नहीं हैं। परीक्षण के दौरान केवल एप्लिकेशन या सॉफ़्टवेयर में त्रुटियों की उपस्थिति निर्धारित की जा सकती है। परीक्षण का मुख्य लक्ष्य किसी भी दोष को ढूंढना है जो विभिन्न तरीकों और परीक्षण तकनीकों का उपयोग करके उत्पाद को ग्राहक की जरूरतों को पूरा करने से रोक सकता है। चूंकि पूरे परीक्षण को ग्राहक की आवश्यकता के लिए वापस पता लगाने में सक्षम होना चाहिए।

   परीक्षण किसी भी एप्लिकेशन में बग की संख्या को कम करता है, लेकिन इसका मतलब यह नहीं है कि एप्लिकेशन दोष-मुक्त है क्योंकि कभी-कभी सॉफ्टवेयर व्यापक परीक्षण के बावजूद बग-मुक्त प्रतीत होता है। हालाँकि, यदि कोई अंतिम उपयोगकर्ता बग में चलता है जो परीक्षण के दौरान खोजा नहीं गया था, तो यह तब होता है जब सर्वर को उत्पादन में तैनात किया जाता है।

2. **संपूर्ण परीक्षण संभव नहीं है**

   वास्तविक परीक्षण प्रक्रिया के दौरान, इनपुट डेटा के प्रभावी और अप्रभावी संयोजनों का उपयोग करके सभी मॉड्यूल और उनकी विशेषताओं का परीक्षण करना बहुत मुश्किल प्रतीत हो सकता है।

   इसलिए, चूंकि इसके लिए अंतहीन दृढ़ संकल्प की आवश्यकता होती है और अधिकांश कड़ी मेहनत असफल होती है, इसके बजाय व्यापक परीक्षण को प्राथमिकता दी जाती है। नतीजतन, हम मॉड्यूल के महत्व के अनुसार इस प्रकार की विविधताओं को पूरा कर सकते हैं क्योंकि इस तरह के परीक्षण परिदृश्यों को करने से उत्पाद समयसीमा का उल्लंघन होगा।

3. **प्रारंभिक परीक्षण**

यहां, प्रारंभिक परीक्षण इस विचार को संदर्भित करता है कि दोषों की पहचान करने के लिए सभी परीक्षण गतिविधियों को सॉफ्टवेयर विकास जीवन चक्र के आवश्यकता विश्लेषण चरण के प्रारंभिक चरणों में शुरू करना चाहिए। प्रारंभिक बग का पता लगाने से हमें उन्हें संबोधित करने की अनुमति मिलती है, जो परीक्षण प्रक्रिया में बाद में पाए जाने की तुलना में हमें काफी कम खर्च कर सकता है।

परीक्षण करने के लिए, हमें आवश्यकता विनिर्देश दस्तावेजों की आवश्यकता होगी; नतीजतन, यदि आवश्यकताओं को गलत तरीके से परिभाषित किया गया है, तो इसे बाद में ठीक किया जा सकता है, संभवतः विकास चरण के दौरान।

4. **दोष क्लस्टरिंग**

दोष क्लस्टरिंग ने निर्दिष्ट किया कि हम उन बगों की मात्रा की पहचान कर सकते हैं जो परीक्षण प्रक्रिया के दौरान सीमित संख्या में मॉड्यूल से संबंधित हैं। हमारे पास इसके लिए कई स्पष्टीकरण हैं, जिनमें जटिल मॉड्यूल, जटिल कोडिंग और बहुत कुछ शामिल हैं।

इस प्रकार के सॉफ़्टवेयर या एप्लिकेशन पेरेटो सिद्धांत का पालन करेंगे, जिसमें कहा गया है कि 20% मॉड्यूल में लगभग 80% जटिलता मौजूद है। यह हमें अस्पष्ट मॉड्यूल का पता लगाने की अनुमति देता है, लेकिन इसकी सीमाएं हैं यदि एक ही परीक्षण अक्सर चलाए जाते हैं क्योंकि वे किसी भी नए पेश किए गए दोषों को नहीं देख पाएंगे।

5. **Pesticide paradox**

इस नियम में कहा गया है कि यदि परीक्षण मामलों का एक ही सेट एक निर्धारित अवधि में बार-बार चलाया गया था, तो परीक्षण सॉफ़्टवेयर या एप्लिकेशन में किसी भी नए बग को खोजने में सक्षम नहीं होंगे। इन कीटनाशक विरोधाभासों पर काबू पाने के लिए सभी परीक्षण मामलों की अक्सर समीक्षा करना महत्वपूर्ण है। इसके अतिरिक्त, एप्लिकेशन या सॉफ़्टवेयर के कई घटकों को लागू करने के लिए, नए और अलग-अलग परीक्षण लिखे जाने चाहिए, जो अतिरिक्त बग की खोज में सहायता करते हैं।

6. **परीक्षण संदर्भ-निर्भर है**

परीक्षण एक संदर्भ-निर्भर सिद्धांत है जिसमें कहा गया है कि ई-कॉमर्स वेबसाइटों, व्यावसायिक वेबसाइटों और अन्य सहित कई बाजार उपलब्ध हैं। प्रत्येक एप्लिकेशन की अपनी आवश्यकताएं, विशेषताएं और कार्यक्षमता होती है, इसलिए वाणिज्यिक और ई-कॉमर्स वेबसाइटों दोनों का परीक्षण करने का एक स्पष्ट तरीका है। हम इस तरह के आवेदन की जांच करने के लिए विभिन्न परीक्षण विधियों, साथ ही विभिन्न दृष्टिकोणों, तकनीकों और विधियों का उपयोग करेंगे। नतीजतन, परीक्षण आवेदन संदर्भ पर निर्भर है।

7. **त्रुटियों की अनुपस्थिति भ्रम**

एक बार जब एप्लिकेशन का पूरी तरह से परीक्षण हो जाता है और रिलीज से पहले कोई बग नहीं पाया जाता है, तो हम कह सकते हैं कि यह 99.9% बग-मुक्त है। हालांकि, एक मौका है कि गलत आवश्यकताओं के साथ एप्लिकेशन का परीक्षण करना, खामियों का पता लगाना और उन्हें पूर्व निर्धारित समय सीमा के भीतर ठीक करना सफल नहीं होगा क्योंकि परीक्षण गलत विनिर्देश का उपयोग करके किया जाता है, जो क्लाइंट की आवश्यकताओं पर लागू नहीं होता है। बग ढूंढना और ठीक करना मदद नहीं करेगा यदि एप्लिकेशन अप्रभावी है और क्लाइंट की जरूरतों और आवश्यकताओं को पूरा करने में असमर्थ है, त्रुटि भ्रम की अनुपस्थिति का दावा करता है।

2.1.9 गुणवत्ता आश्वासन के विभिन्न पहलू

किसी उत्पाद या सेवा की गुणवत्ता को इस रूप में वर्णित किया जा सकता है कि यह अपेक्षाओं की तुलना में कितना अच्छा प्रदर्शन करता है। जब आउटपुट इन पूर्व निर्धारित मानकों को पूरा नहीं करता है, तो गुणवत्ता नियंत्रण प्रक्रिया के हिस्से के रूप में सुधारात्मक कार्रवाई की जाती है, जो आउटपुट की तुलना एक मानक से करती है। इसलिए, ग्राहकों के संदर्भ में गुणवत्ता नियंत्रण यह सुनिश्चित करने की सतत प्रक्रिया होगी कि उत्पादों को ग्राहकों की जरूरतों को पूरा करने और यहां तक कि उससे अधिक करने के लिए बनाया गया है। गुणवत्ता नियंत्रण चिंताएं अनुबंध कार्य के लिए अनुबंध को नवीनीकृत नहीं करने के मुख्य कारणों में से हैं, खासकर सरकारी संगठनों द्वारा दिए गए काम के लिए।

इस रणनीति के साथ, तीन तत्वों पर प्रकाश डाला गया है:

- नियंत्रण, नौकरी प्रबंधन, स्पष्ट रूप से परिभाषित प्रक्रियाएं, प्रदर्शन और अखंडता मानक, और रिकॉर्ड पहचान कुछ घटक हैं।
- ज्ञान, कौशल, अनुभव और योग्यता जो क्षमता प्रदर्शित करती है
- नरम कारकों में टीम वर्क, नैतिक चरित्र, संगठनात्मक संस्कृति, प्रेरणा और अच्छे रिश्ते शामिल हैं।

2.1.10 गुणवत्ता आश्वासन के सिद्धांत

गुणवत्ता आश्वासन के दो सिद्धांत हैं। ensOne करना "उद्देश्य के लिए उपयुक्त" है, जिसका अर्थ है कि अच्छा या सेवा उस उद्देश्य को पूरा करती है जिसके लिए इसे बनाया गया था। विकल्प "पहली बार सही" है, जिसमें त्रुटियों को तुरंत ठीक किया जाता है। विकल्प "पहली बार सही" है, जिसमें त्रुटियों को तुरंत ठीक किया जाता है। इसका उद्देश्य उपरोक्त दो सिद्धांतों को ध्यान में रखते हुए परियोजना के चर का प्रबंधन करके उत्पाद या सेवा को हर समय सही ढंग से संचालित करना है। इस प्रकार, कच्चे माल, असेंबली, उत्पादों और घटकों की गुणवत्ता का प्रबंधन गुणवत्ता आश्वासन का एक हिस्सा है, जैसा कि उत्पादन, प्रबंधन, उत्पादन और निरीक्षण प्रक्रियाओं से संबंधित सेवाएं हैं।

2.1.11 गुणवत्ता आश्वासन दृष्टिकोण

- **विफलता परीक्षण:** विफलता परीक्षण, जिसे तनाव परीक्षण के रूप में भी जाना जाता है, अंतर्निहित कमजोरियों को प्रकट करने के लिए कंपन, तापमान, आर्द्रता आदि को बढ़ाकर उत्पाद को अपनी पूर्ण सीमा तक धकेलने की एक तकनीक है। परिणामों का उपयोग तब उच्च मानक बनाए रखने के लिए उत्पाद को बेहतर बनाने के लिए किया जाता है।
- **सांख्यिकीय नियंत्रण**: इस प्रकार का गुणवत्ता आश्वासन उद्देश्य और व्यक्तिपरक डेटा के विश्लेषण पर आधारित होता है, और यह गुणवत्ता डेटा को ट्रैक करता है और इसे एक सामान्य कारण विचरण के खिलाफ चार्ट करता है।
- **कुल गुणवत्ता प्रबंधन:** अंतिम उत्पाद की गुणवत्ता इसमें शामिल घटकों पर निर्भर करती है, जिनमें से कुछ टिकाऊ और नियंत्रणीय हैं और अन्य जो नहीं हैं। गुणवत्ता की गारंटी नहीं है यदि विनिर्देश वास्तविक गुणवत्ता आवश्यकताओं के अनुरूप नहीं है।
- **मॉडल और मानक**: प्रवीणता के लिए इन बुनियादी आवश्यकताओं को इस अंतरराष्ट्रीय मानक में रखा गया है। एक मान्यता प्राप्त प्रयोगशाला में, 15 प्रबंधन आवश्यकताओं और 10 तकनीकी आवश्यकताओं के साथ उत्तीर्ण करने के लिए परीक्षाएं हैं।
- **कंपनी की गुणवत्ता:** यह विचार पहली बार 1980 के दशक में सामने आया और यह इस बात पर केंद्रित है कि गुणवत्ता सुधार के लिए एक प्रक्रिया बनाने के लिए प्रबंधन के निर्देशन में सभी विभागों को गुणवत्ता से कैसे संपर्क करना चाहिए। इसे प्राप्त करने के लिए बुनियादी ढांचे के साथ-साथ नियंत्रण, नौकरी प्रबंधन, प्रक्रिया, प्रदर्शन, ज्ञान, कौशल और अनुभव का उपयोग किया जाता है।

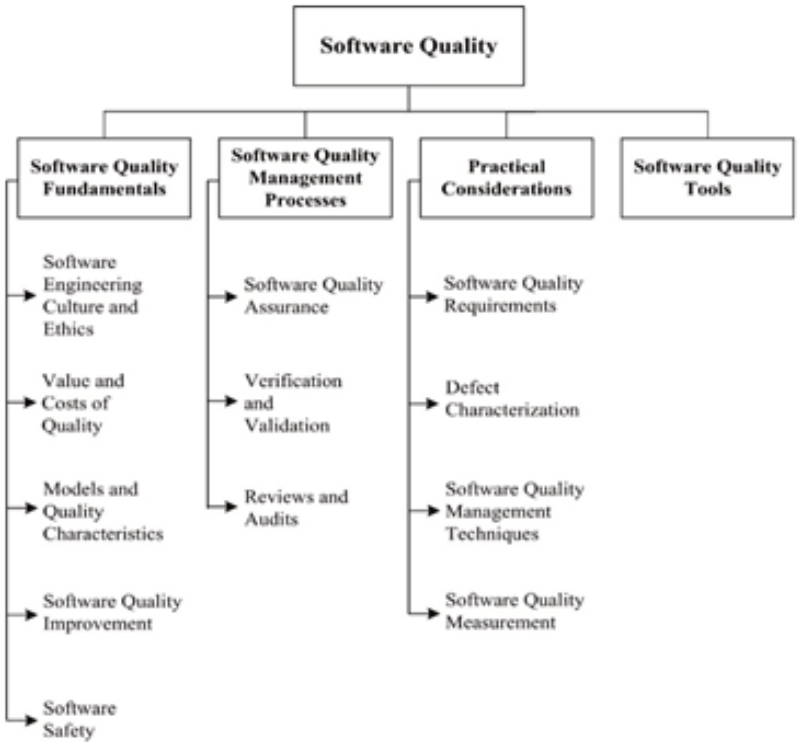


Fig.2.1.4 सॉफ्टवेयर गुणवत्ता

2.1.12 उद्योगों में व्यवहार में गुणवत्ता आश्वासन

लगभग सभी उद्योग गुणवत्ता आश्वासन का उपयोग करते हैं, और गुणवत्ता आश्वासन को संभालने के लिए ठेकेदारों या सलाहकारों का उपयोग करना असामान्य नहीं है।

- **चिकित्सा उद्योग**
  चिकित्सा क्षेत्र में गुणवत्ता नियंत्रण महत्वपूर्ण है क्योंकि त्रुटियों के परिणामस्वरूप मानव जीवन का नुकसान हो सकता है। लगभग हर जगह, अस्पतालों से लेकर प्रयोगशालाओं तक, एजेंसियों का उपयोग यह सुनिश्चित करने के लिए करता है कि उनके मानक इस क्षेत्र के अनुरूप हैं।

- **एयरोस्पेस उद्योग**
  गुणवत्ता आश्वासन, जिसे एयरोस्पेस क्षेत्र में उत्पाद आश्वासन के रूप में भी जाना जाता है, मानव लागत के साथ विनाशकारी विफलताओं को रोकने के लिए परियोजना प्रबंधन और इंजीनियरिंग के साथ सहयोग करता है। उत्पाद आश्वासन के लिए जिम्मेदार विभाग प्रबंधन के उच्चतम स्तर पर रिपोर्ट करता है और स्वतंत्र रूप से वित्त पोषित होता है।

- **सॉफ्टवेयर उद्योग**
  गुणवत्ता आश्वासन सॉफ्टवेयर विकास के दौरान सॉफ्टवेयर इंजीनियरिंग प्रक्रियाओं पर नजर रखने का एक तरीका है। यह आईएसओ 9000 या क्षमता परिपक्वता मॉडल एकीकरण (CMMI) जैसे मॉडल का उपयोग करने सहित विभिन्न तरीकों से किया जा सकता है। सॉफ्टवेयर का उपयोग कभी-कभी समस्याओं को ठीक करने के लिए किया जाता है।

2.1.13 सॉफ्टवेयर परीक्षक कौशल सेट

एक व्यक्ति जो सॉफ्टवेयर का परीक्षण करता है उसे अक्सर सॉफ्टवेयर टेस्ट इंजीनियर के रूप में जाना जाता है। वह परीक्षण योजना बनाने और प्रयोज्य मुद्दों को समझने में सक्षम होना चाहिए। सॉफ्टवेयर परीक्षक के इस स्तर से परीक्षण निष्पादन और डिजाइन दृष्टिकोण में कुशल होने की उम्मीद है। विकास टीम के साथ प्रभावी ढंग से काम करने के लिए एक सॉफ्टवेयर परीक्षक के पास उत्कृष्ट संचार कौशल होना चाहिए। प्रयोज्य सॉफ्टवेयर परीक्षक की भूमिकाएं और कर्तव्य निम्नलिखित हैं:

- प्रयोज्य परीक्षण के लिए परीक्षण परिदृश्यों को डिजाइन करना एक सॉफ्टवेयर परीक्षक की जिम्मेदारी है।
- वह परीक्षण चलाने, परिणामों की जांच करने और फिर विकास टीम को अपने निष्कर्षों की रिपोर्ट करने के प्रभारी हैं।
- उत्पाद के लिए उनकी आवश्यकताओं को बेहतर ढंग से समझने के लिए या यदि डिज़ाइन में कोई बदलाव आवश्यक है, तो उन्हें ग्राहकों से बात करने की आवश्यकता हो सकती है।
- सॉफ़्टवेयर परीक्षक अक्सर परीक्षण-उत्पाद दस्तावेज़ीकरण का उत्पादन करते हैं और परीक्षण से संबंधित पूर्वाभ्यास में भाग लेते हैं।

एक सॉफ्टवेयर परीक्षक विभिन्न कार्यों के लिए जिम्मेदार है। सॉफ्टवेयर परीक्षण की उनकी समझ व्यापक होनी चाहिए। उसे तकनीकी (जीयूआई या गैर-जीयूआई मानव बातचीत) के साथ-साथ सिस्टम के व्यावहारिक पहलुओं से अच्छी तरह वाकिफ होना चाहिए। सॉफ्टवेयर परीक्षक को विभिन्न परीक्षण पद्धतियों के बारे में जानकार होना चाहिए और परीक्षण मामलों को विकसित करने के लिए किसी दिए गए सिस्टम के लिए कौन सी रणनीति सबसे अच्छा काम करती है। उसे विभिन्न सॉफ्टवेयर परीक्षण चरणों के बारे में पता होना चाहिए और प्रत्येक के दौरान परीक्षण कैसे किया जाना चाहिए।

सॉफ्टवेयर परीक्षक की जिम्मेदारियों में शामिल हैं:

- परीक्षण डिजाइन, परीक्षण प्रक्रियाओं, परीक्षण मामलों और परीक्षण डेटा का निर्माण.परीक्षण योजनाओं, परीक्षण प्रक्रियाओं, परीक्षण मामलों और परीक्षण जानकारी का विकास।
- परीक्षण मामलों, परीक्षण जानकारी, परीक्षण योजनाओं और परीक्षण प्रक्रियाओं की तैयारी।
- परीक्षण प्रक्रिया करते समय दिशानिर्देशों का पालन करें।
- परीक्षण प्रक्रियाओं के पूर्वाभ्यास में भाग लें।
- किए गए सॉफ्टवेयर परीक्षण के लिए सभी आवश्यक रिपोर्ट बनाएं।
- सत्यापित करें कि सभी परीक्षण-संबंधी कार्य स्वीकृत मानकों और दिशानिर्देशों के अनुसार किए जाते हैं।
- उसे कोडिंग, सॉफ्टवेयर विकास और परीक्षण कार्यक्रमों को लागू करने में कुशल होना चाहिए।
- स्रोत कोड का निर्माण
- स्रोत कोड का प्रबंधन

2.1.14 एक टेस्ट इंजीनियर के लिए कोडिंग कौशल

यहां तक कि परीक्षकों को भी कोडिंग की मजबूत समझ होनी चाहिए। निम्न कारक इस केरी के प्रतिसाद को प्रभावित करते हैं:

- ब्लैक बॉक्स परीक्षण के लिए परीक्षक को कोडिंग में जानकार होने की आवश्यकता नहीं होती है। परीक्षक केवल उत्पाद में डेटा इनपुट करते हैं, परिणामों की जांच करते हैं, और आवश्यकतानुसार दोहराते हैं।
- व्हाइट बॉक्स परीक्षण के लिए परीक्षकों के पास कोडिंग कौशल होना चाहिए। इस वजह से, इस परीक्षण के रूप में भी जाना जाता है "कोड-संचालित परीक्षण." C, C #, C ++, DBMS और RDBMS जैसी विभिन्न अवधारणाओं का ज्ञान होना उपयोगी हो सकता है।
- क्योंकि उनमें स्टेटमेंट कवरेज, कोड कवरेज, साइक्लोमैटिक जटिलता आदि शामिल हैं, ऑटोमेशन टेस्टिंग और व्हाइट बॉक्स टेस्टिंग के लिए परीक्षकों को कोडिंग की ठोस समझ रखने की आवश्यकता होती है। ये विचार उपयुक्त डेटाबेस और प्रोग्रामिंग ज्ञान और क्षमताओं की मांग करते हैं।
- परीक्षण के लिए डेटाबेस सत्यापन आवश्यक हो सकता है। नतीजतन, परीक्षकों को बुनियादी SQL कमांड जैसे चयन, निर्माण, अपडेट आदि से परिचित होना चाहिए।
- SQL इंजेक्शन अनधिकृत कमांड डालकर डेटाबेस में सेंध लगाने की एक विधि है। परीक्षण चरण शुरू होने से पहले यह निर्धारित करने के लिए कि क्या किसी भी बग को पहले पहचाना जा सकता है, इस तरह के सुरक्षा खतरों के खिलाफ उत्पाद की रक्षा के लिए SQL और Javascript कमांड की ठोस समझ की आवश्यकता होती है। यह हेप्स परीक्षण टीम सक्रिय रूप से कोड समीक्षा में भाग लेती है।
- परीक्षक यह सुनिश्चित करने के लिए प्रभारी है कि सॉफ्टवेयर फुर्तीली परीक्षण में उच्च गुणवत्ता का है। परीक्षक विभिन्न प्रोग्रामर के साथ पूरी प्रक्रिया में कोडिंग पर सहयोग करता है। इस प्रकार, एक स्वचालित परीक्षण बनाने के लिए, परीक्षकों को कोडिंग से परिचित होना चाहिए।
- कोड की समीक्षा करना। कोड समीक्षा को स्थैतिक परीक्षण के एक घटक के रूप में माना जाता है, एक गतिविधि जो मुख्य रूप से गुणवत्ता विश्लेषकों द्वारा यह निर्धारित करने के लिए की जाती है कि परीक्षण चरण शुरू होने से पहले किसी भी बग की पहचान की जा सकती है या नहीं। यह हेप्स परीक्षण टीम सक्रिय रूप से कोड समीक्षा में भाग लेती है।

2.1.15 गुणवत्ता नियंत्रण

गुणवत्ता नियंत्रण का उद्देश्य यह सुनिश्चित करना है कि विशेष प्रक्रियाएं कंपनी के स्थापित मानदंडों का पालन करती हैं।

यह रणनीति तीन कारकों पर जोर देती है:

- **नियंत्रण, कार्य प्रबंधन, अच्छी तरह से परिभाषित और प्रबंधित प्रक्रियाओं, प्रदर्शन और अखंडता आवश्यकताओं और रिकॉर्ड पहचान सहित** तत्व।
- **क्षमता, ज्ञान, क्षमताओं, अनुभव और साख सहित**
- **नरम तत्व,** जिनमें लोगों की ईमानदारी, आत्म-आश्वासन, कंपनी संस्कृति, प्रेरणा, टीम भावना और गुणवत्ता कनेक्शन शामिल हैं

बाहरी बाजार पर किसी उत्पाद की पेशकश करने से पहले, इसका निरीक्षण नेत्रहीन रूप से किया जाता है, अक्सर सबसे छोटे विवरणों का अध्ययन करने के लिए एक स्टीरियोमाइक्रोस्कोप की सहायता से। निरीक्षकों को निषिद्ध उत्पाद दोषों की सूचियों और स्पष्टीकरणों के साथ प्रस्तुत किया जाएगा, जैसे कि दरारें और सतह की खामियां।

द्वितीय विश्व युद्ध के दौरान, गुणवत्ता नियंत्रण पर अधिक जोर दिया गया था। उस समय, गुणवत्ता नियंत्रण गुणवत्ता आश्वासन में विकसित हुआ, जिसे आज एक रणनीतिक दृष्टिकोण के रूप में जाना जाता है, न केवल वस्तुओं को बल्कि प्रक्रियाओं और सेवाओं को बढ़ाने के लिए एक विधि। उत्पादों और सेवाओं की गुणवत्ता का मूल्यांकन आयामों के एक अलग सेट के आधार पर किया जाना चाहिए। समग्र गुणवत्ता की जिम्मेदारी कार्यकारी टीम के साथ टिकी हुई है। प्रबंधन को रणनीति निर्धारित करनी चाहिए, गुणवत्ता कार्यक्रमों को लागू करना चाहिए और प्रबंधकों और कर्मचारियों को प्रेरित करना चाहिए। आमतौर पर, प्रबंधक पूरे संगठन की गुणवत्ता में सुधार या बनाए रखने का प्रयास करते हैं; इसे कुल गुणवत्ता प्रबंधन (TQM) के रूप में जाना जाता है। TQM निरंतर आधार पर गुणवत्ता सुधार के लिए एक संगठन-व्यापी प्रतिबद्धता पर जोर देता है। एक कंपनी को अपने गुणवत्ता नियंत्रण लक्ष्यों तक पहुंचने और उससे अधिक करने के लिए पूरी आपूर्ति श्रृंखला को शामिल करना चाहिए।

सारांश

- ट्रैक रखने की प्रक्रिया कि आपका उत्पाद आपके ग्राहकों के लिए उच्चतम संभव गुणवत्ता का है, गुणवत्ता आश्वासन (QA) परीक्षण के रूप में जाना जाता है।
- कार्यात्मक और गैर-कार्यात्मक सॉफ्टवेयर गुणवत्ता दो मूलभूत स्तर हैं।
- एक उपयुक्त और अनुकूल परीक्षण वातावरण बनाने के लिए शर्तों या लक्ष्यों का एक सेट जिसे पूरा किया जाना चाहिए, प्रवेश मानदंड के रूप में जाना जाता है।
- QA टीम ने निर्धारित समय सीमा और बजट का पालन सुनिश्चित करने के लिए निकास मानदंड नामक एक महत्वपूर्ण दस्तावेज बनाया।
- उपयोगकर्ता अनुसंधान अंतिम उपयोगकर्ता को ध्यान में रखते हुए निष्पक्ष रूप से डिजाइन करने में हमारी सहायता करने के लिए है। अनुसंधान हमें अपने लिए डिजाइन करने और डिजाइन में हमारी व्यक्तिगत प्राथमिकताओं को शामिल करने से रोकता है।
- तनाव परीक्षण में चुनौतीपूर्ण परिस्थितियों में लगातार या स्वीकार्य प्रदर्शन बनाए रखने के लिए सॉफ्टवेयर की क्षमता का मूल्यांकन करना शामिल है।
- सॉफ्टवेयर परीक्षण त्रुटियों या बग को खोजने के लिए सॉफ्टवेयर या एप्लिकेशन का उपयोग करने की प्रक्रिया है।
- एक सॉफ्टवेयर परीक्षक परीक्षण सूट डिजाइन करने और प्रयोज्य समस्याओं को समझने में सक्षम होना चाहिए।
- परीक्षण के लिए डेटाबेस सत्यापन आवश्यक हो सकता है। इसलिए, परीक्षकों को SQL कमांड जैसे चयन, निर्माण, अपडेट आदि की बुनियादी समझ होनी चाहिए।
- व्हाइट बॉक्स परीक्षण के लिए परीक्षकों के पास कोडिंग कौशल होना चाहिए। यही कारण है कि इस परीक्षण को कोड-संचालित परीक्षण भी कहा जाता है। C, C #, C ++, DBMS और RDBMS जैसी विभिन्न अवधारणाओं का ज्ञान होना उपयोगी हो सकता है।

एक्टिविटी

Con-Vid Session h ttps://www.youtube.com/watch?v=veG_WE2n-Lk

- इस सत्र में, ट्रेनर एक वीडियो चलाएगा।
- वीडियो परीक्षण के मूल सिद्धांतों की एक झलक देगा।
- वीडियो के लिए यूट्यूब लिंक है: https://www.youtube.com/watch?v=veG_WE2n-Lk
- प्रशिक्षु पिन ड्रॉप साइलेंस के साथ वीडियो का अवलोकन करेंगे।
- वे वीडियो से पॉइंटर्स को नोट कर सकते हैं जो उन्हें प्रासंगिक लग सकते हैं।
- प्रशिक्षु कक्षा में शिष्टाचार बनाए रखेंगे और कक्षा में बात नहीं करेंगे, कानाफूसी नहीं करेंगे या चर्चा नहीं करेंगे।
- किसी भी प्रश्न या भ्रम के मामले में, प्रशिक्षु उन्हें अपनी नोटबुक में लिखेंगे।

चर्चा सत्र

- प्रशिक्षक पूछताछ करेगा कि क्या प्रशिक्षुओं ने अध्याय को समझा है।
- इस गतिविधि में, प्रशिक्षक प्रशिक्षु से विषय से संबंधित कुछ प्रश्न पूछेगा।
- प्रशिक्षु विषय की अपनी समझ के अनुसार प्रश्नों का उत्तर देंगे।
- यदि प्रशिक्षुओं के मन में प्रश्न और भ्रम हैं; वे ट्रेनर के सामने उन लोगों को सामने रख सकते हैं।
- प्रशिक्षक यह सुनिश्चित करेगा कि वह प्रशिक्षुओं द्वारा पूछे गए प्रश्नों के उचित उत्तर दे सके।

अभ्यास

1. **बहुविकल्पीय प्रश्न:**

I. pesticide paradox क्या है?

a. परीक्षण दिखा सकता है कि दोष मौजूद हैं , लेकिन यह साबित नहीं कर सकता कि कोई दोष नहीं हैं

b. यदि एक ही परीक्षण बार-बार दोहराया जाता है, तो अंततः परीक्षण मामलों के एक ही सेट को अब कोई नया बग नहीं मिलेगा

c. दोषों को ढूंढना और ठीक करना मदद नहीं करता है यदि निर्मित सिस्टम अनुपयोगी है और उपयोगकर्ताओं की जरूरतों और अपेक्षाओं को पूरा नहीं करता है।

d. मॉड्यूल की एक छोटी संख्या में अधिकांश दोष होते हैं

II. निम्नलिखित में से किस परीक्षण को व्हाइट-बॉक्स परीक्षण भी कहा जाता है?

a. अनुमान लगाने में त्रुटि तकनीक

b. डिजाइन आधारित परीक्षण

c. संरचनात्मक परीक्षण

d. उपरोक्त में से कोई नहीं

III. परीक्षण के निकास मानदंड को परिभाषित करने के लिए निम्नलिखित परीक्षण दस्तावेज़ का उपयोग किस में किया जाता है?

a. परीक्षण सारांश रिपोर्ट

b. परीक्षण का मामला

c. परीक्षण योजना

d. दोष रिपोर्ट

IV. ----------क्या वे सॉफ़्टवेयर त्रुटियाँ हैं जो कोडिंग चरण के दौरान हुई थीं?

a. विफलताओं

b. त्रुटियों

c. दोष

d. बग्स

2. **सॉफ्टवेयर परीक्षण के सात आवश्यक सिद्धांतों की सूची बनाएं।**

नोट्स

संबंधित वीडियो देखने के लिए QR कोड को स्कैन करें

https://www.youtube.com/watch?v=LKUtzEHYK2I

QA परीक्षण और संस्करण नियंत्रण

Skill India
कौशल भारत-कुशल भारत

सत्यमेव जयते
GOVERNMENT OF INDIA
MINISTRY OF SKILL DEVELOPMENT
& ENTREPRENEURSHIP

N·S·D·C
National
Skill Development
Corporation
Transforming the skill landscape

3. सॉफ्टवेयर उत्पादों/ ऍप्लिकेशन्स/मॉड्यूल के लिए डिजाइन Testing

IT - ITeS SSC
nasscom

यूनिट 3.1 - सॉफ्टवेयर उत्पादों/ऍप्लिकेशन्स/मॉड्यूलों के लिए डिजाइन Testing

SSC/N1301

प्रमुख सीखने के परिणाम

इस मॉड्यूल के अंत तक, प्रतिभागी निम्न में सक्षम होंगे:

1. कोड का उपयोग करके Testing के लिए सॉफ़्टवेयर आवश्यकताओं के साथ समस्याओं की पहचान करने का तरीका प्रदर्शित करें।
2. आवश्यकताओं से संबंधित Test cases को संशोधित करने की प्रक्रिया की जांच करें।

यूनिट 3.1: सॉफ्टवेयर उत्पादों/ऍप्लिकेशन्स के लिए डिजाइन Testing/मॉड्यूल

यूनिट के उद्देश्य

इस यूनिट के अंत तक, प्रतिभागी निम्न में सक्षम होंगे:

1. स्रोत कोड की अवधारणा और एप्लीकेशन विकास में इसके उपयोग पर चर्चा करें।
2. Test योजना, Test cases और/या स्वचालित स्क्रिप्ट के संशोधन की प्रक्रिया पर चर्चा करें।
3. तकनीकी मुद्दों के लिए टीम को डिजाइन करने में वृद्धि के पदानुक्रम पर चर्चा करें।
4. संबंधित हितधारकों के परामर्श से Testing के लिए आवश्यकताओं के साथ मुद्दों की पहचान करने की प्रक्रिया पर चर्चा करें।
5. सॉफ्टवेयर डिजाइनिंग के लिए राइटिंग सोर्स कोड, रिव्यू कोड आदि का प्रदर्शन करें।
6. स्वचालन के लिए आवश्यकताओं के लिए प्रासंगिक Test cases को संशोधित करें।
7. सभी आवश्यकताओं को कवर करने के लिए एक Testing पद्धति विकसित करने के लिए चरणों की जांच करें।
8. आवश्यकताओं के लिए प्रासंगिक स्वचालित स्क्रिप्ट के लिए डिज़ाइन प्रक्रिया लागू करें।
9. विभिन्न ऍप्लिकेशन्स के लिए स्रोत कोड विकसित करने का तरीका प्रदर्शित करें।

3.1.1 स्रोत कोड

स्रोत कोड कंप्यूटर प्रोग्रामिंग भाषाओं में प्रोग्रामर द्वारा लिखे गए निर्देशों का एक संग्रह है। एक बार जब प्रोग्रामर ने स्रोत कोड की एक पंक्ति या सेट लिख लिया है, तो वे इसे वेबसाइट, एप्लिकेशन या अन्य प्रकार के कंप्यूटर प्रोग्राम में लागू कर सकते हैं ताकि इसे संचालित करने का निर्देश दिया जा सके। उदाहरण के लिए, कोई व्यक्ति स्रोत कोड की एक पंक्ति लिख सकता है जो किसी वेबसाइट को एक विशेष तरीके से प्रतिक्रिया देने का निर्देश देता है जब कोई उपयोगकर्ता किसी विशिष्ट तत्व, जैसे कि बटन या लिंक, को वेब पेज पर हिट करता है।

आमतौर पर, प्रोग्रामर एक टेक्स्ट-आधारित एप्लिकेशन में स्रोत कोड की रचना करते हैं, जैसे कि वर्ड प्रोसेसर, और फिर इसे एक प्रारूप में बदलने के लिए एक कंपाइलर का उपयोग करते हैं जिसे कंप्यूटर प्रोग्राम समझ सकते हैं। जब स्रोत कोड को इस तरह से अनुवादित किया जाता है, तो यह ऑब्जेक्ट कोड बन जाता है। हालांकि, अनुवाद से पहले, स्रोत कोड आमतौर पर समझ में आता है, क्योंकि यह उस फ़ंक्शन की व्याख्या करता है जिसे प्रोग्रामर किसी वेबसाइट या प्रोग्राम में जोड़ना चाहता है। एक प्रोग्रामर भविष्य के संदर्भ के लिए कंप्यूटर पर स्रोत कोड स्टोर कर सकता है, या तो डेटाबेस में या इसकी हार्ड डिस्क पर, अनुवाद के बाद। इसके अतिरिक्त, प्रोग्रामर अपने स्रोत कोड की भौतिक प्रतियां प्रिंट कर सकते हैं।

3.1.2 स्रोत कोड के उपयोगकर्ता

आमतौर पर, स्रोत कोड को विकसित और उपयोग करने वाली भूमिका कंप्यूटर प्रोग्रामर है, एक प्रौद्योगिकी विशेषज्ञ जो नई वेबसाइटों और ऍप्लिकेशन्स के लिए कोड लिखता है। वे जिस संगठन के लिए काम करते हैं और जो कार्य वे करते हैं, उसके आधार पर, कंप्यूटर प्रोग्रामर लगभग किसी भी प्रकार के एप्लिकेशन या प्रोग्राम में स्रोत कोड उत्पन्न और कार्यान्वित करने में सक्षम होते हैं। वे नए स्रोत कोड (IDE) लिखने के लिए विभिन्न प्रकार के उपकरणों का भी उपयोग कर सकते हैं, जैसे कि एक दृश्य प्रोग्रामिंग उपकरण, एक Text संपादक, या एक एकीकृत विकास वातावरण।

आमतौर पर, कंप्यूटर प्रोग्रामर सरल कार्यक्रमों के लिए कोड का एक टुकड़ा लिखकर स्रोत कोड को नियोजित करते हैं, लेकिन यदि एप्लिकेशन व्यापक है तो वे एक ही प्रोजेक्ट के लिए स्रोत कोड के कई सेट लिख सकते हैं। ऐसी स्थितियों में, जटिल कमांड की सुविधा प्रदान करने वाली अलग-अलग प्रोग्रामिंग भाषाओं को नियोजित करने की आवश्यकता हो सकती है। स्रोत कोड लिखने के लिए पेशेवरों द्वारा उपयोग की जाने वाली कुछ सबसे सामान्य कंप्यूटर प्रोग्रामिंग भाषाएं निम्नलिखित हैं:

- C
- C++
- Java
- JavaScript
- Pascal
- Python
- Basic
- PHP

3.1.3 सामान्य स्रोत कोड एप्लीकेशन

स्रोत कोड के कुछ लगातार एप्लीकेशन यहां दिए गए हैं:

1. **एक वेबसाइट के आधार का निर्माण करें**

   स्रोत कोड के सबसे आम उपयोगों में से एक एक ढांचा स्थापित करना है जिस पर वेब डेवलपर्स और वेब डिजाइनर नई वेबसाइट बना सकते हैं। स्रोत कोड को कभी-कभी एक वेबसाइट की नींव के रूप में संदर्भित किया जाता है क्योंकि इसमें निर्देश होते हैं जो वेबसाइट को चलाने और उपयोगकर्ता इंटरैक्शन का जवाब देने की अनुमति देते हैं।

   प्रोग्रामर द्वारा उपयोग की जाने वाली प्रोग्रामिंग भाषा स्रोत कोड के साथ वेबसाइट की नींव बनाने का एक अलग हिस्सा है। ऐसा इसलिए है क्योंकि HTML, एक मार्कअप भाषा, अक्सर वेबसाइट की नींव बनाने के लिए उपयोग की जाती है, जबकि अधिकांश प्रोग्रामर स्रोत कोड लिखने के लिए प्रोग्रामिंग भाषाओं का उपयोग करते हैं। HTML कोडर्स को वेबसाइट की Attributes को परिभाषित करने और संशोधित करने की अनुमति देता है, जैसे कि प्राथमिक सामग्री पैराग्राफ, शीर्षक, और हाइलाइटिंग या जोर देने के किसी भी उदाहरण, जैसे बोल्ड या इटैलिक टेक्स्ट, जिससे यह वेबसाइट नींव विकसित करने के लिए एक अत्यधिक उपयोगी उपकरण बन जाता है।

2. **एक कार्यक्रम में एक विशिष्ट फ़ंक्शन जोड़ें**
   कंप्यूटर प्रोग्राम, एप्लिकेशन या वेबसाइट में एक विशिष्ट फ़ंक्शन जोड़ना स्रोत कोड का एक और विशिष्ट उपयोग है। यह कई प्रौद्योगिकी पेशेवरों द्वारा स्रोत कोड के लिए प्रमुख उद्देश्य माना जाता है, क्योंकि यह विकास के तहत कार्यक्रमों के कामकाज और प्रस्तुति को महत्वपूर्ण रूप से प्रभावित कर सकता है। उदाहरण के लिए, वीडियो गेम पर काम करने वाला एक प्रोग्रामर, स्रोत कोड में एक निर्देश जोड़ सकता है जो सॉफ़्टवेयर को एक विशेष स्क्रीन प्रदर्शित करने का निर्देश देता है जब खिलाड़ी का अवतार गेम में एक विशिष्ट बिंदु तक पहुंच जाता है।
3. **एल्गोरिदम का संचार करें**
   इसके अतिरिक्त, प्रोग्रामर स्रोत कोड के माध्यम से एल्गोरिदम साझा कर सकते हैं। यह संभव है क्योंकि अधिकांश प्रोग्रामर वर्ड प्रोसेसर या अन्य टेक्स्ट-आधारित सॉफ़्टवेयर में स्रोत कोड बनाते हैं, जो प्रोग्रामर को अपने कोड की हार्ड कॉपी प्रिंट करने या दस्तावेज़ फ़ाइल के रूप में संग्रहीत करने की अनुमति देता है। फिर, वे इन फ़ाइलों को अन्य प्रोग्रामर के साथ साझा कर सकते हैं जो अपने काम में रुचि रखते हैं या जो अपनी परियोजनाओं में से एक में एक विशिष्ट स्रोत कोड शामिल करना चाहते हैं। कुछ प्रोग्रामर दूसरों को कोड लिखने का तरीका सिखाने के लिए निर्देश मैनुअल और संदर्भ पुस्तकों जैसे आइटमों में मुद्रित स्रोत कोड का उपयोग करते हैं।

3.1.4 संकलक और Interpretor

कंप्यूटर को प्रोग्रामर के स्रोत कोड को आगे बढ़ाने के लिए, दोनों के बीच एक अनुवाद होना चाहिए, जो एक अतिरिक्त प्रोग्राम का आकार लेता है। सहायता के लिए यह अपील एक संकलक या Interpretor का आकार ले सकती है।
कंपाइलर: इस प्रकार का प्रोग्राम स्रोत कोड को निष्पादन योग्य कोड में परिवर्तित (संकलित) करता है जिसे प्रक्रिया समझती है। यह मशीन कोड एक निष्पादन योग्य फ़ाइल स्वरूप में सहेजा गया है।
एक Interpretor स्रोत कोड लाइन द्वारा लाइन का अनुवाद करता है और सीधे इसे निष्पादित करता है। अनुवाद प्रक्रिया एक कंपाइलर की तुलना में काफी तेज है, लेकिन निष्पादन धीमा है और मेमोरी आवश्यकताएं अधिक हैं।

3.1.5 स्रोत Text की संरचना

प्रोग्राम लिखते समय, आप प्रोग्रामिंग भाषा के आधार पर विशेष मानकों तक ही सीमित होते हैं। हालांकि, बहुत कम प्रोग्रामिंग भाषाएं कुछ भी नहीं से उत्पन्न होती हैं, और विशाल बहुमत एक दूसरे पर निर्मित होते हैं। नतीजतन, कई कंप्यूटर कोड में निम्नलिखित तत्वों का उपयोग किया जाता है:

- **आदेश:** निर्देश आमतौर पर सभी ऐप्स की नींव होते हैं। यहां, प्रोग्रामर भविष्य के कार्यक्रम के लिए स्पष्ट करता है कि क्या पूरा किया जाना चाहिए। उदाहरण के लिए, ऐसे आदेश कुछ गणना शुरू कर सकते हैं या Text दिखा सकते हैं।
- **वेरिएबल:** वेरिएबल खाली स्लॉट हैं जिन्हें जानकारी से भरा जा सकता है। इन्हें बार-बार स्रोत कोड के भीतर एक निर्दिष्ट नाम के साथ संदर्भित किया जाता है।

- **तुलना:** अधिकांश कार्यक्रम संरचनाएं उन प्रश्नों पर आधारित होती हैं जो अगर-तब योजना, या प्रस्तावात्मक तर्क को नियोजित करते हैं। यदि एक निश्चित सत्य मान दर्ज किया जाता है, तो एक समाधान दूसरे के बजाय सक्रिय होता है।
- **लूप:** केरी स्रोत कोड में लूप के आधार के रूप में भी काम कर सकती हैं। एक निर्दिष्ट Boundary तक पहुंचने तक एक आदेश दोहराया जाता है। जब प्रोग्राम लूप को पूरा करता है और शेष कोड निष्पादित करता है।
- **टिप्पणियाँ:** सभी सामान्य प्रोग्रामिंग भाषाओं में, कोड की पंक्तियों को टिप्पणियों के साथ एनोटेट किया जा सकता है। यह स्रोत कोड में Text को शामिल करने की अनुमति देता है जिसे एप्लिकेशन द्वारा अनदेखा किया जाता है। उदाहरण के लिए, स्रोत कोड में टिप्पणियाँ जोड़ी जाती हैं ताकि आप या अन्य डेवलपर भविष्य में कोड के भागों को समझना जारी रख सकें.

हर बार जब कोई स्रोत कोड बनाया जाता है, तो एक समस्या का समाधान किया जाना चाहिए। डेवलपर्स समाधान देने के लिए सॉफ्टवेयर बनाते हैं। हालांकि, ऐसा करने की विधि पत्थर में सेट नहीं है। भले ही वे एक ही प्रोग्रामिंग भाषा का उपयोग कर रहे हों, एक ही समस्या पर काम करने वाले दो प्रोग्रामर के लिए स्रोत कोड का उत्पादन करना संभव है जो एक दूसरे से काफी भिन्न हैं।

यहां तक कि जब हमेशा एक भी सही समाधान नहीं होता है, तब भी सभी प्रोग्रामिंग कार्यों में एक बात समान होती है: एक सभ्य स्रोत कोड अनावश्यक Text से बचा जाता है। जोड़ा गया Text समस्या को अधिक परेशान, धीमा और त्रुटि-प्रवण बनाता है। चूंकि स्पेगेटी कोड की संरचना झूलते नूडल्स की प्लेट के रूप में मुड़ सकती है, इसका उपयोग विशेष रूप से गुप्त स्रोत कोड को संदर्भित करने के लिए किया जाता है जिसे विशेषज्ञ भी समझ नहीं सकते हैं।

3.1.6 कोड स्रोत बनाएं

स्रोत कोड लिखने के लिए, आपको केवल एक साधारण Text संपादक की आवश्यकता होती है, जैसे कि विंडोज पर नोटपैड या मैक पर टेक्स्टएडिट। इस प्रकार, स्रोत कोड को प्रोग्रामिंग भाषा के लिए उपयुक्त फ़ाइल नाम एक्सटेंशन के साथ सादे Text (जैसे, ASCII या UTF-8 में) के रूप में सहेजा जा सकता है। इसलिए, यदि आप अपनी हार्ड ड्राइव पर ".cpp" एक्सटेंशन वाली फ़ाइल खोजते हैं, तो यह संभवतः C ++ प्रोग्रामिंग कोड वाली एक टेक्स्ट फ़ाइल है।

3.1.7 सॉफ्टवेयर Testing

सॉफ्टवेयर विकसित करने की प्रक्रिया में एक महत्वपूर्ण कदम के रूप में Testing शामिल है। इसमें यह सुनिश्चित करने के लिए सॉफ़्टवेयर का गहन मूल्यांकन करना शामिल है कि यह आपके ग्राहक की आवश्यकताओं और उद्देश्यों को पूरा करता है। कार्यान्वयन चरण से पहले, Testing का मुख्य उद्देश्य सॉफ्टवेयर में सभी खामियों और त्रुटियों का पता लगाना है। सॉफ़्टवेयर त्रुटियाँ क्लाइंट के व्यवसाय को नुकसान पहुँचा सकती हैं यदि उन्हें परिनियोजन से पहले ठीक नहीं किया जाता है। उन मुद्दों को हल करने में उच्च लागत शामिल होगी। दूसरी ओर, Testing सॉफ्टवेयर की गुणवत्ता को बनाए रखने और अपने ग्राहकों का विश्वास जीतने में सक्षम बनाता है। तैयार उत्पाद भी सटीक, लगातार और भरोसेमंद रूप से प्रदर्शन करेगा, जिसके परिणामस्वरूप रखरखाव की लागत कम होगी। यह कहने के बाद, सॉफ्टवेयर Testing के लिए विभिन्न दृष्टिकोण हैं। वह तरीका जो Testing प्रक्रिया को शीघ्रता से निष्पादित करता है और फुर्तीली सिद्धांतों का पालन करता है, सबसे अच्छा है। हम इस यूनिट में विभिन्न सॉफ्टवेयर Testing शब्दावली और Testing डिजाइन की जांच करेंगे।

3.1.8 Test परिदृश्य

सॉफ्टवेयर Testing के बिना सॉफ्टवेयर विकास अधूरा है। Test cases का विकास Test परिदृश्यों द्वारा बहुत सहायता प्राप्त है। सॉफ़्टवेयर Testing के दौरान उपयोगकर्ता किसी एप्लिकेशन के साथ कैसे इंटरैक्ट करेगा, इसका गहन विवरण या रिकॉर्ड Test परिदृश्य कहलाता है। इसे टेस्ट कंडीशन या टेस्ट पॉसिबिलिटी के नाम से भी जाना जाता है। यह दिखाने के लिए कि Tester ने क्या किया है; Test परिदृश्यों का उपयोग किया जाता है। Test परिदृश्यों की सहायता से Test cases में सुधार किया जाता है, जो परीक्षकों को सभी संभावित परिणामों की पहचान करने में मदद करता है - अपेक्षित और अप्रत्याशित दोनों - ताकि बग को जल्द से जल्द रिपोर्ट किया जा सके। हम Test परिदृश्यों का उपयोग करके अंतिम उपयोगकर्ता के दृष्टिकोण से ऍप्लिकेशन्स के प्रदर्शन का मूल्यांकन करते हैं। Test परिदृश्यों का निर्माण करते समय, परीक्षकों को वास्तविक दुनिया के परिदृश्यों को समझने के लिए खुद को अंतिम उपयोगकर्ता के परिप्रेक्ष्य में रखना चाहिए, जिसे सॉफ़्टवेयर को रिलीज़ के बाद संभालने की आवश्यकता होगी।

3.1.9 उच्च स्तरीय परिदृश्यों का Test करने के लिए डिजाइनिंग तरीके

सॉफ़्टवेयर एप्लिकेशन का Test करने के लिए Test cases का उपयोग करने के बजाय, परिदृश्य Testing सॉफ्टवेयर Testing में उपयोग की जाने वाली एक विधि है।

उपयोग के मामलों से बनाए गए परिदृश्यों का उपयोग करने वाले Testing को परिदृश्य Testing के रूप में जाना जाता है। इसके अलावा, परिदृश्य Testing का उपयोग करके, जटिल एप्लीकेशन तर्क का Test करने के लिए कठिन-से-मूल्यांकन Testing परिदृश्यों का उपयोग किया जा सकता है।

परिदृश्य Testing की कुछ विशेषताएं हैं:

- **सुसंगत:** Testing परिदृश्य इस बात पर आधारित होना चाहिए कि सॉफ़्टवेयर कैसे नियोजित किया जाता है।
- उन्हें विश्वसनीय होना चाहिए और एक परिदृश्य पर ध्यान केंद्रित करना चाहिए जो वास्तविकता में हो सकता है।
- **प्रेरक:** Testing परिदृश्य में विफल होने की स्थिति में, उन्हें हितधारकों को समस्याओं को ठीक करने के लिये प्रोत्साहित करना चाहिये।
- **जटिलता:** जटिल प्रोग्राम या एप्लीकेशन प्रवाह आमतौर पर Testing परिदृश्यों का एक हिस्सा होते हैं।
- **मूल्यांकन करने में आसान:** चूंकि Testing परिदृश्य में जटिल तर्क शामिल है, इसलिए परीक्षा परिणाम का आकलन करना आसान होना चाहिए।

3.1.10 Testing परिदृश्य टेम्पलेट

एक Testing परिदृश्य दस्तावेज़ में निम्न फ़ील्ड हो सकते हैं:

- **मॉड्यूल:** एप्लिकेशन का मॉड्यूल या घटक।
- **आवश्यकताएँ:** ID एक वैकल्पिक फ़ील्ड है जिसे एसआरएस से जोड़ा जा सकता है।
- **Testing ScenarioId:** Testing परिदृश्य इस फ़ील्ड द्वारा पहचाने जाते हैं।
- **वर्णन:** Testing परिदृश्य का उद्देश्य वर्णन फ़ील्ड में वर्णन किया गया है।

Template for a Test Scenario			
Module	**RequirementId**	**TestScenarioId**	**Test Scenario Description**
लॉगिन	US0001	TS_01	यह देखने के लिए जांचें कि क्या उपयोगकर्ता मान्य जानकारी का उपयोग करके लॉग इन कर सकता है ..
	US0002	TS_02	सुनिश्चित करें कि उपयोगकर्ता दोषपूर्ण जानकारी का उपयोग करके लॉग इन नहीं कर सकता है।
	US0003	TS_03	लॉगिन पृष्ठ के अनिवार्य फ़ील्ड चेक सत्यापन की पुष्टि करें।

तालिका 3.1.1. Testing परिदृश्य टेम्पलेट

Gmail के लिए Test Case - इनबॉक्स कार्यक्षमता

1. यह देखने के लिए जाँचें कि क्या हाल ही में प्राप्त ईमेल हाइलाइट किया गया है और इनबॉक्स अनुभाग में दिखाया गया है.
2. यह देखने के लिए जाँचें कि क्या हाल ही में प्राप्त ईमेल में प्रेषक का ईमेल पता या नाम, संदेश का विषय, और संदेश का मुख्य भाग (एक पंक्ति में छंटनी किया गया) सही ढंग से प्रदर्शित है.
3. सुनिश्चित करें कि उपयोगकर्ता को ईमेल सामग्री पर निर्देशित किया जाता है जब वे नए प्राप्त ईमेल पर क्लिक करते हैं।
4. सुनिश्चित करें कि ईमेल की सामग्री वांछित स्रोत स्वरूपण के साथ सही ढंग से प्रदर्शित होती है।
5. जांचें कि कोई भी ईमेल अटैचमेंट संलग्न और डाउनलोड करने योग्य दोनों हैं।
6. डाउनलोड करने से पहले, सुनिश्चित करें कि अटैचमेंट वायरस-स्कैन किए गए हैं।
7. यह सुनिश्चित करने के लिए जांचें कि पढ़े गए के रूप में चिह्नित कोई भी ईमेल हाइलाइट नहीं किया गया है।
8. जांचें कि इनबॉक्स अनुभाग में प्रदर्शित ईमेल सूची में प्रत्येक ईमेल के अंत में मेल पढ़ने का समय शामिल है, दोनों पढ़े और अपठित।
9. यह देखने के लिए जांचें कि क्या Gmail के बाएं साइडबार में "इनबॉक्स" Text के बगल में अपठित ईमेल की संख्या दिखाई गई है।
10. जांचें कि हर बार जब आप एक नया ईमेल प्राप्त करते हैं, तो अपठित ईमेल की संख्या एक से बढ़ जाती है।
11. यह देखने के लिए जांचें कि जब कोई ईमेल पढ़ा जाता है (पढ़ा गया के रूप में चिह्नित), अपठित ईमेल की संख्या एक से कम हो जाती है।
12. सुनिश्चित करें कि सभी उपयोगकर्ता प्रतिScript शुल्क में ई-मेल पते देख सकते हैं.
13. यह सुनिश्चित करने के लिए जांचें कि बीसीसी में ईमेल प्राप्तकर्ता उपयोगकर्ता से छिपे हुए हैं।
14. यह देखने के लिए जांचें कि सभी आने वाले ईमेल "इनबॉक्स" अनुभाग में ढेर हो गए हैं और उपलब्ध स्थान के आधार पर समय-समय पर हटा दिए जाते हैं।
15. यह देखने के लिए जांचें कि क्या ईमेल Gmail के अलावा अन्य ईमेल पतों से प्राप्त किया जा सकता है, जैसे कि Yahoo, Hotmail, आदि।

3.1.11 Test Case क्या है?

एक Test Case को परिस्थितियों के एक सेट के रूप में वर्णित किया जाता है जिसके तहत एक Tester यह निर्धारित करता है कि ग्राहक की आवश्यकताओं के अनुसार कोई सॉफ़्टवेयर एप्लिकेशन काम कर रहा है या नहीं। पूर्व शर्तें, केस नाम, इनपुट आवश्यकताएं और प्रत्याशित परिणाम सभी Test Case डिजाइन में शामिल हैं। पहले स्तर पर एक कार्रवाई, Test Case Testing परिदृश्यों से प्राप्त होते हैं।

एक Test Case को परिस्थितियों के एक सेट के रूप में वर्णित किया जाता है जिसके तहत एक Tester यह निर्धारित करता है कि ग्राहक की आवश्यकताओं के अनुसार कोई सॉफ़्टवेयर एप्लिकेशन काम कर रहा है या नहीं। एक Test Case के डिजाइन में पूर्व शर्तें, मामले का नाम, इनपुट स्थितियां और प्रत्याशित परिणाम शामिल हैं। एक Test case पहले स्तर पर एक क्रिया है और Testing परिदृश्यों से लिया गया है।

- Test case Testing दृष्टिकोण, Testing प्रक्रिया, पूर्व शर्तों और प्रत्याशित परिणामों पर व्यापक जानकारी प्रदान करता है। ये Testing चरण के दौरान यह निर्धारित करने के लिए किए जाते हैं कि क्या सॉफ़्टवेयर एप्लिकेशन उस फ़ंक्शन को निष्पादित करने में सक्षम है जिसके लिए इसे बनाया गया था।
- डिफेक्ट को Test Case ID के साथ जोड़कर, Test Case Tester को डिफेक्ट रिपोर्टिंग में सहायता करता है। यदि कोई डेवलपर कुछ चूक गया है, तो इसे इन पूर्ण-प्रूफ Test cases के निष्पादन के दौरान खोजा जा सकता है, इसलिए विस्तृत Test Case प्रलेखन Testing टीम के लिए पूर्ण प्रूफ गार्ड के रूप में कार्य करता है।
- Test cases को लिखा जाना चाहिए ताकि हम Testing के लिए किसी भी Attributes को याद न करें, जिसके लिए आवश्यक है कि हमारे पास इनपुट प्राप्त करने के लिए आवश्यकताएं हों। फिर, एकरूपता बनाए रखने के लिए, हमारे पास Test cases के लिए एक टेम्पलेट होना चाहिए जिसका उपयोग सभी Testing इंजीनियर Testing दस्तावेज़ बनाने के लिए कर सकते हैं।

3.1.12 हम Test Case क्यों लिखते हैं?

हम निम्नलिखित कारणों से Test Case लिखते हैं:

- हम Test Case को देखेंगे और Test Case के निष्पादन में स्थिरता की मांग करने के लिए एप्लीकेशन का Test शुरू करेंगे।
- बेहतर Testing कवरेज सुनिश्चित करने के लिए, हमें सभी संभावित परिदृश्यों को कवर करना चाहिए और उन्हें दस्तावेज करना चाहिए ताकि हमें उन सभी पर नज़र न रखनी पड़े।
- एक विशिष्ट व्यक्ति के बजाय, यह प्रक्रिया पर निर्भर करता है: कंपनी छोड़ने से पहले पहले, दूसरे और तीसरे रिलीज के दौरान एक Testing इंजीनियर द्वारा एक एप्लीकेशन का Test किया गया था। Testing इंजीनियर ने कई मूल्यों को प्राप्त करके एप्लीकेशन का पूरी तरह से Testing किया क्योंकि उन्होंने एक मॉड्यूल को समझा। यह नए व्यक्ति के लिए चुनौतीपूर्ण हो जाता है यदि व्यक्ति तीसरी रिलीज से चूक जाता है। नतीजतन, सभी व्युत्पन्न मूल्यों को भविष्य में उपयोग के लिए प्रलेखित किया जाता है।
- उत्पाद पर प्रत्येक नए Testing इंजीनियर को प्रशिक्षित करने से रोकने के लिए: Testing इंजीनियर जानकारी और परिदृश्यों के धन के साथ प्रस्थान करता है। इन परिदृश्यों को रिकॉर्ड किया जाना चाहिए ताकि नया Testing इंजीनियर प्रदान किए गए उदाहरणों का उपयोग करके Testing कर सके और नए परिदृश्य भी बना सके।

एप्लीकेशन की दक्षता एक Test Case लिखने का मुख्य लक्ष्य है।

3.1.13 Test Case टेम्पलेट

विभिन्न Testing परिदृश्यों के लिए Test cases का एक संगठित सेट एक दस्तावेज़ में निहित होता है जिसे Test case टेम्पलेट के रूप में जाना जाता है. इन Test cases का उपयोग यह निर्धारित करने के लिए किया जाता है कि प्रोग्राम में अपेक्षित कार्यक्षमता है या नहीं। नीचे दिए गए Test Case टेम्पलेट का उपयोग यह जांचने के लिए किया जा सकता है कि कोई एप्लिकेशन उपयोगकर्ता के उपयोगकर्ता नाम को सही तरीके से बदलता है या नहीं।

Test Case 1: उपयोगकर्ता नाम बदलना

विवरण:

- गेम को कस्टमाइज़ करते समय उपयोगकर्ता मान्य मानों का चयन करता है।

इस Test Case के लिए पूर्व शर्तें

- उपयोगकर्ता एप्लिकेशन गेम में लॉग इन है
- उपयोगकर्ता के पास सक्रिय इंटरनेट कनेक्शन है

उपयोगकर्ता नए उपयोगकर्ता नाम की पुष्टि करता है

Test Case						
Steps	**Step Description**	**Data/Value**	**Expected Result**	**Actual Result (if different from expected)**	**Successful/ Failed**	**Log Number (if failed)**
1	User types in new username in text box	Bob	The characters the user is typing should appear in the text box		Successful	
2	User presses 'Confirm' button		A prompt informs the user their username has been successfully updated		Successful	
			Test Case Status	*Successful*		

User cancels changing username

Test Case						
Steps	**Step Description**	**Data/Value**	**Expected Result**	**Actual Result (if different from expected)**	**Successful/ Failed**	**Log Number (if failed)**
1	User presses 'Cancel' button		The application returns to Home screen with no change made to the username	The application does not respond	Failed	1
			Test Case Status	*Failed*		

सारणी 3.1.2. Test Case टेम्पलेट

3.1.14 Test Case डिजाइन तकनीक

सॉफ्टवेयर Testing तकनीक बेहतर Test cases को बनाना आसान बनाती है। संपूर्ण Testing की असंभवता के कारण, मैनुअल Testing तकनीक Testing कवरेज को बढ़ाते हुए Test cases की संख्या को कम करने में मदद करती है। वे Testing की स्थिति को पहचानना आसान बनाते हैं जो अन्यथा स्पॉट करना मुश्किल होगा।

1. **Bsoundary मूल्य विश्लेषण**

   BVA एक ऐसी तकनीक है जिसमें Test डेटा की Boundary निर्धारित करना शामिल है। अधिकतम, न्यूनतम, सीमाओं के अंदर या बाहर, और त्रुटियों जैसे मान सभी शामिल हैं। यह Strategy इस धारणा पर आधारित है कि डेवलपर्स पूर्व निर्धारित बाधाओं के भीतर काम करते समय त्रुटियां करने की संभावना रखते हैं। संक्षेप में, इनपुट डोमेन के केंद्र के बजाय, किनारों पर कई त्रुटियां होती हैं। अपने मूल सिद्धांत के अनुसार, यदि कोई प्रणाली इन विशेष Boundary मूल्यों के लिए अच्छा प्रदर्शन करती है, तो यह उनके बीच किसी भी मूल्य के लिए भी अच्छा प्रदर्शन करेगी।

संक्षेप में, इनपुट डोमेन के केंद्र के बजाय, किनारों पर कई त्रुटियां होती हैं। अपने मूल सिद्धांत के अनुसार, यदि कोई प्रणाली इन विशेष Boundary मूल्यों के लिए अच्छा प्रदर्शन करती है, तो यह उनके बीच किसी भी मूल्य के लिए भी अच्छा प्रदर्शन करेगी।

उदाहरण के लिए: एक टेक्स्ट फ़ील्ड Test Case के रूप में 1 से 100 तक इनपुट की अनुमति देता है।

Boundary मान विश्लेषण Test Case - 0,1,2 और 99,100,101

2. तुल्यता वर्ग विभाजन

समानता वर्ग विभाजन अभी तक एक और ब्लैक-बॉक्स Testing Strategy है। यह विधि इनपुट डोमेन को कई वर्गों या डिवीजनों में विभाजित करती है। यह माना जाता है कि सॉफ्टवेयर इस परिदृश्य में एक ही वर्ग से डेटा का Test करने के लिए लगातार प्रतिक्रिया देगा। यह सुनिश्चित करने के लिए कि कक्षा में अन्य सभी संभावित इनपुट शामिल हैं, प्रति वर्ग केवल एक इनपुट का Test किया जाना चाहिए। प्रत्येक वर्ग के लिए Test cases का निर्माण Next कदम है।

Test Scenerio	Description	Outcome
1	यदि उपयोगकर्ता 0 और 6 के बीच Characters दर्ज करता है	वे इसे स्वीकार नहीं करेंगे।
2	यदि उपयोगकर्ता 7 और 12 के बीच Characters दर्ज करता है	सिस्टम इसे स्वीकार करेगा।
3	यदि उपयोगकर्ता 12 से अधिक Characters दर्ज करता है	सिस्टम इसे स्वीकार नहीं करेगा।

सारणी. 3.1.3. समतुल्यता वर्ग विभाजन

उदाहरण के लिए: एक सॉफ़्टवेयर एप्लिकेशन में पासवर्ड फ़ील्ड पर विचार करें जो न्यूनतम 7 Characters और अधिकतम 12 Characters को स्वीकार करता है।

उपरोक्त स्थिति का विश्लेषण करने के बाद हम 3 विभाजन कर सकते हैं: 0-6, 7-12, 13-17

1. **निर्णय-आधारित तालिका Test**

Test cases को विकसित करने के लिए एक उपकरण एक निर्णय तालिका है, जिसे कारण और प्रभाव तालिका के रूप में भी जाना जाता है। सिस्टम ने इनपुट और प्रत्याशित परिणामों का एक सारणीबद्ध प्रतिनिधित्व पेश किया ताकि उपयोगकर्ता देख सकें कि सिस्टम विभिन्न इनपुट संयोजनों पर कैसे प्रतिक्रिया करता है। जब सॉफ्टवेयर के निष्कर्ष को निर्धारित करने के लिए इनपुट का उपयोग किया जाता है, तो यह Testing की स्थिति बनाने और Testing कवरेज को बढ़ाने के लिए एक व्यवस्थित तरीका प्रदान करता है।

उदाहरण के लिए: सॉफ्टवेयर का विकास जो गारंटी देता है कि केवल वास्तविक व्यक्तियों को ही कोविड19- टीकाकरण मिलता है।

नियम - 'केवल 60 वर्ष से अधिक आयु के किसी भी व्यक्ति या मधुमेह या उच्च रक्तचाप के इतिहास वाले 45 वर्ष से अधिक आयु के किसी भी व्यक्ति को टीकाकरण की अनुमति दी जानी चाहिए।

	आयु>=60	आयु>=45	डायबेटीज	उच्च रक्तचाप	अपेक्षित परिणाम
1	Y	-	-	-	अनुमति
2	N	Y	Y	Y	अनुमति
3	N	Y	Y	N	अनुमति
4	N	Y	N	Y	अनुमति
5	N	Y	N	N	अनुमति नहीं है
6	N	N	-	-	अनुमति नहीं है

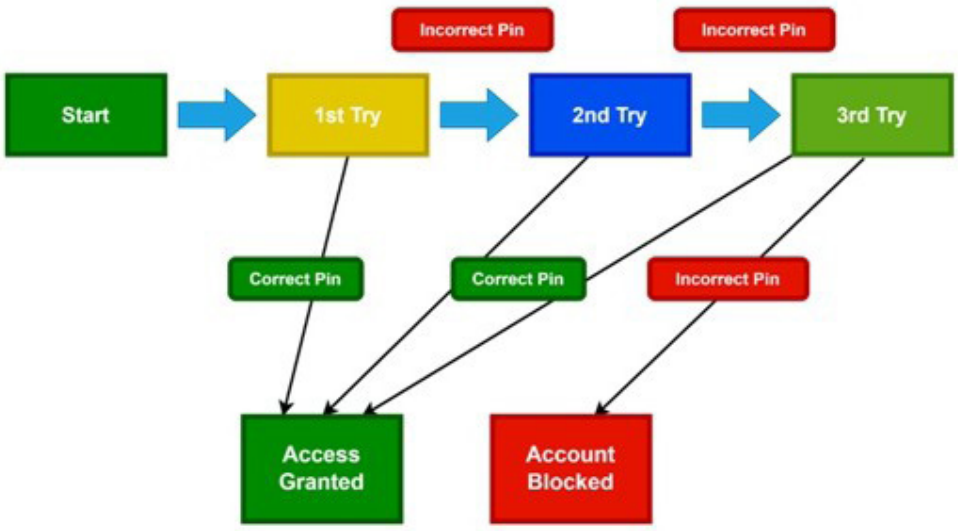
सारणी. 3.1.4. Test Case टेम्पलेट

2. State Transition Testing

इस पद्धति में, कार्यक्रम में States की एक सीमित संख्या अनुमानित रूप से मौजूद है। एप्लिकेशन अंडर टेस्ट (AUT) एक State या किसी अन्य में है, यह इस बात पर निर्भर करता है कि यह उपयोगकर्ता कार्यों का जवाब कैसे देता है। Tester इस पद्धति का उपयोग करके AUT के व्यवहार का Test कर सकता है, जो उन्हें क्रिया-आधारित इनपुट स्थितियों में प्रवेश करने में सक्षम बनाता है। सिस्टम व्यवहार का मूल्यांकन सकारात्मक और नकारात्मक इनपुट दोनों मानों का उपयोग करके किया जा सकता है।

उदाहरण के लिए: एक एटीएम सिस्टम फ़ंक्शन पर विचार करें जिसमें उपयोगकर्ता द्वारा तीन बार गलत पासवर्ड दर्ज करने पर खाता लॉक हो जाता है।

इस प्रणाली में, उपयोगकर्ता को सफलतापूर्वक साइन इन किया जाएगा यदि पहले तीन प्रयासों में से किसी में भी पासवर्ड सही ढंग से दर्ज किया गया है। यदि पासवर्ड पहली या दूसरी बार गलत दर्ज किया गया है, तो उपयोगकर्ता को इसे फिर से इनपुट करना होगा। तीन बार गलत पासवर्ड डालने पर यूजर का अकाउंट डिलीट हो जाता है।



चित्र 3.1.1 state transition Testing

इस आरेख में, जब भी उपयोगकर्ता सही पिन दर्ज करता है, तो उसे एक्सेस दी गई स्थिति में ले जाया जाता है। यदि वह गलत पासवर्ड दर्ज करता है तो वह अगले प्रयास पर जाता है।

इस आरेख में, जब भी उपयोगकर्ता सही पिन दर्ज करता है, तो उसे एक्सेस दी गई स्थिति में ले जाया जाता है। यदि वह गलत पासवर्ड दर्ज करता है तो वह अगले प्रयास पर जाता है।

Testing का अनुमान लगाने में त्रुटि

यह Strategy व्यक्तिगत अनुभव पर आधारित है। इसकी सफलता मुख्य रूप से Testing विश्लेषक की प्रवीणता के स्तर और सॉफ्टवेयर Testing की समझ के साथ-साथ इसकी कार्यक्षमता और व्यवहार पर आधारित है। Testing विश्लेषक की प्रवीणता और सॉफ्टवेयर Testing की समझ के साथ-साथ इसकी कार्यक्षमता और व्यवहार का स्तर, परियोजना की सफलता को बहुत प्रभावित करता है। Tester अपने अनुभव, किनारे के मामलों की समझ और कार्यक्रम प्रवाह में अपवाद परिदृश्यों के ज्ञान पर ड्राइंग करके संभावित मुद्दों की भविष्यवाणी करता है। विधि में संभावित त्रुटियों या प्रोग्राम अनुभागों की एक सूची शामिल है जो गलतियों से ग्रस्त हैं। Tester इन स्थितियों के लिए Test cases को विकसित करता है। नतीजतन, Tester पिछले अनुभव के आधार पर Test cases को विकसित करने के लिए इस पद्धति का उपयोग कर सकता है।

उदाहरण के लिए: कल्पना कीजिए कि एक सॉफ्टवेयर प्रोग्राम है और उस प्रोग्राम में एक आवश्यक फ़ील्ड है जहां उपयोगकर्ता को अपना मोबाइल नंबर दर्ज करना होगा और उस फ़ील्ड की कुछ सीमाएँ हैं। मजबूरियां यह हैं कि

1. मोबाइल संख्यात्मक होना चाहिए
2. यह 10 अंकों का होना चाहिए अब, यहाँ त्रुटि अनुमान लगाने की तकनीक का उपयोग आता है

इस तकनीक में, Tester कोने के मामलों का विश्लेषण करेगा जैसे:

1. यदि फ़ील्ड को खाली छोड़ दिया जाए तो परिणाम क्या होगा?
2. यदि 10 से कम अंक दर्ज किए जाते हैं तो परिणाम क्या होगा?
3. यदि कोई गैर-संख्यात्मक Characters दर्ज किया जाता है तो परिणाम क्या होगा?

3.1.15 एक Testing स्क्रिप्ट

Testing के तहत एक सिस्टम पर निर्देशों का एक सेट निष्पादित करना यह सुनिश्चित करने के लिए कि यह अपेक्षित रूप से प्रदर्शन करता है, Testing स्क्रिप्ट चलाने के रूप में जाना जाता है। मैन्युअल Testing के लिए, उन्हें मानव भाषा में लिखा जा सकता है, जबकि स्वचालित Test के लिए उन्हें स्क्रिप्टिंग या प्रोग्रामिंग भाषा में लिखा जाना आवश्यक है। एक Test Case में एक Testing स्क्रिप्ट या एकाधिक Testing स्क्रिप्ट हो सकती है, और एक Testing स्क्रिप्ट एक Test Case का एक घटक है। एकल Test Case से जुड़ी कई Testing स्क्रिप्ट हैं जब:

- Test Case में परिदृश्य को Testing स्क्रिप्ट के अनुसार विभिन्न तरीकों से Testing किया जा सकता है। उदाहरण के लिए, विभिन्न Testing वातावरणों में परिदृश्य का Test करने के लिए Test Case के लिए कई स्क्रिप्ट की आवश्यकता हो सकती है।

Test Case को चलाने के लिए मैनुअल और स्वचालित स्क्रिप्ट दोनों हैं।

एक Testing स्क्रिप्ट का एक उदाहरण नीचे दिया गया है।

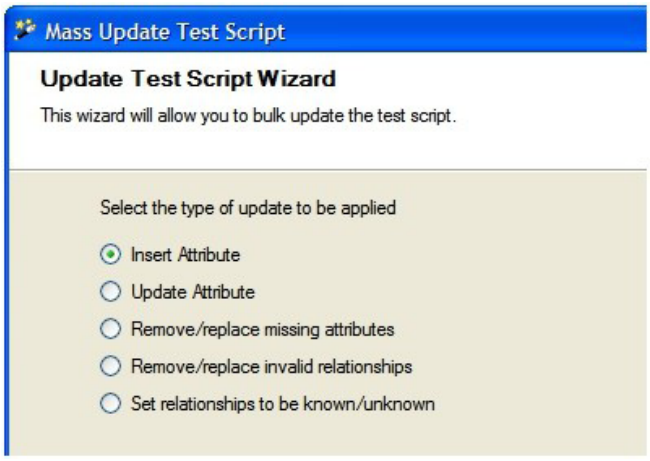
- उदाहरण के लिए, आपकी Testing स्क्रिप्ट किसी वेबसाइट की लॉगिन सुविधा का Test करने के लिए निम्न चला सकती है:
- वह स्थान सेट करें जहां स्वचालन उपकरण को लॉगिन स्क्रीन पर "उपयोगकर्ता नाम" और "पासवर्ड" फ़ील्ड देखना चाहिए। आइए मान लें कि हम उनके CSS एलिमेंट ID का उपयोग करेंगे।
- वेबसाइट के होम पेज पर "लॉगिन" बटन पर क्लिक करें। सत्यापित करें कि "उपयोगकर्ता नाम" और "पासवर्ड" कॉलम, साथ ही लॉगिन स्क्रीन, सभी स्पष्ट रूप से दिखाई दे रहे हैं।
- Next, पासवर्ड "123456" दर्ज करें और "चार्ल्स" लॉगिन करें, फिर "cofirm" बटन देखें और चुनें।
- उन्हें यह बताना होगा कि लॉग इन करने के बाद उपयोगकर्ता वेलकम स्क्रीन का शीर्षक कैसे ढूंढ सकता है, उदाहरण के लिए, इसकी CSS तत्व ID द्वारा।
- सुनिश्चित करें कि स्वागत स्क्रीन का शीर्षक दिखाई दे रहा है.
- स्वागत स्क्रीन का शीर्षक पढ़ें.
- शीर्षक Text में, "वेलकम चार्ल्स" टाइप करें।
- Testing सफल रहा यदि शीर्षक का शब्द प्रत्याशित से मेल खाता है। यदि नहीं, तो एल्बम Testing में विफल रहता है।

3.1.16 एक Testing स्क्रिप्ट संशोधित

नियम आधार में परिवर्तनों को ध्यान में रखते हुए, Test cases की अक्सर समीक्षा करने या उन्हें बदलने की आवश्यकता होती है. Test Case संपादक का उपयोग करके अलग-अलग Test cases को संशोधित किया जा सकता है, और Testing स्क्रिप्ट अपडेट Wizard का उपयोग एक बार में एकाधिक Testing स्क्रिप्ट और Test cases में परिवर्तन करने के लिए किया जा सकता है।

कई Testing स्क्रिप्ट और Test cases को एक साथ संशोधित करने के लिए: कई Testing स्क्रिप्ट और Test cases को एक साथ संशोधित करने के लिए:

1. Testing स्क्रिप्ट या Oracle नीति मॉडलिंग में Testing स्क्रिप्ट वाले फ़ोल्डर पर राइट-क्लिक करें और Testing स्क्रिप्ट Wizard Update करें चुनें। यह Mass Update Test Script प्रदर्शित करता है।



2. बाद के चार विकल्पों में से एक का चयन करें, जो नीचे अधिक विस्तार से विस्तृत हैं।

- Insert Attribute
- Update Attribute
- Remove/replace missing attributes
- Remove/replace invalid relationships
- Set relationships to be known/unknown

1. **Insert Attribute**

   यह विकल्प आपको एक Test Case Attributes के लिए एक मान सम्मिलित करने की अनुमति देता है जिसे अभी तक नहीं जोड़ा गया है। सामान्यतया, यह तब होता है जब पिछली बार Test cases Update किए गए थे के बाद नियम आधार के लिए एक नई Attributes प्रस्तुत किया गया था।

   कोई Attributes सम्मिलित करने के लिए:

   - Wizard की पहली स्क्रीन पर **Insert Attribute** विकल्प का चयन करें और Next क्लिक करें।
   - उन Test cases का चयन करें जिनमें Attributes जोड़ी जानी चाहिए। जोड़ी जाने वाली Attributes का चयन करने के लिए ब्राउज़ बटन का उपयोग करें, और वह मान दर्ज करें जिसे आप Attributes के लिए सम्मिलित करना चाहते हैं, यदि कोई हो। Next पर क्लिक करें.
   - परिवर्तनों का सारांश स्क्रीन **पर अपने परिवर्तनों की समीक्षा करें**. यदि आवश्यक हो तो अपने परिवर्तनों में संशोधन करने के लिए वापस क्लिक करें, फिर परिवर्तनों को लागू करने के लिए Next पर क्लिक करें।
   - Wizard द्वारा परिवर्तन लागू करने के बाद, कोई अन्य परिवर्तन करने के लिए हाँ विकल्प का चयन करें, अन्यथा नहीं विकल्प का चयन करें और समाप्त क्लिक करें.

2. **Update Attribute**

   यह विकल्प आपको उस Attributes के लिए मान को Update करने की अनुमति देता है जो आपके Test cases में पहले से मौजूद है।

   Attributes के लिए मान Update करने के लिए:

   - Wizard की पहली स्क्रीन पर Update Attributes विकल्प का चयन करें और Next क्लिक करें।
   - उन Test cases का चयन करें जिनमें Attributes जोड़ी जानी चाहिए। जोड़ी जाने वाली Attributes का चयन करने के लिए ब्राउज़ बटन का उपयोग करें, और वह नया मान दर्ज करें जिसे आप Attributes के लिए सेट करना चाहते हैं। Next पर क्लिक करें.
   - परिवर्तनों का सारांश स्क्रीन पर अपने परिवर्तनों की समीक्षा करें. यदि आवश्यक हो तो अपने परिवर्तनों में संशोधन करने के लिए वापस क्लिक करें, फिर परिवर्तनों को लागू करने के लिए Next पर क्लिक करें।
   - Wizard द्वारा परिवर्तन लागू करने के बाद, कोई अन्य परिवर्तन करने के लिए हाँ विकल्प का चयन करें, अन्यथा नहीं विकल्प का चयन करें और समाप्त क्लिक करें.

3. **अनुपलब्ध एट्रिब्यूट्स को हटाना/बदलना**

   यह विकल्प आपको एक Attributes को हटाने की अनुमति देता है जो अभी भी आपके Test cases में मौजूद है, लेकिन नियमबेस से हटा दिया गया है। वैकल्पिक रूप से, आप एक Attributes मान निर्दिष्ट कर सकते हैं जिसे इसे बदलना चाहिए।

अनुपलब्ध Attributes को हटाने या बदलने के लिए:

- Wizard की **पहली स्क्रीन पर अनुपलब्ध atributes** निकालें / बदलें विकल्प का चयन करें और Next क्लिक करें।
- Wizard यह पता लगाएगा कि क्या आपके Test cases में कोई Attributes मौजूद है जो अब नियमबेस में मौजूद नहीं हैं। उस Attributes का चयन करें जिसे आप त्रुटियों के साथ विशेषताएँ सूची से बदलना चाहते हैं। यदि आप अपने Test cases से केवल Attributes मान निकालना चाहते हैं, या इसे अनचेक करें और इसे बदलने के लिए एक Attributes का चयन करने के लिए ब्राउज़ बटन का उपयोग करें और नई Attributes के लिए मान दर्ज करें, तो केवल निकालें चेकबॉक्स को छोड़ दें।
- परिवर्तनों का सारांश स्क्रीन पर अपने परिवर्तनों की समीक्षा करें. यदि आवश्यक हो तो अपने परिवर्तनों में संशोधन करने के लिए वापस क्लिक करें, फिर परिवर्तनों को लागू करने के लिए Next पर क्लिक करें।
- Wizard द्वारा परिवर्तन लागू करने के बाद, कोई अन्य परिवर्तन करने के लिए हाँ विकल्प का चयन करें, अन्यथा नहीं विकल्प का चयन करें और समाप्त क्लिक करें.

4. **अमान्य संबंधों को हटाना/बदलना**

यह विकल्प आपको अपने Test cases में किसी भी संबंध को हटाने या बदलने की अनुमति देता है जो अब नियमबेस में मौजूद नहीं है।

अमान्य संबंधों को निकालने या बदलने के लिए:

- Wizard की पहली स्क्रीन पर अमान्य संबंधों को हटाने/बदलने का विकल्प चुनें और Next क्लिक करें।
- Wizard यह पता लगाएगा कि क्या आपके Test cases में कोई संबंध मौजूद है जो अब नियमबेस में मौजूद नहीं हैं। प्रत्येक अमान्य संबंध का पता लगाने के लिए, आप या तो हटाएँ चेकबॉक्स का चयन करके Test Case से इसे निकाल सकते हैं, या आप इसे बदलने के लिए ड्रॉप डाउन सूची से एक मान्य संबंध का चयन कर सकते हैं। एक बार जब आप प्रत्येक अमान्य संबंध के लिए ऐसा कर लेते हैं, तो Next क्लिक करें।
- परिवर्तनों का सारांश स्क्रीन पर अपने परिवर्तनों की समीक्षा करें. यदि आवश्यक हो तो अपने परिवर्तनों में संशोधन करने के लिए वापस क्लिक करें, फिर परिवर्तनों को लागू करने के लिए Next पर क्लिक करें।
- Wizard द्वारा परिवर्तन लागू करने के बाद, कोई अन्य परिवर्तन करने के लिए हाँ विकल्प का चयन करें, अन्यथा नहीं विकल्प का चयन करें और समाप्त क्लिक करें.

5. **संबंधों को known/unknown होने के लिए सेट करें**

यह विकल्प आपको संबंधों को ज्ञात या अज्ञात पर सेट करने की अनुमति देता है.

किसी संबंध की नई स्थिति सेट करने के लिए:

- Wizard की पहली स्क्रीन पर known/unknown होने के लिए संबंध सेट करें विकल्प का चयन करें और Next क्लिक करें।
- बाएँ pane में, उन Test cases का चयन करें जिन पर परिवर्तन लागू होना है (या सभी आइटम चेक करें चेकबॉक्स पर सही का निशान लगाकर सही का निशान लगाकर चुनें कि आप सभी Test cases को Update से प्रभावित करना चाहते हैं).
- दाएँ pane में, निकाय, संबंध और वर्तमान स्थिति का चयन करें. फिर नई संबंध स्थिति और प्रभावित आवृत्तियों का चयन करें.

4. Next पर क्लिक करें.
5. परिवर्तनों का सारांश स्क्रीन पर अपने परिवर्तनों की समीक्षा करें. यदि आवश्यक हो तो अपने परिवर्तनों में संशोधन करने के लिए वापस क्लिक करें, फिर **परिवर्तनों को लागू करने के लिए** Next पर क्लिक करें।
6. Wizard द्वारा परिवर्तन लागू करने के बाद, कोई अन्य परिवर्तन करने के लिए हाँ विकल्प का चयन करें, अन्यथा नहीं विकल्प का चयन करें और समाप्त क्लिक करें .

3.1.17 Testing नीति

"Testing नीति" के रूप में संदर्भित एक दस्तावेज़ जो Test गतिविधियों के लिए संगठनात्मक अंतर्दृष्टि प्रदान करता है, संगठन के स्तर पर वर्णित है।

यह संगठन के वरिष्ठ प्रबंधन द्वारा तय किया जाता है और उन मानकों को स्थापित करता है जिनका कंपनी को पालन करना चाहिए। Testing नीति एक बहुत ही महत्वपूर्ण दस्तावेज है जो Testing प्रलेखन के पदानुक्रम में सबसे पहले आता है।

इसे एक अलग दस्तावेज़ में प्रकाशित करने के अलावा, कुछ संगठन अपनी Testing नीति को एक वाक्य में प्रकाशित करना पसंद करते हैं। वे इस नीति को निर्माण और रखरखाव परियोजनाओं दोनों पर भी लागू कर सकते हैं।

Testing नीति निम्नलिखित का वर्णन करेगी:

- "संगठन के लिए Testing का क्या अर्थ है?" की पूछताछ के लिए एक विशिष्ट प्रतिक्रिया।
- Testing के उद्देश्य जो संगठन के पास हैं।
- विकसित सॉफ्टवेयर के कैलिबर को बढ़ाने के लिए संगठन की Testing प्रक्रिया का विवरण।
- उद्देश्यों को पूरा करते समय संगठन Testing की प्रभावकारिता और दक्षता का आकलन कैसे करेगा
- संगठन की अपनी Testing प्रक्रियाओं को बढ़ाने की योजना है।

3.1.18 Testing Strategy

कार्यक्रम स्तर पर एक Testing Strategy दस्तावेज बनाया जाता है और इसमें सॉफ्टवेयर पर विस्तृत Testing करने के लिए सामान्य Testing Strategy, प्रबंधन अवधारणाएं, प्रक्रियाएं और दृष्टिकोण शामिल होते हैं। Testing Strategy दस्तावेज़, जिसे कार्यक्रम स्तर पर विकसित किया गया है, में सॉफ्टवेयर पर किए जाने वाले विशिष्ट परीक्षणों के लिए समग्र Testing Strategy, प्रबंधन सिद्धांत, तरीके और कार्यप्रणाली शामिल हैं।

एक शीर्ष-स्तरीय संगठन में, Testing प्रबंधक और परियोजना प्रबंधक अक्सर Testing Strategy दस्तावेज़ लिखते हैं, जो एक उच्च-स्तरीय दस्तावेज़ भी है। बड़े पैमाने पर परियोजनाएं आमतौर पर इसे तैयार करती हैं, और इसे शायद ही कभी Update करने की आवश्यकता होती है। Testing Strategies और Testing दृष्टिकोणों को छोटी परियोजनाओं में Test योजना में शामिल किया जा सकता है, और Testing Strategy दस्तावेज भी अलग से नहीं लिखा जा सकता है।

Testing Strategy दस्तावेज़ का Test दृष्टिकोण और Testing प्रक्रियाएं संगठन की Testing नीतियों के अनुरूप होनी चाहिए:

- Testing का उद्देश्य /
- Testing के लिए इन-स्कोप /
- Testing स्तर (यूनिट, सिस्टम, एकीकरण, सिस्टम एकीकरण)
- टेस्ट प्रकार (फंक्शनल /
- Testing के लिए प्रवेश/निर्गमन/रोकना/बहाली मानदंड (विभिन्न स्तरों/चरणों के लिए)
- संबोधित किए जाने वाले जोखिम
- Testing वातावरण
- Test Case डिजाइन पद्धति
- Testing पद्धति (टॉप-डाउन/बॉटम-अप/जोखिम आधारित)
- Testing नियंत्रण और रिपोर्टिंग
- Testing स्वचालन दृष्टिकोण
- उपयोग किए जाने वाले Testing उपकरण
- डिफेक्ट प्रबंधन दृष्टिकोण
- डिफेक्ट वर्गीकरण
- पुन: Testing और प्रतिगमन दृष्टिकोण

3.1.19 Testing दिशानिर्देश

सॉफ़्टवेयर का Test करते समय, निम्नलिखित Testing मानकों का पालन किया जाना चाहिए:

- **सॉफ्टवेयर का Test कभी भी विकास टीम द्वारा नहीं किया जाना चाहिए; इसके बजाय, Testing टीम को इसका प्रभारी होना चाहिए।** सॉफ्टवेयर डेवलपमेंट टीम को कभी भी खुद ही इसका Test न करने दें। ऐसा इसलिए है, क्योंकि सॉफ़्टवेयर बनाने में कई घंटे लगाने के बाद, यह अनजाने में बहुत अधिक स्वामित्व में बढ़ सकता है, जिससे डिज़ाइन में किसी भी डिफेक्ट का पता लगाना मुश्किल हो सकता है।
- परीक्षकों को विनाशकारी दिमाग सेट के साथ उत्पाद से संपर्क करना चाहिए। डेवलपर्स यूनिट और एकीकरण Testing कर सकते हैं, लेकिन सॉफ्टवेयर Testing Testing टीम द्वारा नियंत्रित किया जाना चाहिए।**परीक्षण कभी भी यह साबित करने में सक्षम नहीं होगा कि सॉफ्टवेयर का एक टुकड़ा 100% बग-मुक्त है।** दूसरे शब्दों में, कई Test Case बनाने के बाद भी, यह प्रदर्शित करना असंभव है कि सॉफ्टवेयर त्रुटि मुक्त है।
- **जितनी जल्दी हो सके शुरू करें: आवश्यकता विश्लेषण प्रक्रिया और Testing प्रक्रिया हमेशा समवर्ती रूप से शुरू होनी चाहिए।** डिफेक्ट प्रवास के मुद्दे को रोकने के लिए, यह आवश्यक है। गुंजाइश और Testing वस्तुओं को जल्द से जल्द तय किया जाना चाहिए।
- **अनुभागों को प्राथमिकता दें: यदि कोई महत्वपूर्ण खंड हैं, तो यह सुनिश्चित करना महत्वपूर्ण है कि उनका जल्द से जल्द और अत्यंत तात्कालिकता के साथ Testing किया जाए।**
- **समय की एक सीमित मात्रा है**: सॉफ्टवेयर Testing समय सीमित है। Testing प्रक्रिया शुरू करने से पहले, एक ठोस Test योजना तैयार करना और ध्यान रखना महत्वपूर्ण है कि Testing का समय सीमित है। Testing प्रक्रिया को कब रोकना है, यह तय करने के लिए, कुछ मानदंड होने चाहिए। इस मानदंड को पहले से चुनना आवश्यक है। उदाहरण के लिए, समय Boundary या वित्तीय प्रतिबंधों के अनुसार, या जब सिस्टम को जोखिम के स्वीकार्य स्तर के साथ छोड़ दिया जाता है।
- **Testing अप्रत्याशित और नकारात्मक इनपुट के साथ किया जाना चाहिए:** यह सुनिश्चित करने के लिए कि प्रणाली लीक-प्रूफ है, Testing सही डेटा और Test cases और दोषपूर्ण Test cases दोनों का उपयोग करके किया जाना चाहिए। मूल्यांकन c होना चाहिए यह सटीक परिभाषाओं और पारित इनपुट और अपेक्षित आउटपुट के विवरण के साथ Test cases को शामिल करने की मांग करता है। Testing के दौरान, सटीक डेटा और Test cases और दोषपूर्ण Test cases दोनों का उपयोग किया जाना चाहिए। बाद के Testing चरणों में उनकी पुन: प्रयोज्यता सुनिश्चित करने के लिए Test cases को अच्छी तरह से प्रलेखित किया जाना चाहिए। यह आवश्यक है कि Test cases को परिभाषाओं और पारित इनपुट और अपेक्षित आउटपुट के विवरण के साथ गणना की जाए। सॉफ्टवेयर की फंक्शनल और गैर-फंक्शनल आवश्यकताओं दोनों को Testing के लिए रखा जाना चाहिए।
- **परीक्षा परिणामों का सही मूल्यांकन करना**: टेस्ट और उनके परिणामों का मात्रात्मक मूल्यांकन किया जाना चाहिए। उचित Testing सुनिश्चित करने के लिए, Test Case के परिणामों को मान्य करते समय प्रलेखन को ठीक से उद्त किया जाना चाहिए। जितना संभव हो, Testing को स्वचालित उपकरणों और विधियों द्वारा समर्थित किया जाना चाहिए। यह सुनिश्चित करने के अलावा कि सिस्टम अपने इच्छित कार्य करता है, परीक्षकों को यह जांचना चाहिए कि सिस्टम उन कार्यों को नहीं करता है जो इरादा नहीं हैं।
- **मान्यताओं को मान्य करना:** Test cases को हमेशा पर्याप्त रूप से मान्य किया जाना चाहिए और कभी भी मान्यताओं के आधार पर नहीं बनाया जाना चाहिए। मामलों को कभी भी अनुमानों या अनुमानों पर नहीं बनाया जाना चाहिए। उन्हें हमेशा उचित रूप से मान्य किया जाना चाहिए। उदाहरण के लिए, इस धारणा के तहत Test cases को डिजाइन करना कि सॉफ्टवेयर उत्पाद बग-मुक्त है, बहुत ही कमजोर Test Case प्रदान कर सकता है।

3.1.20 डिजाइन टेस्ट डेटा

Test Case सॉफ्टवेयर प्रोग्राम का मूल्यांकन करने के लिए Test डेटा का उपयोग करते हैं, जो उत्पादन जैसा डेटा है। जब Test cases और Testing स्क्रिप्ट निष्पादित किए जाते हैं, तो Test डेटा आमतौर पर Test डेटा दस्तावेज़ नामक दस्तावेज़ में एकत्र किया जाता है।

Test Case पूरी तरह से सभी संभावित परिदृश्यों को कवर नहीं कर सकते हैं यदि Test डेटा की पहले से योजना नहीं बनाई गई है, जो अंततः सॉफ़्टवेयर की गुणवत्ता से समझौता कर सकता है।

Test डेटा की दो श्रेणियां मान्यता प्राप्त हैं:

1. **वैध Test डेटा**: यह उस सकारात्मक डेटा को संदर्भित करता है जिसका उपयोग सिस्टम प्रत्याशित परिणामों का उत्पादन करने के लिए करता है। यह सुनिश्चित करने के लिए सभी संभावित इनपुट का संश्लेषण है कि एप्लिकेशन विनिर्देशों के अनुसार काम कर रहा है।
2. **गलत Test डेटा:** गलत Test डेटा नकारात्मक डेटा है जिसका उपयोग झूठी या नकारात्मक स्थितियों और अपवादों का Test करने के लिए किया जाता है। कुछ उदाहरण निम्नलिखित हैं:
3. **वैध Test डेटा:** यह उस सकारात्मक डेटा को संदर्भित करता है जिसका उपयोग सिस्टम प्रत्याशित परिणामों का उत्पादन करने के लिए करता है। यह सुनिश्चित करने के लिए सभी संभावित इनपुट का संश्लेषण है कि एप्लिकेशन विनिर्देशों के अनुसार काम कर रहा है।
4. **अविश्वसनीय Test डेटा:** प्रतिकूल Test डेटा का उपयोग करके अप्रत्याशित या प्रतिकूल परिस्थितियों और अपवादों का Test किया जाता है। कुछ उदाहरण निम्नलिखित हैं:
   - Null value - अनिवार्य क्षेत्रों के मामले में
   - Boundary से बाहर मान
   - विशेष Characters जिनकी अनुमति नहीं है
   - अमान्य डेटा प्रारूप - उदाहरण के लिए. mobile# with alphabet

वास्तविक और गलत डेटा का एक संयोजन जो सकारात्मक और नकारात्मक दोनों Testing परिदृश्यों को कवर करता है, अच्छा Test डेटा माना जाता है।

Test डेटा-राइटिंग

Testing शुरू होने से पहले और आमतौर पर Test cases को डिजाइन करते समय, नमूना डेटा बनाया जाना चाहिए। क्योंकि Test वातावरण को कॉन्फ़िगर करने से Test डेटा बनाते समय कई Test परिवेशों में कई चरण शामिल हो सकते हैं। इसके अतिरिक्त, यदि Testing निष्पादन के दौरान Test डेटा निर्माण किया जाता है, तो इससे Test धीमा चल सकता है।

सॉफ्टवेयर के लिए Test डेटा बनाने वाले लोग Tester हैं। व्यापार विश्लेषक विशिष्ट परिस्थितियों में Test डेटा के रूप में नकाबपोश उत्पादन डेटा की पेशकश कर सकते हैं, जैसे कि बैंकिंग या चिकित्सा ऍप्लिकेशन्स के साथ, जब डेटा अधिक संवेदनशील होता है।

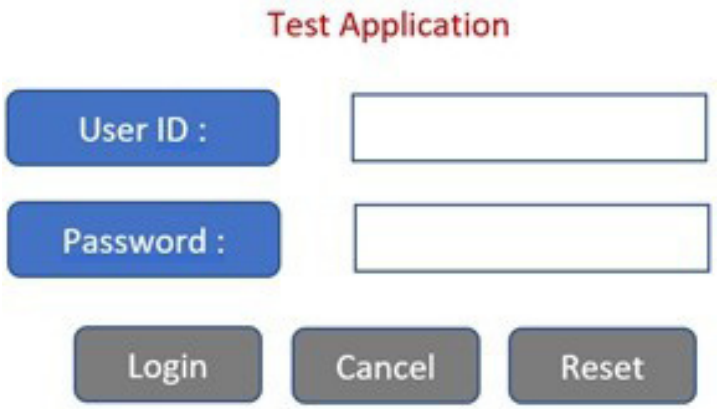
Test डेटा बनाया जा सकता है:

1. **Tester दो में से एक तरीके से Test डेटा जेनरेट कर सकते हैं**: मैन्युअल रूप से या स्वचालित रूप से. मैन्युअल रूप से Test डेटा उत्पन्न करने में अक्सर इसे एक्सेल स्प्रेडशीट में मैन्युअल रूप से बनाना शामिल होता है। Test डेटा बनाने के लिए, कोई वर्ड, टेक्स्ट और XML फाइलों का भी उपयोग कर सकता है।
2. **उत्पादन से कॉपी करें**: लोड, तनाव और प्रदर्शन के लिए Testing करते समय, आमतौर पर पर्याप्त मात्रा में डेटा आवश्यक होता है। स्वचालित डेटा उत्पादन उपकरणों का उपयोग करना, एक विन्यास योग्य उपयोगिता जो डेटा बनाती है, तालिकाएं उत्पन्न होती हैं (दृश्य, प्रक्रियाएं, आदि)। इस जानकारी का उपयोग प्रदर्शन Testing, डेटाबेस Testing, लोड Testing, फंक्शनल Testing और प्रयोज्य Testing के दौरान किया जाता है।

उदाहरण:

Test डेटा को समझने के लिए नीचे दिए गए Test एप्लीकेशन पर विचार करें। यहां यूजर ID और पासवर्ड की आवश्यकताएं दी गई हैं: **उपयोगकर्ता ID आवश्यकताएं:**

1. न्यूनतम लंबाई 3 Characters है, अधिकतम 15 है।
2. केवल विशेष Characters की अनुमति है '_'
3. उपयोगकर्ता ID अद्वितीय होनी चाहिए.



चित्र 3.1.2 Test एप्लीकेशन

पासवर्ड आवश्यकताएँ:

User ID	Password	Comments
John_12	Excellent@56$	Valid Input
Anna_Jose_Samul	passWORD#12345	Valid Higher Boundry Vlaues
Pet	Pst_98	Valid Lower Boundry Values
John_12	%XYZabc%	Invalid- Duplicate User ID
%^&*(uo	Password#123	Invalid- Special Characters in User ID
NULL	Pst_9890	Invalid- Blank User ID
Jippsy_45	tigress	Invalid- No Special charactors in password
Rose_merry9	68778#$	Invalid- No Alphabet

1. न्यूनतम लंबाई 5 Characters है, अधिकतम **15 है,**
2. कम से कम **1 अपर-केस** वर्णमाला,
3. कम से कम **1 लोअर-केस** अल्फाबेट,
4. न्यूनतम **एक संख्या,**
5. केवल विशेष Characters जिनकी अनुमति है: **! # $ % * ? @ &**

तो, उपरोक्त आवश्यकताओं के अधिकतम Testing कवरेज के लिए बनाया गया Test डेटा नीचे दिया गया है

Test Data Template				**Test Data**		**Optional**
Module Name	Test Scenario ID	Test Case ID	Test Case Name	User ID	Password	Comments
लॉगिन मॉड्यूल	TS_001	TC_001	सत्यापित करें कि उपयोगकर्ता सही यूजर ID और पासवर्ड के साथ लॉगिन करने में सक्षम है	John_12	Excellent@56$	मान्य
				Anna_Jose_Samul	पासवर्ड12345#	उच्च बाउंड्री व्लाउस
				Pet	Pst_98	कम Boundary मान
		TC_002	सत्यापित करें कि उपयोगकर्ता गलत के साथ लॉगिन करने में सक्षम नहीं है यूजर ID और सही पासवर्ड	John_12	%XYZabc%	डुप्लीकेट यूजर ID
				%^&*(uo	पासवर्ड123#	यूजर ID में विशेष Characters
				Null	Pst_9890	रिक्त उपयोगकर्ता परिचय
		TC_003	सत्यापित करें कि उपयोगकर्ता सही के साथ लॉगिन करने में सक्षम नहीं है यूजर ID और गलत पासवर्ड	Jippsy_45	tigress	अमान्य- में कोई विशेष Characters नहीं पासवर्ड
				Rose_merry9	68778#$	अमान्य- में कोई वर्णमाला नहीं पासवर्ड

तालिका 3.1.5 Test डेटा टेम्पलेट

परिणाम:

- डेटा का Test करने के लिए नियमित अपडेट किया जाना चाहिए।
- Test Case के निष्पादन से पहले, Test Case को डिजाइन करते समय, इसे बनाया जाना चाहिए।
- नकारात्मक परिदृश्यों का Test करने के लिए, Test डेटा में अमान्य इनपुट होना चाहिए।
- लोड या तनाव Testing के लिए बड़ी मात्रा में Test डेटा बनाने के लिए स्वचालन तकनीकों का उपयोग करें।
- Test डेटा बनाते समय डेवलपर / बीए की सहायता का लाभ उठाएं।
- अंत में, आमतौर पर Test डेटा को शामिल करने की सलाह दी जाती है जिसमें समर्थित और असमर्थित स्वरूपों का हर संभव संयोजन होता है। यह पूर्ण Testing कवरेज की गारंटी देता है।

सारांश

- Testing का प्राथमिक लक्ष्य कार्यान्वयन चरण से पहले सॉफ्टवेयर में सभी डिफेक्ट्स और त्रुटियों की पहचान करना है।
- सॉफ़्टवेयर Testing के दौरान, Testing परिदृश्य विस्तृत विवरण या लॉग होते हैं कि उपयोगकर्ता किसी एप्लिकेशन के साथ कैसे इंटरैक्ट करेगा। इसे Testing संभावना और Testing स्थिति के रूप में भी जाना जाता है ..
- एक Test case उन स्थितियों का एक समूह है जिसके तहत एक Tester यह तय करता है कि कोई सॉफ्टवेयर प्रोग्राम ग्राहक के विनिर्देशों को पूरा करता है या नहीं।
- विकास प्रक्रिया के दौरान, Testing किया जाता है। Testing करने के लिए प्रदर्शन मेट्रिक्स की योजना बनाना, डिजाइन करना और विकसित करना महत्वपूर्ण है।
- Test योजनाएं, Test Case और Testing रिपोर्ट जो अच्छी तरह से लिखी गई हैं, Testing गतिविधि का सटीक वर्णन और दस्तावेजीकरण करने के लक्ष्य को प्राप्त करने में Tester की सहायता करती हैं।
- परियोजना ब्रीफिंग में एक व्यापक Testing Strategy शामिल होनी चाहिए। आपका Test दस्तावेज़ पारदर्शी, संक्षिप्त और आपके शेड्यूल या परिवेश में परिवर्तनों के अनुकूल होना चाहिए।

एक्टिविटी

Con-vid Session

- इस सत्र में, ट्रेनर एक वीडियो चलाएगा।
- वीडियो टेस्ट प्लान की एक झलक देगा और कैसे लिखना है?
- वीडियो के लिए यू ट्यूब लिंक है : https:// www.youtube.com/ watch? v=S2_ AJP9Oeg0
- प्रशिक्षु पिन ड्रॉप साइलेंस के साथ वीडियो का अवलोकन करेंगे।
- वे वीडियो से पॉइंटर्स को नोट कर सकते हैं जो उन्हें प्रासंगिक लग सकते हैं।
- प्रशिक्षु कक्षा में शिष्टाचार बनाए रखेंगे और कक्षा में बात नहीं करेंगे, कानाफूसी नहीं करेंगे या चर्चा नहीं करेंगे।

किसी भी प्रश्न या भ्रम के मामले में, प्रशिक्षु अपनी नोटबुक में लिख लेंगे।

चर्चा सत्र

- प्रशिक्षक पूछताछ करेगा कि क्या प्रशिक्षुओं ने अध्याय को समझा है।
- इस गतिविधि में, प्रशिक्षक प्रशिक्षु से विषय से संबंधित कुछ प्रश्न पूछेगा।
- प्रशिक्षु विषय की अपनी समझ के अनुसार प्रश्नों का उत्तर देंगे।
- यदि प्रशिक्षुओं के मन में प्रश्न और भ्रम हैं; वे ट्रेनर के सामने उन लोगों को सामने रख सकते हैं।
- प्रशिक्षक यह सुनिश्चित करेगा कि वह प्रशिक्षुओं द्वारा रखे गए प्रश्नों के उचित उत्तर दे सके।

अभ्यास

1. बहुविकल्पीय प्रश्न:

1. निम्नलिखित में से किसे व्हाइट-बॉक्स Testing के रूप में भी जाना जाता है?
   a) अनुमान लगाने में त्रुटि तकनीक
   b) संरचनात्मक Testing
   c) डिजाइन आधारित Testing
   d) उपरोक्त में से कोई नहीं
2. Testing के विभिन्न स्तर क्या हैं?
   a) एकीकरण Testing
   b) यूनिट Testing
   c) सिस्टम Testing
   d) उपरोक्त सभी
3. निम्नलिखित में से कौन Test योजना में शामिल नहीं है?
   a) घटना की रिपोर्ट
   b) प्रवेश और निकास मानदंड
   c) अनुसूची
   d) जोखिम
4. Testing के निकास मानदंड को परिभाषित करने के लिए निम्नलिखित में से किस Testing दस्तावेज का उपयोग किया जाता है?
   a) Testing सारांश रिपोर्ट
   b) डिफेक्ट रिपोर्ट
   c) Test योजना
   d) Test Case
5. Test cases पर किसके द्वारा हस्ताक्षर किए जाते हैं?
   a) टीम लीड
   b) डेवलपर
   c) व्यापार विश्लेषक
   d) परियोजना प्रबंधक
6. Test योजना के लिए इनपुट क्या हैं
   a) अनुरोध दस्तावेज़
   b) Testing Strategy
   c) (a) और (b) दोनों
   d) उपरोक्त में से कोई नहीं
7. _____कोड की जांच के लिए Testing का उपयोग किया जाता है?
   a) ब्लैक बॉक्स Testing
   b) ग्रे बॉक्स Testing
   c) रेड बॉक्स Testing
   d) व्हाइट-बॉक्स Testing

B. निम्नलिखित प्रश्न का संक्षेप में उत्तर दीजिए:

a) निकास मानदंड का उद्देश्य क्या है?

b) Test Case क्या है?

c) Testing स्क्रिप्ट निर्धारित करें

नोट्स

संबंधित वीडियो देखने के लिए क्यूआर कोड को स्कैन करें

https://www.youtube.com/watch?v=RDBzZ-9XnDE

सॉफ्टवेयर उत्पादों/ऍप्लिकेशन्स/मॉड्यूल के लिए डिजाइन Testing

Skill India
कौशल भारत-कुशल भारत

सत्यमेव जयते
GOVERNMENT OF INDIA
MINISTRY OF SKILL DEVELOPMENT
& ENTREPRENEURSHIP

N·S·D·C
National
Skill Development
Corporation
Transforming the skill landscape

4. सॉफ्टवेयर उत्पादों/ ऍप्लिकेशन्स /मॉड्यूल पर स्वचालित Test करें

IT - ITeS SSC
nasscom

यूनिट 4.1 - सॉफ्टवेयर उत्पादों/ऍप्लिकेशन्स/मॉड्यूलों पर स्वचालित Test करना

SSC/N1302

प्रमुख सीखने के परिणाम

इस मॉड्यूल के अंत तक, प्रतिभागी निम्न में सक्षम होंगे:

1. किए जाने वाले Testing की प्रकृति और उपयोग किए जाने वाले Test प्रबंधन उपकरण की पहचान करने के ज्ञान का प्रदर्शन करें।
2. ऍप्लिकेशन्स पर फंक्शनल , प्रयोज्यता, संगतता, प्रदर्शन और प्रतिगमन Testing की प्रक्रिया की जांच करें।

यूनिट 4.1: सॉफ्टवेयर उत्पादों/ऍप्लिकेशन्स/मॉड्यूलों पर स्वचालित Test करना

यूनिट के उद्देश्य

इस यूनिट के अंत तक, प्रतिभागी निम्न में सक्षम होंगे:

1. Test cases और स्वचालित स्क्रिप्ट के नवीनतम संस्करणों की पहचान करने की प्रक्रिया पर चर्चा करें।
2. Testing के लिए आवश्यक एप्लीकेशन और डेटा स्रोतों के सही संस्करणों की पहचान करने के तरीके पर चर्चा करें।
3. कार्यक्षमता, प्रयोज्य और प्रतिगमन विधि के प्रमुख तत्वों का वर्णन करें।
4. चर्चा कीजिए कि प्रोग्रामिंग भाषाएँ जैसे Java, SQL आदि किस प्रकार साफ्टवेयर मॉड्यूल के विकास में सहायता करती हैं।
5. स्वचालित Test स्क्रिप्ट करने के लिए निर्देशों की व्याख्या करें।
6. फंक्शनल , प्रयोज्यता, संगतता, प्रदर्शन और प्रतिगमन Testing की प्रक्रिया का प्रदर्शन करें।
7. सॉफ्टवेयर मॉड्यूल विकसित करने के लिए प्रोग्रामिंग भाषा लागू करें, जैसे C, C ++, SQL, Java, आदि।

4.1.1 Test प्रबंधन उपकरण क्या हैं और आपको उनकी आवश्यकता क्यों है?

Test cases के विकास, परीक्षणों के निष्पादन और Testing परिणामों की रिकॉर्डिंग सहित दैनिक Test गतिविधियों के प्रबंधन की प्रक्रिया को Test प्रबंधन के रूप में जाना जाता है।

Test Case निर्माण, Testing निष्पादन, Testing परिणाम संग्रह, रिपोर्ट generation, Test Case निष्पादन के साथ समवर्ती डिफेक्ट ट्रैकिंग, और एक्सेल से Test प्रबंधन उपकरणों के लिए परीक्षणों का आयात / निर्यात सभी Test प्रबंधन उपकरणों द्वारा प्रबंधित किए जाते हैं।

4.1.2 सही Test प्रबंधन उपकरण चुनना

- **मूल्य निर्धारण:** Test प्रबंधन उपकरण चुनते समय, लाइसेंस लागत को ध्यान में रखा जाना चाहिए। लाइसेंस की लागत लगातार, समवर्ती या व्यक्तिगत उपयोगकर्ताओं की संख्या से निर्धारित की जा सकती है। कार्यान्वयन और प्रशिक्षण की लागत को अग्रिम लाइसेंस लागत के अतिरिक्त माना जाना चाहिए।
- **एकीकरण समर्थन:** Testing स्वचालन उपकरण और विभिन्न CI/CD उपकरण Test प्रबंधन उपकरणों के साथ अच्छी तरह से एकीकृत होने चाहिए। टूल में बग ट्रैकिंग टूल जैसे बगजिला, मैंटिस आदि के साथ एकीकृत करने की क्षमता होनी चाहिए।
- **विभिन्न विकास प्रक्रियाओं के लिए अनुकूलन:** कई टीमों ने विभिन्न संगठनों में काम करने के एजाइल और DevOps तरीकों को अपनाया है, इसलिए टूल को एजाइल, वाटरफॉल या दोनों के संयोजन के साथ मैप करने में सक्षम होना चाहिए।
- **एकाधिक उपयोगकर्ताओं का समर्थन करना:** जब कई उपयोगकर्ता Test cases के प्रबंधन के लिए टूल का उपयोग करते हैं, तो एक बात यह देखने के लिए है कि क्या एक विशेष Test Case को एक ही समय में दो उपयोगकर्ताओं द्वारा एक्सेस किया जाता है या नहीं।

- **STLC के सभी चरणों का उपयोग करना और सुविधा प्रदान करना:** Test प्रबंधन उपकरण को STLC के सभी चरणों को पूरा करना आसान बनाना चाहिए, जिसमें Test cases को बनाए रखना, डिफेक्ट्स को लॉग करना, बग को Test Case रन से जोड़ना, Testing निष्पादन और व्यक्तिगत रिपोर्ट तैयार करना शामिल है।

4.1.3 Test प्रबंधन उपकरण की आवश्यकता

Test प्रबंधन उपकरणों की आवश्यकता क्यों है, इसके कुछ कारण निम्नलिखित हैं:

- **Testing असाइन करने की विशेषताएं:** Test प्रबंधन उपकरण परीक्षणों के बारे में विवरण ट्रैक करता है और उपयोगकर्ता विशिष्ट लोगों को कार्य सौंप सकते हैं। यह "Test किसने चलाया?", "कौन से Test चलाए गए?", "Test किसने पास किए?" , "किन आवश्यकताओं को कवर किया गया है?" जैसे विवरणों को भी ट्रैक करता है।
- **Test गतिविधियों का प्रबंधन और ट्रैकिंग :** STLC चरण के दौरान पाए जाने वाले परीक्षणों और डिफेक्ट्स के प्रबंधन और ट्रैकिंग के लिए एक Test प्रबंधन उपकरण उपयोगी है।
- **जटिलता और दोहराव को हटा दें:** डेटा के दोहराव से बचने के लिए और एंड-एंड-एंड परिप्रेक्ष्य से किसी एप्लिकेशन के Testing की जटिलता को प्रबंधित करने के लिए, किसी भी संगठन के लिए एक Test प्रबंधन उपकरण होना चाहिए।
- **परीक्षणों के लिए पूर्ण दृश्यता और पता लगाने की क्षमता:** Testing Lifeचक्र के प्रत्येक चरण का पता एक Test प्रबंधन उपकरण के भीतर लगाया जा सकता है।

4.1.4 Test प्रबंधन उपकरण के लाभ

- प्रयोग करने में आसान और सरल UI।
- विभिन्न परियोजनाओं के लिए विशिष्ट अनुमतियों के साथ कई उपयोगकर्ताओं तक पहुंच की अनुमति दें।
- पूर्ण दृश्यता और पता लगाने की क्षमता
- Testing स्वचालन उपकरण और CI/CD उपकरण जैसे jenkins, bamboo, आदि के लिए एकीकरण।
- टूल मैपिंग को एजाइल और वाटरफॉल पद्धति की अनुमति देना।
- परियोजना की समग्र प्रगति और मैट्रिक्स की निगरानी करना
- Test गतिविधियों के सभी चरणों के लिए वन-स्टॉप शॉप।

4.1.5 प्रबंधन उपकरण जो व्यापक रूप से उपयोग किए जाते हैं

1. **इन्फ्लेक्ट्रा द्वारा स्पाइराटेस्ट**

   आज बाजार पर सबसे प्रभावी एंड-टू-एंड Test प्रबंधन उपकरण, स्पाइराटेस्ट उपयोगकर्ताओं को एक ही स्थान पर ग्राफिकल रिपोर्टिंग के साथ Test cases, आवश्यकताओं और डिफेक्ट्स का प्रबंधन करने में सक्षम बनाता है।

   सुविधाऐं:

- उपयोगकर्ता पूरी तरह से अनुकूलन रिपोर्ट का उपयोग करके परीक्षणों की स्थिति और समग्र परियोजना की स्थिति की जांच कर सकते हैं।
- सभी Testing-मैनुअल और स्वचालित-उपयोगकर्ताओं द्वारा एक ही स्थान से आसानी से बनाए और चलाए जा सकते हैं।

- डिफेक्ट्स की निर्बाध रिपोर्टिंग और ट्रैकिंग।
- बेहतर पता लगाने की क्षमता के लिए, उपयोगकर्ता डिफेक्ट्स को Testing निष्पादन से जोड़ सकते हैं।
- यह CI और CD टूल, J I R A, और Azure Dev Ops टूल के साथ-साथ Selenium जैसे स्वचालित Test टूल के साथ सहज एकीकरण को सक्षम बनाता है।
- या तो एक SaaS/ क्लाउड सेवा या ऑन-प्रिमाइसेस समाधान सुलभ है।

2. **टेस्ट रेल**

उपयोगकर्ता टेस्ट रेल नामक वेब-आधारित GUI इंटरफ़ेस का उपयोग करके Test योजनाएं, Test Case बना सकते हैं और Testing निष्पादन का प्रबंधन कर सकते हैं। सॉफ्टवेयर Testing प्रक्रिया के हर चरण को नियंत्रित और मॉनिटर करें।

सुविधाऐं:

- Test सूट, पिछले Test परिणामों और Test Case के इतिहास को ट्रैक करने की अनुमति देता है।
- Test निष्पादन परिणामों को वेब-आधारित इंटरफ़ेस में रिकॉर्ड करें।
- परिणामों को प्रभावी ढंग से समूह, क्रमबद्ध और फ़िल्टर करें।
- आपके परीक्षणों के लिए संग्रह और ऑडिटिंग विकल्प.
- मोबाइल उपकरणों, ब्राउज़रों और कई OS पर परीक्षणों को ट्रैक करता है
- Test योजनाओं और Test रन के लिए आसानी से अनुकूलन रिपोर्ट और मेट्रिक्स।
- Jira Cloud के साथ सहज एकीकरण, स्वचालन Testing उपकरण जैसे Ranorex, BitBucket, Frogbugz, Mantis, Git Hub, और Bit Bucket।
- टिप्पणियों, प्रतिक्रिया पर प्रभावी ढंग से सहयोग करने के लिए टीमें।
- नए असाइन किए गए परीक्षणों के लिए वैयक्तिकृत टू-डू सूचियाँ और ईमेल सूचनाएँ।

3. **X-Ray – JIRA के लिए टेस्ट प्रबंधन**

JIRA के लिए एक क्लाउड ऐप है जो एटलसियन मार्केटप्लेस पर उपलब्ध सभी Test गतिविधियों के लिए JIRA इश्यू का उपयोग करता है।

सुविधाऐं:

- JIRA के लिए एक्सरे स्थापित करते समय, आपको टेस्ट प्लान, टेस्ट सेट, टेस्ट रन और टेस्ट निष्पादन के लिए नए मुद्दे देखने को मिलते हैं।
- आवश्यकता पता लगाने की क्षमता के लिए सुविधा जोड़ें जो आपको आवश्यकताओं, परीक्षणों, डिफेक्ट्स और निष्पादन को ट्रैक करने देती है।
- Selenium, cucumber, JUnit और NUnit के साथ सहज एकीकरण।
- CI/CD पाइपलाइनों के साथ REST API का एकीकरण।
- आवश्यकताओं, डिफेक्ट्स, परीक्षणों और निष्पादन के लिए उन्नत कवरेज।
- अनुकूलन योग्य XRay समग्र Testing प्रगति के साथ Testing निष्पादन और Testing सेट पर नज़र रखने की रिपोर्ट।
- बेहतर दृश्यता और पारदर्शिता के लिए Jira Agile बोर्डों का उपयोग करके टीमों के काम को ट्रैक करने के लिए जोड़ा गया सुविधा।

4. Zephyr स्केल

पूरे Testing Life Cycle के अंदर सर्वश्रेष्ठ Test प्रबंधन उपकरण JIRA द्वारा प्रबंधित किया जाएगा। इसका उपयोग करना आसान और आत्म-व्याख्यात्मक है।

सुविधाऐं:

- हमें एक टीम की जरूरतों के अनुसार कॉन्फ़िगर करने की अनुमति देता है। उदाहरण के लिए, आप J ira व्यवस्थापक की आवश्यकता के बिना प्रोजेक्ट स्तर पर कस्टम फ़ील्ड और स्थिति जोड़ सकते हैं।
- कुछ उपयोगकर्ताओं को किसी एप्लीकेशन के भीतर विभिन्न फ़ंक्शंस जैसे Test Case, Test cycles, निष्पादन और रिपोर्ट तक पहुँच की अनुमति देने के लिए अनुमतियाँ सेट करें.
- अपने परीक्षणों को स्वचालन Test, Smoke Test और प्रदर्शन Test जैसी आवश्यकताओं के आधार पर सबफ़ोल्डर्स और फ़ोल्डरों में समूहीकृत करें.
- प्रदर्शन समस्याओं से बचने के लिए विभिन्न प्रोजेक्ट, मॉड्यूल और Testing लाइब्रेरीज़ में परीक्षणों का पुन: उपयोग करें।
- एक ही समय में Test cases के कई संस्करणों को प्रबंधित करने से हम संस्करणों के बीच परिवर्तनों की तुलना कर सकते हैं।
- J ira Issue view आपको Testing निष्पादन, Test Case, परिणाम देखने की अनुमति देता है।
- Zephyr Scale रिपोर्ट को महाकाव्यों, कहानियों और संस्करणों द्वारा फ़िल्टर किया जा सकता है जो आपको किसी विशिष्ट कार्यक्षेत्र या संस्करण के लिए पूर्ण Testing प्रगति देखने की अनुमति देता है।
- cucumber BDD, CI/CD, और स्वचालन उपकरणों को एकीकृत करके अपनी Test गतिविधियों को बढ़ाएं।

5. Zephyr squad

एक लचीला और निर्बाध रूप से एकीकृत Test प्रबंधन उपकरण रखने के लिए जो एक देशी JIRA एप्लिकेशन के रूप में कार्य करता है, फुर्तीली टीमों को Zephyr Squad का उपयोग करना चाहिए। Zephyr Squad का उपयोग करके कोई Test Case बना सकता है, Testing चला सकता है और Testing निष्पादन रिपोर्ट देख सकता है।

सुविधाऐं:

- JIRA के साथ निर्बाध एकीकरण।
- उपयोग में आसान, देखो और महसूस करना फुर्तीली टीमों के लिए एक बेहतर विकल्प बनाता है।
- कहानी दृश्य द्वारा Testing निष्पादन और परिणाम देखें।
- JIRA डैशबोर्ड आपको Test cycles और परीक्षकों द्वारा Testing निष्पादन देखने की अनुमति देता है।
- Selenium और CI/CD टूल जैसे jenkins और bamboo जैसे ऑटोमेशन टेस्टिंग टूल के साथ एकीकृत करें।

6. **प्रैक्टिटेस्ट**

Test cases को शुरू से अंत तक प्रबंधित करने के लिए यह उपकरण सास-आधारित है। उपयोगकर्ता कुशलता से Testing बना सकते हैं, उन्हें चला सकते हैं, बग का ट्रैक रख सकते हैं और इसका उपयोग करके रिपोर्ट तैयार कर सकते हैं।

सुविधाऐं:

- यह मुद्दों, परीक्षणों, चरणों और आवश्यकताओं को आयात और निर्यात करने की क्षमता प्रदान करता है।
- उपयोगकर्ताओं को 3 पार्टी टूल के साथ एकीकृत किए बिना मैनुअल, खोजपूर्ण और स्वचालन Testing करने का विकल्प देता है।
- प्रोजेक्ट के लिए प्रासंगिक एक कस्टम फ़ील्ड बनाएं जिसका उपयोग विभिन्न Testing विज्ञापन समस्याओं के साथ किया जा सकता है।
- यह क्रोम, फायर फॉक्स, IEE, एज और सफारी जैसे कई ब्राउज़रों का समर्थन करता है।
- मैनुअल और स्वचालन Testing परिणामों के लिए बेहतर दृश्यता कई रिपोर्टिंग विकल्पों की पेशकश करके समर्थित है।
- JIRA, Pivotal, Azure Dev Ops, Jenkins, Git Hub, Bugzilla और Slack के साथ सहज एकीकरण प्रदान करता है।
- jenkins और bamboo जैसे CI/CD टूल और उनके API के साथ सहज एकीकरण आपके स्वयं के कस्टम एकीकरण को भी जोड़ने की सुविधा देता है।
- किसी भी API कोड का उपयोग किए बिना XML Testing के परिणामों को प्रैक्टिटेस्ट में आयात करने के लिए फायर क्रैकर टूल के साथ एकीकरण।

7. **टेस्ट लिंक**

उपयोगकर्ता Testing लिंक, एक वेब-आधारित ओपन-सोर्स Test प्रबंधन उपकरण की मदद से Test cases, Testing सूट, Testing परियोजनाओं और उपयोगकर्ता प्रबंधन का प्रबंधन कर सकते हैं।

सुविधाऐं:

- मैनुअल और स्वचालित Test निष्पादन दोनों का समर्थन करता है।
- एकाधिक उपयोगकर्ता अपने क्रेडेंशियल्स और असाइन की गई भूमिकाओं के साथ टूल की कार्यक्षमता तक पहुंच सकते हैं।
- Word, Excel और HTML स्वरूपों जैसे विभिन्न स्वरूपों में Testing निष्पादन रिपोर्ट का निर्माण।
- Test cases का आसान आयात/निर्यात।
- Bugzilla, Mantis के साथ सहज एकीकरण।
- Test cases को डिफेक्ट्स से जोड़ना।
- Test case ID, संस्करण के आधार पर Test cases को फ़िल्टर और सॉर्ट करें.

8. **Q टेस्ट**

क्यू टेस्ट नामक एक Test प्रबंधन उपकरण QA सिम्फनी द्वारा बनाया गया था और यह एजाइल विकास के अनुकूल है। परियोजना की आवश्यकताओं को जोड़ा जा सकता है, Test Case बनाए जा सकते हैं, Testing चलाए जा सकते हैं, और Testing के परिणाम संग्रहीत किए जा सकते हैं।

सुविधाऐं:

- आसान उपयोगकर्ता इंटरफ़ेस उपयोगकर्ताओं को Test गतिविधियों को ट्रैक करने में सक्षम बनाता है।
- Bugzilla या JIRA के साथ QTest का एकीकृत।
- एक्सेल स्प्रेडशीट से Test cases का आसान आयात /
- दिनांक या फ़ील्ड पर फ़िल्टर और सॉर्ट विकल्पों का उपयोग करके आपके लिए उपयोगी डेटा प्रदर्शित करने के लिए अनुकूलन योग्य रिपोर्ट्स.
- Test cases और आवश्यकताओं में परिवर्तन ट्रैक करें.
- कई रिलीज में Test cases और Testing सूट का पुन: उपयोग करें।

9. QMetry Test प्रबंधन

शीर्ष Test प्रबंधन उपकरणों में से एक QMetry है। शीर्ष Test प्रबंधन उपकरणों में से एक QMetry है। JIRA के साथ-साथ jenkins, bamboo और ऑटोमेशन फ्रेमवर्क जैसे CI / ट्रैकिंग आवश्यकताएं, Test cases का प्रबंधन, Testing चलाना, रिपोर्टिंग करना, उपयोगकर्ताओं को प्रबंधित करना और समस्याओं का प्रबंधन करना इसकी कुछ प्रमुख विशेषताएं हैं।

Attributes में शामिल:

- Testing निष्पादन और एंड-टू-एंड Testing का नियंत्रण।
- स्वचालन और CI/CD उपकरणों का एकीकरण।
- उत्पाद की गुणवत्ता बढ़ाने में DevOps और Agile टीमों की सहायता करता है।

Kualitee

इसका उद्देश्य एक ही मंच पर सभी Test गतिविधियों का समन्वय करना है। आप Test चक्र चला सकते हैं, Test Case रिपॉजिटरी प्रबंधित कर सकते हैं और त्रुटियों को रिकॉर्ड कर सकते हैं।

सुविधाऐं:

- एकाधिक लोग एक ही आइटम पर कार्य कर सकते हैं और सभी प्रतिक्रियाएँ एक ही स्थान पर साझा की जाती हैं. सरल और उपयोग में आसान यूजर इंटरफेस। Test Case प्रबंधन और समस्या ट्रैकिंग के लिए वन-स्टॉप शॉप।
- उपयोगकर्ताओं के लिए परियोजनाएं, मॉड्यूल, Test Case, Test cycles बनाना, Test Case चलाना, त्रुटियों को रिकॉर्ड करना और रिपोर्ट तैयार करना आसान बनाता है।
- उपयोगकर्ताओं के लिए परियोजनाएं, मॉड्यूल, Test Case, Test cycles बनाना, Test Case चलाना, त्रुटियों को रिकॉर्ड करना और रिपोर्ट तैयार करना आसान बनाता है।
- विभिन्न फ़ाइल प्रकारों (एक्सेल, वर्ड, सीएसवी) में Test cases का सरल आयात /

4.1.6 Testing Life Cycle

- **पूर्वापेक्षा:** पूर्व शर्तें जो Testing निष्पादित करने से पहले प्राप्त की जानी चाहिए।
- **योजना:** Testing का दायरा, पर्यावरण जिसके तहत Testing करने की आवश्यकता है, Testing चरण और उपयोग की जाने वाली पद्धतियां, मैनुअल और स्वचालन Testing, बग प्रबंधन, कॉन्फ़िगरेशन प्रबंधन, देयता प्रबंधन, मूल्यांकन और पहचान - Testing उपकरण, Testing शेड्यूलिंग, संसाधन साझाकरण।
- **डिजाइनिंग:** टेस्ट परिदृश्य पहचान, टेस्ट कवरेज और ट्रेसेबिलिटी मैट्रिक्स और टेस्ट स्क्रिप्ट तैयारी, Test Case तैयारी और टेस्ट डेटा, Test Case समीक्षा और अनुमोदन, कॉन्फ़िगरेशन प्रबंधन के तहत बेस लाइनिंग।
- **टेस्ट पर्यावरण सेटअप:** नेटवर्क कनेक्टिविटी, टेस्ट बेड इंस्टॉलेशन और कॉन्फ़िगरेशन, सभी उपकरण या सॉफ़्टवेयर इंस्टॉलेशन और कॉन्फ़िगरेशन, व्यापारियों और अन्य लोगों के साथ समन्वय।
- **स्वचालन:** स्वचालन उपकरण पहचान और मूल्यांकन, विश्लेषण और डिजाइनिंग फ्रेमवर्क और स्क्रिप्टिंग, स्क्रिप्ट आत्मसात, लेखा परीक्षा और अनुमोदन, विन्यास प्रबंधन के तहत दिशानिर्देश।
- **निष्पादन और बग ट्रैकिंग:** Test Case निष्पादन, टेस्ट स्क्रिप्ट Testing, अधिग्रहण, ऑडिट और Testing परिणामों का मूल्यांकन, बग उठाएं और इसकी समाप्ति के लिए खोजें।
- **स्वीकृति और रिपोर्ट generation**: टेस्ट मेट्रिक्स, टेस्ट रिपोर्ट और प्रक्रिया एन्हांसमेंट किया गया, रिलीज का निर्माण करें, स्वीकृति प्राप्त करें।

4.1.7 जेनेरिक सॉफ्टवेयर Testing शर्तें

- **सॉफ्टवेयर Testing:** उत्पाद की वास्तविक deployment से पहले सॉफ्टवेयर में सभी त्रुटियों और बगों को खोजने के लिए सॉफ्टवेयर Testing एक आवश्यक गतिविधि है।
- **सत्यापन:** सॉफ्टवेयर दस्तावेजों, कोड, डिजाइन और प्रोग्राम की जांच। इसमें Testing निष्पादन शामिल नहीं है। यह ऑडिटिंग, निरीक्षण, वॉक-थ्रू आदि जैसी विधियों का उपयोग करता है।
- **सत्यापन:** यह वास्तविक उत्पाद को मान्य और Testing करने की गतिशील विधि है। इसमें Exection शामिल है। यह सफेद बॉक्स, ब्लैक बॉक्स आदि जैसे तरीकों का उपयोग करता है।
- **गुणवत्ता आश्वासन (QA):** यह सुनिश्चित करने के लिए गतिविधियों का एक समूह कि रखरखाव या / और विकास प्रक्रिया यह सुनिश्चित करने के लिए पर्याप्त है कि सिस्टम अपने उद्देश्यों को पूरा करता है। पर्याप्त विश्वास प्रदान करने के लिए आवश्यक गतिविधियों का एक मानकीकृत और योजनाबद्ध सेट कि आवश्यकताएं ठीक से स्थापित हैं और सेवाएं या उत्पाद निर्दिष्ट आवश्यकताओं के अनुरूप हैं। इसमें Exection शामिल नहीं है।
- **गुणवत्ता नियंत्रण (QC):** वह प्रक्रिया जिसके द्वारा उत्पाद की गुणवत्ता को लागू मानकों के साथ सहसंबद्ध किया जाता है और गैर-एकरूपता का पता चलने पर कार्रवाई की जाती है। इसमें हमेशा Exection शामिल होती है।

4.1.8 Testing के प्रकार

Testing के तीन व्यापक प्रकार हैं:

- **फंक्शनल Testing:** Testing का प्रकार जो यह जांचता है कि सॉफ्टवेयर एप्लिकेशन का प्रत्येक कार्य आवश्यकता विनिर्देश के अनुसार काम करता है, फंक्शनल Testing के रूप में जाना जाता है। संक्षेप में, इसमें उपयोगकर्ता स्वीकृति Testing, यूनिट Testing और एकीकरण Testing शामिल हैं। कुछ इनपुट (वैध और अमान्य दोनों) प्रदान करके और उत्पादित संबंधित आउटपुट को देखकर, सिस्टम की कार्यक्षमता का Test किया जाता है। इस तरह का Test मैन्युअल रूप से करना आसान है।
- **गैर-फंक्शनल Testing:** गैर-फंक्शनल Testing Testing के प्रकार को संदर्भित करता है जो सॉफ्टवेयर एप्लीकेशन के गैर-फंक्शनल पहलुओं की जांच करता है, जैसे कि इसकी मापनीयता, प्रयोज्यता, धीरज, आदि। फंक्शनल और गैर-फंक्शनल Testing दोनों आवश्यक हैं। संक्षेप में, इसमें प्रयोज्य, प्रदर्शन और लोड Testing शामिल हैं। इस प्रकार का Test मैन्युअल रूप से किया जाना थोड़ा मुश्किल है। रखरखाव Testing प्रक्रिया सॉफ्टवेयर एप्लिकेशन को deploy किए जाने के बाद और किसी भी परिवर्तन या सुधार के बाद की जाती है। संक्षेप में, इसमें रखरखाव और प्रतिगमन Testing दोनों शामिल हैं।

4.1.9 मैनुअल बनाम स्वचालित Test

मानव द्वारा मैन्युअल रूप से किसी एप्लिकेशन का Test करना मैन्युअल Testing के रूप में जाना जाता है। एक गुणवत्ता आश्वासन विशेषज्ञ (Tester) सत्यापित करता है कि लिखित परीक्षा मामलों को चलाकर एक एप्लीकेशन सही ढंग से कार्य करता है। किसी एप्लिकेशन के विभिन्न घटकों पर क्लिक करके, Tester एप्लिकेशन के प्रदर्शन, डिज़ाइन और कार्यक्षमता का मूल्यांकन करता है। मैन्युअल Testing तब उपयोगी होता है जब स्वचालन Testing एक विकल्प नहीं होता है। विशिष्ट प्रकार के Testing में खोजपूर्ण Testing, प्रयोज्य Testing, तदर्थ Testing आदि शामिल हैं। Testing जो पूर्व-स्क्रिप्टेड और स्वचालित रूप से निष्पादित होता है, उसे स्वचालित Test के रूप में जाना जाता है। प्रत्याशित परिणामों की तुलना करके Testing परिणामों को सत्यापित करने के लिए Testing चलाए जाते हैं। स्वचालित Test की आवश्यकता तब होती है जब एक ही Testing को विभिन्न संदर्भों में कई बार निष्पादित करने की आवश्यकता होती है।

उदाहरण के लिए: स्वीकृति Testing, यूनिट Testing, एकीकरण Testing और सिस्टम परीक्षण। नीचे दी गई तालिका स्वचालित और मैन्युअल Testing प्रक्रिया के बीच तुलनात्मक अध्ययन प्रस्तुत करती है।

स्वचालित Test	मैनुअल Testing
स्वचालित Test अधिक विश्वसनीय है। यह मानव त्रुटि को कम करते हुए, समय की एक ही Testing N number करता है।	मैनुअल Testing कम विश्वसनीय है। मानवीय त्रुटियों के कारण, मैनुअल Testing हर बार सटीक नहीं हो सकता है।
स्वचालित Test की लागत Testing के लिए उपयोग किए जाने वाले Testing उपकरण पर निर्भर करती है	मैनुअल Testing करने के लिए आवश्यक श्रम की मात्रा इसकी लागत निर्धारित करता है।
स्क्रिप्ट के कारण स्वचालित Test तेज और कम है समय लेने वाला	मानव हस्तक्षेप के कारण मैनुअल Testing धीमा समय लेने वाला है और
स्वचालित Test का उपयोग तब किया जाता है जब Test cases बार-बार निष्पादित किया	मैनुअल Testing का उपयोग तब किया जाता है जब दो या तीन Test Case निष्पादित किया जाना होते हैं
स्वचालित Test एप्लीकेशन उपयोगकर्ता की मित्रता सुनिश्चित करता है	मैनुअल Testing applicationउपयोगकर्ता की मित्रता सुनिश्चित नहीं करता है
स्वचालित Test समानांतर में निष्पादित किया जा सकता है।	मैनुअल Testing समानांतर में भी किया जा सकता है लेकिन यह Testing लागत का प्रचार करें
प्रोग्रामिंग ज्ञान स्वचालित परीक्षण में जरूरी है	प्रोग्रामिंग ज्ञान स्वचालित परीक्षण में आवश्यक नहीं है

तालिका 4.1.2 मैनुअल और स्वचालित Test के बीच तुलना

4.1.10 Test cases और स्वचालित Scripts के लिए सॉफ़्टवेयर Testing उपकरण के नवीनतम संस्करण

यद्यपि कई Testing स्वचालन उपकरण उपलब्ध हैं, उनमें से हर एक आपकी परियोजना के लिए आदर्श नहीं होगा। विभिन्न स्तरों के Testing इंजीनियर विभिन्न प्लेटफार्मों के लिए विभिन्न भाषाओं में बनाए गए उत्पादों का Test करने के लिए इस सूची के उपकरणों का उपयोग कर सकते हैं।

Tool	Description
Selenium	Selenium एक Testing उपकरण है, जिसका उपयोग वेब ब्राउज़र पर किए जाने वाले परीक्षणों को स्वचालित करने के लिए किया जाता है। इसे कई ब्राउज़रों में निष्पादित किया जा सकता है। यह कई प्रोग्रामिंग भाषाओं के अनुकूल है।
Cucumber	cucumber एक ओपन सोर्स टूल है जो बिहेवियर ड्रिवेन डेवलपमेंट (BDD) का समर्थन करता है। इसे Testing ढांचे के रूप में परिभाषित किया जा सकता है, जो सादे अंग्रेजी भाषा (गेरकिन) का उपयोग करता है यह कई प्लेटफॉर्म संगत है, उदाहरण के लिए- Ruby ऑन रेल्स, Selenium, पिकोकंटेनर, आदि।
Ranorex	Ranorex मोबाइल, वेब और डेस्कटॉप Testing के लिए एक उपकरण में सभी है। यह शुरुआती लोगों के लिए एक आसान क्लिक और गो इंटरफ़ेस और स्वचालन विशेषज्ञों के लिए एक शक्तिशाली IDE प्रदान करता है। यह एक लाइसेंस प्राप्त सॉफ्टवेयर है।
Testsigma	टेस्टसिग्मा बाजार में उपलब्ध सर्वोत्तम स्वचालन उपकरणों में से एक है। यह DevOps और Agile बाजार के लिए सबसे उपयुक्त है। यह एक एआई-संचालित उपकरण है जो सरल अंग्रेजी का उपयोग करके जटिल परीक्षणों को स्वचालित करता है। कोई प्रोग्रामिंग नहीं CI/CD समर्थन।
LamdaTest	LamdaTest सर्वश्रेष्ठ क्रॉस ब्राउज़र टेस्ट ऑटोमेशन टूल में से एक है। Selenium Selenium ग्रिड आधारित सुरक्षित, स्केलेबल और विश्वसनीय क्लाउड पर स्वचालन परीक्षण चलाएं

तालिका 4.1.1 सॉफ्टवेयर Testing उपकरण

सॉफ्टवेयर Testing एक सतत प्रक्रिया है जो ऍप्लिकेशन्स, मॉड्यूल, सबसिस्टम और यूनिट प्रक्रियाओं का Test करके वृद्धिशील रूप से प्रभावित होती है जो निम्नतम स्तर पर हैं। Testing पद्धति सॉफ्टवेयर त्रुटि, गलती और गुणवत्ता प्राप्ति के सामान्य मुद्दे पर हमला करने में विफलता को अलग करती है। त्रुटि को विसंगति के रूप में वर्णित किया गया है, गलती को खराबी के रूप में वर्णित किया गया है, और विफलता को अक्षमता के रूप में वर्णित किया गया है। इनमें से प्रत्येक परिदृश्य में, सॉफ़्टवेयर का Test किया जाना चाहिए; Testing इस तरह से किया जाना चाहिए जिससे सॉफ़्टवेयर की कमियों का पता चलता है। जबकि स्वचालित Test उपकरण Testing की लागत और समय को कम करते हैं, मैनुअल Testing एक श्रम-गहन प्रक्रिया है और इसलिए कम लागत प्रभावी है।

4.1.11 लॉगिंग की विधि: Testing प्रगति, परिणाम, डिफेक्ट और Agreed Test प्रबंधन उपकरण के उपयोग

सॉफ्टवेयर सिस्टम बनाने, Testing करने और चलाने वाली टीमों को आधुनिक लॉग एकत्रीकरण और खोज टूल द्वारा दी जाने वाली महत्वपूर्ण नई क्षमताओं से बहुत लाभ होता है। हम लॉगिंग को कोर सिस्टम घटक के रूप में मानकर और अद्वितीय ईवेंट ID, लेनदेन अनुरेखण और संरचित लॉग आउटपुट, विशेष रूप से क्रॉस-घटक दृश्यता जैसी तकनीकों का उपयोग करके एप्लीकेशन व्यवहार और स्वास्थ्य में गहरी अंतर्दृष्टि प्राप्त करते हैं।

पृष्ठ दर पृष्ठ, समीक्षक द्वारा समीक्षक, और या तो लेखक द्वारा या एक मुंशी द्वारा, कठिनाइयों, जैसे कि खामियों, जो तैयारी के दौरान पाए गए थे, प्रलेखित हैं। निरीक्षण जैसे औपचारिक समीक्षा प्रकारों के लिए, लॉगिंग करने के लिए एक अलग व्यक्ति (एक scribe) विशेष रूप से सहायक होता है।

दक्षता और प्रगति सुनिश्चित करने के लिए लॉगिंग करते समय किसी वास्तविक चर्चा की अनुमति नहीं है। चर्चा का चरण वह है जहां किसी मुद्दे को संभाला जाता है यदि उस पर चर्चा करने की आवश्यकता होती है। यह एक समस्या है या नहीं, इस बारे में लंबी चर्चा करने के बजाय किसी मुद्दे को लॉग इन करना और अगले एक पर आगे बढ़ना अधिक कुशल है। इसके अतिरिक्त, टीम की राय के बावजूद, एक चर्चा और अस्वीकार किए गए डिफेक्ट बहुत अच्छी तरह से काम के दौरान वास्तविक हो सकते हैं। हर डिफेक्ट को उसकी गंभीरता के साथ दर्ज किया जाना चाहिए। जो प्रतिभागी डिफेक्ट पाता है वह गंभीरता की डिग्री का सुझाव देता है। गंभीरता श्रेणियों में शामिल हैं:

- **विवेचनात्मक:** केवल निरीक्षण किए जा रहे दस्तावेज़ से अधिक प्रभावित करने वाले डिफेक्ट्स का डाउनस्ट्रीम प्रभाव होगा।
- महत्वपूर्ण खामियों का भविष्य में प्रभाव पड़ सकता है (उदाहरण के लिए किसी डिजाइन में गलती के परिणामस्वरूप कार्यान्वयन में त्रुटि हो सकती है)।
- मामूली खामियों के परिणामस्वरूप और नुकसान होने की संभावना नहीं है (उदाहरण के लिए मानकों और टेम्पलेट्स का अनुपालन न करना)।

वर्तनी की गलतियों को समीक्षाओं के अतिरिक्त मूल्य को संरक्षित करने के लिए डिफेक्ट वर्गीकरण में शामिल नहीं किया गया है। वर्तनी त्रुटियों को प्रतिभागियों द्वारा समीक्षा के तहत दस्तावेज़ में नोट किया जाता है और बैठक के समापन पर लेखक को दिया जाता है, या उन्हें एक अलग प्रूफरीडिंग अभ्यास में संबोधित किया जा सकता है। लॉगिंग चरण का लक्ष्य एक निर्धारित समय के भीतर अधिक से अधिक डिफेक्ट्स को रिकॉर्ड करना है। मॉडरेटर इसकी गारंटी के लिए उच्च लॉगिंग दर (प्रति मिनट रिपोर्ट किए गए डिफेक्ट्स की संख्या) को बनाए रखने का प्रयास करता है। लॉगिंग दर एक अच्छी तरह से चलने वाली और व्यवस्थित औपचारिक समीक्षा बैठक में प्रति मिनट 1-2 डिफेक्ट्स के बीच होनी चाहिए।

4.1.12 डिफेक्ट प्रबंधन

सॉफ्टवेयर विकास के दृष्टिकोण से एक डिफेक्ट केवल एक त्रुटि संदेश या कोडिंग त्रुटि द्वारा लाए गए सिस्टम क्रैश से अधिक है। वास्तविक और प्रत्याशित परिणाम के बीच कोई अंतर एक डिफेक्ट है, जैसे कि जब सॉफ़्टवेयर त्रुटिपूर्ण रूप से काम करने के बावजूद बिल्कुल वैसा ही प्रदर्शन करता है।

एक सॉफ्टवेयर आवश्यकताओं विनिर्देश की अपेक्षाओं को पूरा नहीं किया जाता है जब कोई डिफेक्ट होता है। जब पूर्व-उत्पादन परीक्षणों द्वारा फंक्शनल या प्रदर्शन के मुद्दों का पता नहीं लगाया जाता है, तो लाइव सॉफ़्टवेयर में डिफेक्ट भी दिखाई देते हैं। यहाँ छह उदाहरण हैं:

- **मूल सॉफ़्टवेयर विनिर्देशों में त्रुटियां, चूक या छेद** ये खामियां एक आवश्यकता की अनदेखी, खराब वाक्यांश, हितधारकों द्वारा गलत समझे जाने या डेवलपर्स द्वारा गलत व्याख्या किए जाने के परिणामस्वरूप दिखाई दे सकती हैं।
- **सॉफ़्टवेयर के आर्किटेक्चर या डिज़ाइन में त्रुटियाँ** ये समस्याएँ तब उत्पन्न होती हैं जब सॉफ़्टवेयर डेवलपर अप्रभावी सॉफ़्टवेयर एल्गोरिदम या प्रक्रियाओं को डिज़ाइन करते हैं, या जब वे एल्गोरिदम या प्रक्रियाएँ आवश्यक सटीकता के साथ परिणाम नहीं देती हैं।
- **कोडिंग गलतियाँ या कार्यान्वयन की गलतियाँ**। लापता ब्रैकेट और अनजेंटल एरर हैंडलिंग जैसी पारंपरिक त्रुटियां इन खामियों में से हैं। इन डिफेक्ट्स में पारंपरिक बग शामिल हैं जो लापता ब्रैकेट से लेकर अशोभनीय त्रुटि से निपटने तक सब कुछ के कारण होते हैं।

- **Test योजना या Test गतिविधियों में त्रुटियां।** ये डिफेक्ट अपर्याप्त Testing सुविधाओं और कार्यों से उपजी हैं। ये खामियां उन Attributes और कार्यों के परिणामस्वरूप होती हैं जिनका पूरी तरह से Testing नहीं किया गया था।
- **deployment के दौरान की गई गलतियाँ या चूक।** इन डिफेक्ट्स का एक उदाहरण तब होगा जब एक टीम अपर्याप्त वीएम संसाधनों का प्रावधान करती है।
- **प्रक्रिया या नीतियों में त्रुटियां जो एक टीम विकास चक्र को नियंत्रित करने के लिए उपयोग करती है**। ये डिफेक्ट तब सामने आते हैं, जब उदाहरण के लिए, एक टीम पर्याप्त डिजाइन, कोडिंग या Testing समीक्षा के बिना साइनऑफ या अनुमोदन प्राप्त करती है।

टीम डिफेक्ट को ठीक करने के लिए सक्रिय उपाय कर सकती है और मूल कारण विश्लेषण से समस्या की पहचान करने के बाद इसे फिर से होने से रोक सकती है। यदि, उदाहरण के लिए, एक डिज़ाइन गलती के परिणामस्वरूप समस्या हुई, तो डेवलपर्स डिज़ाइन और आवश्यकताओं के कागजात का अध्ययन कर सकते हैं और संशोधन कर सकते हैं। यदि Testing प्रक्रिया में किसी डिफेक्ट के कारण कोई समस्या हुई थी, तो डेवलपर Test cases और मीट्रिक को बदल सकते हैं।

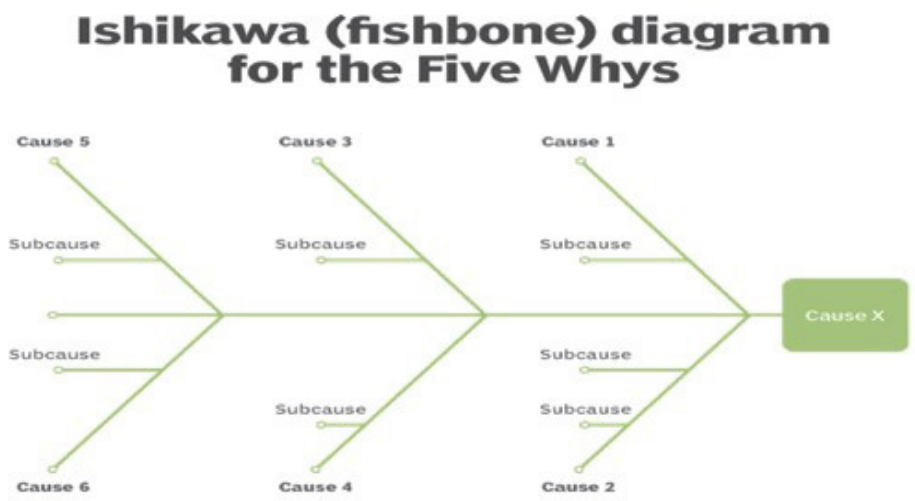
4.1.13 सॉफ्टवेयर डिफेक्ट्स के मूल कारण विश्लेषण के संचालन के लिए तरीके

सॉफ्टवेयर टीमें मूल कारण विश्लेषण कार्यों तक पहुंचने के लिए कई उपकरणों से आकर्षित कर सकती हैं, जिनमें शामिल हैं:

- Fishbone diagrams
- Five Whys
- Scatter plots
- Failure Mode and Effects Analysis (FMEA)
- Pareto charts

Fishbone diagram और Five Whys सबसे लोकप्रिय तकनीकें हैं।

Fishbone diagram। फिशबोन विश्लेषण का लक्ष्य, जिसे इशिकावा आरेख या कारण-और-प्रभाव आरेख के रूप में भी जाना जाता है, संभावित कारणों को उपश्रेणियों में समूहित करना है जो विश्लेषकों को मूल कारण की कल्पना करने में सहायता करने के लिए मुख्य समस्या से दूर शाखा करते हैं। परिणाम देने वाला आंकड़ा मछली के कंकाल के समान है, इसलिए नाम। परिणाम देने वाला आंकड़ा मछली के कंकाल के समान है, इसलिए नाम। वास्तविकता में, मछली का "सिर" वह जगह है जहां अंतर्निहित मुद्दा या समस्या लिखी जाती है। छवि की "हड्डियां" संभावित कारणों की श्रेणियों से बनी हैं। Next कदम विश्लेषकों के लिए प्रत्येक श्रेणी के लिए प्राथमिक कारणों को निर्धारित करना है; यदि अतिरिक्त माध्यमिक और तृतीयक कारणों की आवश्यकता होती है, तो उन्हें जोड़ा जा सकता है।



चित्र 4.1.1 5 Whys के लिए इशिकावा आरेख

- 5 Whys विश्लेषण। लोग "Whys' पूछकर एक समस्या में आगे बढ़ सकते हैं, जो नई संभावनाओं को खोलता है। प्रत्येक क्यों की प्रतिक्रिया निम्नलिखित जांच के लिए आधार के रूप में कार्य करती है। यह प्रक्रिया एक बच्चा के बराबर है जो क्यों प्रश्नों की एक श्रृंखला पूछ रहा है; हर बार जब कोई वयस्क जवाब देता है, तो नौजवान एक नया सवाल पूछने के लिए उस प्रतिक्रिया पर बनाता है। विधि बुद्धिशीलता का उपयोग करती है।

केवल चार "Whys' के साथ एक उदाहरण देखें:

- लॉग फ़ाइल अनुपलब्ध क्यों है?
- लॉग फ़ाइल अपेक्षित तार्किक यूनिट संख्या या फ़ोल्डर में नहीं मिली है।
- लॉग फ़ाइल वहाँ क्यों नहीं थी?
- सॉफ़्टवेयर एप्लीकेशन लॉग फ़ाइल सक्षम नहीं किया गया था।
- लॉग सुविधा चालू क्यों नहीं की गई?
- सॉफ़्टवेयर एप्लीकेशन का कॉन्फ़िगरेशन ग़लत था.
- सॉफ़्टवेयर ठीक से कॉन्फ़िगर क्यों नहीं किया गया था?

किसी टीम ने सॉफ़्टवेयर स्थापित करने और संचालित करने की प्रक्रिया पूरी नहीं की, या उन्होंने एप्लीकेशन को ठीक से दस्तावेज़ीकृत नहीं किया. सबसे अच्छा समाधान लॉग चालू करना और बेहतर दस्तावेज़ीकरण और उपयोगकर्ता निर्देश प्रदान करना हो सकता है।

4.1.14 प्रोग्रामिंग भाषा और इसके उपयोग

एक शक के बिना, संगठनों द्वारा उपयोग की जाने वाली पारंपरिक मैनुअल Testing विधियों को आधुनिक युग में स्वचालन Testing द्वारा प्रतिस्थापित किया जा रहा है। यह सब इसलिए हो रहा है क्योंकि स्वचालन Testing, मैनुअल Testing की तुलना में, बेहतर प्रदर्शन प्रदान करने और कम परिचालन लागत वहन करने के साथ-साथ अधिक श्रमदायी, कुशल और स्केलेबल है। स्वचालन Testing मुख्य रूप से विभिन्न प्रकार के उपकरणों और सॉफ्टवेयर का उपयोग करके Test cases को विकसित करने की प्रक्रिया पर केंद्रित है और फिर पूर्व-निर्धारित कार्यों को ध्यान में रखते हुए उन Test cases को निष्पादित करता है।

Python

एक शक के बिना, संगठनों द्वारा उपयोग की जाने वाली पारंपरिक मैनुअल Testing विधियों को आधुनिक युग में स्वचालन Testing द्वारा प्रतिस्थापित किया जा रहा है। यह सब इसलिए हो रहा है क्योंकि स्वचालन Testing, मैनुअल Testing की तुलना में, बेहतर प्रदर्शन प्रदान करने और कम परिचालन लागत वहन करने के साथ-साथ अधिक श्रम प्रभावी, कुशल और स्केलेबल है। स्वचालन Testing मुख्य रूप से विभिन्न प्रकार के उपकरणों और सॉफ्टवेयर का उपयोग करके Test cases को विकसित करने की प्रक्रिया पर केंद्रित है और फिर पूर्व-निर्धारित कार्यों को ध्यान में रखते हुए उन Test cases को निष्पादित करता है।

Javascript

Javascript फ्रंट-एंड डेवलपमेंट के माध्यम से ग्राहकों की अपेक्षाओं को फिर से परिभाषित करने में उत्कृष्टता प्राप्त करता है और Testing स्वचालन पर उच्च जोर देता है। Instagram, Accenture, Slack, और Air bnb सहित कई वेब एप्लिकेशन, instauto, ATOM Accenture Test Automation Open source Modular Libraries), Botkit, और Mavericks सहित Javascript ऑटोमेशन के साथ बनाए गए पुस्तकालयों का समर्थन करते हैं। अपरिहार्य रूप से, स्लैक, इंस्टाग्राम, एक्सेंचर और एयरबीएनबी सहित कई वेब एप्लिकेशन, Javascript ऑटोमेशन लाइब्रेरी प्रदान करते हैं, जिनमें इंस्टोटो, एटीओएम एक्सेंचर टेस्ट ऑटोमेशन ओपन सोर्स मॉड्यूलर लाइब्रेरी), मावेरिक्स और बॉट किट शामिल हैं।

C#

स्टैक ओवर फ्लो सर्वेक्षण के अनुसार, 60% से अधिक उपयोगकर्ता वाणिज्यिक उद्यम के विकास और स्वचालन की जरूरतों को पूरा करने के लिए C # प्रोग्रामिंग भाषा का समर्थन करते हैं।

C # अपने स्वचालन Testing ढांचे के लिए लगातार फलफूल रहा है, जिसे माइक्रोसॉफ्ट के कुशल डेवलपर्स द्वारा विकसित किया गया था। Microsoft के अनुभवी डेवलपर्स ने C# बनाया, जो अपने स्वचालन Testing उपकरणों की बदौलत लगातार फलफूल रहा है।

Ruby

एमवीसी आर्किटेक्चर द्वारा समर्थित एक और प्रोग्रामिंग भाषा Ruby है, जो व्यावसायिक क्षेत्रों में लोकप्रियता में बढ़ रही है जिन्हें स्वचालन की आवश्यकता है। यह संभव है कि उन विषयों को जोखिम प्रबंधन, अनुपालन, रसद, या भर्ती के साथ करना होगा। Ruby का एक और पेचीदा पहलू Selenium ऑटोमेशन टेस्टर्स के कार्य वातावरण को सुविधाजनक बनाने की इसकी क्षमता है, जिससे उन्हें कोड की कम लाइनों के साथ Boundary पार Testing और इसकी संबंधित प्रक्रियाओं को लागू करने में सक्षम बनाया जा सके।

Java

स्वचालन Testing के क्षेत्र में नौकरी की संभावनाओं को बढ़ाने के लिए बड़ी कंपनियां Java को अपनी दूसरी सबसे आवश्यक तकनीक के रूप में चुन रही हैं। ओरेकल कॉर्पोरेशन द्वारा आयोजित सामान्य प्रयोजन Java कोड (ओं) के लिए धन्यवाद, 2 बिलियन से अधिक डिवाइस इस बहुउद्देशीय भाषा के "एक बार लिखें, कहीं भी चलाएं" आधार द्वारा सक्षम विभिन्न स्वचालन लाभों को सिंक्रनाइज़ कर सकते हैं।

सारांश

- Test प्रबंधन दैनिक Test गतिविधियों को नियंत्रित करने की एक प्रक्रिया है, Test cases को विकसित करने से लेकर उन्हें बाहर ले जाने और निष्कर्षों को रिकॉर्ड करने तक।
- Test प्रबंधन उपकरण Test गतिविधियों के प्रबंधन और ट्रैकिंग के लिए आवश्यक हैं, Testing असाइन करने की सुविधा, पता लगाने की क्षमता और जटिलता और दोहराव को खत्म करने के लिए है।
- कुछ बेहतरीन Test प्रबंधन उपकरण हैं; स्पाइराटेस्ट, टेस ट्रेल, एक्स-रे, Zephyr स्केल, Zephyr Squad, प्रैक्टिटेस्ट, टेस्टलिंक, क्यूमेट्री आदि।
- प्री-स्क्रिप्टेड Testing जो स्वचालित रूप से चलता है उसे स्वचालित Test के रूप में जाना जाता है। अपेक्षित परिणाम के साथ वास्तविक परिणाम की तुलना करने के लिए Testing परिणामों पर जोर देने के लिए चलाए जाते हैं
- सॉफ्टवेयर Testing उपकरण के नवीनतम संस्करण Test Case और स्वचालित स्क्रिप्ट Selenium, cucumber, Ranorex आदि हैं।
- एक डिफेक्ट एक परिणाम के बीच कोई अंतर है जो वास्तव में होता है और एक जो प्रत्याशित था, जैसे कि जब सॉफ़्टवेयर त्रुटिपूर्ण रूप से काम करते हुए इरादा के अनुसार प्रदर्शन करता है।
- सॉफ्टवेयर टीमें विभिन्न तरीकों का उपयोग कर सकती हैं, जैसे कि Fishbone diagram और फाइव व्हाईज़ मेजर हैं, मूल कारण विश्लेषण असाइनमेंट को संबोधित करने के लिए।
- मैनुअल Testing की तुलना में, स्वचालन Testing अधिक श्रम-प्रभावी, कुशल और स्केलेबल है, बेहतर प्रदर्शन प्रदान करता है, और इसमें परिचालन खर्च कम होता है।

एक्टिविटी

लैब सत्र

- इस गतिविधि में, ट्रेनर आपस में जोड़ी बनाने के लिए कहता है।
- प्रत्येक जोड़ी को एक विषय के साथ दिया जाएगा जिस पर उन्हें प्रबंधन उपकरणों से नीचे सूचीबद्ध विषयों से प्रस्तुति देनी होगी।
    - o SpiraTest by Inflectra/TestRail/XRay – Test Management For Jira/Zephyr Scale/Zephyr Squad/PractiTest/TestLink/QTest/QMetry Test Management/Kualitee
- ट्रेनर नीचे दी गई सूची से समस्या चुनने के लिए कहते हैं या अपनी इच्छा के अनुसार चुनने के लिए खुले हैं:
- सक्रिय भागीदारी के लिए छात्र को प्रोत्साहित करें।
- कार्य पूरा होने के बाद वे इसे ट्रेनर को जमा करेंगे।
- ट्रेनर सर्वोत्तम भरे हुए लॉगिंग नमूने की सराहना करेगा और सर्वोत्तम प्रयासों के लिए प्रशंसा करेगा।

चर्चा सत्र

- प्रशिक्षक पूछताछ करेगा कि क्या प्रशिक्षुओं ने अध्याय को समझा है।
- इस गतिविधि में, प्रशिक्षक प्रशिक्षु से विषय से संबंधित कुछ प्रश्न पूछेगा।
- प्रशिक्षु विषय की अपनी समझ के अनुसार प्रश्नों का उत्तर देंगे।
- यदि प्रशिक्षुओं के मन में प्रश्न और भ्रम हैं; वे ट्रेनर के सामने उन लोगों को सामने रख सकते हैं।
- प्रशिक्षक यह सुनिश्चित करेगा कि वह प्रशिक्षुओं द्वारा रखे गए प्रश्नों के उचित उत्तर दे सके।

अभ्यास

A. **निम्नलिखित प्रश्नों के उत्तर दीजिए:**

1. प्रमुख प्रोग्रामिंग भाषा और इसके उपयोगों की सूची बनाएं।
2. मूल कारण विश्लेषण कार्यों तक पहुंचने के लिए नीचे दिए गए टूल की व्याख्या करें
   a) Fishbone diagram
   b) Five Whys
3. Test cases और स्वचालित स्क्रिप्ट के लिए सॉफ़्टवेयर Testing टूल के नवीनतम संस्करण:
4. Test प्रबंधन उपकरण की आवश्यकता के बारे में बताएं।

B. **सही उत्तर का चयन कीजिए:**

1. शेड्यूलिंग, ट्रैकिंग और विश्लेषण के प्रबंधन के लिए किस Testing उपकरण का उपयोग किया जाता है
   a) Test प्रबंधन उपकरण
   b) Testing तुलनित्र उपकरण
   c) Testing निष्पादन उपकरण
   d) प्रदर्शन Testing उपकरण
2. संकेत क्या हैं जो बताते हैं कि एक सॉफ्टवेयर प्रोजेक्ट मुश्किल में है
   a) उत्पाद क्षेत्र अनुचित रूप से निर्धारित किया गया है।
   b) समय Boundary अवास्तविक हैं।
   c) परिवर्तनों को खराब तरीके से प्रबंधित किया जाता है।
   d) उपरोक्त सभी

नोट्स

Skill India
कौशल भारत-कुशल भारत

सत्यमेव जयते
GOVERNMENT OF INDIA
MINISTRY OF SKILL DEVELOPMENT
& ENTREPRENEURSHIP

N·S·D·C
National
Skill Development
Corporation
Transforming the skill landscape

5. परियोजनाओं की गुणवत्ता आश्वासन में योगदान

IT - ITeS SSC
nasscom

यूनिट 5.1 - परियोजनाओं की गुणवत्ता आश्वासन में योगदान

SSC/N1302

प्रमुख सीखने के परिणाम

इस मॉड्यूल के अंत तक, प्रतिभागी निम्न में सक्षम होंगे:

1. उन Checkpoints की पहचान करने की क्षमता का प्रदर्शन करें जिनका परियोजना के प्रत्येक चरण के दौरान एक परियोजना को पालन करना चाहिए।
2. किए गए चेक, एकत्र किए गए डेटा और जानकारी और पहचाने गए जोखिमों पर रिपोर्ट तैयार करें।

यूनिट 5.1: प्रोजेक्ट्स की गुणवत्ता आश्वासन में योगदान

यूनिट के उद्देश्य

इस यूनिट के अंत तक, प्रतिभागी निम्न में सक्षम होंगे:

1. डेटा पर प्रभावी गुणवत्ता Testing के सिद्धांतों पर चर्चा करें।
2. मानक टेम्पलेट्स और उपकरणों का उपयोग करके प्रमुख संकेतकों के खिलाफ आवश्यक डेटा/जानकारी एकत्र करें।
3. गुणवत्ता टीम में अनुमोदन प्रक्रिया के पदानुक्रम पर चर्चा करें।
4. विशेषज्ञों के साथ चर्चा करें, प्रोजेक्ट डेटा से संबंधित किसी भी मुद्दे, जहां आवश्यक हो।
5. प्रोजेक्ट्स के जोखिमों की सटीक पहचान करने के लिए डेटा के प्रतिगामी / प्रगतिशील विश्लेषण का प्रदर्शन करें।
6. Test Plan, Test Cases और/या स्वचालित स्क्रिप्ट की समीक्षा प्रक्रिया लागू करें।
7. गुणवत्ता पर डेटा की विफलता Test या तनाव Test के प्रभाव की जांच करें।
8. प्रबंधन समीक्षकों, आंतरिक लेखा Tester और तकनीकी समीक्षकों को माइलस्टोन प्रोजेक्ट करने के लिए चार्ट/बार आरेखों के उपयोग का प्रदर्शन करें।
9. स्वचालित स्क्रिप्ट की निगरानी के लिए विभिन्न Test Plans, Test Cases की पहचान करें।

5.1.1 सॉफ्टवेयर Testing में गुणवत्ता आश्वासन का दायरा

Testing विषयों की एक विस्तृत श्रृंखला को कवर कर सकता है, जिसमें सिस्टम प्रशासन, व्यावसायिक आवश्यकताओं, डिजाइन आवश्यकताओं, प्रोग्रामर के कोड और हार्डवेयर कॉन्फ़िगरेशन के लिए मानक और सीमाएं शामिल हैं। Testing के दायरे में पेशेवर सर्वोत्तम प्रथाओं और उद्योग मानकों के संबंध में Testing भी शामिल हो सकते हैं। नतीजतन, Testing सॉफ्टवेयर इंजीनियरिंग के सभी पहलुओं को मान्य और पुष्टि करने का मौका प्रदान करता है। सॉफ्टवेयर Testing इस प्रकार एक संगठन को कई फायदे प्रदान करता है। डिफेक्ट्स का शीघ्र पता लगाने के कारण, समय और धन की बचत होती है। Testing यह सुनिश्चित करता है कि उत्पाद के लिए कम डाउनटाइम है, जिससे ग्राहकों की संतुष्टि बढ़ जाती है।

सॉफ्टवेयर गुणवत्ता आश्वासन के प्रभारी व्यक्ति का कार्य वर्तमान सॉफ्टवेयर विकास प्रक्रियाओं का विश्लेषण और आकलन करना है और बग की घटना से बचने के लक्ष्य के साथ इसे बेहतर बनाने के तरीके खोजना है। एक सॉफ्टवेयर गुणवत्ता आश्वासन समूह की जिम्मेदारियां और पहुंच एक सॉफ्टवेयर Testing समूह की तुलना में व्यापक हैं।

एक गुणवत्ता आश्वासन समूह की जिम्मेदारी यह सुनिश्चित करना है कि आश्वासन की परिभाषा के अनुसार उत्पाद की गुणवत्ता अच्छी है, जो "गारंटी या प्रतिज्ञा" या "संदेह से स्वतंत्रता" है। एक सॉफ्टवेयर प्रोजेक्ट के मानकों और कार्यप्रणाली को गुणवत्ता आश्वासन (QA) समूह द्वारा जांचा जा सकता है। यह प्रक्रिया की समस्याओं के लिए प्रतिक्रिया समाधान प्रदान करने के लिए सॉफ्टवेयर विकास प्रक्रिया की निगरानी और मूल्यांकन भी करता है और यह तय करने के लिए Testing प्रक्रिया करता है कि उत्पाद बाजार में रिलीज के लिए तैयार है या नहीं।

निम्नलिखित प्रमुख गुणवत्ता आश्वासन कार्य हैं:

- सॉफ्टवेयर विकास और गुणवत्ता नियंत्रण के लिए एक मानकीकृत प्रक्रिया बनाने के लिए और यह सुनिश्चित करने के लिए कि मानकों से प्रस्थान नहीं किया जाता है।
- गुणवत्ता कारकों, गुणवत्ता मानदंडों और गुणवत्ता मैट्रिक्स को विकसित करने और गुणवत्ता कारकों के एक पूर्ण सेट को परिभाषित करने के लिए।
- प्रक्रिया के प्रत्येक चरण में मानकों को स्थापित करने के लिए, जैसे कि आवश्यकता टेम्पलेट, डिजाइन पद्धतियां, कोडिंग मानक, आदि।
- प्रक्रिया के प्रत्येक चरण के लिए चेकलिस्ट बनाने और बाद के दिशानिर्देशों और चेकलिस्ट के खिलाफ प्रत्येक चरण के परिणामों को सत्यापित करने के लिए।

5.1.2 प्रोजेक्ट माइलस्टोन समीक्षा बैठक

विशिष्ट प्रोजेक्ट जीवन चक्र माइलस्टोन/चरणों में समग्र माइलस्टोन की प्रगति और उपलब्धियों की समीक्षा करना माइलस्टोन समीक्षा बैठक का लक्ष्य है। प्रोजेक्ट टीम इस बैठक के दौरान प्रोजेक्ट से संबंधित मुद्दों पर चर्चा करने, सीखे गए पाठों को साझा करने और आगामी माइलस्टोन के लिए सुधार की पेशकश करने के लिए उच्च प्रबंधन और अन्य हितधारकों के साथ मिल सकती है।

कार्य

- प्रोजेक्ट प्रबंधक माइलस्टोन प्रगति की गणना करने और एक माइलस्टोन समीक्षा रिपोर्ट बनाने के लिए माइलस्टोन समीक्षा रिपोर्ट टेम्पलेट का उपयोग करता है।
- योजना के अनुसार, प्रोजेक्ट प्रबंधक उपयुक्त हितधारकों को बैठक अनुरोध, एजेंडा और माइलस्टोन समीक्षा रिपोर्ट भेजता है।
- एजेंडे के अनुसार, प्रोजेक्ट प्रबंधक माइलस्टोन की समीक्षा करने के लिए बैठक आयोजित करता है।
- चर्चा की जाने वाली कार्यसूची की प्रत्येक वस्तु, और समस्याओं, सुधार के लिए सुझाव, सीखे गए सबक और कार्रवाई आइटम समस्या प्रबंधन प्रणाली में दर्ज किए जाते हैं।
- प्रोजेक्ट और ग्राहक संपर्क व्यक्ति प्रोजेक्ट प्रबंधक से एक माइलस्टोन समीक्षा रिपोर्ट प्राप्त करता है।
- प्रोजेक्ट प्रबंधक ने उचित हितधारकों को कार्रवाई सौंपी और उनकी प्रतिबद्धता को सुरक्षित किया।
- प्रोजेक्ट प्रबंधक कार्रवाई वस्तुओं के पूरा होने की निगरानी करता है।

5.1.3 माइलस्टोन समीक्षा बैठक के लिए दिशानिर्देश

1. माइलस्टोन समीक्षा बैठक साप्ताहिक उच्च प्रबंधन स्थिति बैठक के साथ समवर्ती रूप से या माइलस्टोन के पूरा होने के बाद की तारीख में आयोजित की जा सकती है।
2. माइलस्टोन समीक्षा बैठक के प्रासंगिक हितधारकों में आमतौर पर निम्नलिखित शामिल होते हैं, लेकिन अन्य हितधारकों को आवश्यकतानुसार आमंत्रित किया जा सकता है:
   - विकास प्रबंधक
   - QA प्रबंधक
   - प्रोसेस इंजीनियरिंग लीड
   - कॉन्फ़िगरेशन नियंत्रक
   - प्रोजेक्ट टीम (गुणवत्ता आश्वासन और विकास)
3. आम तौर पर, माइलस्टोन समीक्षा बैठक के एजेंडे में निम्नलिखित आइटम शामिल होते हैं (लेकिन यदि आवश्यक हो तो अन्य आइटम जोड़े जा सकते हैं):
   - प्रयास और अनुसूची विचरण के संदर्भ में प्रत्येक माइलस्टोन के समग्र विकास की जांच करें।
   - ग्राहक प्रतिक्रिया का विश्लेषण (यदि कोई हो)।
   - एक माइलस्टोन के दौरान टीम द्वारा सामना की जाने वाली समस्याएं।
   - माइलस्टोन के दौरान सीखे गए सबक और माइलस्टोन का पालन करने के लिए सुधार के विचार।

5.1.4 QA प्रक्रिया और विकास चरण

प्रारंभ में, QA सॉफ्टवेयर विकास प्रक्रिया की योजना और प्रबंधन में इस तरह से योगदान देता है जो पूरे प्रोजेक्ट में बड़ी समस्याओं को टालने में मदद करता है। इसे प्राप्त करने के लिए, QA इंजीनियर प्रोजेक्ट पर सॉफ्टवेयर विकास टीम के एक महत्वपूर्ण हिस्से के रूप में पूरे सॉफ्टवेयर विकास चक्र में उत्पाद स्वामी, प्रोजेक्ट प्रबंधक, व्यापार विश्लेषक और डेवलपर्स के अन्य विशेषज्ञों के साथ सहयोग करते हैं। कार्य की प्रकृति और प्रोजेक्ट के उद्देश्यों के आधार पर, गुणवत्ता आश्वासन गतिविधियों की संख्या और क्रम प्रोजेक्ट से प्रोजेक्ट में भिन्न हो सकते हैं।

विकास प्रक्रिया

- आवश्यकताओं का विश्लेषण
- डिज़ाइन
- कार्यान्वयन
- सत्यापन या Testing
- अनुरक्षण

QA प्रक्रिया

- आवश्यकताओं की समीक्षा
- Test Plan/लेखन Test Cases
- यूनिट Testing
- एकीकरण Testing
- प्रणाली Testing
- प्रदर्शन Testing
- सुरक्षा Testing
- क्रॉस-ब्राउज़र Testing / क्रॉस-प्लेटफ़ॉर्म Testing
- Test Cases को अद्यतन करना
- प्रतिगमन Testing

1. **चरण एक:** आवश्यकताओं और दस्तावेज़ीकरण की समीक्षा करें

   प्रलेखन के निर्माण के समानांतर, QA इंजीनियर प्रोजेक्ट पर काम करना शुरू करते हैं। वे इसके लिए आवश्यक शर्तें और सहायक कागजी कार्रवाई की जांच करते हैं:

- संपूर्णता
- अतिरेक
- स्पष्टता
- स्थिरता
- निष्पादन योग्यता

सत्यापनीयता इसका उद्देश्य Anamolies के लिए सिस्टम आर्किटेक्चर और प्रौद्योगिकियों की जांच करना है।

विकास प्रक्रिया के लिए मुख्य लाभ:

- जल्दी पता चलने पर त्रुटियों की लागत कम होती है।
- बेहतर दस्तावेज़ीकरण के परिणामस्वरूप कम श्रम इनपुट और अधिक सटीक अनुमानों के साथ उच्च गुणवत्ता वाली प्रोजेक्ट होती है।

  इस चरण के लिए दस्तावेज़ीकरण की समीक्षा के लिए विशेष सॉफ़्टवेयर का उपयोग करने पर विचार करें, जैसे कि संगम। आप संपूर्ण प्रोजेक्ट के लिए उपयोग की जाने वाली सभी प्रासंगिक कंटेंट को संकलित कर सकते हैं और आंतरिक ज्ञानकोष रख सकते हैं। जब कोई आवश्यकता या दस्तावेज़ समायोजित, जोड़ा, अद्यतन या हटाया जाता है, तो टीम के सभी सदस्यों को तुरंत सूचित किया जाता है।

2. **दूसरा चरण:** Test Cases की योजना बनाएं और तैयार करें

एक बार विनिर्देश स्थापित हो जाने के बाद, Test Cases को विकसित करना शुरू करने का समय आ गया है, जो उन प्रक्रियाओं को निर्दिष्ट करते हैं जिनका उपयोग QA इंजीनियर यह सुनिश्चित करने के लिए करते हैं कि सॉफ्टवेयर इरादा के अनुसार काम करता है। आप टेस्ट केस लिखने के लिए टेस्टरेल या ज़ेफिर जैसे विशेष टूल का भी उपयोग कर सकते हैं यदि इन मामलों की मात्रा काफी बड़ी हो जाती है। दोनों कार्यक्रम परीक्षणों के निर्माण और संशोधन और परिणामों की निगरानी के लिए मैट्रिक्स के उपयोग को सक्षम करते हैं।

3. **तीसरा चरण:** डिजाइन Test Cases

QA टीम विकास चरण पूरा होने के बाद Test Cases को चलाती है। इस चरण का प्राथमिक उद्देश्य यह जांचना है कि क्या समाधान तकनीकी रूप से सही ढंग से उत्पादित किया गया था और यदि यह मूल उत्पाद स्वामी के विनिर्देशों को पूरा करता है।

गुणवत्ता आश्वासन की प्राथमिक गतिविधियों और उनके संबंधित उद्देश्यों का विवरण नीचे दिया गया है:

- Smoke परीक्षण। QA इंजीनियर सत्यापित करते हैं कि सॉफ्टवेयर या उसका मॉड्यूल इरादा के अनुसार संचालित होता है। पारित होने पर, अतिरिक्त जांच शुरू होगी।
- एकीकरण Testing सत्यापन कि अलग-अलग तत्व एक एकीकृत प्रणाली के रूप में कार्य करते हैं।
- प्रदर्शन Testing सामान्य और अनुमानित पीक लोड के तहत सिस्टम के व्यवहार का Testing करें।
- तनाव Testing महत्वपूर्ण भार स्थापित करता है जिसके आगे सिस्टम विफल हो जाता है।
- सुरक्षा Testing यह सुनिश्चित करना कि समाधान में पर्याप्त स्तर की सुरक्षा है।
- क्रॉस-ब्राउज़र Testing/क्रॉस-प्लेटफ़ॉर्म Testing कई ब्राउज़रों (क्रोम, मोज़िला और सफारी) या प्लेटफ़ॉर्म (एंड्रॉइड, आईOS, विंडोज फोन) के साथ किसी एप्लिकेशन की संगतता सत्यापित करना। यह वेब और हाइब्रिड ऐप्स के लिए विशेष रूप से महत्वपूर्ण है।
- प्रतिगमन Testing पहले Testing किए गए कोड के भीतर समस्याओं की पहचान करना। आमतौर पर नई सुविधाओं को जोड़ते समय या पुराने सिस्टम को अपडेट करते समय आवश्यक होता है। फिर, Testing को स्वचालित करना संभव है (जैसे, यूनिट Testing, प्रतिगमन Testing)। Thumb rule: एक प्रोजेक्ट जितनी अधिक समय तक चलती है, स्वचालित Testing उतना ही महत्वपूर्ण हो जाता है।

4. **चरण चार:** रिपोर्ट करें और मापें

   जब एक गुणवत्ता आश्वासन इंजीनियर को बग मिलता है, तो वे इसे बग ट्रैकिंग सिस्टम में लॉग करते हैं, जो प्रोजेक्ट प्रबंधन उपकरण के रूप में दोगुना हो जाता है। आप इसके लिए रेडमाइन या जीरा का उपयोग कर सकते हैं, जो दोनों अत्यधिक विन्यास योग्य कार्यक्रम हैं। टीम के सभी सदस्य रीयल-टाइम कार्य अपडेट देख सकते हैं, और वे टूटे हुए लॉगिन फ़ॉर्म से लेकर सुरक्षा समस्याओं तक, किसी भी गंभीरता के मुद्दों को ट्रैक करना आसान बनाते हैं। यह टीम संचार को आसान बनाता है और सुधार प्रक्रिया की स्पष्ट तस्वीर बनाए रखने में सहायता करता है।

   प्रत्येक मुद्दे को उच्च से निम्न तक का प्राथमिकता स्तर सौंपा गया है, और विकास टीम उपलब्ध समय और संसाधनों के आधार पर उनका समाधान करती है।

5. **चरण पांच:** फ़िक्सेस सत्यापित करना

   जिम्मेदार QA इंजीनियरों को सूचित किया जाता है जब कोई डेवलपर किसी समस्या का समाधान करता है, और वे इसकी पुष्टि करते हैं। जब कोई समस्या नहीं मिलती है, तो बग ट्रैकिंग सिस्टम टिकट बंद कर देता है। इस नियम के अनुसार, किसी भी बग को तब तक मरम्मत के रूप में चिह्नित नहीं किया जा सकता जब तक कि उसे मान्य नहीं किया जाता है।

   आपकी QA प्रक्रिया को बेहतर बनाने के लिए संकेत और सुझाव

   एक Waterfall विकास पद्धति ऊपर वर्णित सभी Testing चरणों के साथ संगत है। जटिल, दीर्घकालिक प्रोजेक्ट्स और/या संवेदनशील उद्योगों में, जैसे कि स्वास्थ्य सेवा या रसद, जहां समस्याएं और त्रुटियां उत्पन्न होने की संभावना है, पूरी तरह से दस्तावेज तैयार करने में समय और ऊर्जा समर्पित करना उचित है। गुणवत्ता आश्वासन प्रक्रियाओं जैसे आवश्यकताओं की समीक्षा और Test Plan/Testing मामला लेखन व्यापक रूप से नियोजित हैं।

   जब समय और संसाधन सीमित होते हैं, तो लंबे दस्तावेज़ीकरण और संपूर्ण Testing प्रक्रियाओं के उत्पादन की तुलना में सॉफ्टवेयर स्थिरीकरण और सुधार पर ध्यान केंद्रित करना बेहतर होता है। हर कोई चाहता है कि उसका प्रोजेक्ट सफल हो। हालाँकि, जब प्रोजेक्ट ऑडिट का समय होता है, तो हम इसे ठुकरा देते हैं। हालाँकि, प्रोजेक्ट प्रबंधन समीक्षा करना आपकी क्षमताओं और आपके प्रोजेक्ट नियंत्रण के स्तर को दिखाने का एक शानदार तरीका है। परिणामस्वरूप आपका ग्राहक, प्रायोजक और हितधारक अधिक सुरक्षित महसूस करेंगे। इसके अतिरिक्त, यह सच्ची उत्कृष्टता प्राप्त करने के लिए अपनी टीम और प्रोजेक्ट को सीखने और प्रोत्साहित करने का एक तरीका है।

   सॉफ्टवेयर की समीक्षा

   सॉफ्टवेयर समीक्षा सॉफ्टवेयर विकास जीवन चक्र (SDLC) के शुरुआती चरणों में एक या एक से अधिक लोगों द्वारा किए गए सॉफ्टवेयर का व्यवस्थित अध्ययन है ताकि सॉफ्टवेयर की खामियों और डिफेक्ट्स को खोजा और सही किया जा सके। सॉफ्टवेयर समीक्षा सॉफ्टवेयर इंजीनियरों को सॉफ्टवेयर विकास जीवन चक्र (SDLC) के हिस्से के रूप में सॉफ्टवेयर की गुणवत्ता, कार्यक्षमता और अन्य प्रमुख विशेषताओं और घटकों को मान्य करने में सहायता करती है। यह एक व्यापक प्रक्रिया है जिसमें यह सुनिश्चित करने के लिए सॉफ्टवेयर उत्पाद का Testing शामिल है कि यह ग्राहक की आवश्यकताओं को पूरा करता है।

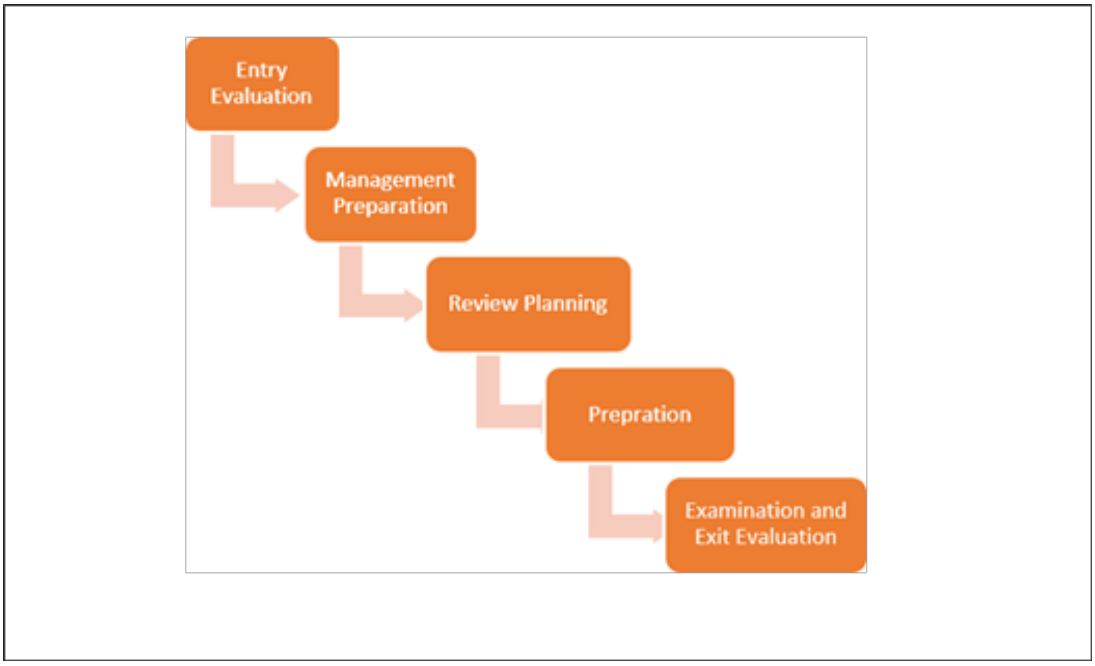
   आमतौर पर, मैनुअल सॉफ़्टवेयर समीक्षा कई दस्तावेजों को मान्य करने के लिए आयोजित की जाती है, जैसे कि आवश्यकताएं, सिस्टम डिज़ाइन, कोड, Test Plan और Test Cases।

सॉफ्टवेयर समीक्षा के उद्देश्य:

सॉफ्टवेयर समीक्षा का उद्देश्य है:

- विकास टीम के उत्पादन को बढ़ाने के लिए।
- Testing प्रक्रिया की लागत में तेजी लाने और कम करने के लिए।
- अंतिम उत्पाद में कम खामियों के साथ सॉफ्टवेयर बनाने के लिए।
- कमियों को दूर करने के लिए।**सॉफ्टवेयर समीक्षा की प्रक्रिया:**

सॉफ्टवेयर समीक्षा की प्रक्रिया:



चित्र 5.1.1 साफ्टवेयर समीक्षा प्रक्रिया

सॉफ्टवेयर समीक्षा के प्रकार:

मुख्य रूप से तीन प्रकार की सॉफ़्टवेयर समीक्षाएं हैं:

1. **सॉफ्टवेयर Peer समीक्षा:**

   Peer समीक्षा किसी उत्पाद की तकनीकी कंटेंट और गुणवत्ता का मूल्यांकन करने की प्रक्रिया है, और यह अक्सर उत्पाद के लेखक और अन्य डेवलपर्स द्वारा की जाती है। Peer समीक्षा एक ऐसी प्रक्रिया है जो उत्पाद की तकनीकी कंटेंट और गुणवत्ता का मूल्यांकन करती है, और यह आमतौर पर कुछ अन्य डेवलपर्स के सहयोग से कार्य उत्पाद के लेखक द्वारा की जाती है।

   सॉफ्टवेयर डिफेक्ट्स की जांच या उन्हें ठीक करने के लिए Peer समीक्षा की जाती है, जिसकी गुणवत्ता टीम के अन्य सदस्यों द्वारा भी सत्यापित की जाती है।

   Peer समीक्षा के कई प्रकार हैं:

   - **कोड** समीक्षा: कंप्यूटर स्रोत कोड का व्यवस्थित रूप से विश्लेषण किया जाता है।
   - **पेयर प्रोग्रामिंग:** यह एक कोड रिव्यू है जिसमें दो डेवलपर्स कोड बनाने के लिए एक ही प्लेटफॉर्म पर सहयोग करते हैं।
   - **पूर्वाभ्यास:** विकास टीम के सदस्यों को लेखक और अन्य इच्छुक पार्टियों द्वारा निर्देशित किया जाता है, जबकि प्रतिभागी प्रश्न उठाते हैं और खामियों पर प्रतिक्रिया प्रदान करते हैं।
   - **तकनीकी समीक्षा:** उच्च योग्य व्यक्तियों की एक टीम अपने ग्राहक के उपयोग के लिए सॉफ्टवेयर उत्पाद की जांच करती है और उत्पाद के विनिर्देशों और मानकों के आधार पर किसी भी तकनीकी समस्या का पता लगाती है।
   - **निरीक्षण:** निरीक्षण में, समीक्षक खामियों की पहचान करने के लिए एक अच्छी तरह से परिभाषित प्रक्रिया का उपयोग करते हैं।

2. **सॉफ्टवेयर प्रबंधन की समीक्षा**
   कार्य स्थिति का मूल्यांकन सॉफ्टवेयर प्रबंधन समीक्षा द्वारा किया जाता है। इस खंड में उन गतिविधियों के बारे में निर्णय लिए जाते हैं जो नीचे की ओर होंगी।
3. **सॉफ्टवेयर ऑडिट समीक्षा**
   "सॉफ्टवेयर ऑडिट समीक्षा" एक प्रकार की बाहरी समीक्षा है जिसमें एक या एक से अधिक आलोचक जो विकास टीम का हिस्सा नहीं हैं, सॉफ्टवेयर उत्पाद और इसकी प्रक्रियाओं पर एक निष्पक्ष नज़र डालते हैं ताकि यह देखा जा सके कि वे बताई गई आवश्यकताओं और मानकों को पूरा करते हैं या नहीं। इसके लिए प्रबंधकीय कर्मचारी जिम्मेदार हैं।

सॉफ्टवेयर समीक्षा के लाभ:

- विकास प्रक्रिया में डिफेक्ट्स का पता लगाया जा सकता है (विशेषकर औपचारिक समीक्षा में)।
- पहले के निरीक्षण में सॉफ्टवेयर रखरखाव की लागत भी कम हो जाती है।
- इसका उपयोग तकनीकी लेखक प्रशिक्षण के लिए किया जा सकता है।
- इसका उपयोग डिफेक्ट्स को बढ़ावा देने वाली प्रक्रिया की कमियों को खत्म करने के लिए किया जा सकता है।

5.1.5 तकनीकी विशेषज्ञ समीक्षा आयोजित करना

सॉफ्टवेयर इंजीनियर औपचारिक तकनीकी समीक्षा (FTR) करके तार्किक और कार्यात्मक त्रुटियों को जल्दी पा सकते हैं। FTR में, विशिष्ट भूमिकाओं में प्रतिभागी यह सुनिश्चित करते हैं कि विकसित किया जा रहा सॉफ्टवेयर स्थापित मानकों और आवश्यकताओं का अनुपालन करता है।

सॉफ्टवेयर इंजीनियर प्रारंभिक चरण में कार्यात्मक और तार्किक डिफेक्ट्स की पहचान करने के लिए एक औपचारिक तकनीकी समीक्षा (FTR) करते हैं। FTR में, निर्दिष्ट भूमिकाओं में प्रतिभागी गारंटी देते हैं कि विकसित सॉफ्टवेयर बताए गए मानकों और आवश्यकताओं के अनुरूप है।

जब आपका उत्पाद पूरी तरह से विकसित हो जाता है, तो FTR चलाने का यह आदर्श समय होता है। हालाँकि, यह समीक्षा के प्रकार के साथ भिन्न होता है। एक विशिष्ट FTR के लिए वक्ताओं, समीक्षकों या उत्पादकों के रूप में निर्दिष्ट जिम्मेदारियों वाले इंजीनियरों की एक टीम आवश्यक है। निम्नलिखित मुद्दों को हल करने के लिए प्रत्येक बैठक के बाद एक समीक्षा रिपोर्ट तैयार की जानी चाहिए:

- क्या समीक्षा की गई थी?
- इसकी समीक्षा किसने की?
- क्या निष्कर्ष और निर्णय किए गए थे?

FTR समीक्षाओं की एक श्रेणी को इंगित करता है जिसमें निम्नलिखित शामिल हैं:

- किसी उत्पाद का लेखक औपचारिक मूल्यांकन या समीक्षा बैठक में अपना काम प्रस्तुत करता है। प्राथमिक उद्देश्य शेष समीक्षकों को उत्पाद पेश करना है। नतीजतन, सभी प्रतिभागियों को आउटपुट स्वीकार करना चाहिए, समायोजन की पेशकश करनी चाहिए और समय सीमा पर चर्चा करनी चाहिए।
- पूर्वाभ्यास एक मीटिंग है जिसमें समीक्षक संदर्भित उत्पाद के स्रोत कोड, डिज़ाइन और बताई गई आवश्यकताओं का मूल्यांकन करते हैं। कोड डिफेक्ट्स की पहचान करने के लिए एक पूर्वाभ्यास बैठक आयोजित की जाती है। अक्सर, कोड का लेखक प्रश्नों का उत्तर देने के लिए मौजूद होता है।
- एक निरीक्षण एक समीक्षा सत्र है जो विनिर्देशों के अनुसार उत्पाद के अतिरिक्त गुणों को निर्धारित करता है। जबकि औपचारिक समीक्षाओं और पूर्वाभ्यास का उपयोग खामियों को उजागर करने के लिए किया जाता है, प्रारंभिक मानकों का विस्तार करने या यह सुनिश्चित करने के लिए निरीक्षण किए जाते हैं कि पहले के बग वापस नहीं आए हैं।
- औपचारिक तकनीकी समीक्षा त्रुटि की रोकथाम में सहायता करती है और तार्किक और कार्यान्वयन गलतियों की संभावना को पहले से कम करती है। इसके अलावा, यह पूरे उत्पाद की विशेषताओं को देखने में एक उत्पादन टीम की सहायता करता है, जिससे विकास अधिक नियंत्रणीय हो जाता है।

5.1.6 RCA (Root Cause Analysis)

RCA (Root Cause Analysis) डिफेक्ट्स के मूल कारण की पहचान करने की एक तकनीक है। हम यह निर्धारित करने के लिए विचार-मंथन, पढ़ने और खुदाई का उपयोग करते हैं कि क्या कोई डिफेक्ट " testing miss," "development miss," or "requirement or design miss."" के कारण हुआ था।

सटीक RCA डिफेक्ट्स को बाद के रिलीज या चरणों में प्रदर्शित होने से रोकने में मदद करता है। यदि यह पता चला है कि डिज़ाइन त्रुटि के कारण कोई डिफेक्ट था, तो हम डिज़ाइन दस्तावेज़ीकरण की समीक्षा कर सकते हैं और आवश्यक कार्रवाई कर सकते हैं। इसी तरह, हम अपने Test Cases या मीट्रिक की समीक्षा कर सकते हैं और उचित अपडेट कर सकते हैं यदि हमें पता चलता है कि Testing चूक के कारण कोई डिफेक्ट हुआ था।

केवल डिफेक्ट्स का Testing RCA का एकमात्र फोकस नहीं होना चाहिए। हम उत्पादन डिफेक्ट्स पर RCA भी कर सकते हैं। हम अपने टेस्ट बेड में सुधार कर सकते हैं और RCA के निष्कर्ष के आधार पर उन उत्पादन टिकटों को प्रतिगमन टेस्ट मामलों के रूप में जोड़ सकते हैं। यह डिफेक्ट या समान प्रकृति के अन्य डिफेक्ट्स को फिर से होने से रोकेगा।

Root Cause Analysis करने की विधि एक रोगी का इलाज करने वाले चिकित्सक के समान है। चिकित्सक शुरू में लक्षणों की पहचान करेगा। फिर, बीमारी के अंतर्निहित कारणों को निर्धारित करने के लिए, वह प्रयोगशाला Testing करेगा।

यदि बीमारी का अंतर्निहित कारण अज्ञात है, तो चिकित्सक स्कैन परीक्षणों के लिए संदर्भित करेगा। वह तब तक निदान और शोध करना जारी रखेगा जब तक कि वह रोगी की स्थिति के मूल कारण की पहचान नहीं कर लेता। किसी भी उद्योग में, Root Cause Analysis पर भी यही अवधारणा लागू होती है।

नतीजतन, RCA का उद्देश्य प्रक्रियाओं और साथ के उपकरणों के एक परिभाषित सेट का उपयोग करके, लक्षण का इलाज करने के विपरीत मूल कारण की पहचान करना है। यहडिफेक्ट विश्लेषण, समस्या निवारण और अन्य समस्या-समाधान तकनीकों से अलग है क्योंकि ये तकनीकें किसी विशेष समस्या के समाधान की तलाश करती हैं, जबकि RCA समस्या की जड़ की तलाश करता है।

5.1.7 प्रभावी दस्तावेज़ीकरण का अभ्यास करना

आइए शुरू करें कि प्रभावी दस्तावेज़ीकरण इतना महत्वपूर्ण क्यों है। टीम प्रलेखन कौशल के विकास को प्राथमिकता देने के बदले में आपकी कंपनी को क्या लाभ होगा? चार विशिष्ट लाभ हैं जिन्हें हम इस लेख में शामिल करना चाहते हैं:

1. **उत्पादकता में वृद्धि**

   प्रभावी दस्तावेज़ीकरण काम पर उत्पादकता और दक्षता को बढ़ाता है। इसके बारे में सोचें: नए कर्मचारी स्पष्ट दस्तावेज़ीकरण के साथ अपनी नौकरी अधिक तेज़ी से सीख सकते हैं। इसके अतिरिक्त, यदि महत्वपूर्ण कंपनी दस्तावेज़ लगातार एक ही सुरक्षित स्थान पर संग्रहीत किए जाते हैं, तो आपकी टीम को उनकी तलाश में समय बर्बाद नहीं करना पड़ेगा।

   हालाँकि अनगिनत अन्य उदाहरण हैं जिनका हम उपयोग कर सकते हैं, ये दोनों अच्छी तरह से हमारी बात को स्पष्ट करते हैं। दस्तावेज़ पहुँच योग्य और समझने योग्य होने पर टीमें अधिक प्रभावी ढंग से कार्य करती हैं.

2. **Beter प्रगति निगरानी**

   यदि आप प्रभावी दस्तावेज़ीकरण बनाए रखते हैं तो आप इस बात पर नज़र रखने में सक्षम होंगे कि आपकी टीम विशिष्ट प्रोजेक्ट्स और पहलों पर कैसा प्रदर्शन कर रही है। ताजा कंटेंट विपणन पहल के साथ स्टुअर्ट की भागीदारी के स्तर को निर्धारित करना आसान होगा और जेनी के माध्यम से सॉर्ट करने में सक्षम ऍप्लिकेशन्स की संख्या।

   यदि आप इस तरह से उनकी प्रगति को ट्रैक करते हैं तो आपकी टीम निम्नलिखित लक्ष्य की ओर काम करने के लिए केंद्रित और प्रेरित रहेगी। यह आपको समय सीमा को बेहतर ढंग से प्रबंधित करने में भी मदद करेगा।

3. **बेहतर ग्राहक सेवा**

   यदि आप अपने व्यवसाय के ग्राहकों के लिए प्रलेखन बना रहे हैं, तो आप प्रथम श्रेणी के ग्राहक सहायता प्रदान करने में सक्षम होंगे। जब कोई ग्राहक आपकी टीम को एक प्रश्न के साथ ईमेल करता है, तो लंबे ईमेल टाइप करने के बजाय, वे उन्हें केवल एक विशिष्ट दस्तावेज़ की दिशा में इंगित कर सकते हैं।

   यह आपकी टीम और ग्राहकों दोनों को ग्राहक सेवा आसान और अधिक प्रभावी बनाता है। इसे दूसरे तरीके से रखने के लिए, हर कोई जीतता है।

4. **बूस्टेड कंपनी वैल्यू**

   अंत में, प्रभावी दस्तावेज़ीकरण किसी कंपनी के मूल्य को बढ़ा सकता है। यदि आप उस कंपनी के मालिक हैं जिसे आप चलाते हैं, तो आप इसे एक दिन बेचना चाह सकते हैं और अपने प्रयासों को भुना सकते हैं। यदि आपने रास्ते में सटीक रिकॉर्ड नहीं रखा है तो इसे पूरा करना मुश्किल होगा।

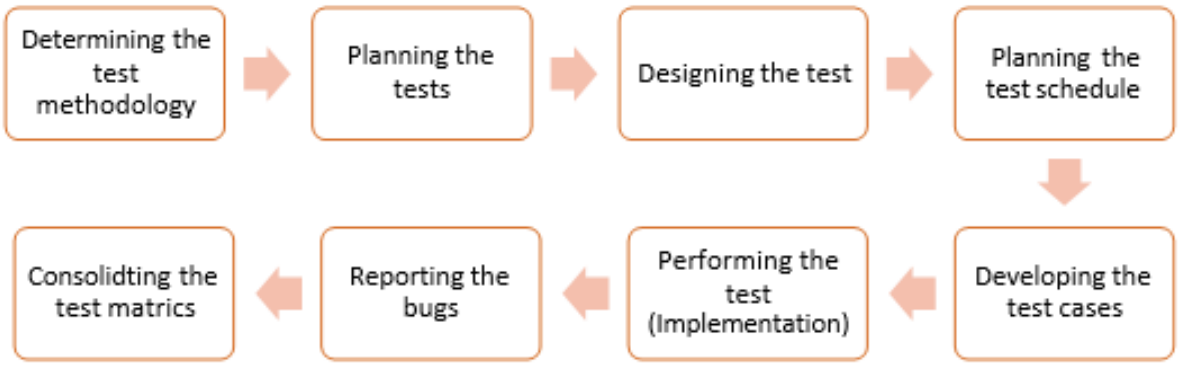
   किसी भी संभावित खरीदार को उन तकनीकों में दिलचस्पी होगी जिन्हें आपने सफलता हासिल करने के लिए बनाया है। यदि आप इन तथ्यों को साबित कर सकते हैं, तो आप उनका उपयोग उच्च बिक्री मूल्य पर बातचीत करने के लिए कर सकते हैं।

5.1.8 Test Plan

एक Test Plan सॉफ्टवेयर का Testing करते समय संदर्भित मौलिक दस्तावेज है। एक प्रोजेक्ट की व्यापक Test Plan Testing प्रबंधक का कर्तव्य है। टेस्ट प्लानिंग से पूरे प्रोजेक्ट के टेस्टिंग शेड्यूल और दृष्टिकोण का पता चलता है। इसलिए, प्रयास की मात्रा और Test Plan में जाने वाली जानकारी की मात्रा जानना महत्वपूर्ण है। एक अच्छी Test Plan अच्छी तरह से निर्मित Test Plans, Test Cases और Testing रिपोर्टों के साथ Testing प्रयास को कुशलतापूर्वक संवाद और दस्तावेज करने में मदद करेगी। यह Tester को Testing किए जा रहे उत्पाद के लिए वांछित गुणवत्ता प्राप्त करने में भी मदद करेगा।

5.1.9 Testing अनुसूची

विकास प्रक्रिया के दौरान, Testing किया जाता है। Testing आयोजित करने के लिए, प्रदर्शन उपायों की योजना, डिजाइन और परिभाषित करना आवश्यक है। Testing प्रक्रिया में शामिल गतिविधियों को चरणों में विभाजित किया जा सकता है, जो डिजाइन चरण में शुरू होते हैं और ग्राहक की साइट पर सॉफ़्टवेयर स्थापित होने पर समाप्त होते हैं।



चित्र 5.1.2 एक विशिष्ट नियोजन

Testing को लागू करने का मुख्य उद्देश्य सॉफ्टवेयर की प्रभावशीलता और दक्षता का Testing करना है। यह Testing किए जा रहे सिस्टम या सॉफ़्टवेयर में मौजूद अनिर्धारित त्रुटियों की संख्या को कम करने का एक प्रयास है। त्रुटियों को पहचानने और हटाने के लिए किए गए सभी उपायों के बावजूद, डिफेक्ट्स से मुक्त सॉफ़्टवेयर प्राप्त करना अभी भी एक अवास्तविक लक्ष्य है। इस चुनौती के लिए Tester को सॉफ्टवेयर में Testing की उच्च गुणवत्ता बनाए रखने की आवश्यकता होती है। Testing प्रक्रिया में सुधार के लिए अपनाए गए दो तरीके Testing के दौरान लागू Test Cases की प्रभावशीलता को उन्नत करना और स्वचालित सॉफ्टवेयर Testing उपकरण विकसित करना है। एक Tester अच्छी तरह से लिखित Test Plans, Test Cases और Testing रिपोर्टों की सहायता से Testing गतिविधि को सटीक रूप से संप्रेषित करने और दस्तावेज करने के कार्य को सफलतापूर्वक पूरा कर सकता है।

5.1.10 एक Test Plan लिखना

1. **सॉफ्टवेयर सीखें**

सॉफ्टवेयर का मूल्यांकन करने से पहले, इसके बारे में हर संभव जानकारी सीखना आवश्यक है। इसे कैसे बनाया गया था, इसके इच्छित उपयोग, यह कैसे कार्य करता है, और विवरण इकट्ठा करने के लिए जो आपको इसकी कार्यक्षमता को समझने में मदद कर सकता है, इसके बारे में अधिक जानने के लिए, इसके बारे में प्रश्न पूछें कि यह कैसे बनाया गया था। आप Test Cases को विकसित कर सकते हैं जो आपके सॉफ़्टवेयर को ठीक से समझकर आपके उत्पाद का मूल्यांकन करने के लिए प्रासंगिक और सहायक हैं।

3. Testing के दायरे को परिभाषित करें

Testing प्रलेखन बनाना जो उत्पाद से अधिक लंबा है, बेकार है। स्थापित करें कि प्रक्रिया के दौरान विशेष रूप से क्या Testing किया जाएगा, कौन से मॉड्यूल या कार्यों को विस्तार से कवर करने की आवश्यकता है, और किसी भी अन्य महत्वपूर्ण विवरण के बारे में आपको कुछ और करने से पहले पता होना चाहिए।

3. Test Cases बनाएँ

Test Cases बनाना सॉफ्टवेयर Testing दस्तावेज़ विकसित करने के महत्वपूर्ण चरणों में से एक है। एक Testing मामला आपके Testing के दौरान पालन की जाने वाली प्रक्रियाओं का एक लिखित खाता है। इसमें विवरण होना चाहिए जैसे:

- Testing की क्या आवश्यकता है
- इसका मूल्यांकन कैसे किया जाएगा?
- Testing कौन आयोजित करेगा?
- अपेक्षित परिणाम

Test Case Type	Description	Test Step	Expected Result	Status
Functionality	Area should accommodate up to 20 characters	Input up to 20 characters	All 20 characters in the request should be appropriate	Pass or Fail
Security	Verify password rules are working	Create a new password in accordance with rules	The user's password will be accepted if it adheres to the rules	Pass or Fail
Usability	Ensure all links are working properly	Have users click on various links on the page	Links will take users to another web page according to the on-page URL	Pass or Fail

तालिका 5.1.1 Spreadsheet for SeEng up a Tet Case

4. एक Test Startegy विकसित करें

Test Startegy आपकी Testing कार्यान्वयन Strategies की रूपरेखा तैयार करती है। सुनिश्चित करें कि टीम में हर कोई इस बात से अवगत है कि उन्हें किसी भी समय क्या करना है। आपके Tester को उसी गेम प्लान के अनुसार काम करना चाहिए।

5. Testing उद्देश्य को परिभाषित करें

एक Testing उद्देश्य प्रत्येक Test Cases से जुड़ा होना चाहिए। लक्ष्य सुनिश्चित करता है कि हर कार्रवाई प्रासंगिक है और ग्राहकों के लिए आपके सॉफ़्टवेयर के मूल्य को बढ़ाने में मदद करती है। Testing लक्ष्यों में निम्नलिखित शामिल हो सकते हैं:

- ज्ञात सुविधाओं का Testing
- नई कार्यान्वित सुविधाओं का Testing
- खोजपूर्ण Testing करना
- पूरे उत्पाद जीवन चक्र में स्थिरता सुनिश्चित करना

6. **Testing उपकरण चुनें**

अपने Testing कार्यों को करने के लिए, आपको यह सुनिश्चित करना होगा कि आपके पास उपयुक्त सॉफ़्टवेयर Testing समाधान है। इनमें से कुछ उपकरण कंप्यूटर प्रोग्राम हो सकते हैं, जबकि अन्य को Testing मशीनों जैसे वास्तविक हार्डवेयर की आवश्यकता हो सकती है। सामान्यीकृत दृष्टिकोण पर भरोसा करने के बजाय प्रत्येक व्यक्तिगत कार्य के लिए उचित उपकरण का चयन करना महत्वपूर्ण है।

7. बग जल्दी खोजें

अपने नियोजन दस्तावेज़ में, "बग फिक्सिंग" सत्रों के लिए समय शामिल करें। ये आपको सॉफ़्टवेयर समस्याओं को ठीक करने में बहुत कठिन या महंगे होने से पहले जल्दी स्पॉट करने में सक्षम बनाते हैं। इससे उनके साथ व्यवहार करना सरल और कम खर्चीला हो जाता है। किसी भी ऐप सुरक्षा सुविधाओं को सत्यापित करें, सभी सुविधाओं का उपयोग करें, और ऐसी किसी भी चीज़ की तलाश करें जो ठीक से काम न करे।

8. अपने Testing मानदंड को परिभाषित करें

भले ही यह Test Cases का एक हिस्सा होना चाहिए, फिर भी इसे और अधिक विस्तार से वर्णन करना एक अच्छा विचार है। आपके उद्देश्यों को अनिवार्य रूप से छोटे घटकों में विभाजित किया जाता है जिन्हें Testing मानदंड कहा जाता है। वे इस बारे में विस्तृत जानकारी प्रदान करते हैं कि प्रत्येक उद्देश्य कैसे पूरा किया जाएगा, जिससे आपके लिए अपनी Testing प्रगति पर नज़र रखना आसान हो जाता है।

ऐसे मानक हैं जिन्हें Testing निलंबित करने से पहले पूरा किया जाना चाहिए। उदाहरण के लिए, हो सकता है कि आप Testing बंद करना चाहें जब किसी विशिष्ट संख्या में बग की खोज की गई हो या जब प्रदर्शन समस्याओं के कारण सॉफ़्टवेयर का उपयोग नहीं किया जा सके।

निकास मानदंड ऐसी आवश्यकताएं हैं जिन्हें Testing पूरा होने से पहले पूरा किया जाना चाहिए। उदाहरण के लिए, सभी लक्ष्यों को प्राप्त करने और सभी बग ठीक हो जाने के बाद Test Cases को पूरा किया जाना चाहिए।

9. संसाधन योजना

अपने सॉफ़्टवेयर Testing दस्तावेज़ में Testing प्रक्रिया के लिए आवश्यक व्यक्तियों की संख्या का विवरण देने वाली एक संसाधन Strategies शामिल करें। इसमें प्रत्येक व्यक्ति की जिम्मेदारियों और किसी भी आवश्यक प्रशिक्षण का विवरण शामिल होना चाहिए।

चित्र 5.1.3 प्रोजेक्ट प्रबंधन में संसाधन नियोजन

10. अपने Testing परिवेश की योजना बनाएं

अपनी Test Plan में उस वातावरण के बारे में विवरण शामिल करें जहां Testing होगा, जैसे:

- उत्पाद Testing के लिए आवश्यक Testing उपकरण।
- सॉफ्टवेयर और सर्वर के आकार के लिए आवश्यकताएँ।
- प्लेटफ़ॉर्म जो उत्पाद का समर्थन करता है।
- पर्यावरण से संबंधित अतिरिक्त विवरण जो आपकी Testing प्रक्रिया को प्रभावित कर सकते हैं।

11. प्लान टेस्ट टीम लॉजिस्टिक्स

प्रक्रिया कार्यान्वयन के सबसे महत्वपूर्ण तत्वों में से एक Testing प्रबंधन है। आपका Testing दस्तावेज़ उतना उपयोगी नहीं होगा जितना कि यह हो सकता है यदि आप अपने Tester के साथ प्रभावी ढंग से संवाद नहीं कर सकते हैं। इससे उनके विकास में बाधा आएगी।

12. अनुसूची & अनुमान

एक समयरेखा शामिल करें जो आपको अपनी Test Plan में विशिष्ट Testing माइलस्टोन और समय सीमा निर्दिष्ट करने में सक्षम बनाती है। माइलस्टोन में पहला सार्वजनिक बीटा Testing, आंतरिक Testing सत्र, उत्पाद की प्रारंभिक रिलीज़, या कोई अन्य महत्वपूर्ण मोड़ शामिल हो सकते हैं जहां आपकी टीम को Testing पर अपने प्रयासों को केंद्रित करने की आवश्यकता होती है।

13. टेस्ट डिलिवरेबल्स

Testing के लिए आवश्यक प्रत्येक वितरण योग्य की एक सूची आपके Testing दस्तावेज में शामिल की जानी चाहिए। यह सुनिश्चित करने के लिए कि प्रत्येक व्यक्ति उस विशिष्ट समय से अवगत है जिस पर उन्हें कार्रवाई करने के लिए तैयार होना चाहिए, इन्हें कार्यक्रम के कदमों से जोड़ा जाना चाहिए।

14. Testing स्वचालन

- सॉफ़्टवेयर Testing स्वचालन पर विचार करें यदि आपका सॉफ़्टवेयर विशेष रूप से जटिल है और बहुत सारे Test Cases की आवश्यकता है।
- प्रक्रिया को स्वचालित करके, Tester कम समय में अधिक काम पूरा कर सकते हैं, उत्पादकता बढ़ा सकते हैं और समग्र रूप से Testing लागत को तेजी से कम कर सकते हैं। यहां तक कि एक मोबाइल बॉट भी Testing कार्यों में तेजी लाने के लिए उपयोगी हो सकता है।
- एक प्रभावी Test Startegy प्रोजेक्ट ब्रीफिंग का एक महत्वपूर्ण घटक है। आपकी Test Plan स्पष्ट, संक्षिप्त, अनुकूलनीय होनी चाहिए और आपके परिवेश या शेड्यूल में परिवर्तनों को ध्यान में रखना चाहिए।
- जब आपको उनकी आवश्यकता हो तो आवश्यक संसाधन उपलब्ध कराने के लिए, कई Testing-संबंधी पहलुओं में योजना और तैयारी आवश्यक है। लोगों और स्थानों सहित कुछ संसाधनों को बहुत अधिक योजना की आवश्यकता हो सकती है। इन संसाधनों को Test Plan में Testing आवश्यकताओं के साथ वर्णित किया गया है।
- Testing के कई तत्वों में योजना और तैयारी आवश्यक है ताकि आपको आवश्यकता पड़ने पर संसाधन उपलब्ध हों। कुछ संसाधनों, जैसे लोगों और स्थानों को व्यापक योजना की आवश्यकता हो सकती है। Test Startegy इन संसाधनों के साथ-साथ Testing विनिर्देशों की रूपरेखा तैयार करती है।
- बस ध्यान रखें कि कोई भी Test Plan निर्दोष नहीं है, लेकिन अभ्यास के साथ, Test Plan बनाना सरल हो जाता है।

सारांश

- Testing उत्पाद की स्थिरता का आश्वासन देता है और डाउनटाइम को कम करता है, जिसके परिणामस्वरूप ग्राहकों की संतुष्टि में वृद्धि होती है।
- आश्वासन की परिभाषा के अनुसार " a promise or pledge " या " freedom from question " के रूप में, एक गुणवत्ता आश्वासन समूह का काम यह सुनिश्चित करना है कि उत्पाद की गुणवत्ता उच्च है।
- माइलस्टोन समीक्षा बैठक का उद्देश्य चयनित प्रोजेक्ट मील के पत्थरों/चरणों में समग्र माइलस्टोन की प्रगति और उपलब्धियों की समीक्षा करना है।
- पूरे सॉफ्टवेयर विकास चक्र के दौरान, QA इंजीनियर सॉफ्टवेयर डेवलपमेंट टीम का एक महत्वपूर्ण सदस्य हैं, जो उत्पाद मालिक, प्रोजेक्ट प्रबंधक, व्यापार विश्लेषक और डेवलपर्स जैसे अन्य पेशेवरों के साथ काम कर रहे हैं।
- सॉफ्टवेयर समीक्षा सॉफ्टवेयर विकास जीवन चक्र के शुरुआती चरणों के दौरान एक सॉफ्टवेयर का एक या अधिक व्यक्तियों द्वारा व्यवस्थित निरीक्षण है जो डिफेक्ट्स और समस्याओं (SDLC) का पता लगाने और उन्हें ठीक करने के लिए सहयोग करते हैं।
- सॉफ्टवेयर इंजीनियर प्रारंभिक चरण में कार्यात्मक और तार्किक डिफेक्ट्स की पहचान करने के लिए एक औपचारिक तकनीकी समीक्षा (FTR) करते हैं।
- RCA (Root Cause Analysis) डिफेक्ट्स के मूल कारण की पहचान करने की एक विधि है। हम यह निर्धारित करने के लिए विचार-मंथन, पढ़ने और खुदाई का उपयोग करते हैं कि क्या कोई डिफेक्ट "Testing मिस," "विकास मिस," या "आवश्यकता या डिज़ाइन मिस" के कारण हुआ था।
- प्रभावी दस्तावेज़ीकरण अभ्यास के चार अलग-अलग लाभ हैं:
- बढ़ी हुई उत्पादकता, बेहतर प्रगति ट्रैकिंग, बढ़ी हुई ग्राहक सेवा और बढ़ी हुई कंपनी मूल्य।

एक्टिविटी

लैब सत्र

- इस गतिविधि में, प्रशिक्षक कक्षा को 10 बराबर जोड़े में विभाजित करेगा।
- सभी समूहों को गुणवत्ता आश्वासन प्रक्रिया पर कक्षा को संक्षिप्त करने की आवश्यकता होगी। प्रत्येक जोड़ी को नीचे दिए गए विषय के साथ दिया जाएगा जिसमें उन्हें लाभ भी बताना होगा।

  टीम 1: आवश्यकताओं की समीक्षा

  टीम 2: Test Plan/लेखन Test Cases

  टीम 3: यूनिट Testing

  टीम 4: एकीकरण Testing

  टीम 5: सिस्टम Testing

  टीम 6: प्रदर्शन Testing

  टीम 7: सुरक्षा Testing

  टीम 8: क्रॉस-ब्राउज़र Testing / क्रॉस-प्लेटफ़ॉर्म Testing

  टीम 9: Test Cases को अपडेट करना

  टीम 10: प्रतिगमन Testing
- दिए गए कार्य को सर्वोत्तम तरीके से पूरा करने वाले समूह को विजेता घोषित किया जाएगा और प्रशंसा के साथ कक्षा में सराहा जाएगा।

चर्चा सत्र

- प्रशिक्षक पूछताछ करेगा कि क्या प्रशिक्षुओं ने अध्याय को समझा है।
- इस गतिविधि में, प्रशिक्षक प्रशिक्षु से विषय से संबंधित कुछ प्रश्न पूछेगा।
- प्रशिक्षु विषय की अपनी समझ के अनुसार प्रश्नों का उत्तर देंगे।
- यदि प्रशिक्षुओं के मन में प्रश्न और भ्रम हैं; वे ट्रेनर के सामने उन लोगों को सामने रख सकते हैं।
- प्रशिक्षक यह सुनिश्चित करेगा कि वह प्रशिक्षुओं द्वारा रखे गए प्रश्नों के उचित उत्तर दे सके।

अभ्यास

A. **प्रत्येक कथन के सामने सही विकल्प पर निशान लगाएँ**

I. कौन पहचानता है, दस्तावेज करता है, और सत्यापित करता है कि सॉफ्टवेयर में सुधार किए गए हैं?

a) प्रोजेक्ट प्रबंधक
b) प्रोजेक्ट टीम
c) SQA समूह
d) सभी उल्लिखित

2. सॉफ्टवेयर गुणवत्ता आश्वासन में प्रबंधन के ऑडिटिंग और रिपोर्टिंग कार्य शामिल हैं।

a) True
b) False

3. QA और QC का क्या अर्थ है?

a) Quality Assurance and Queuing Control
b) Quality Adjustment and Quality completion
c) Quality Assurance and Quality control
d) Quality Adjustment and Queuing control

4. QA क्या है?

a) यह उस डिग्री का माप है जिस पर कोई उत्पाद आवश्यकता को पूरा करता है
b) प्रक्रिया में गुणवत्ता सुनिश्चित करने के लिए इस्तेमाल की जाने वाली कोई व्यवस्थित प्रक्रिया
c) डिफेक्ट्स की पहचान करने की प्रक्रिया
d) यह एक सुधारात्मक उपकरण है

5. QA के चरणों को आरोही क्रम में व्यवस्थित करें?

a) आवश्यकताओं और दस्तावेज़ीकरण की समीक्षा करें, Test Cases की योजना बनाएं और तैयार करें, Test Cases को डिजाइन करें, रिपोर्ट करें और मापें, सुधारों की पुष्टि करें
b) सुधारों का सत्यापन करना, Test Cases की योजना बनाना और तैयार करना, Test Cases को डिजाइन करना, आवश्यकताओं और दस्तावेज़ीकरण, रिपोर्ट और माप की समीक्षा करना
c) आवश्यकताओं और दस्तावेज़ीकरण की समीक्षा करें, सुधारों की पुष्टि करें, Test Cases की योजना बनाएं और तैयार करें, Test Cases को डिजाइन करें, रिपोर्ट करें और मापें
d) रिपोर्ट और उपाय, समीक्षा आवश्यकताओं और प्रलेखन, सुधारों की पुष्टि करना, Test Cases की योजना बनाना और तैयार करना, Test Cases को डिजाइन करना,

B. **नीचे दिए गए प्रश्नों के उत्तर दें:**

a) FTR का पूर्ण रूप क्या है?
b) RCA (Root Cause Analysis) को परिभाषित करें?
c) सॉफ्टवेयर समीक्षा प्रक्रिया के चरणों को सूचीबद्ध करें।

नोट्स

संबंधित वीडियो देखने के लिए क्यूआर कोड को स्कैन करें

https://www.youtube.com/watch?v=TAjwPW2iULY

एक टेस्ट इंजीनियर के लिए गुणवत्ता आश्वासन के दायरे पर चर्चा करें

Skill India
कौशल भारत-कुशल भारत

सत्यमेव जयते
GOVERNMENT OF INDIA
MINISTRY OF SKILL DEVELOPMENT
& ENTREPRENEURSHIP

N·S·D·C
National
Skill Development
Corporation
Transforming the skill landscape

IT - ITeS SSC
nasscom

6. सॉफ्टवेयर ऍप्लिकेशन्स के लिए प्रमुख संकेतक

यूनिट 6.1 - सॉफ्टवेयर ऍप्लिकेशन्स के लिए मुख्य संकेतक

SSC/N1303

प्रमुख सीखने के परिणाम

इस मॉड्यूल के अंत तक, प्रतिभागी निम्न में सक्षम होंगे:

1. गुणवत्ता Testing के लिए प्रमुख संकेतकों के प्राथमिक स्रोतों की पहचान करें।
2. गुणवत्ता जांच के लिए Third Party को प्रदान किए गए डेटा या जानकारी के उद्देश्य की जांच करें।

यूनिट 6.1: सॉफ्टवेयर ऍप्लिकेशन्स के लिए मुख्य संकेतक

यूनिट के उद्देश्य

इस यूनिट के अंत तक, प्रतिभागी निम्न में सक्षम होंगे:

1. प्रोजेक्ट्स की गुणवत्ता आश्वासन को प्रभावित करने वाले प्रमुख कारकों की पहचान करें और प्रोजेक्ट्स को इनका अनुपालन क्यों करना चाहिए।
2. जोखिमों के प्रकार और उनके संकेतकों जैसे अप्रमाणित प्रौद्योगिकियों, उपयोगकर्ता और कार्यात्मक आवश्यकताओं, एप्लीकेशन और सिस्टम आर्किटेक्चर आदि पर चर्चा करें।
3. गुणवत्ता आश्वासन में Third Party के प्रदर्शन के लिए चौकियों को परिभाषित करें।
4. सॉफ्टवेयर की गुणवत्ता को प्रभावित करने वाले कारकों की व्याख्या करें जैसे कि अपर्याप्त Third-Party प्रदर्शन, बौद्धिक संपदा की सुरक्षा में मुकदमेबाजी, अप्रचलित सॉफ्टवेयर और गलत सॉफ्टवेयर कार्यक्षमता।
5. विश्लेषण करें कि प्रबंधन समीक्षक, आंतरिक लेखा Tester, तकनीकी समीक्षक सॉफ्टवेयर ऍप्लिकेशन्स के लिए साझा किए गए डेटा / जानकारी के उद्देश्य को कैसे प्रभावित करते हैं।

6.1.1 सॉफ्टवेयर Testing में प्रभाव विश्लेषण की मूल बातें

सॉफ्टवेयर विकास एक सतत प्रक्रिया है जहां हम लगातार नई सुविधाओं को जोड़ते हैं या मौजूदा कार्यक्षमता को बढ़ाते हैं। हालांकि, किसी उत्पाद में किए गए प्रत्येक संशोधन का एक विशिष्ट घटक या यहां तक कि पूरे उत्पाद पर प्रभाव पड़ सकता है। हालांकि, किसी उत्पाद में किए गए प्रत्येक संशोधन का एक विशिष्ट घटक या यहां तक कि पूरे उत्पाद पर प्रभाव पड़ सकता है। इसके अलावा, किसी उत्पाद में हमारे द्वारा किए गए परिवर्तनों के प्रभावों को ट्रैक करना कठिन हो जाता है क्योंकि हम उनमें से अधिक बनाते हैं।

इसलिए, इसकी उचित सुरक्षा और डिफेक्टरहित प्रदर्शन की गारंटी के लिए उद्योग की सर्वोत्तम प्रथाओं के अनुसार अपने अपडेट किए गए उत्पाद का Testing करना पर्याप्त नहीं हो सकता है। कुछ कोड अनुभागों को दो बार सत्यापित करने की आवश्यकता हो सकती है, अधिक गहन परीक्षा दी जा सकती है, या एक अलग Strategies का उपयोग करके Testing किया जा सकता है। कोड के कुछ अनुभागों के लिए एक अलग Test Startegy, एक गहन विश्लेषण या डबल-चेकिंग आवश्यक हो सकती है। और संपूर्ण प्रभाव विश्लेषण करना आपके उत्पाद के इन घटकों को अलग करने का सबसे अच्छा तरीका है।

एक सॉफ्टवेयर Testing Startegy जिसे प्रभाव विश्लेषण कहा जाता है, Testing उत्पाद में किए गए किसी भी परिवर्तन से जुड़े सभी जोखिमों को परिभाषित करने में सहायता करती है।

एक प्रभाव विश्लेषण आयोजित किया जाना चाहिए जब:

- उत्पाद को बदलने के लिए अनुरोध किया गया है।
- उत्पादों के लिए विनिर्देश बदल गए हैं।
- वर्तमान मॉड्यूल या सुविधाओं में परिवर्तन हुए हैं
- आप नए मॉड्यूल या सुविधाएँ रखना चाहते हैं।

हालांकि यह प्रक्रिया उत्पाद विकास की समग्र लागत बढ़ा सकती है, अतिरिक्त लागत आसानी से उचित है।

Testing में प्रभाव विश्लेषण क्या है?

तीन प्रकार के प्रभाव विश्लेषण हैं जो इस प्रक्रिया के विभिन्न तत्वों पर ध्यान केंद्रित करते हैं और विभिन्न उद्देश्यों को आगे बढ़ाते हैं:

- निर्भरता प्रभाव विश्लेषण का मुख्य लक्ष्य निर्भरताओं की पहचान करना है, जो संभावित Party हो सकता है ईपरिवर्तनों या उत्पाद घटकों के प्रभाव जिन्हें इन परिवर्तनों के परिणामस्वरूप फिर से डिज़ाइन करने की आवश्यकता होती है।
- अनुभवात्मक प्रभाव विश्लेषण का उद्देश्य संपूर्ण विकास प्रक्रिया के संदर्भ में उत्पाद परिवर्तनों से जुड़े जोखिमों का अनुमान लगाना है, जिसमें विकास के लिए अतिरिक्त समय और संसाधनों की आवश्यकता शामिल है।
- ISTQB शब्दावली की परिभाषा के अनुसार, ट्रेसबिलिटी प्रभाव विश्लेषण, यह आकलन करता है कि उत्पाद में किसी विशेष परिवर्तन को लागू करने के लिए विभिन्न दस्तावेज़ीकरण स्तरों पर क्या बदला जाना चाहिए।

TEI (टेस्ट एफिशिएंसी इंडिकेटर) की गणना

एक डिफेक्ट गंभीरता बिंदु प्रणाली और सिस्टम Testing बनाम UAT में कुल डिफेक्ट्स की गणना Testing दक्षता संकेतक की गणना करने के लिए की जाती है। लॉग किए गए प्रत्येक डिफेक्ट को एक गंभीरता रेटिंग दी जाती है, जो एक बिंदु रैंकिंग से मेल खाती है।

- Critical = 4
- Serious = 3
- Medium = 2
- Low =1

पूरे सिस्टम और UAT Testing चक्रों में खोजी गई खामियों की कुल संख्या तब गिनी जाती है, और संबंधित बिंदु रैंकिंग निर्धारित की जाती है। UAT के दौरान खोजी गई कोई भी खामी जिसे प्रारंभिक Testing सेटिंग में पुन: प्रस्तुत नहीं किया जा सकता है, विश्लेषण से बाहर ले जाया जाता है। उसके बाद, हम सिस्टम Testing के दौरान खोजे गए डिफेक्ट्स की कुल संख्या को सिस्टम और UAT Testing के दौरान खोजे गए डिफेक्ट्स की कुल संख्या से विभाजित करते हैं। अंतिम प्रतिशत समग्र रूप से Testing प्रक्रिया की ई प्रभावशीलता का प्रतिनिधित्व करता है।

TEI समीकरण: SIT Points / SIT + UAT Points = TEI

आम तौर पर, अच्छी Testing प्रक्रियाओं में लगभग 90% TEI होता है, जिसमें केवल 10-12% डिफेक्ट Leakage होता है। हालाँकि, जैसा कि ऊपर कहा गया है, QA मेंटर का लक्ष्य 5% डिफेक्ट Leakage के लिए उससे अधिक है।

6.1.2 सॉफ्टवेयर Testing मेट्रिक्स का आधार

सॉफ्टवेयर Testing मेट्रिक्स, जिसे सॉफ्टवेयर Testing माप भी कहा जाता है, एक सॉफ्टवेयर प्रक्रिया की चौड़ाई, गहराई, ऊंचाई और क्षमता दिखाता है और इसकी प्रभावशीलता और दक्षता को तुरंत बढ़ाने का प्रयास करता है। सॉफ्टवेयर Testing जीवन चक्र के दौरान Tester की टीम द्वारा किए गए विभिन्न Testing कार्यों का आकलन और ट्रैक रखने का सबसे प्रभावी तरीका Testing मेट्रिक्स के उपयोग के माध्यम से है। इसके अतिरिक्त, यह डेटा के संयोजन के आधार पर भविष्यवाणी के परिणाम को संप्रेषित करने में सहायता करता है। नतीजतन, दुनिया भर में सॉफ्टवेयर इंजीनियर निम्नलिखित विभिन्न सॉफ्टवेयर Testing मैट्रिक्स का उपयोग करते हैं:

सॉफ्टवेयर गुणवत्ता संकेतकों का अनुपालन

मेट्रिक्स और संकेतक आमतौर पर सॉफ्टवेयर गुणवत्ता और परिपक्वता में महत्वपूर्ण भूमिका निभाते हैं। वे प्रोजेक्ट और जिस दिशा में आगे बढ़ रही है, उस पर एक त्वरित अवलोकन प्रदान करते हैं। मीट्रिक के साथ सुधारों को लक्षित करना और उनका आकलन करना आसान है। हालांकि, मेट्रिक्स का डेवलपर्स और कोड पर भी नकारात्मक प्रभाव पड़ सकता है। समस्या तब होती है जब लोग अपने मूल लक्ष्य को भूल जाते हैं और झूठे प्रॉक्सी ट्रैप (नीचे उद्धृत) में पड़ जाते हैं। आमतौर पर, किसी उत्पाद की सीधे गुणवत्ता को मापना असंभव है, इसलिए हम कुछ आसान मापने के लिए चुनते हैं e.g यह उपाय उस मूल माप का सन्निकटन बन जाता है जिसके लिए हमने लक्षित किया था।

गुणवत्ता संकेतकों की मदद से प्रबंधन की चिंताओं को भी संबोधित किया जा सकता है। अनुशंसित गुणवत्ता संकेतकों में से कुछ हैं:

- **जारी रहना:** यह प्रत्येक चरण में डेवलपर द्वारा किए गए कार्य की मात्रा को मापता है।
- **स्थिरता:** यह मूल्यांकन करता है कि प्रत्येक चरण के उत्पाद अगले चरण में आगे बढ़ने के लिए पर्याप्त रूप से स्थिर हैं या नहीं।
- **प्रक्रिया अनुपालन:** यह प्रोजेक्ट की शुरुआत के दौरान अनुमोदित विकास प्रक्रियाओं के साथ डेवलपर की आज्ञाकारिता को मापता है।
- **गुणवत्ता मूल्यांकन प्रयास:** यह डेवलपर के प्रयास को मापता है जो आंतरिक गुणवत्ता मूल्यांकन गतिविधियों पर खर्च किया जा रहा है।
- **डिफेक्ट का पता लगाने की दक्षता:** यह मापता है कि किसी विशेष चरण में कितने डिफेक्ट्स का पता चला था
- **डिफेक्ट हटाने की दर:** यह समय की अवधि में पता लगाए गए डिफेक्ट्स की कुल संख्या को मापता है और हल किया गया है।
- **डिफेक्ट age प्रोफाइल:** यह उन डिफेक्ट्स की कुल संख्या को मापता है जिन्हें समय की अवधि में हल नहीं किया गया है।
- **डिफेक्ट घनत्व:** यह सिस्टम के डिफेक्ट-प्रवण भागों की पहचान करता है।
- **जटिलता**: यह किसी विशेष प्रोग्राम में कोड की जटिलता को मापता है।

6.1.3 गुणवत्ता आश्वासन के लिए सॉफ्टवेयर Testing प्रदर्शन संकेतक (KPI)

प्रदर्शन मापन का एक प्रकार, संगठन और Tester दोनों मापा जा सकने वाला डेटा एकत्रित करने के लिए प्रदर्शन संकेतकों या KPI का उपयोग करते हैं. KPI सटीक चश्मा हैं जिन्हें सॉफ़्टवेयर Testing टीम मापती है और यह सुनिश्चित करने के लिए जांच करती है कि प्रक्रिया कंपनी के लक्ष्यों का अनुपालन करती है। इसके अतिरिक्त, वे कोई भी आवश्यक कार्रवाई करने में टीम का समर्थन करते हैं यदि उत्पाद का प्रदर्शन पूर्व निर्धारित लक्ष्यों से कम हो जाता है।

संक्षेप में, प्रमुख प्रदर्शन संकेतक (KPI) महत्वपूर्ण मीट्रिक हैं जो सॉफ्टवेयर Testing दल यह सुनिश्चित करने के लिए गणना करते हैं कि प्रोजेक्ट ठीक से प्रगति कर रही है और प्रभावी रूप से लक्ष्य प्राप्त कर रही है, जिसे योजना, Strategies और/या बजट सत्रों के दौरान स्थापित किया गया था। ये सॉफ्टवेयर Tester के लिए कई महत्वपूर्ण KPI में से कुछ हैं:

सक्रिय डिफेक्ट: सक्रिय डिफेक्ट: यह सीधा लेकिन महत्वपूर्ण KPI यह निर्धारित करने में मदद करता है कि कोई समस्या नई, खुली या ठीक की गई है या नहीं और टीम को उसके समाधान के लिए उचित कार्रवाई करने में सक्षम बनाती है। इनका मूल्यांकन टीम की सीमा का उपयोग करके किया जाता है, और यदि वे इससे अधिक हो जाते हैं, तो उन्हें त्वरित कार्रवाई के लिए ध्वजांकित किया जाता है।

1. **स्वचालित Testing:** Testing प्रबंधक के लिए प्रमुख प्रदर्शन संकेतकों को ट्रैक और जांचते समय स्वचालित परीक्षणों को पहचानना महत्वपूर्ण है। ट्रिकी टीम को स्वचालित परीक्षणों की संख्या पर नज़र रखने में सक्षम बनाता है जिसका उपयोग सॉफ़्टवेयर वितरण स्ट्रीम में पेश की जा रही महत्वपूर्ण और उच्च प्राथमिकता वाली खामियों को खोजने और रोकने के लिए किया जा सकता है
2. **कवर की गई आवश्यकताएँ:** टीम इस प्रमुख प्रदर्शन संकेतक की मदद से कम से कम एक Testing द्वारा कवर की गई आवश्यकताओं के प्रतिशत की निगरानी कर सकती है। पूर्ण Testing और आवश्यकता कवरेज की गारंटी देने के लिए, Testing प्रबंधक हर दिन इस KPI पर नज़र रखता है।
3. **अधिकृत Testing:** एक और महत्वपूर्ण KPI, लेखक परीक्षणों की जांच Testing प्रबंधक और Testing इंजीनियर द्वारा की जाती है ताकि उनके व्यापार विश्लेषकों और Testing टीम की Testing डिजाइन गतिविधि का आकलन किया जा सके।
4. **उत्तीर्ण Testing:** टीम एक Testing के भीतर निहित प्रत्येक कॉन्फ़िगरेशन के निष्पादन का ट्रैक रखकर उत्तीर्ण परीक्षणों के अनुपात का आकलन/माप करती है। यह टीम की समझ में सहायता करता है कि Testing प्रक्रिया के दौरान Testing कॉन्फ़िगरेशन कितनी अच्छी तरह खामियों का पता लगाते हैं और उन्हें पकड़ते हैं।
5. **निष्पादित Testing उदाहरण:** टीम इस मुख्य प्रदर्शन संकेतक का उपयोग यह दिखाने के लिए करती है कि Testing सेट में कुल उदाहरणों का अनुपात वास्तव में निष्पादित किया गया है। यह Testing निष्पादन योजना की गति से जुड़ा हुआ है। हालाँकि, यह KPI निर्माण गुणवत्ता के बारे में जानकारी प्रदान नहीं करता है।
6. **Testing निष्पादित:** Test Cases का चयन करने के बाद, टीम विभिन्न Testing निष्पादन विधियों को देखती है, जिसमें मैनुअल, स्वचालित आदि शामिल हैं। बस Testing उदाहरण निष्पादित की तरह, यह भी एक वेग KPI है.
7. **प्रति दिन हल किए गए डिफेक्ट:** इस KPI की निगरानी करके, Testing प्रबंधक प्रत्येक दिन तय किए गए डिफेक्ट्स की संख्या के साथ-साथ इन समस्याओं के समाधान के लिए टीम के प्रयासों पर नज़र रख सकता है। इसके अतिरिक्त, यह उन्हें प्रोजेक्ट के विकास और Testing कार्यों का पालन करने में सक्षम बनाता है।

8. **प्रत्यक्ष कवरेज:** यह KPI मानव या स्वचालित सुविधा या घटक कवरेज का समर्थन करता है और गारंटी देता है कि सभी सुविधाओं और उनकी कार्यात्मकताओं का संपूर्ण मूल्यांकन किया जाता है। एक घटक को अधूरा माना जाएगा यदि किसी दिए गए स्प्रिंट के दौरान इसका Testing नहीं किया जाता है और इसे तब तक स्थानांतरित नहीं किया जाएगा जब तक कि यह न हो।
9. **गंभीर और बच निकले समस्याओं का प्रतिशत:** सॉफ़्टवेयर Tester को इस महत्वपूर्ण KPI पर पूरा ध्यान देना चाहिए, जो महत्वपूर्ण और बच गए डिफेक्ट्स के प्रतिशत को मापता है। यह गारंटी देता है कि टीम के Testing प्रयास उत्पाद की सबसे महत्वपूर्ण समस्याओं और खामियों को ठीक करने पर केंद्रित हैं, जो बदले में उत्पाद और समग्र Testing प्रक्रिया दोनों के कैलिबर को सुनिश्चित करने में सहायता करता है।
10. **Testing करने का समय:** इस प्रमुख प्रदर्शन संकेतक का लक्ष्य सॉफ़्टवेयर Testing टीम को इस बारे में जानकारी प्रदान करना है कि "Testing" चरण से "पूर्ण" चरण तक जाने में कितना समय लगता है। यह Testing की जा रही सुविधा की जटिलता को समझने के साथ-साथ Tester की प्रभावशीलता और दक्षता की गणना करने में मदद करता है।
11. **डिफेक्ट समाधान समय:** इस मीट्रिक का उपयोग यह आकलन करने के लिए किया जाता है कि सॉफ़्टवेयर डिफेक्ट्स की पहचान करने और मरम्मत का Testing और पुष्टि करने के लिए एक टीम को कितना समय लगता है। इसके अलावा, यह संकल्प समय का ट्रैक रखते हुए Tester की जिम्मेदारी और उनके बग के स्वामित्व को मापता है और योग्य बनाता है। दूसरे शब्दों में, यह KPI डिफेक्ट्स को ट्रैक करने से लेकर यह सुनिश्चित करने तक सब कुछ गारंटी देता है कि समस्या को तुरंत बंद करने के उद्देश्य से उन्हें ठीक किया गया है।
12. **सफल स्प्रिंट गणना अनुपात:** हालांकि एक सॉफ्टवेयर Testing मीट्रिक, एक बार सभी सफल स्प्रिंट आँकड़े एकत्र हो जाने के बाद, सॉफ़्टवेयर Tester इसे KPI के रूप में उपयोग करते हैं। वे सफल होने वाले स्प्रिंट के अनुपात की गणना करने के लिए निम्नलिखित गणना का उपयोग कर सकते हैं: सफल स्प्रिंट गणना अनुपात: (सफल स्प्रिंट / स्प्रिंट की कुल संख्या) x 100।
13. **गुणवत्ता अनुपात:** गुणवत्ता अनुपात, जो एक सॉफ्टवेयर Testing मीट्रिक और एक KPI दोनों के रूप में कार्यरत है, सॉफ्टवेयर Tester द्वारा किए गए सभी परीक्षणों की उत्तीर्ण या असफल दरों पर आधारित है। इसकी गणना निम्न सूत्र का उपयोग करके की जाती है: गुणवत्ता अनुपात = (सफल Test Cases/कुल Test Cases) x 100।
14. टेस्ट केस की गुणवत्ता एक सॉफ्टवेयर Testing आँकड़ा और KPI है जो स्थापित मानकों के अनुसार लिखित Test Cases का आकलन और रेटिंग करने में सहायता करता है। उच्च-गुणवत्ता वाले Test Cases परिदृश्य बनाकर या नमूनाकरण की सहायता से, यह सुनिश्चित करता है कि सभी Test Cases का मूल्यांकन किया जाए।

इसके अलावा, Test Cases की गुणवत्ता सुनिश्चित करने के लिए, टीम द्वारा कुछ कारकों पर विचार किया जाना चाहिए, जैसे:

- त्रुटियों और खामियों के लिए उनकी समीक्षा की जानी चाहिए।
- पूर्ण Testing और आवश्यकता कवरेज की आवश्यकता है।
- जिन क्षेत्रों में खामियां हैं aFFECT को इंगित और निर्दिष्ट किया जाना चाहिए।
- Testing डेटा सटीक होना चाहिए और व्यापक रूप से सभी संभावित परिदृश्यों का प्रतिनिधित्व करना चाहिए।
- सफलता और विफलता दोनों स्थितियों को शामिल किया जाना चाहिए।
- अपेक्षित परिणाम सटीक और समझने योग्य तरीके से बताए जाने चाहिए।

- **डिफेक्ट समाधान सफलता अनुपात:** सॉफ्टवेयर Testing टीम यह निर्धारित कर सकती है कि इस KPI की गणना करके कितने बग बंद किए गए हैं और फिर से खोले गए हैं। %100 रिज़ॉल्यूशन सफलता प्राप्त होती है यदि कोई भी डिफेक्ट फिर से नहीं खोला जाता है। डिफेक्ट समाधान सफलता अनुपात का मूल्यांकन करने के लिए निम्नलिखित सूत्र का उपयोग किया जाता है: डिफेक्ट समाधान सफलता अनुपात = = [(Total Number of Resolved Defects) - (Total Number of Reopened Defects) / (Total Number of Resolved Defects)] / (Total Number of Resolved Defects) x 100
- **प्रक्रिया पालन और सुधार:** इस KPI का उपयोग सॉफ़्टवेयर Testing टीम के प्रयासों को पुरस्कृत करने के लिए किया जा सकता है यदि वे किसी भी अवधारणा या सुधार के साथ आते हैं जो Testing प्रक्रिया को समझने में आसान, अधिक Agile और अधिक सटीक बनाते हैं।

ये सॉफ्टवेयर विकास जीवनचक्र में एक महत्वपूर्ण भूमिका निभाते हैं, उत्पाद की गुणवत्ता को मान्य करने से लेकर Tester द्वारा किए गए कई परीक्षणों की सटीकता की गारंटी देने तक। नतीजतन, आप अपने Testing प्रयासों की दक्षता और सटीकता में सुधार कर सकते हैं और इन सॉफ़्टवेयर Testing मैट्रिक्स और प्रदर्शन संकेतकों को लागू और उपयोग करके उल्लेखनीय गुणवत्ता प्राप्त कर सकते हैं।

6.1.4 सॉफ्टवेयर प्रोजेक्ट प्रबंधन में जोखिम

- अधिकांश सॉफ्टवेयर विकास प्रोजेक्ट्स के लिए पांच प्राथमिक जोखिम प्रभाव क्षेत्रों की पहचान की जा सकती है:
  - उभरती, अपरीक्षित प्रौद्योगिकियां
  - उपयोगकर्ता और कार्यात्मक आवश्यकताएं
  - सिस्टम और एप्लिकेशन आर्किटेक्चर
  - प्रदर्शन
  - संगठनात्मक
- उभरती हुई, अपरीक्षित प्रौद्योगिकियां. नई तकनीकों का उपयोग अधिकांश सॉफ्टवेयर प्रोजेक्ट्स के लिए एक आवश्यकता है। संभावना है कि प्रौद्योगिकी जोखिम व्यावहारिक रूप से किसी भी महत्वपूर्ण सॉफ्टवेयर इंजीनियरिंग में भौतिक हो जाएगा eउपकरण, तकनीकों, प्रोटोकॉल, मानकों और विकास प्रणालियों में निरंतर परिवर्तन के कारण बढ़ जाती है। प्रशिक्षण और ज्ञान महत्वपूर्ण हैं, और नई तकनीक के अनुचित एप्लीकेशन के परिणामस्वरूप अक्सर प्रोजेक्ट विफलता होती है।
- उपयोगकर्ताओं और कार्यक्षमता के लिए विनिर्देश। सॉफ्टवेयर सिस्टम द्वारा प्रदान की जाने वाली सुविधाओं, कार्यों और सेवा के स्तर के संबंध में सभी उपयोगकर्ता की जरूरतों को सॉफ्टवेयर आवश्यकताओं द्वारा कैप्चर किया जाता है। आवश्यकताओं को परिभाषित करने की प्रक्रिया बहुत बार खींची जाती है, थकाऊ और जटिल होती है। इसके अतिरिक्त, आवश्यकताएँ आमतौर पर एकीकरण, प्रोटोटाइप और खोज गतिविधियों के परिणामस्वरूप बदल जाती हैं। उपयोगकर्ता आवश्यकताओं में संशोधन कार्यात्मक आवश्यकताओं में परिवर्तन में अनुवाद नहीं कर सकते हैं, और मौलिक आवश्यकताओं में परिवर्तन संभवतः पूरे प्रोजेक्ट में फैल जाएंगे। इन असफलताओं के परिणामस्वरूप अक्सर खराब विचार वाले सॉफ़्टवेयर विकास प्रोजेक्ट की एक या अधिक गंभीर विफलताएँ होती हैं।

- ऍप्लिकेशन्स और प्रणालियों के लिए वास्तुकला। एक मंच, घटक, या architecture बुरी तरह से गलत हो सकती है यदि इसे गलत तरीके से विकसित किया गया है। यह महत्वपूर्ण है कि टीम में ऐसे विशेषज्ञ शामिल हों जो architecture से परिचित हों और बुद्धिमान डिजाइन निर्णय लेने की क्षमता रखते हों, ठीक वैसे ही जैसे तकनीकी जोखिमों के साथ है।
- प्रदर्शन. किसी भी जोखिम प्रबंधन Strategies को भागीदारों और उपयोगकर्ताओं से प्रदर्शन अपेक्षाओं को ध्यान में रखना चाहिए। यह सुनिश्चित करने के लिए कि कार्य उत्पाद सही दिशा में जा रहे हैं, पूरे प्रोजेक्ट में बेंचमार्क और थ्रेशोल्ड Testing को ध्यान में रखा जाना चाहिए।
- संगठनात्मक. प्रोजेक्ट प्रबंधन को प्रोजेक्ट के प्रभावी निष्पादन के लिए योजना बनानी चाहिए और विकास टीम और ग्राहकों की मांगों के बीच संतुलन बनाना चाहिए। प्रोजेक्ट प्रबंधन को प्रोजेक्ट के प्रभावी निष्पादन के लिए योजना बनानी चाहिए और विकास टीम और ग्राहकों की मांगों के बीच संतुलन बनाना चाहिए। बेशक, पर्याप्त स्टाफिंग में कौशल सेट के साथ टीम के सदस्यों को चुनना शामिल है जो प्रोजेक्ट के साथ एक अच्छा मैच हैं।

6.1.5 सॉफ्टवेयर प्रोजेक्ट में जोखिम

"जोखिम अप्रत्याशित भविष्य की घटनाएं हैं जिनमें नुकसान होने की उच्च संभावना है। सॉफ्टवेयर जोखिमों का प्रभावी ढंग से विश्लेषण किया जा सकता है, जो जोखिम विश्लेषण में सहायता करेगा और भविष्य के लिए कार्य योजना बनाने में सहायता करेगा। हम इस लेख में विभिन्न जोखिम प्रकारों को समझेंगे। सॉफ्टवेयर विकास में जोखिम-आधारित Testing का उपयोग किया जाता है।

यदि वे सभी संभावित इनपुट के साथ सॉफ़्टवेयर का Testing नहीं करते हैं तो कोई उचित मात्रा में जोखिम उठा सकता है क्योंकि एक मौका है कि वे कुछ ऐसे इनपुट को छोड़ देंगे जो सही हैं। इस बिंदु पर विफल होने का जोखिम चलता है, जिसके परिणामस्वरूप मौद्रिक हानि, सुरक्षा की हानि, या यहां तक कि जीवन की हानि भी हो सकती है। इसके परिणामस्वरूप एक सॉफ्टवेयर Tester अत्यधिक दबाव में है। यह माना जाता है कि सॉफ्टवेयर Testing अभ्यास का एक जोखिम-आधारित शासन है, जहां कोई यह पा सकता है कि:

- अत्यधिक Testing विकास की कीमत बढ़ा सकता है।
- अपर्याप्त Testing के कारण विकसित सॉफ़्टवेयर विफल होने पर एक संगठन को महत्वपूर्ण लागत लग सकती है।
- मिस्ड बग, ओवरटेस्टिंग और अंडरटेस्टिंग के कारण Testing की कुल लागत अधिक है।

विकास जोखिम: विकास जोखिम किसी भी प्रोजेक्ट में निहित हैं। उनकी व्यापकता को समझने के लिए, और उन्हें कैसे नियंत्रित किया जा सकता है, हमें पहले अवधारणा को परिभाषित करना चाहिए। एक विकास जोखिम "एक विकास कार्य या पर्यावरण की एक स्थिति या संपत्ति है, जिसे अगर नजरअंदाज कर दिया जाता है, तो प्रोजेक्ट की विफलता की संभावना बढ़ जाएगी"।

- **तकनीकी अंतराल:** विकास अनुबंध की मांगों को पूरा करने के लिए पर्याप्त और पर्याप्त पेशेवर ज्ञान और अनुभव का अभाव।
- **कर्मचारियों की कमी:** पेशेवर कर्मचारियों की अप्रत्याशित कमी।
- **संगठनात्मक तत्वों की अन्योन्याश्रय:** संभावना है कि विशेष हार्डवेयर या सॉफ्टवेयर उपठेकेदारों के आपूर्तिकर्ता, उदाहरण के लिए, समय पर अपने दायित्वों को पूरा नहीं करेंगे।

जोखिम प्रबंधन प्रक्रिया में निम्नलिखित गतिविधियां शामिल हैं:

जोखिम की पहचान, जोखिम मूल्यांकन, जोखिम प्रबंधन कार्यों (RMA) की योजना, RMAS का कार्यान्वयन, और RMAS की निगरानी। सॉफ्टवेयर RMAS विकास योजना में शामिल हैं।

साहित्य में संभावित सॉफ्टवेयर विकास जोखिमों ("सॉफ़्टवेयर जोखिम आइटम" या SRI) की कई सूचियों का उल्लेख किया गया है। रोपोनेन और लियटिनन (2000) ने सॉफ्टवेयर जोखिम वस्तुओं को निम्नलिखित छह वर्गों में वर्गीकृत किया है:

- शेड्यूलिंग और समय जोखिम
- सिस्टम कार्यक्षमता जोखिम
- उप-अनुबंध जोखिम
- आवश्यकता प्रबंधन जोखिम
- संसाधन उपयोग और प्रदर्शन जोखिम
- कार्मिक प्रबंधन जोखिम।

Sr. No.	Software Risk	No.	Software Risk Item	Description
1	Personnel	1	कार्मिक की कमी	प्रबंधन जोखिमों का अभाव और कारोबार
2	Scheduling and timing	2	अवास्तविक कार्यक्रम	गलत तरीके से अनुमानित (बहुत कम) विकास का समय और बजट
3	System functionality	3	गलत सॉफ्टवेयर फ़ंक्शन विकसित करना	सॉफ्टवेयर कार्यों का विकास जिनकी आवश्यकता नहीं है या गलत तरीके से निर्दिष्ट हैं
		4	गलत विकसित करना प्रयोक्ता इंटरफ़ेस	अपर्याप्त या कठिन उपयोगकर्ता इंटरफ़ेस (GUI)
4	Requirement management	5	Gold plating	पेशेवर हितों, गर्व या उपयोगकर्ता मांगों के कारण अनावश्यक सुविधाओं ("whistles and bells ") का जोड़
		6	आवश्यकता परिवर्तनों की सतत धारा	सिस्टम सुविधाऐं फ़ंक्शंस में अनियंत्रित और अप्रत्याशित परिवर्तन और
5		7	बाहरी रूप से कमी सुसज्जित घटक	बाहरी रूप से वितरित की खराब गुणवत्ता सिस्टम घटक
		8	बाहरी रूप से किए गए कार्यों में कमी	बाहरी रूप से कार्य किए खराब गुणवत्ता या अप्रत्याशित उपलब्धि
6	Resource usage and performance	9	वास्तविक प्रदर्शन की कमी	खराब सिस्टम प्रदर्शन
		10	कंप्यूटर विज्ञान क्षमताओं पर दबाव	कारण प्रणाली को लागू करने में असमर्थता तकनीकी समाधान और/या कंप्यूटिंग शक्ति की कमी के लिए

तालिका 6.1.1 शीर्ष 10 सॉफ्टवेयर जोखिम आइटम

6.1.6 प्रदर्शन Testing कठिनाइयों और उन पर

1. **गलत प्रदर्शन Testing उपकरणों का चयन**

   यह एक लगातार समस्या है, और अक्सर सबसे अच्छा प्रदर्शन Testing उपकरण नहीं चुना जाता है। उपकरण का चुनाव कई Variables से प्रभावित होता है, जिसमें एप्लिकेशन संचार प्रोटोकॉल, एप्लिकेशन टेक्नोलॉजी स्टैक, प्रदर्शन Tester का कौशल स्तर और टूल का लाइसेंस शुल्क शामिल है।

   गलत प्रदर्शन उपकरण के चयन के परिणामस्वरूप Testing स्क्रिप्ट को चलाने की कोशिश में Testing समय खो सकता है, इसलिए यह महत्वपूर्ण है कि यह Testing किए जा रहे एप्लिकेशन के नियंत्रणों की पहचान कर सके।

   Solution:

   Testing प्रक्रिया की सफलता की गारंटी के लिए सर्वोत्तम प्रदर्शन Testing उपकरण का चयन करने से पहले Testing के तहत एप्लीकेशन (AUT) और संबंधित लाइसेंसिंग लागतों का QA प्रबंधक और QA टीम द्वारा सावधानीपूर्वक मूल्यांकन किया जाना चाहिए।

2. **Third-Party प्रदर्शन-3rd Party एप्लीकेशन निर्भरता**

   वेब ऍप्लिकेशन्स का प्रदर्शन अब Third-Party वेब घटकों से काफी प्रभावित है। आपका सिस्टम पूरी तरह से खराब हो सकता है यदि यह किसी Third Party के डेटा पर निर्भर है जो अनुपलब्ध हो जाता है। Third Party के सबपर प्रदर्शन के परिणामस्वरूप आपके सिस्टम का प्रदर्शन प्रभावित हो सकता है।

3. **आउटसोर्सिंग चुनौतियां**

   असफल सॉफ्टवेयर विकास आउटसोर्सिंग प्रोजेक्ट्स के प्राथमिक कारणों में प्रोजेक्ट की आवश्यकताओं और दायरे को अपर्याप्त रूप से परिभाषित करना, गलत विक्रेताओं और विकास पद्धतियों का चयन करना, और विकास प्रक्रिया के सभी चरणों में विभिन्न संगठनात्मक स्तरों से हितधारकों और अंतिम उपयोगकर्ताओं को शामिल करने में विफल होना शामिल है। अपर्याप्त माइलस्टोन और प्रगति ट्रैकिंग, उपयुक्त और सक्रिय आपूर्तिकर्ता-ग्राहक संचार की कमी, और कार्य दल की संगठनात्मक संरचना की एक घटिया परिभाषा अतिरिक्त कारक हैं (Selleo, 2016 a)।

   Solution:

   इसलिए, इस शोध का लक्ष्य इस अंतर को बंद करने के लिए आउटसोर्सिंग प्रक्रियाओं के आयोजन और प्रबंधन के लिए चरणों का प्रवाह विकसित करना है। उपरोक्त प्रक्रियाओं और सर्वोत्तम प्रथाओं के विश्लेषण और वर्गीकरण ने आउटसोर्सिंग प्रबंधन के लिए निम्नलिखित चरणों का विकास किया: योजना, आपूर्तिकर्ता चयन, संबंध प्रबंधन, विकास और पुनर्विचार।

4. **प्रदर्शन Testing परिणामों का अनुचित विश्लेषण:**

   यह कई Tester के लिए एक बड़ी चुनौती है क्योंकि प्रदर्शन Testing परिणामों के गहन विश्लेषण के लिए पर्याप्त प्रणाली और एप्लीकेशन ज्ञान की आवश्यकता होती है।

Solution:

Testing प्रक्रिया एक अनुभवी प्रदर्शन Tester द्वारा की जानी चाहिए जो परिदृश्यों का मूल्यांकन कर सकता है, परीक्षणों में लगातार सुधार कर सकता है, और उन्हें सुसंगत बनाने के लिए Testing जोड़ता रहता है। एप्लिकेशन आर्किटेक्चर प्रदर्शन Tester के लिए अच्छी तरह से जाना जाना चाहिए।

प्रदर्शन Tester को नेटवर्किंग, डेटा संरचनाओं, OSआई मॉडल, क्लाइंट-साइड और सर्वर-साइड प्रदर्शन, साथ ही OS अवधारणाओं, वेब आर्किटेक्चर और OS अवधारणाओं का ज्ञान भी होना चाहिए। इन प्रदर्शन विशेषज्ञों के पास Testing परिणामों के विश्लेषण तक त्वरित पहुंच होगी।

6.1.7 Third Party QA Testing

सॉफ्टवेयर विकास में गुणवत्ता आश्वासन और सॉफ्टवेयर Testing का महत्व कोड के बराबर है। फिर भी, सॉफ़्टवेयर गुणवत्ता आश्वासन (QA) प्रक्रिया आमतौर पर प्रोजेक्ट के रिलीज़ होने से ठीक पहले अंतिम रूप से की जाती है।

इस तथ्य के बावजूद कि QA Testing प्रक्रिया अक्सर थकाऊ और समय लेने वाली होती है, यह संपूर्ण Testing की सफलता सुनिश्चित करने में महत्वपूर्ण भूमिका निभातीहै ई फोर्ट, इसलिए विश्वसनीय सॉफ्टवेयर और सर्वोत्तम उत्पादों के वितरण की सुविधा प्रदान करता है। यदि अच्छी तरह से योजनाबद्ध और प्रदर्शन किया जाता है, तो गुणवत्ता आश्वासन प्रक्रिया न केवल उत्पाद की सफलता और उच्च गुणवत्ता सुनिश्चित कर सकती है, बल्कि व्यवसाय संचालन की निरंतरता भी सुनिश्चित कर सकती है।

Third-Party QA Testing से अनुकूल परिणाम सुनिश्चित करने के लिए सबसे चतुर तरीकों में से एक कुछ Testing प्रबंधन सर्वोत्तम प्रथाओं को नियोजित करना है।

Third-Party QA Testing के लिए सर्वोत्तम तरीके

उच्च गुणवत्ता वाले उत्पाद का उत्पादन करने का एकमात्र तरीका एक कुशल QA प्रक्रिया को लागू करना है। यहां कुछ Third-Party QA Testing सर्वोत्तम अभ्यास दिए गए हैं जो आपको गुणवत्ता आश्वासन में उच्च सफलता प्राप्त करने में मदद करेंगे:

1. **Testing से पहले, एक मजबूत उत्पाद और Third-Party एप्लिकेशन विशेषज्ञता विकसित करें**

   Testing के दौरान, Third-Party एप्लीकेशन संबंधी चिंताएँ अक्सर वास्तविक उत्पाद डिफेक्ट्स के साथ भ्रमित होती हैं। यह Testing और आपके समय-समय पर बाजार पर पर्याप्त प्रभाव डाल सकता है।

   Testing परिदृश्य आयोजित करने से पहले, यह महत्वपूर्ण है कि QA टीम को सभी डोमेन (एपीआई, OS, मोबाइल, आदि) की व्यापक समझ हो।

2. **Testing और गुणवत्ता आश्वासन प्रक्रिया की गहराई से योजना बनाएं**

   सॉफ्टवेयर Testing प्रक्रिया की योजना, परिभाषित और दस्तावेजीकरण समग्र उत्पाद की गुणवत्ता बढ़ाने के लिए सबसे प्रभावी तरीकों में से एक है।

अच्छा प्रलेखन एक सॉफ्टवेयर टीम के भीतर अच्छे संचार को बढ़ावा देने का एक कुशल साधन है और एक प्रोजेक्ट के लिए गुणवत्ता और Test Plans के रखरखाव को सक्षम बनाता है। ग्राहकों या हितधारकों के साथ साझा किए जा सकने वाले सबसे महत्वपूर्ण कागजात में से हैं:

- Test Plan
- समर्थित सिस्टम
- लॉग Test Cases और चेकलिस्ट बदलें

3. गुणवत्ता बढ़ाने पर ध्यान दें

किसी भी गुणवत्ता आश्वासन प्रक्रिया का प्राथमिक उद्देश्य सभी हितधारकों को आश्वासन प्रदान करना है कि आपके सॉफ़्टवेयर के निर्माण में नियोजित प्रक्रियाओं और गतिविधियों को उत्पाद की उच्च गुणवत्ता को बनाए रखने के लिए सावधानीपूर्वक योजनाबद्ध किया गया है।

इसलिए, QA Testing को अंतिम उत्पादों की गुणवत्ता को अनुकूलित करने के लिए सॉफ्टवेयर विकास प्रक्रिया को बढ़ाने पर ध्यान केंद्रित करना चाहिए।

4. कुशल पद्धतियों का उपयोग करें

सॉफ्टवेयर के जीवनकाल की पूरी लागत को कम करने के लिए एक कुशल QA Testing तकनीक आवश्यक हो सकती है। यह आश्वासन देता है कि सॉफ्टवेयर सभी आवश्यकताओं और मानकों का अनुपालन करता है, जो जीवन-महत्वपूर्ण वस्तुओं के निर्माण के लिए महत्वपूर्ण हैं।

इसके अलावा, उत्पाद के विकास में नियोजित की जाने वाली उपयुक्त प्रक्रियाओं की पहचान करना और यह सत्यापित करना आवश्यक है कि इन प्रक्रियाओं का ठीक और बिना किसी बदलाव के पालन किया जाता है।

5. गुणवत्ता आश्वासन से ऊपर जोखिम प्रबंधन वरीयता दें

आम गलत धारणा के विपरीत, गुणवत्ता आश्वासन Testing की तुलना में काफी व्यापक शब्द है। अपने उत्पाद की गुणवत्ता सुनिश्चित करने के लिए, प्रक्रियाओं, गतिविधियों और जोखिम प्रबंधन की एक विशाल सरणी की आवश्यकता होती है। वास्तव में, गुणवत्ता जांच के साथ मिलकर जोखिम प्रबंधन ई प्रभावी गुणवत्ता आश्वासन के स्तंभों में से एक है।

6. आवश्यकताओं के आधार पर Testing विधियों को नियोजित करें

सुरक्षा, मोबाइल, फुर्तीला, बड़ा डेटा और एनालिटिक्स जैसे विभिन्न प्रकार के Testing करने में सक्षम होने के लिए, आपको विभिन्न Test Startegyयों को नियोजित करना होगा जो विशेष प्रकार के Testing प्रबंधन Framework के अनुरूप हैं। सीधे शब्दों में व्यक्त किया जाए, तो आवश्यकता-आधारित Test Startegy का पालन करना आवश्यक है जो सिस्टम/एप्लिकेशन सत्यापन और Testing प्रयासों के अनुमान में सहायता कर सकती है।

7. **जल्दी और बार-बार Testing करें**

   गुणवत्ता आश्वासन Testing का उद्देश्य समस्याओं को जल्दी उजागर करने के लिए प्रोजेक्ट के जीवनचक्र के प्रत्येक चरण में Testing को शामिल करना है। Testing के प्रत्येक स्तर के भीतर, QA इंजीनियर नई जोड़ी गई सुविधाओं की व्यवहार्यता निर्धारित करने के लिए उत्पाद का Testing और पुन: Testing करते हैं। यह उन्हें खोजने और इस प्रक्रिया के दौरान प्रस्तुत किया गया हो सकता है जो किसी भी समस्या का पता लगाने के लिए सक्षम करता है।

   प्रारंभिक और लगातार Testing के लिए निरंतर प्रक्रिया निगरानी की आवश्यकता होती है ताकि यह सुनिश्चित किया जा सके कि विकास प्रक्रिया के दौरान सभी सहमत मानकों और प्रक्रियाओं का पालन किया जा रहा है।

8. **मशीनी परिचालन**

   दोहराए जाने वाले Test Cases को स्वचालित करने के लिए उपन्यास दृष्टिकोण खोजना हमेशा एक अच्छा विचार है। यह किसी भी तरह से सब कुछ स्वचालित करने के लिए बहुत अधिक धन का निवेश करने का मतलब नहीं है।

   इष्टतम समाधान एक Testing स्वचालन सेवा प्रदाता की सेवाओं को प्राप्त करना और स्वचालित परीक्षणों को कुशलतापूर्वक प्रबंधित करने में सक्षम सर्वर स्थापित करना है।

9. **पहली बार पूर्णता प्राप्त करने के लिए अभ्यास करें**

   किसी भी डिलीवरी प्रोजेक्ट में गलतियाँ अपरिहार्य हैं, भले ही विनिर्देश कितने भी सख्त क्यों न हों। सॉफ्टवेयर Testing में, इसे पहली बार सही करने का तात्पर्य है कि ऐसी सभी संभावित खामियों को मिटाया जाना चाहिए।

   उद्देश्य एक निश्चित डिग्री की सटीकता के साथ परिभाषित करना है कि आप प्रोजेक्ट में क्या हासिल करने का प्रयास कर रहे हैं, एक समाधान डिजाइन करें जो स्पष्ट रूप से आपके वांछित परिणाम को दर्शाता है, और यह सुनिश्चित करता है कि किसी भी बदलाव को पहचाना और स्वीकार किया जाए।

सारांश

- उद्योग की सर्वोत्तम प्रथाओं के अनुसार अपने अपडेट किए गए उत्पाद का Testing करना इसकी उचित सुरक्षा और निर्दोष प्रदर्शन सुनिश्चित करने के लिए पर्याप्त नहीं हो सकता है। कोड के कुछ अनुभागों के लिए एक अलग Test Startegy, एक गहन विश्लेषण या डबल-चेकिंग आवश्यक हो सकती है।
- प्रभाव विश्लेषण नामक एक सॉफ्टवेयर Testing तकनीक Testing उत्पाद में किए गए किसी भी परिवर्तन से जुड़े सभी जोखिमों की पहचान करने में मदद करती है।
- एक डिफेक्ट गंभीरता बिंदु प्रणाली और सिस्टम Testing बनाम UAT में कुल डिफेक्ट्स की गणना Testing दक्षता संकेतक की गणना करने के लिए की जाती है।
- सॉफ्टवेयर Testing मेट्रिक्स, जिसे सॉफ्टवेयर Testing माप भी कहा जाता है, एक सॉफ्टवेयर प्रक्रिया की चौड़ाई, गहराई, ऊंचाई और क्षमता दिखाता है और इसकी प्रभावशीलता और दक्षता को तुरंत बढ़ाने का प्रयास करता है।
- संगठन और Tester दोनों मापा जा सकने वाला डेटा एकत्रित करने के लिए प्रदर्शन संकेतक या KPI का उपयोग करते हैं.
- हम सॉफ्टवेयर विकास प्रोजेक्ट्स के बहुमत के लिए पांच मुख्य जोखिम प्रभाव क्षेत्रों को निम्नानुसार वर्गीकृत कर सकते हैं: नई, अप्रमाणित प्रौद्योगिकियां, उपयोगकर्ता और कार्यात्मक आवश्यकताएं, एप्लीकेशन और सिस्टम architecture, प्रदर्शन और संगठनात्मक।

- जोखिम भविष्य में नुकसान और आसान घटना की संभावना के साथ अज्ञात घटनाएं हैं। सॉफ्टवेयर जोखिमों का प्रभावी ढंग से विश्लेषण किया जा सकता है, जो जोखिम विश्लेषण में सहायता करेगा और भविष्य के लिए कार्य योजना बनाने में सहायता करेगा।
- जोखिम मूल्यांकन, जोखिम की पहचान, जोखिम प्रबंधन कार्यों (RMA) की योजना, RMA का निष्पादन और RMA की निगरानी जोखिम प्रबंधन प्रक्रिया के सभी घटक हैं। सॉफ्टवेयर RMA विकास योजना में शामिल हैं
- Third Party के सबपर प्रदर्शन के परिणामस्वरूप आपके सिस्टम का प्रदर्शन प्रभावित हो सकता है।

एक्टिविटी

चर्चा सत्र

- प्रशिक्षक प्रशिक्षुओं से निम्नलिखित प्रश्न पूछता है: "प्रमुख प्रदर्शन आश्वासन Indu cater क्या हैं?"
- इस गतिविधि में उम्मीदवार को स्वयंसेवक और बोलने के लिए अपने हाथ उठाने की जरूरत है।
- सत्र के दौरान बोली जाने वाली भाषा कक्षा में अधिकांश प्रशिक्षुओं द्वारा जानी जानी चाहिए।
- जबकि सत्र चल रहा है, ट्रेनर को एक मार्कर की मदद से व्हाइटबोर्ड पर महत्वपूर्ण बिंदुओं को संक्षेप में लिखना चाहिए।
- सबसे अच्छे उत्तर की सराहना ट्रेनर द्वारा पूरी कक्षा के सामने की जाएगी।

चर्चा सत्र

- प्रशिक्षक पूछताछ करेगा कि क्या प्रशिक्षुओं ने अध्याय को समझा है।
- इस गतिविधि में, प्रशिक्षक प्रशिक्षु से विषय से संबंधित कुछ प्रश्न पूछेगा।
- प्रशिक्षु विषय की अपनी समझ के अनुसार प्रश्नों का उत्तर देंगे।
- यदि प्रशिक्षुओं के मन में प्रश्न और भ्रम हैं; वे ट्रेनर के सामने उन लोगों को सामने रख सकते हैं।
- प्रशिक्षक यह सुनिश्चित करेगा कि वह प्रशिक्षुओं द्वारा रखे गए प्रश्नों के उचित उत्तर दे सके।

अभ्यास

1. सही उत्तर चुनें:

I. QA क्या है?

a) डिफेक्ट्स की पहचान करने की प्रक्रिया

b) यह एक सुधारात्मक उपकरण है

c) कोई उत्पाद किस हद तक किसी आवश्यकता को पूरा करता है, इसका अनुमान इस मीट्रिक से लगाया जाता है।

d) प्रक्रिया की गुणवत्ता की गारंटी के लिए लागू कोई भी व्यवस्थित प्रक्रिया

II. किसी प्रोजेक्ट की क्षमता बढ़ाने के लिए मुद्दों की पहचान करना और उन्हें संबोधित करना लक्ष्य है

_____

a) सॉफ्टवेयर बग

b) सॉफ्टवेयर जटिलता

c) सॉफ्टवेयर Testing

d) सॉफ्टवेयर विकास

III. संकेतक का एक उदाहरण कौन सा है?

a) परीक्षणों की संख्या

b) कर्मचारियों के घंटों की संख्या

c) वास्तविक बनाम नियोजित कार्य पूर्णता

d) कोड की प्रति हजार लाइनों में डिफेक्ट

IV. Reducing_____ सॉफ्टवेयर आश्वासन का मुख्य उद्देश्य है।

a) Risks

b) Time

c) Quality

5. FTR का पूर्ण रूप क्या है?

a) File Transfer

b) Formal Telephonic Review

c) Formal Technical Review

d) Formal Telegraphic Revie

6. प्रोजेक्ट प्रबंधन को प्रभावित करने वाले कारक हैं.

a) Time

b) Cost

c) Scope

d) All of the above

2. Testing में प्रदर्शन चुनौतियों की सूची बनाएं।

नोट्स

संबंधित वीडियो देखने के लिए क्यूआर कोड को स्कैन करें

https://www.youtube.com/watch?v=I1uT7mhalio

प्रभाव संकेतक का उद्देश्य, Testing में दक्षता संकेतक

Skill India
कौशल भारत-कुशल भारत

सत्यमेव जयते
GOVERNMENT OF INDIA
MINISTRY OF SKILL DEVELOPMENT
& ENTREPRENEURSHIP

N·S·D·C
National
Skill Development
Corporation
Transforming the skill landscape

7. मैनुअल टेस्ट के लिए तकनीकी कौशल

IT - ITeS SSC
nasscom

यूनिट 7.1 - मैनुअल टेस्ट के लिए तकनीकी कौशल

SSC/N1303

प्रमुख सीखने के परिणाम

इस मॉड्यूल के अंत तक, प्रतिभागी निम्न में सक्षम होंगे:

1. 100% Testing कवरेज के साथ Testing चरण के दौरान Testing Suite या मामलों को डिज़ाइन करें।
2. गुणवत्ता आश्वासन को संभालने के लिए स्रोत कोडिंग मानकों, और उपयोगिताओं/उपकरणों के महत्व की जांच करें।

यूनिट 7.1: मैनुअल टेस्ट के लिए तकनीकी कौशल

यूनिट के उद्देश्य

इस यूनिट के अंत तक, प्रतिभागी निम्न में सक्षम होंगे:

1. Testing Suite डिजाइन करने के क्षेत्र में नवीनतम परिवर्तनों, प्रक्रियाओं और प्रथाओं पर चर्चा करें।
2. सॉफ्टवेयर Testing तत्वों के उद्देश्य की जांच करें, जैसे Static Testing, Dynamic Testing, व्हाइट बॉक्स Testing, ब्लैक बॉक्स Testing, आदि।
3. मैन्युअल Testing में सही कार्रवाई निर्धारित करने के लिए विभिन्न मूल्य और डेटा के उपयोग पर चर्चा करें।
4. इनपुट और/या डेटा को सटीक रूप से निकालने के लिए सूचना प्रौद्योगिकी का सही उपयोग लागू करें।
5. Static Testing, Dynamic Testing, व्हाइट बॉक्स Testing, ब्लैक बॉक्स Testing आदि का प्रदर्शन करें।
6. संबंधित कोडिंग भाषा में लिखे गए कोड को समझने के लिए .Net, SQL, Java, Oracle, VB Script, आदि जैसे एप्लिकेशन का उपयोग करें।
7. स्वचालित चेतावनियों और Testing सेवा अनुरोधों की निगरानी, पहुँच और सत्यापन करें.

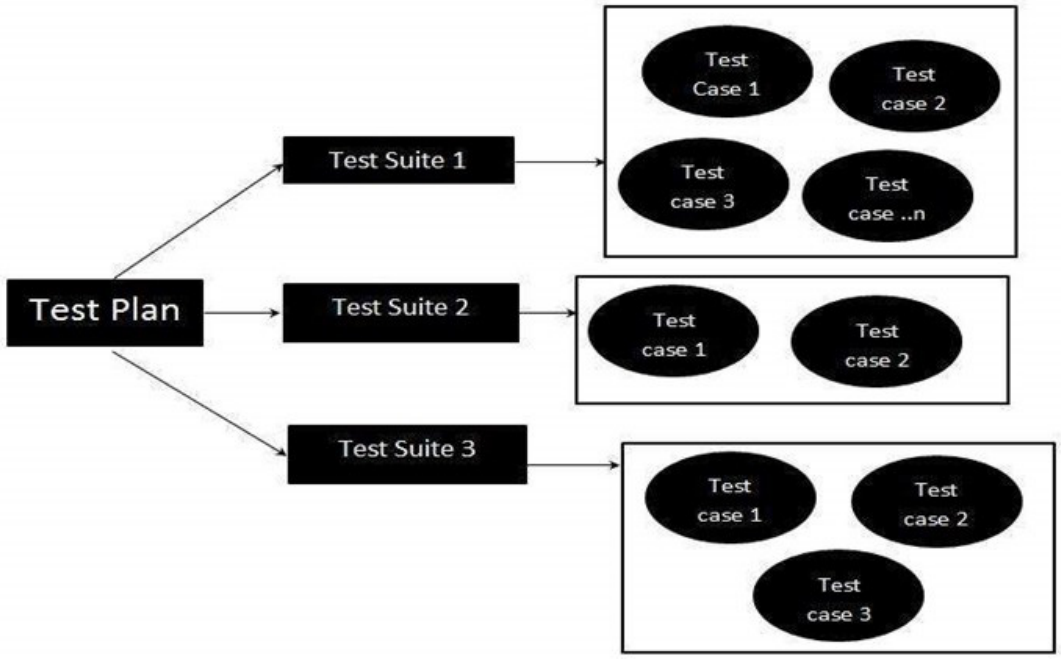
7.1.1 परिचय

सॉफ्टवेयर विकास और सिस्टम वैश्विक अर्थव्यवस्था का एक क्षेत्र है जो तेजी से बढ़ रहा है। कंप्यूटर सॉफ्टवेयर के बिना निष्क्रिय हैं। यदि इस सॉफ्टवेयर और जिस हार्डवेयर पर यह काम करता है, उसका रखरखाव नहीं किया जाता है, तो पूरी सभ्यताएं बिजली और जीवन यापन के लिए अपनी बुनियादी जरूरतों को खो सकती हैं।

7.1.2 Test Suite

Test Suite एक कंटेनर है जिसमें परीक्षणों का एक संग्रह होता है जो Testing निष्पादन की स्थिति को निष्पादित करने और रिपोर्ट करने में Tester की सहायता करता है। यह सक्रिय, प्रगतिशील, या पूर्ण तीन चरणों में से किसी में भी मौजूद हो सकता है। एकाधिक Testing Suite और Test Plans में एक Testing मामला शामिल हो सकता है। एक Test Plan तैयार करने के बाद, किसी भी संख्या में परीक्षणों सहित Testing Suite विकसित किए जाते हैं।

Test Suite चक्र या दायरे के अनुसार उत्पन्न होते हैं। इसमें कार्यात्मक और गैर-कार्यात्मक दोनों Testing शामिल हो सकते हैं।



चित्र 7.1.1 Test Suite

7.1.3 सूचना स्टोरेज और पुनर्प्राप्ति

किसी दस्तावेज़ प्रबंधन सिस्टम (DMS) में दस्तावेज़ सहेजते समय आमतौर पर डिजिटल दस्तावेज़ का उपयोग किया जाता है. अलमारियाँ दाखिल करने के बजाय, हम आज इलेक्ट्रॉनिक स्टोरेज का उपयोग करते हैं। यह समय और कमरे बचाता है। जब कोई दस्तावेज़ संग्रहीत किया जाता है, तो इसे आमतौर पर किसी विशेष प्रकार या वर्गीकरण को निर्दिष्ट करने के लिए मेटाडेटा के एक सेट के साथ चिह्नित किया जाता है। फिर इसे डिजिटल फाइल में रखा जाता है।

अनुरोधकर्ता को आवश्यक कंटेंट से जोड़ने की प्रक्रिया को दस्तावेज़ पुनर्प्राप्ति (ओं) के रूप में जाना जाता है। एक पेशेवर विशेष दस्तावेजों और प्रबंधन प्रणालियों से निपटने के दौरान रिकॉर्ड की खोज और पुनर्प्राप्ति में मदद कर सकता है।

सूचना स्टोरेज और पुनर्प्राप्ति डेटा को व्यवस्थित रूप से इकट्ठा करने और सूचीबद्ध करने की प्रक्रिया है ताकि इसे मांग पर स्थित और दिखाया जा सके। कंप्यूटर और डेटा प्रोसेसिंग तकनीकों ने सरकारी, वाणिज्यिक और शैक्षणिक ऍप्लिकेशन्स के लिए बड़ी मात्रा में डेटा की तेजी से चयनात्मक पुनर्प्राप्ति को सक्षम किया है।

विभिन्न प्रकार की सूचना स्टोरेज-और-पुनर्प्राप्ति प्रणालियाँ हैं।

- दस्तावेज़ पुनर्प्राप्ति प्रणाली पूर्ण दस्तावेज़ रखती है, जिसे अक्सर दस्तावेज़ के शीर्षक या संबंधित कीवर्ड का उपयोग करके पुनर्प्राप्त किया जाता है।
- कुछ प्रणालियों में, दस्तावेज़ पाठ को डेटा के रूप में रखा जाता है। यह दस्तावेज़ में किसी भी शब्द के आधार पर पूर्ण पाठ खोज और पुनर्प्राप्ति को सक्षम करता है। अन्य मामलों में, दस्तावेज़ की एक डिजिटल छवि सहेजी जाती है, आमतौर पर एक बार लिखने वाली ऑप्टिकल डिस्क पर। डेटाबेस सिस्टम असतत क्षेत्रों (जैसे, नाम, पता और टेलीफोन नंबर) में विभाजित असतत रिकॉर्ड के अनुक्रम के रूप में जानकारी संग्रहीत करते हैं; फ़ील्ड की कंटेंट के आधार पर रिकॉर्ड खोजे और पुनर्प्राप्त किए जा सकते हैं (उदाहरण के लिए, वे सभी लोग जिनके पास एक विशेष टेलीफोन क्षेत्र कोड है)। जानकारी कंप्यूटर पर सहेजी जाती है, या तो प्राथमिक स्टोरेज या द्वितीयक स्टोरेज में, आसान पहुंच के लिए।
- संदर्भ पुनर्प्राप्ति प्रणाली वास्तविक दस्तावेजों के विपरीत दस्तावेजों के संदर्भ को बनाए रखती है। एक खोज क्वेरी के जवाब में, ये सिस्टम दस्तावेज़ नाम और अक्सर, उनके वास्तविक स्थान प्रदान करते हैं।

7.1.4 बैकअप और पुनर्स्थापना

इलेक्ट्रॉनिक अभिलेखागार सहित इलेक्ट्रॉनिक दस्तावेजों को संग्रहीत करते समय आवधिक बैकअप किया जाना चाहिए। संग्रहीत इलेक्ट्रॉनिक रिकॉर्ड तक पहुँचने के लिए समान दिशानिर्देश बैकअप (ओं) पर लागू किए जाने चाहिए, भले ही ये बैकअप संग्रहीत रिकॉर्ड के रूप में योग्य न हों और सिस्टम विफलता की स्थिति में इन्हें पुनर्स्थापित करने की आवश्यकता हो सकती है। इन अभिलेखागार की दीर्घकालिक पठनीयता आमतौर पर कोई समस्या नहीं है क्योंकि बैकअप आमतौर पर दीर्घकालिक स्टोरेज या संग्रह के बजाय अल्पकालिक स्टोरेज के लिए अभिप्रेत होते हैं; हालांकि, बैकअप की बहाली को भी नियमित आधार पर जांचना चाहिए। सुविधा की बैकअप और पुनर्स्थापना नीति को अध्ययन के प्रयोगात्मक चरण के दौरान बनाए गए किसी भी डेटा को भी कवर करना चाहिए।

डेटा निष्कर्षण अंतर्दृष्टिपूर्ण व्यावसायिक जानकारी प्राप्त करने या केंद्रीय डेटा वेयरहाउस में स्टोरेज के लिए बाद में प्रसंस्करण और विश्लेषण के लिए विभिन्न डेटा स्रोतों से डेटा प्राप्त करने की प्रक्रिया है। असंरचित, अर्ध-संरचित या संरचित डेटा विभिन्न स्रोतों से एकत्र किया जा सकता है।

डेटा अक्सर संगठनों, लोगों या कंपनियों द्वारा विश्लेषण के लिए BI उपकरणों में उपयोग के लिए निकाला जाता है, किसी रिपॉसिटरी में माइग्रेशन, या बैकअप के रूप में प्रतिकृति।

डेटा निष्कर्षण और जनरेशन को समझना

- ETL (एक्सट्रैक्ट, ट्रांसफॉर्म और लोड) प्रक्रिया में पहला चरण डेटा निष्कर्षण है। आप डेटा को केवल उन गंतव्यों में रूपांतरित और लोड कर सकते हैं जिन्हें आप डेटा को ठीक से निकालने के बाद आगे डेटा विश्लेषण के लिए उपयोग करना चाहते हैं।
- डेटा निष्कर्षण, इसे सीधे शब्दों में कहें, तो डेटा वेयरहाउस वातावरण में इसका उपयोग करने के लिए स्रोत सिस्टम से डेटा प्राप्त करने की प्रक्रिया है। डेटा निष्कर्षण प्रक्रिया को आमतौर पर तीन चरणों में विभाजित किया जा सकता है:
    - परिवर्तनों की पहचान करें: आपको अपने डेटा के किसी भी अपडेट की निगरानी करनी चाहिए। उदाहरण के लिए, एक नई तालिका या कॉलम शामिल किया जा सकता है।

- **निकाले जाने वाले डेटा को निर्दिष्ट करें**: आपको यह तय करना चाहिए कि आपके डेटा के किन हिस्सों को निकालने की आवश्यकता है और फिर उन हिस्सों को निर्दिष्ट करें। डेटा का पूरा सेट पूर्ण निष्कर्षण विधि का उपयोग करके एक ही बार में निकाला जाता है।
- **प्रक्रिया डेटा निष्कर्षण:** इस बिंदु पर, आपने सभी पूर्वापेक्षा चरणों को पूरा कर लिया है और मैन्युअल रूप से बनाई गई स्क्रिप्ट या स्वचालित टूल का उपयोग करके डेटा निकालने के लिए तैयार हैं।

7.1.5 डेटा निष्कर्षण के तरीके

विभिन्न डेटा निष्कर्षण उपकरणों द्वारा बाद में विश्लेषण के लिए डेटा निष्कर्षण प्रक्रिया के दौरान विभिन्न डेटा स्रोतों से डेटा एकत्र किया जाता है।

डेटा स्रोत भौतिक या डिजिटल हो सकते हैं, जिसमें मुद्रित या भौतिक मीडिया जैसे किताबें, समाचार पत्र, चालान, स्प्रेडशीट और डेटाबेस शामिल हैं। विश्लेषण के लिए, कई डेटा निष्कर्षण उपकरण कुछ या सभी स्रोतों का उपयोग करते हैं।

भौतिक स्रोतों से डेटा निकालने की प्रक्रिया आमतौर पर मैन्युअल रूप से किए जाने पर श्रमसाध्य, महंगी और समय लेने वाली होती है, लेकिन ऑप्टिकल कैरेक्टर रिकॉग्निशन (ओसीआर) जैसी आधुनिक तकनीकों ने इस प्रक्रिया को स्वचालित करना संभव बना दिया है।

7.1.6 डेटा निष्कर्षण में उपयोग की जाने वाली डेटा संरचनाओं के प्रकार

डेटा निष्कर्षण विधियों को मुख्य रूप से तार्किक और भौतिक में विभाजित किया जा सकता है। इसके अतिरिक्त, इनमें से कई प्रकार हैं, जो इस प्रकार हैं:

1. **तार्किक डेटा निष्कर्षण**

   सबसे लोकप्रिय डेटा निष्कर्षण तकनीक तार्किक निष्कर्षण है। इसे आगे दो समूहों में विभाजित किया गया है:

   - **पूर्ण निष्कर्षण:** यह प्रक्रिया आमतौर पर प्रारंभिक लोड के दौरान होती है। यहां, सभी डेटा सीधे स्रोत से एक बार में निकाले जाते हैं। चूंकि यह निष्कर्षण स्रोत सिस्टम पर वर्तमान में उपलब्ध सभी डेटा को दर्शाता है, इसलिए सबसे हालिया सफल निष्कर्षण के बाद डेटा स्रोत में परिवर्तनों का ट्रैक रखने की कोई आवश्यकता नहीं है।
   - **वृद्धिशील निष्कर्षण:** डेटा में डेल्टा परिवर्तन इस पद्धति का फोकस हैं। आपको पहले डेटा इंजीनियर के रूप में स्रोत सिस्टम पर जटिल निष्कर्षण तर्क लागू करना होगा और डेटा अपडेट और परिवर्तनों का ट्रैक रखना होगा। अद्यतन डेटा निष्कर्षण टाइमस्टैम्प इस तकनीक का उपयोग करके दर्ज किए जाते हैं।

2. **भौतिक डेटा निष्कर्षण**

पुराने डेटा स्टोरेज सिस्टम से तार्किक निष्कर्षण का उपयोग करके डेटा निष्कर्षण चुनौतीपूर्ण हो सकता है। यह जानकारी केवल भौतिक निष्कर्षण के माध्यम से प्राप्त की जा सकती है। इसे दो श्रेणियों में भी विभाजित किया जा सकता है:

- **ऑनलाइन निष्कर्षण:** आप इस प्रक्रिया का उपयोग करके डेटा स्रोत से सीधे वांछित डेटा वेयरहाउस में डेटा निकाल सकते हैं। इस विधि के कार्य करने के लिए निष्कर्षण उपकरण को सीधे स्रोत प्रणाली से लिंक करना होगा। इसे सीधे स्रोत से जोड़ने के बजाय, आप इसे संक्रमणकालीन प्रणाली से जोड़ सकते हैं, जो स्रोत प्रणाली का एक सटीक डुप्लिकेट है लेकिन अधिक संरचित डेटा के साथ।
- **ऑफ़लाइन निष्कर्षण:** मूल स्रोत से सीधे लिए जाने के बजाय, डेटा को इस पद्धति में इसके बाहर स्पष्ट रूप से मंचित किया जाता है। इस प्रक्रिया में, डेटा या तो संरचित होता है या डेटा निष्कर्षण के लिए दिनचर्या का उपयोग करके संरचित किया जा सकता है। एक फ्लैट फ़ाइल, एक डंप फ़ाइल, या डेटाबेस लेनदेन लॉग से एक दूरस्थ निष्कर्षण फ़ाइल संरचनाओं में से कुछ हैं जो इसे ध्यान में रखते हैं।

  समय किसी भी संगठन के लिए पैसा है। इसलिए, आपको डेटा निष्कर्षण उपकरण का उपयोग करने के बारे में सोचना चाहिए जो आपके वर्कफ़्लोज़ में सुधार कर सकते हैं और आपका समय बचा सकते हैं। जब ठीक से उपयोग किया जाता है, तो डेटा निष्कर्षण उपकरण आपकी टीम का समय बचा सकते हैं और स्टाफ के सदस्यों को उच्च प्राथमिकता वाले कार्यों पर काम करने के लिए मुक्त कर सकते हैं।

7.1.7 डेटा जनरेट करना

यहां आप सीखेंगे कि सर्वोत्तम परिणाम प्राप्त करने के लिए Testing डेटा कैसे उत्पन्न करें, प्रबंधित करें और उपयोग करें।

- **मैनुअल टेस्ट डेटा जनरेशन**

  मैनुअल Testing डेटा generation में आपकी QA टीम की सहायता से, या डेवलपर्स द्वारा स्वतंत्र रूप से Testing डेटा का उत्पादन करना शामिल है। मैन्युअल दृष्टिकोण के साथ, आपको पहले Testing करने के लिए चीजों की एक सूची तैयार करनी होगी और फिर उन चीजों में से प्रत्येक के लिए नमूना डेटा तैयार करना होगा।

  यह Testing डेटा एकत्र करने का सबसे सरल और सबसे प्रत्यक्ष तरीका है। मैनुअल Testing डेटा अक्सर एक प्रोजेक्ट के कार्यान्वयन की शुरुआत में बनाया जाता है और ऐसा इस तरह से किया जाता है जिसमें सभी संभावित इनपुट-आउटपुट संयोजन शामिल होते हैं। यह विधि सबसे अधिक समय लेने वाली है और आमतौर पर एज टेस्ट मामलों और इनपुट/आउटपुट संयोजनों पर लागू होती है जो प्रोजेक्ट-विशिष्ट होते हैं और स्वचालित रूप से उत्पन्न नहीं होते हैं।
- **स्वचालित Testing डेटा जनरेशन**

  स्वचालित Testing डेटा जनरेशन सॉफ्टवेयर टूल की मदद से किया जाता है जो पूरी प्रक्रिया को शुरू से अंत तक स्वचालित करता है। इस पद्धति का मुख्य लाभ विशाल generation की गति है, साथ ही उत्पन्न डेटा की सटीकता भी है। यह मैन्युअल Testing डेटा का उपयोग करने की तुलना में परीक्षणों को विकसित करने, बनाए रखने और निष्पादित करने में लगने वाले समय को कम करने का एक प्रभावी तरीका है, जो मानवीय त्रुटि से ग्रस्त है।
- **सर्वोत्तम Testing परिणाम प्राप्त करने के लिए Testing डेटा का प्रबंधन करना**

  पहला चरण केवल नमूना Testing डेटा बनाना है। प्रतिगमन से रिलीज तक प्रत्येक प्रतिगमन चक्र में प्रत्येक Test Cases को चलाने से एकत्र किए गए Testing डेटा का प्रबंधन करना वास्तविक चुनौती है।

  प्रभावी सॉफ्टवेयर Testing कुशल Testing डेटा प्रबंधन का परिणाम है, लेकिन यह सरल नहीं है। हाल के एक अध्ययन में केवल 45% Testing टीमों ने कहा कि Testing वातावरण और डेटा का प्रबंधन करना मुश्किल था।

सर्वोत्तम Testing डेटा प्रबंधन टूल में शामिल हैं:

- Solix Enterprise Data Management Suite
- DATPROF
- CA Test Data Manager (Datamaker)
- Compuware
- Informatica
- Microfocus Data Express
- IBM Test Data Management
- Appsurify
- PractiTest
- Quality

7.1.8 एक Anamoly क्या है?

सॉफ्टवेयर Testing में एक Anamoly एक परिणाम है जो प्रत्याशित से भिन्न होता है। यह व्यवहार किसी दस्तावेज़ या Tester की धारणाओं और अनुभवों का परिणाम हो सकता है।

चूंकि Testing वेयर विनिर्देश के अनुसार व्यवहार कर सकता है लेकिन फिर भी सुधार के लिए प्रयोज्य कक्ष है, एक Anamoly एक प्रयोज्य समस्या का भी उल्लेख कर सकती है। Anamoly का वर्णन करने के लिए एक डिफेक्ट या बग का भी उपयोग किया जा सकता है।

एक Anamoly रिपोर्ट क्या है?

Anamoly रिपोर्ट: एक Anamoly रिपोर्ट, जिसे बग रिपोर्ट या डिफेक्ट रिपोर्ट के रूप में भी जाना जाता है, का उपयोग प्रोजेक्ट टीम या विकास टीम को उन Anamolies के बारे में सूचित करने के लिए किया जाता है जो पाई गई हैं और आमतौर पर इसमें निम्नलिखित घटक शामिल होते हैं:

- - डिफेक्ट पहचानकर्ता: प्रत्येक खोजे गए डिफेक्ट को सौंपा गया एक अद्वितीय नंबर।
- डिफेक्ट विवरण में डिफेक्ट्स का विस्तार से वर्णन करता है।
- **डिफेक्ट की स्थिति:** अभी डिफेक्ट की स्थिति:
- त्रुटियों को डुप्लिकेट करने के चरण
- तेज।
- प्राथमिकता।
- डिफेक्ट रिपोर्टिंग की तिथि।
- Testing की स्थिति।
- दस्तावेज़, स्क्रीन कैप्चर और अन्य अनुलग्नक

डेटा प्रवाह Anamolies: एक Anamoly को क्रियाओं के दो-वर्ण अनुक्रम द्वारा दर्शाया जाता है। उदाहरण के लिए, ku का अर्थ है कि वस्तु को मार दिया जाता है और फिर उपयोग किया जाता है, जबकि dd का अर्थ है कि वस्तु को बिना किसी हस्तक्षेप के दो बार परिभाषित किया गया है। एक Anamoly क्या है यह एप्लीकेशन पर निर्भर करता है। d, k और u के लिए नौ संभावित दो-अक्षर संयोजन हैं। कुछ बग हैं, कुछ संदिग्ध हैं, और कुछ ठीक हैं।

1. DD: शायद हानिकारक नहीं है, लेकिन परेशान करने वाला। जब बीच में कोई उपयोग नहीं होता है तो ऑब्जेक्ट को दो बार परिभाषित क्यों करें? उपयोग की अनुपस्थिति में, वस्तु को दो बार परिभाषित क्यों करें?
2. DK : ? यदि आप इसका उपयोग नहीं करने जा रहे हैं तो किसी ऑब्जेक्ट को परिभाषित क्यों करें?
3. DU : सामान्य मामला। ऑब्जेक्ट को परिभाषित किया जाता है और फिर उपयोग किया जाता है।
4. KD: सामान्य स्थिति। एक वस्तु को मार दिया जाता है और फिर फिर से परिभाषित किया जाता है।
5. KK: हानिरहित लेकिन शायद छोटी गाड़ी। क्या आप यह सुनिश्चित करना चाहते थे कि यह वास्तव में मारा गया था?
6. KU: एक बग। ऑब्जेक्ट मौजूद नहीं है।
7. UD: आमतौर पर बग नहीं होता है क्योंकि भाषा लगभग किसी भी समय पुन: असाइनमेंट की अनुमति देती है।
8. UK: सामान्य स्थिति।
9. UU: सामान्य स्थिति

7.1.9 Anamoly का पता लगाने के तरीके

Anamolies के प्रकार हैं:

- **बिंदु Anamolies:** ज्ञान का एक भी उदाहरण असामान्य है यदि यह शेष से बहुत दूर है।
- **व्यावसायिक उपयोग का मामला:** "खर्च की गई राशि" के आधार पर मास्टरकार्ड धोखाधड़ी का पता लगाना।
- **प्रासंगिक Anamolies**: असामान्यता विशिष्ट संदर्भ पर आधारित है। इस तरह की Anamoly समय श्रृंखला डेटा में पाई जा सकती है। कंपनी का उपयोग मामला: कार्यदिवसों के दौरान हर दिन गैस पर $ 100 खर्च करना विशिष्ट है, लेकिन छुट्टी पर रहते हुए ऐसा करना अजीब माना जाएगा।
- **सामूहिक Anamolies:** ज्ञान उदाहरणों का एक सेट एक साथ रखा गया है जो Anamolies का पता लगाने में मदद करता है। उदाहरण: कोई व्यक्ति अनपेक्षित रूप से किसी क्षेत्र होस्ट के लिए एक दूर की मशीन टाइप करने की जानकारी को दोहराने का प्रयास कर रहा है, एक Anamoly जिसे संभावित साइबर हमले के रूप में चिह्नित किया जाएगा।

7.1.10 Testing .Net, SQL, Java, Oracle, V B Script जैसे सामान्य ऍप्लिकेशन्स का उपयोग क्या है?

1. **Net?**

   माइक्रोसॉफ्ट। Net, जिसे पहले के नाम से जाना जाता था। Net कोर, 2016 में पेश किया गया था और यह पहले के लिए एक ओपन-सोर्स, क्रॉस-प्लेटफॉर्म प्रतिस्थापन है। Net फ्रेमवर्क.

   डेस्कटॉप, मोबाइल, गेम, वेब और IoT उपकरणों सहित ऍप्लिकेशन्स की एक विस्तृत श्रृंखला का उपयोग करके बनाया जा सकता है। Net फ्रेमवर्क। .NET विभिन्न पुस्तकालयों, भाषाओं और संपादकों का उपयोग कर सकता है क्योंकि यह ओपन-सोर्स है।

Net फ्रेमवर्क क्या करता है?

Net फ्रेमवर्क ने शुरू में माइक्रोसॉफ्ट विंडोज विकास में सहायता के लिए एक मानकीकृत सॉफ्टवेयर विकास Framework की पेशकश की थी जब इसे पहली बार कल्पना और जारी किया गया था। वही। Net उत्पाद अब एक मल्टीचैनल ओपन-सोर्स डेवलपमेंट फ्रेमवर्क प्रदान करता है। .NET फ्रेमवर्क, द्वारा बनाया गया। Net फाउंडेशन और MIT लाइसेंस के तहत उपलब्ध कराया गया, विकासशील को सरल, तेज और अधिक विश्वसनीय बनाने के लिए डिज़ाइन किया गया है।

एक विशाल पारिस्थितिकी तंत्र के साथ जुड़ा हुआ है। Net, जिसमें शामिल हैं:

- WPF (विंडोज प्रेजेंटेशन फाउंडेशन)। विंडोज ऑपरेटिंग सिस्टम और डेस्कटॉप प्रोग्राम के लिए एक उपयोगकर्ता-इंटरफ़ेस डिज़ाइन टूल।
- विंडोज फॉर्म। Windows डेस्कटॉप एप्लीकेशन NET Framework GUI लायब्रेरी का उपयोग कर सकते हैं।
- asp Net फॉर्म। एक वेब एप्लीकेशन Framework जो सुरक्षित और पहुँच योग्य वेब ऍप्लिकेशन्स के सृजन में सहायता करता है.

Net किसके लिए प्रयोग किया जाता है?

- Net के कई महत्वपूर्ण भाग हैं जो विकास को आसान बनाते हैं, जिनमें शामिल हैं:
- ऍप्लिकेशन्स में उपयोग किए जाने वाले Framework और लाइब्रेरी
- भाषा कंपाइलर और रनटाइम घटक।
- Visual Basic, F# और C# जैसी भाषाओं के लिए समर्थन।
- विजुअल स्टूडियो जैसे सॉफ्टवेयर प्रोग्राम
- डेस्कटॉप एप्लिकेशन, वेब एप्लिकेशन, मोबाइल ऐप, क्लाउड-नेटिव एप्लिकेशन, गेम्स और इंटरनेट ऑफ थिंग्स (IoT) गैजेट्स सभी Net के साथ बनाए जा सकते हैं।
- Net की अविश्वसनीय चौड़ाई इसे उद्योगों की एक विस्तृत श्रृंखला में एक लोकप्रिय विकल्प बनाती है। प्रोग्रामर जो Net में विशेषज्ञ हैं, उच्च मांग में होंगे।
- Net केवल एक Framework है, हालांकि, एक ही समय में। अंतिम उत्पाद बनाने के लिए एक डेवलपर को NET विशिष्ट प्रोग्रामिंग भाषाओं में से एक से परिचित होना चाहिए, जैसे कि C#, F#, या Visual Basic।

2. **SQL (स्ट्रक्चर्ड क्वेरी भाषा)**
   - मैनुअल Tester और स्वचालित Tester दोनों के पास SQL ज्ञान होना चाहिए।
   - SQL (स्ट्रक्चर्ड क्वेरी लैंग्वेज) नामक एक डोमेन-विशिष्ट भाषा का उपयोग डेटाबेस में डेटा को स्टोर करने, हेरफेर करने और पुनर्प्राप्त करने के लिए किया जाता है।
   - सीक्वल (संरचित अंग्रेजी क्वेरी भाषा) आईबीएम शोधकर्ताओं रेमंड बॉयस और डोनाल्ड चाम बर्लिन द्वारा ई एफ कॉड के पेपर के अपने अध्ययन के परिणामस्वरूप बनाया गया था। 1970 में, दोनों ने IBM Corporation की सैन जोस रिसर्च सुविधा में SQL बनाया।
   - रिलेशनल डेटाबेस सिस्टम के लिए पसंदीदा भाषा SQL है। SQL सभी रिलेशनल डेटाबेस मैनेजमेंट सिस्टम (R D MS) द्वारा उपयोग की जाने वाली सामान्य डेटाबेस भाषा है, जिसमें My SQL, MS Access, Oracle, Sybase, Informix, Postgress और SQL Server शामिल हैं।
3. **SQL का उपयोग**
   - **डेटा परिभाषा:** यह संग्रहीत डेटा के संगठन और संरचना के साथ-साथ संग्रहीत डेटा आइटम के बीच कनेक्शन का वर्णन करता है।
   - **डेटा पुनर्प्राप्ति:** SQL का उपयोग करके डेटा पुनर्प्राप्ति भी की जा सकती है।
   - **डेटा हेरफेर:** SQL उपयोगकर्ताओं को नया डेटा जोड़ने, डेटा निकालने या मौजूदा डेटा को संशोधित करने की क्षमता भी प्रदान करता है।
   - **अभिगम नियंत्रण:** डेटा तक पहुँचने, जोड़ने और संशोधित करने की उपयोगकर्ता की क्षमता को सीमित करके, SQL संग्रहीत जानकारी को अनधिकृत पहुँच से सुरक्षित कर सकता है।
   - **डेटा साझाकरण:** SQL का उपयोग समवर्ती उपयोगकर्ताओं के बीच डेटा साझाकरण को प्रबंधित करने के लिए किया जाता है, जिससे दो अलग-अलग उपयोगकर्ताओं द्वारा लगभग एक ही समय में किए गए परिवर्तनों के आकस्मिक उन्मूलन को रोका जा सकता है।
4. **Java**
   - Java सबसे लोकप्रिय, व्यापक रूप से इस्तेमाल की जाने वाली ऑब्जेक्ट-ओरिएंटेड प्रोग्रामिंग भाषा है। Java की सुरक्षा विशेषता इसे लोकप्रिय और व्यापक रूप से उपयोग करती है। यह कई Java उत्साही लोगों द्वारा विभिन्न उद्देश्यों के लिए उपयोग किया जाताहै। Java का उपयोग करके, कोई भी विभिन्न प्रकार के एप्लिकेशन विकसित कर सकता है जैसे एंटरप्राइज़ एप्लिकेशन, नेटवर्क एप्लिकेशन, डेस्कटॉप एप्लिकेशन, वेब एप्लिकेशन, गेम्स, एंड्रॉइड ऐप और बहुत कुछ।
   - सबसे व्यापक रूप से इस्तेमाल की जाने वाली और प्रसिद्ध ऑब्जेक्ट-ओरिएंटेड प्रोग्रामिंग भाषा Java है। Java की लोकप्रियता और व्यापक उपयोग काफी हद तक इसकी सुरक्षा सुविधा के कारण है। कई Java उत्साही इसे विभिन्न उद्देश्यों के लिए उपयोग करते हैं। Java का उपयोग एंटरप्राइज़ ऐप, नेटवर्क ऐप, डेस्कटॉप ऐप, वेब ऐप, गेम, एंड्रॉइड ऐप और कई अन्य सहित ऍप्लिकेशन्स की एक विस्तृत श्रृंखला बनाने के लिए किया जा सकता है।
   - Java प्रोग्रामिंग भाषा एप्लीकेशन
   - कथन 3 बिलियन डिवाइस रन Java, जो Java इंस्टॉलेशन के दौरान प्रदर्शित होता है, Java प्रोग्रामिंग भाषा के व्यापक उपयोग को प्रमाणित करता है। Java ओएपीआई का एक व्यापक और समृद्ध सेट है जो डेवलपर्स को एप्लिकेशन बनाने में सहायता करता है। Java हमें विभिन्न उपयोगों के लिए विभिन्न प्रकार के एप्लिकेशन बनाने की अनुमति देता है। हम Java तकनीक का उपयोग करके निम्नलिखित एप्लिकेशन बना सकते हैं।

- मोबाइल ऐप्स का विकास
- डेस्कटॉप GUI के लिए एप्लीकेशन
- वेब के लिए वेब-आधारित एप्लीकेशन
- खेलों के लिए एप्लीकेशन
- बिग डेटा के लिए टेक्नोलॉजीज
- विभिन्न एप्लीकेशन
- क्लाउड का उपयोग करने वाले एप्लिकेशन
- IoT के एप्लीकेशन
- मोबाइल एप्लिकेशन बनाना

5. **Oracle क्लाउड इन्फ्रास्ट्रक्चर (OCI) एप्लीकेशन**
   विश्व स्तर पर वितरित ओरेकल क्लाउड को ओरेकल क्लाउड इन्फ्रास्ट्रक्चर (OCI) एप्लिकेशन डेवलपमेंट सेवाओं का उपयोग करके डेवलपर्स द्वारा निर्मित, प्रबंधित और स्वचालित किया जा सकता है। ग्राहक ओरेकल वेब लॉजिक जैसे एप्लिकेशन सर्वर को अपग्रेड कर सकते हैं, कुबेर नेट्स और कंटेनरों का उपयोग करके मौजूदा ऐप्स का आधुनिकीकरण कर सकते हैं, और माइक्रो सेवाओं, AI, स्वायत्त डेटाबेस और कई अन्य तकनीकों का उपयोग करके नए एप्लिकेशन बना सकते हैं।

Oracle डेटाबेस के फायदे

Oracle डेटाबेस के मुख्य लाभ इस प्रकार हैं:

- **ओरेकल की प्रक्रियाएं और मार्गदर्शक सिद्धांत** हमें डेटाबेस प्रदर्शन के उच्च स्तर को प्राप्त करने में मदद करते हैं। अपने डेटाबेस में प्रदर्शन अनुकूलन तकनीकों का उपयोग करके, हम संचालन और क्वेरी निष्पादन को गति दे सकते हैं। यह विधि त्वरित डेटा पुनर्प्राप्ति और संशोधन की सुविधा प्रदान करती है।
- **अपने किसी भी प्रतिद्वंद्वी की तुलना में अधिक पोर्टेबल होने के नाते,** ओरेकल डेटाबेस का उपयोग विभिन्न प्लेटफार्मों पर किया जा सकता है। इस डेटाबेस का उपयोग 100 से अधिक हार्डवेयर प्लेटफार्मों और लगभग 20 नेटवर्किंग प्रोटोकॉल पर किया जा सकता है। OS और हार्डवेयर में सुरक्षित परिवर्तन करके, यह डेटाबेस ओरेकल एप्लिकेशन लिखना आसान बनाता है।
- **बैकअप और बहाली:** अपने सभी Oracle ऑनलाइन बैकअप और पुनर्स्थापना डेटा का ठीक से बैकअप लेना हमेशा उचित होता है। ओरेकल डेटाबेस का उपयोग करने से रिकवरी को जल्दी से पूरा करना आसान हो जाता है। RMAN (रिकवरी मैनेजर) कार्यक्षमता। डाउनटाइम या पावर आउटेज के दौरान, यह डेटाबेस फ़ाइलों को पुनर्प्राप्त या पुनर्स्थापित कर सकता है। इसका उपयोग संग्रहीत बैकअप, निरंतर संग्रह और ऑनलाइन बैकअप के लिए किया जा सकता है। उपयोगकर्ता-प्रबंधित पुनर्प्राप्ति करने के लिए, हम SQL * PLUS का भी उपयोग कर सकते हैं।
- **PL/SQL:** प्रक्रियात्मक प्रोग्रामिंग के लिए PL/SQL एक्सटेंशन के लिए समर्थन Oracle डेटाबेस का उपयोग करने के मुख्य लाभों में से एक है।
- **एकाधिक डेटाबेस:** ओरेकल डेटाबेस एक ही सर्वर पर कई डेटाबेस इंस्टेंस प्रबंधन की अनुमति देता है। डेटाबेस इंस्टेंस होस्ट करने वाले सर्वर पर सीपीयू आवंटन को नियंत्रित करने के लिए, यहएक इंस्टेंस केजिंग Strategies को आगे बढ़ाता है। डेटाबेस इंस्टेंस होस्ट करने वाले सर्वर पर सीपीयू आवंटन को नियंत्रित करने के लिए, यह एक इंस्टेंस केजिंग Strategies को आगे बढ़ाता है। डेटाबेस संसाधन प्रबंधन और आवृत्ति केजिंग एकाधिक आवृत्तियों में सेवाओं का प्रबंधन करने के लिए एक साथ कार्य कर सकते हैं.

फ्लैशबैक तकनीक: नवीनतम ओरेकल संस्करण यह लाभ प्रदान करता है। यह हमें मानवीय त्रुटि के कारण गलती से हटा दी गई या खो गई जानकारी को पुनः प्राप्त करने में सक्षम बनाता है, जैसे कि तालिका छोड़ना या गलती से महत्वपूर्ण जानकारी हटाना।

V B स्क्रिप्ट

- V B स्क्रिप्ट एक स्क्रिप्टिंग भाषा है।
- प्रोग्रामिंग भाषाओं की व्याख्या की जाती है जिनमें स्क्रिप्टिंग भाषाएं शामिल हैं।
- माइक्रोसॉफ्ट की विजुअल बेसिक प्रोग्रामिंग भाषा V B स्क्रिप्ट के लिए एक नींव के रूप में कार्य करती है।
- यह एक शक्तिशाली उपकरण प्रदान करता है जिसका उपयोग वेब पेजों में सहभागिता जोड़ने के लिए किया जा सकता है, भले ही यह विजुअल बेसिक के समान कार्यक्षमता प्रदान न करे।

V B स्क्रिप्ट का उपयोग करता है

- V B स्क्रिप्ट का उपयोग वेब पेजों की कार्यक्षमता और बातचीत देने के लिए किया जाता है।
- क्लाइंट-साइड स्क्रिप्टिंग V B स्क्रिप्ट का उपयोग करके किया जा सकता है।
- क्लाइंट साइड स्क्रिप्टिंग के लिए वीबीएस का उपयोग करने का डिफेक्ट यह है कि केवल इंटरनेट एक्सप्लोरर ही इसे समझ सकता है। माइक्रोसॉफ्ट का इंटरनेट एक्सप्लोरर माइक्रोसॉफ्ट उत्पाद V B स्क्रिप्ट के लिए आवश्यक है।
- इसके अतिरिक्त, V B एस के साथ सर्वर-साइड स्क्रिप्टिंग एक विकल्प है। यह एक सक्रिय सर्वर पृष्ठ (ASP)-संगत पैकेजिंग और व्यक्तिगत वेब सर्वर (PWS) या इंटरनेट सूचना सर्वर (IIS) जैसे किसी Microsoft वेब सर्वर का उपयोग करने के लिए कहता है।

7.1.11Static Testing, Dynamic Testing, व्हाइट बॉक्स/ ब्लैक बॉक्स/ग्रे बॉक्स Testing करने की प्रक्रिया

1. **Static Testing**

स्टेटिक Testing सॉफ्टवेयर Testing की एक विधि है जिसमें पूरे प्रोग्राम को चलाना या सॉफ़्टवेयर निष्पादित करना शामिल नहीं है। विकास प्रक्रिया में Static Testing जल्दी किया जाता है ताकि Dynamic Testing पर जाने से पहले बग और त्रुटियों को पाया जा सके और ठीक किया जा सके। Dynamic Testing की तुलना में, Static Testing के लिए कम समय और संसाधनों की आवश्यकता हो सकती है। Static Testing का उपयोग कुछ त्रुटियों को भी उजागर कर सकता है जो Dynamic Testing आसानी से पता लगाने में सक्षम नहीं हो सकते हैं।

अनिवार्य रूप से दो प्रकार के Static Testing हैं:

- एक मैन्युअल रूप से किया जाता है, अर्थात, कोड का विश्लेषण मैन्युअल रूप से किया जाता है। एक कोड समीक्षा या एक साधारण समीक्षा इस प्रक्रिया के अन्य नाम हैं।
- स्वचालित विश्लेषण, जिसमें हम उपकरणों का उपयोग करके Testing चलाते हैं, एक और है।

Static Testing का महत्व: Static Testing किए जाने के कई कारण हैं और यह Dynamic Testing को बहुत क्यों बढ़ाता है। Static Testing के उपयोग और महत्व को नीचे दिए गए कुछ बिंदुओं में चित्रित किया गया है।

- इस Testing पद्धति का उपयोग गंभीर होने से पहले त्रुटियों को खोजने के लिए किया जाता है, न कि केवल Dynamic Testing के बाद।
- यह Testing Dynamic Testing के दौरान खोजी जाने वाली त्रुटियों की संख्या को कम करने के लिए किया जाता है, जो विकास के अगले चरणों में कार्यभार को कम करेगा।
- इन लागत और संसाधन क्षमता के कारण लागत को अनुकूलित किया जाता है, और Static Testing तकनीक का उपयोग करके कुल विकास अवधि को भी तेज किया जाता है।

कम प्रयास और अनुभव के साथ गलतियों को देखने में सक्षम होने से उत्पादकता बढ़ जाती है।

विभिन्न घटक या संस्थाएं जिनकी जांच Static Testing प्रक्रिया के दौरान की जाएगी।

- यूनिट टेस्ट उदाहरण।
- व्यावसायिक आवश्यकताओं को रेखांकित करने वाले दस्तावेज़
- सॉफ्टवेयर का एक कार्यशील संस्करण।
- Testing डेटा - उपयोगकर्ता मैनुअल, निर्देश पुस्तिका, या कोई अन्य प्रासंगिक दस्तावेज
- प्रदर्शन Testing के लिए स्क्रिप्ट
- Test Cases
- कार्यों के लिए आवश्यकताएँ
- पता लगाने की क्षमता मैट्रिक्स वाले दस्तावेज़
- Test Startegy

Testing समीक्षा की अवधारणा:

किसी भी प्रोग्राम के डिज़ाइन में संभावित गलतियों या खामियों का पता लगाना एक प्रक्रिया है जिसे Static Testing में समीक्षा के रूप में जाना जाता है। पूरी टीम को प्रक्रिया की प्रगति और Testing चरण के दौरान मौजूद हर जोखिम के बारे में सूचित किया जाता है।

इसलिए एक ऐसा निर्णय लेना बोधगम्य है जो किसी विशेषज्ञ द्वारा किए गए किसी भी व्यक्तिगत निर्णय से काफी बेहतर है क्योंकि सभी विभिन्न मानसिकता एक ही मंच पर एक साथ काम कर सकती हैं और अपनी राय व्यक्त करने के लिए स्वतंत्र हैं। यह उल्लेख किया जाना चाहिए कि चार मुख्य श्रेणियां हैं जिनमें समीक्षा रखी जा सकती है। निम्नानुसार:

- **औपचारिक समीक्षा:** एक औपचारिक समीक्षा में, पूरी प्रक्रिया एक औपचारिक और पूर्व निर्धारित एजेंडे का पालन करती है। इसके अतिरिक्त, इसमें एक सुव्यवस्थित और विनियमित प्रक्रिया है जिसका पालन सॉफ्टवेयर जीवन चक्र के प्रत्येक चरण के बाद किया जाना चाहिए।
- **पूर्वाभ्यास:** पूर्वाभ्यास के दौरान, विकास टीम को सॉफ़्टवेयर उत्पाद की समीक्षा करने और उनके सामने आने वाली किसी भी Anamolies को नोट करने की आवश्यकता होती है।
- **तकनीकी समीक्षा:** प्रकृति में तकनीकी, इसका मतलब है कि केवल तकनीकी प्रशिक्षण प्राप्त करने वालों को सॉफ्टवेयर में त्रुटियों और बग की पहचान करने के लिए अपने ज्ञान और विशेषज्ञता का उपयोग करने की अनुमति है।

जैसा कि नाम से ही स्पष्ट है, तकनीकी, इसलिए केवल तकनीकी रूप से प्रशिक्षित व्यक्तियों को अपने कौशल और अनुभव का उपयोग करके सॉफ़्टवेयर के भीतर त्रुटियों और बगों को खोजने की अनुमति है।

निरीक्षण: इस प्रक्रिया के दौरान, विशेषज्ञों और डेवलपर्स की एक टीम सॉफ़्टवेयर की विशेषताओं की प्रभावशीलता का मूल्यांकन करने के साथ-साथ बग और अन्य खामियों की पहचान करने के लिए मानकों, नियमों और दिशानिर्देशों के एक सेट का उपयोग करती है जिन्हें ठीक किया जा सकता है।

Testing के समीक्षा चरण में प्रतिभागी:

- **मॉडरेटर:** यह टीम प्रवेश समीक्षा का प्रबंधन करती है, फिर से काम का प्रबंधन करती है, टीम को कोच करती है, आवश्यकतानुसार बैठकों की योजना बनाती है, और अन्य सभी पूर्व-Testing समीक्षा कर्तव्यों को पूरा करती है।
- **Scribe:** वे कमी लॉगिंग करते हैं, समीक्षा बैठक में भाग लेते हैं, और उन सभी आवश्यक चीजों को नोट करते हैं जिन्हें ठीक करने की आवश्यकता होती है। एक मुंशी की प्राथमिक जिम्मेदारी पूरी तरह से नोट्स लेना है जबकि एक Testing की समीक्षा की जा रही है।
- **समीक्षक:** यह वह व्यक्ति या लोग हैं जो सामग्री का निरीक्षण करते हैं और खामियों की तलाश करते हैं।
- **प्रबंधक:** यह व्यक्ति सुनिश्चित करता है कि सभी समीक्षा प्रोटोकॉल का पालन किया गया है और Testing समीक्षा पूरी होने के बाद सभी आवश्यकताओं और उद्देश्यों को पूरा किया गया है।
- **लेखक:** यह व्यक्ति खोजी गई किसी भी त्रुटि को ठीक करने का प्रभारी है और यह भी गारंटी देना है कि दस्तावेज़ की गुणवत्ता में वृद्धि हुई है।

2. **Dynamic Testing**

- एक प्रकार का सॉफ्टवेयर Testing जिसे Dynamic Testing कहा जाता है, सॉफ्टवेयर के कोड के गतिशील व्यवहार की पूरी तरह से जांच करता है। संक्षेप में, सत्यापन और सत्यापन पूरे Testing को बनाते हैं, बाद वाले के साथ - सत्यापन प्रक्रिया - को Dynamic Testing के रूप में संदर्भित किया जाता है।
- आइए लॉग-इन पृष्ठ का एक उदाहरण देखें, जहां पासवर्ड और ईमेल दर्ज करने के लिए दो फ़ील्ड हैं। मान लें कि हमें बस लोकल पार्ट जोड़ना है, जैसे स्टार और address domain @gmail.com ऑटोमैटिकली ऐड हो जाएगा।
- इसके अतिरिक्त, पासवर्ड को पासवर्ड के रूप में स्वीकार करने के लिए, पंजीकरण करते समय इसमें कम से कम एक प्रतीक या एक अपरकेस अक्षर होना चाहिए। गतिशील कार्यक्षमता के इस वर्ग की जांच करने के लिए Dynamic Testing का उपयोग किया जाता है।
- Dynamic Testing का प्राथमिक लक्ष्य सॉफ़्टवेयर के प्रत्येक घटक का Test मैन्युअल रूप से उसके UI या API में मान दर्ज करके करना है और यह निर्धारित करना है कि मान स्वीकार किए गए हैं या नहीं और इनपुट के अनुसार वांछित आउटपुट उत्पन्न हुआ है या नहीं।
- सिस्टम का Test अंतिम उपयोगकर्ताओं के इनपुट के साथ किया जाता है, और यह पूरी तरह से उनका दृष्टिकोण है।
- वास्तविक प्रणाली के साथ काम करते समय, Dynamic Testing किया जा सकता है, जिससे वास्तविक समय इनपुट और आउटपुट सत्यापन की अनुमति मिलती है। सीधे शब्दों में कहें, तो यह Testing सिस्टम पर ही किया जाता है ताकि सॉफ्टवेयर और सिस्टम के उपयोग के दौरान किसी भी डिफेक्ट्स या त्रुटि की पहचान की जा सके।
- Dynamic Testing, जैसा कि हमने सीखा है, सॉफ़्टवेयर के सभी ऍप्लिकेशन्स और विशेषताओं को सत्यापित करने की प्रक्रिया है जो यह सुनिश्चित करती है कि वास्तविक इनपुट किए गए हैं और सॉफ़्टवेयर को वास्तविक कार्य वातावरण में बनाए रखा गया है।

Dynamic Testing प्रक्रिया

- Dynamic Testing आमतौर पर एक पूर्व निर्धारित प्रक्रिया का पालन करता है जब दृष्टिकोण और Testing कार्यान्वयन प्रदर्शन तय हो जाते हैं, जिस बिंदु पर टीम विभिन्न Testing गतिविधियों को पूरा करने के लिए आगे बढ़ सकती है।
- टीम इस प्रक्रिया का उपयोग करके दृष्टिकोण और रणनीतियों में किसी भी अनियमितताओं की पहचान कर सकती है, जो सभी Testing चरणों के प्रदर्शन में भी सहायता करती है।
- STLC में Dynamic Testing प्रक्रिया में विभिन्न कार्य शामिल हैं। इसके अतिरिक्त, Testing प्रक्रिया में पहले कार्य की सफलता Dynamic Testing प्रक्रिया में सभी कार्यों के लिए आवश्यक है।

Dynamic Testing प्रक्रिया बनाने वाले चरण इस प्रकार हैं:

- Test case डिजाइन
- Testing पर्यावरण स्टेप-अप
- Test Case निष्पादन
- Testing विश्लेषण और मूल्यांकन
- बग रिपोर्टिंग

सॉफ्टवेयर Testing के Life Cycle में, वास्तविक Dynamic Testing प्रक्रिया Testing केस डिजाइन के साथ शुरू होती है। अब जब हमें Dynamic Testing प्रक्रिया की पूरी समझ है, तो हम प्रत्येक चरण पर व्यक्तिगत रूप से जाएंगे।

चरण 1: Test case डिजाइन

- सॉफ्टवेयर Testing के Life Cycle में, वास्तविक Dynamic Testing प्रक्रिया Testing केस डिजाइन के साथ शुरू होती है। अब जब हमें Dynamic Testing प्रक्रिया की पूरी समझ है, तो हम प्रत्येक चरण पर व्यक्तिगत रूप से जाएंगे।
- इस चरण में, हम उन विशेषताओं को निर्धारित कर सकते हैं जिनके लिए Testing की आवश्यकता होती है, साथ ही Testing की स्थिति भी बना सकते हैं, Test cases को प्राप्त कर सकते हैं और कवरेज आइटम निकाल सकते हैं।

चरण 2: पर्यावरण सेटअप

- क्योंकि Testing सीधे सॉफ्टवेयर उत्पाद पर किया जाता है, हम यह सुनिश्चित करेंगे कि इस चरण के दौरान Testing वातावरण हमेशा उत्पादन वातावरण के समानांतर हो।
- Testing वातावरण स्थापित करना, जो हमें Testing मशीनों में सफल होने में सहायता करता है, इस चरण में Dynamic Testing प्रक्रिया का मुख्य लक्ष्य है।

चरण 3: Testing निष्पादन

- Testing वातावरण सफलतापूर्वक स्थापित होने के बाद Dynamic Testing के प्रारंभिक चरण के दौरान बनाए गए Test cases को निष्पादित किया जाएगा।

चरण 4: विश्लेषण और मूल्यांकन

- Test cases के पूरा होने के बाद, हम परिणामों का विश्लेषण और आकलन करेंगे। और हम उन परिणामों की तुलना करेंगे जो प्रत्याशित थे।
- हम उन Test cases को विफल मान लेंगे और बग रिपॉजिटरी में बग लॉग करेंगे यदि निष्पादन के बाद अपेक्षित और वास्तविक परिणाम समान नहीं हैं।
- Test cases के पूरा होने के बाद, हम परिणामों का विश्लेषण और आकलन करेंगे। और हम उन परिणामों की तुलना करेंगे जो प्रत्याशित थे।
- हम उन Test cases को विफल मान लेंगे और बग रिपॉजिटरी में बग लॉग करेंगे यदि निष्पादन के बाद अपेक्षित और वास्तविक परिणाम समान नहीं हैं।

चरण 5: बग रिपोर्टिंग

- Test cases की जांच करने के बाद, हम वास्तविक परिणाम और प्रत्याशित परिणाम के बीच किसी भी बग या विसंगतियों के उपयुक्त पक्ष को सूचित करेंगे। इसके अतिरिक्त, संबंधित पक्ष यह सुनिश्चित करेगा कि समस्या का समाधान हो गया है और एक उच्च गुणवत्ता वाला उत्पाद वितरित किया गया है।
- Dynamic Testing का एक केस स्टडी
- आइए यह प्रदर्शित करने के लिए एक सरल उदाहरण का उपयोग करें कि Dynamic Testing कैसे कार्य करता है।
- इसलिए हम इस उद्देश्य के लिए किसी भी एप्लिकेशन के लॉगिन मॉड्यूल को समझेंगे, जैसे कि www.Twitter.com ।
- मान लें कि हम एक मजबूत पासवर्ड के साथ एक नया खाता बनाना चाहते हैं, इस स्थिति में पासवर्ड फ़ील्ड को कुछ पूर्व-स्थापित दिशानिर्देशों का पालन करना होगा।
- इसके अलावा आवश्यक आठ character, सभी बड़े अक्षर और पासवर्ड में कम से कम एक विशेष character हैं।
- यदि हम इस कार्यक्षमता का Test कर रहे हैं, तो हम इसका Test करने के लिए सभी इनपुट शर्तों को लेंगे और फिर आउटपुट को सत्यापित करेंगे। यदि हम इस कार्यक्षमता का Test करना चाहते थे, तो हम सभी संभावित इनपुट परिदृश्यों का उपयोग करेंगे और फिर परिणामों की पुष्टि करेंगे।
- हम ऐसे प्रतिबंध भी लगा सकते हैं जो काम नहीं करेंगे, जैसे उपयोगकर्ताओं को 4-character पासवर्ड दर्ज करने की आवश्यकता होती है, और यह देखने के लिए जांचें कि क्या वास्तव में कोई त्रुटि हुई है।

3. White Box Testing

- White Box Testing एक सॉफ्टवेयर Testing तकनीक है जहां, आंतरिक संरचना, डिजाइन, कोडिंग और सभी घटनाओं का Test किया जाता है। जैसा कि नाम से निहित है, White Box सॉफ्टवेयर Tester को बॉक्स के अंदर जो कुछ भी है उसे देखने की अनुमति देता है। जैसा कि नाम से पता चलता है, सफेद बॉक्स, जिसका अर्थ है कि बॉक्स में जो कुछ भी है, सॉफ्टवेयर Tester को दिखाई देता है।
- Testing की यह विधि यह पुष्टि करने पर केंद्रित है कि इनपुट आउटपुट में कैसे प्रवाहित होता है। विभिन्न मॉड्यूल या कार्यों के माध्यम से डेटा प्रवाह का विश्लेषण सॉफ्टवेयर की सुरक्षा, उपयोगिता और प्रस्तुत करने को बढ़ाने पर भी केंद्रित है।
- White Box Testing के लिए कई नाम हैं। उदाहरण के लिए, ओपन बॉक्स Testing या स्पष्ट बॉक्स परीक्षण। इसका मतलब है कि Tester सभी आंतरिक मॉड्यूल को देख या एक्सेस कर सकता है। डेवलपर्स की एक टीम White Box Testing करती है।

White Box Testing के सामान्य चरण

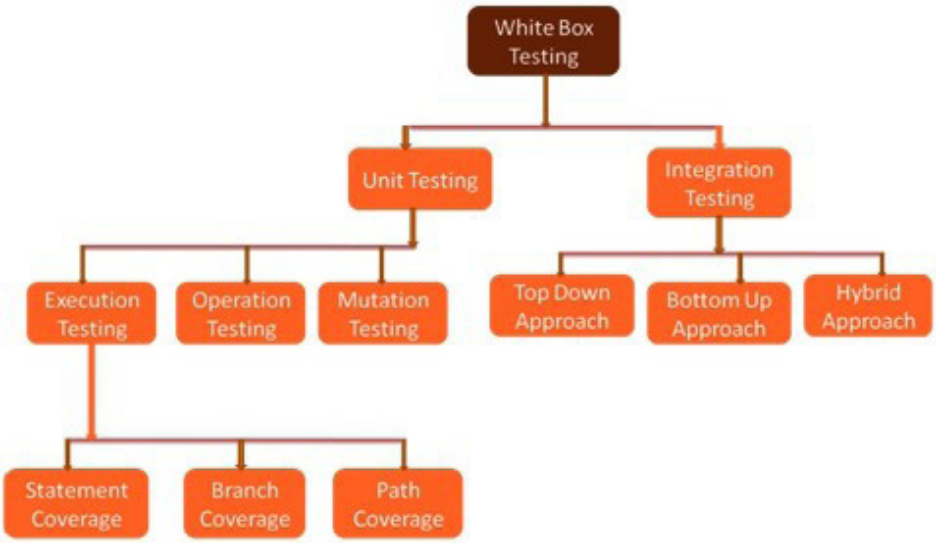
- सभी Testing परिदृश्यों, Test cases को डिज़ाइन करें और उच्च प्राथमिकता संख्या के अनुसार उन्हें प्राथमिकता दें। सभी Test Case और परिदृश्य बनाएं, फिर उन्हें सर्वोच्च प्राथमिकता के क्रम में रैंक करें।
- यह चरण कोड का विश्लेषण करने पर जोर देता है, जबकि यह संसाधन उपयोग, कोड जिसका उपयोग नहीं किया जा रहा है, विभिन्न तरीकों और संचालन में कितना समय लगता है, और अन्य कारकों को देखने के लिए चल रहा है।
- इस चरण के दौरान आंतरिक सबरूटीन्स का Test किया जाता है। आंतरिक सबरूटीन, जैसे कि गैर-सार्वजनिक तरीके और इंटरफेस, अपने डिजाइन के आधार पर सभी प्रकार के डेटा को सही या गलत तरीके से संभाल सकते हैं।
- यह कदम सशर्त और लूप स्टेटमेंट के Testing पर केंद्रित है ताकि यह सुनिश्चित किया जा सके कि वे डेटा इनपुट की एक श्रृंखला के लिए सटीक और कुशल हैं।
- White Box Testing के अंतिम चरण में किसी भी संभावित सुरक्षा खामियों को देखने के लिए सुरक्षा Testing शामिल है कि कोड सुरक्षा का प्रबंधन कैसे करता है।

चित्र 7.1.2 White Box Testing

White Box Testing के लिए तकनीक

Data Flow Testing	घटनाओं के अनुक्रम के संबंध में चर के अनुक्रम का पता लगाने के लिए, डेटा प्रवाह Testing के रूप में जाना जाने वाला Testing तकनीकों का एक समूह कार्यक्रमों के नियंत्रण प्रवाह को देखता है।
Control Flow Testing	नियंत्रण प्रवाह Testing एक नियंत्रण संरचना का उपयोग करके कार्यक्रम के कथन या निर्देश निष्पादन आदेश को स्थापित करता है। प्रोग्राम के लिए एक Test Case प्रोग्राम की नियंत्रण संरचना का उपयोग करके बनाया गया है। इस पद्धति में, Tester Testing पथ सेट करने के लिए एक बड़े कार्यक्रम का एक विशिष्ट खंड चुनता है। कार्यक्रम का नियंत्रण ग्राफ Test cases के प्रतिनिधित्व के रूप में कार्य करता है।
Branch Testing	नियंत्रण प्रवाह ग्राफ की सभी शाखाओं को शाखा कवरेज तकनीक का उपयोग करके कवर किया जाता है। निर्णय बिंदु की प्रत्येक स्थिति को सही और गलत दोनों परिणामों के लिए कम से कम एक बार कवर किया जाता है।
Statement Testing	White Box Test case स्टेटमेंट कवरेज तकनीक का उपयोग करके बनाए जाते हैं। यह विधि प्रत्येक स्रोत कोड कथन को कम से कम एक बार चलाने के लिए कहती है। स्रोत कोड में मौजूद सभी कथनों में से, इसका उपयोग निष्पादित कथनों की कुल संख्या की गणना करने के लिए किया जाता है।
Decision Testing	यह विधि रिपोर्ट करती है कि बूलियन व्यंजकों के परिणाम सही हैं या गलत। हर बार एक बयान, जैसे कि एक जबकि कथन, एक अगर कथन, या एक केस स्टेटमेंट, में दो या अधिक परिणाम होने की क्षमता होती है, इसे निर्णय बिंदु के रूप में संदर्भित किया जाता है क्योंकि केवल दो संभावित परिणाम होते हैं: सही या गलत।

तालिका 7.1.1 White Box Testing तकनीक



चित्र 7.1.3 White Box Testing के प्रकार

4. Black Box Testing

यह Testing की एक विधि है जिसमें एप्लिकेशन के आंतरिक कामकाज का पता नहीं चलता है। यह केवल सिस्टम के मूलभूत घटकों को देखता है और सिस्टम के आंतरिक तार्किक संगठन पर इसका कोई असर नहीं पड़ता है।

Black Box Testing के प्रकार Black Box Testing विभिन्न रूपों में आता है, लेकिन निम्नलिखित सबसे आम हैं।

- सॉफ्टवेयर Tester फंक्शनल Testing करते हैं, एक प्रकार का Black Box Test जो सिस्टम की फंक्शनल आवश्यकताओं से संबंधित है।
- गैर-फंक्शनल Testing - इस प्रकार का Black Box Test विशिष्ट कार्यक्षमता के Testing के बजाय प्रदर्शन, मापनीयता और उपयोगिता जैसी गैर-फंक्शनल आवश्यकताओं पर केंद्रित है।
- प्रतिगमन Testing - प्रतिगमन Testing कोड अपग्रेड, फिक्स, या किसी अन्य सिस्टम रखरखाव के बाद किया जाता है ताकि यह सुनिश्चित किया जा सके कि नए कोड ने मौजूदा कोड पर प्रतिकूल प्रभाव नहीं डाला है।

यह सुनिश्चित करने के लिए कि कोड फंक्शनल और गैर-फंक्शनल आवश्यकताओं को पूरा करता है, Black Box Testing में बाहरी इंटरफेस का Test शामिल है। Black Box Testing में शामिल विभिन्न चरण निम्नलिखित हैं:

Black Box Testing चरण

- Testing योजनाएं विकसित करें। Black Box Testing की तैयारी के लिए, प्राथमिकता वाली Testing योजनाएँ बनाएँ।
- बाहरी इंटरफेस की जांच करें। स्वचालित Testing Suite, जैसे कि NUnit Suites और कस्टम प्रोटोटाइप ऍप्लिकेशन्स का उपयोग करके, विभिन्न प्रकार के इनपुट के लिए बाहरी इंटरफेस का Test करें।
- लोड Testing करें। यह विभिन्न भारों के तहत कैसे व्यवहार करता है की जांच करने के लिए एप्लीकेशन ब्लॉक पर एक लोड Testing चलाएँ। यह गारंटी देता है कि यह उन सभी प्रदर्शन आवश्यकताओं को पूरा करता है जिन्हें आगे निर्धारित किया गया है।
- तनाव Testing लागू करें। विभिन्न बाधाओं का विश्लेषण करने और किसी भी समस्या को स्पॉट करने के लिए जो केवल अत्यधिक उच्च लोड स्थितियों के तहत स्पष्ट होगी, जैसे कि दौड़ की स्थिति और विवाद, एप्लिकेशन ब्लॉक का तनाव परीक्षण।
- सुरक्षा Testing करें। संभावित खतरों के लिए परिनियोजन परिदृश्यों का Test करें। एप्लीकेशन ब्लॉक को मॉक लक्ष्य वातावरण में परिनियोजित करें और मौजूद किसी भी एप्लीकेशन ब्लॉक डिफेक्ट्स का लाभ उठाकर एप्लीकेशन को हैक करने का प्रयास करें.
- वैश्वीकरण Testing का संचालन करें। यह सुनिश्चित करने के लिए Test cases को निष्पादित करें कि एप्लिकेशन ब्लॉक को प्रोग्राम में डिफ़ॉल्ट के रूप में सेट किए गए स्थानों के अलावा अन्य स्थानों के लिए लक्षित कार्यक्रमों के साथ एकीकृत किया जा सकता है।

5. Gray Box Testing पद्धति

यह तकनीक, जिसे "White Box + Black Box = Gray Box" के रूप में जाना जाता है, का उपयोग किसी एप्लिकेशन का Test करने के लिए किया जाता है, जबकि केवल एक बुनियादी समझ होती है कि यह आंतरिक रूप से कैसे संचालित होता है।

Gray Box Testing करते समय Tester को Test cases बनाने की आवश्यकता नहीं होती है। इसके बजाय, Test cases को बनाने के लिए आंतरिक स्थितियों, प्रोग्राम व्यवहार और एप्लीकेशन वास्तुकला ज्ञान का आकलन करने वाले एल्गोरिदम का उपयोग किया जाता है। Tester तब इन Testings को निष्पादित करता है और परिणामों का विश्लेषण करता है।

Gray Box Testing करते समय निम्नलिखित क्रियाएं की जाती हैं:

Gray Box Testing प्रक्रियाएं

Testing किए जा रहे सिस्टम के बारे में वर्तमान में ज्ञात जानकारी के आधार पर एक संरचित मूल्यांकन को Gray Box मूल्यांकन के रूप में जाना जाता है। इसे निम्नानुसार आगे बढ़ना चाहिए:

- Black-Box और White-Box Testing विधियों का उपयोग करके इनपुट निर्धारित करें
- उन आउटपुट का निर्धारण करें जो इन इनपुट को प्रदान किए गए दस्तावेज़ीकरण के प्रकाश में उत्पन्न करना चाहिए।
- उन प्रमुख नियंत्रण प्रवाहों का निर्धारण करें जिन्हें Testing करने की आवश्यकता है।
- निर्धारित करें कि किन महत्वपूर्ण उप-कार्यों को गहन Testing से गुजरना होगा।
- उप-फ़ंक्शन के इनपुट निर्धारित करें।
- आउटपुट निर्धारित करें कि उप-फ़ंक्शन को आपूर्ति किए गए इनपुट के जवाब में उत्पादन करना चाहिए।
- फ़ंक्शन के इस उप-खंड के लिए एक Testing केस बनाएं और चलाएं।
- यह देखने के लिए जांचें कि क्या उप-फ़ंक्शन Test Case के लिए वांछित परिणाम प्रदान करता है।
- सभी उप-कार्यों के लिए, चरण 4-8 दोहराएं।

6. **"Black Box" Testing, "White Box" Testing और "Gray Box" Testing की तुलना**

Sr.No.	Black Box Testing	White Box Testing	Gray Box Testing
1	इस प्रकार के Testing के लिए आंतरिक कार्य संरचना (कोड) का ज्ञान आवश्यक नहीं है। Test cases के लिए, केवल GUI (ग्राफिकल यूजर इंटरफेस) आवश्यक है। Test cases के लिए केवल GUI (ग्राफिकल यूजर इंटरफेस) आवश्यक है।	इस प्रकार के Testing के लिए सॉफ्टवेयर के आंतरिक कामकाज (सॉफ्टवेयर कोडिंग) की समझ की आवश्यकता होती है।	आंतरिक रूप से चीजें कैसे काम करती हैं, इसका कुछ ज्ञान होना आवश्यक है।
2	फंक्शनल Testing, डेटा-संचालित Testing और बंद बॉक्स Testing शब्द Black Box Testing के सभी रूप हैं।	White Box Testing का वर्णन करने के लिए संरचनात्मक Testing, स्पष्ट बॉक्स Testing, कोड-आधारित Testing और पारदर्शी Testing शब्द का भी उपयोग किया जाता है।	शब्द "Gray Box Testing" या "पारभासी Testing" Testing को संदर्भित करता है जहां Tester के पास केवल अल्पविकसित कोडिंग ज्ञान होता है।

3	इस तथ्य के कारण कि Tester को आंतरिक सॉफ़्टवेयर कोडिंग को समझने की आवश्यकता नहीं है, Testing पद्धति में Testing और त्रुटि विधियां शामिल हैं।	चूंकि कोई आंतरिक कोडिंग ज्ञान अंतर नहीं है, White Box Testing के बाद सॉफ्टवेयर में निर्मित सिस्टम सीमाओं और डेटा डोमेन का सत्यापन किया जाता है।	सॉफ्टवेयर को तब इसकी आंतरिक प्रणाली सीमाओं और डेटा डोमेन के लिए मान्य किया जाता है यदि Tester के पास कोडिंग ज्ञान है।
4	सभी Testing स्थानों में सबसे बड़ा इनपुट तालिकाओं के लिए Testing स्थान है, जिसमें Test cases के निर्माण के लिए आवश्यक इनपुट होते हैं।	Black Box Testing की तुलना में, इनपुट की तालिकाओं (Test cases को बनाने के लिए उपयोग किए जाने वाले इनपुट) के लिए Testing स्थान छोटा है।	इनपुट के लिए तालिकाओं (Test cases को बनाने के लिए उपयोग किए जाने वाले इनपुट) में Black Box और White Box Testing की तुलना में एक छोटा Testing स्थान होता है।
5	क्योंकि छिपी हुई सॉफ़्टवेयर त्रुटियाँ आंतरिक कार्यप्रणाली के कारण हो सकती हैं जो Black Box Testing के लिए अज्ञात हैं, उन्हें खोजना बहुत चुनौतीपूर्ण हो सकता है।	इस संभावना के कारण कि वे आंतरिक कामकाज के परिणामस्वरूप हो सकते हैं, जिसकी White Box Testing में पूरी तरह से जांच की जाती है, छिपी हुई त्रुटियां आसानी से पाई जाती हैं।	छिपी हुई त्रुटि को खोजना मुश्किल है। उपयोगकर्ता स्तर के Testing से इसका पता चल सकता है। उपयोगकर्ता स्तर Testing में पाया जा सकता है।
6	एल्गोरिदम का Test करते समय इसे ध्यान में नहीं रखा जाता है।	यह एक अच्छा फिट है और एल्गोरिदम के Testing के लिए सलाह दी जाती है।	एल्गोरिदम का Test करते समय इसे ध्यान में नहीं रखा जाता है।
7	Black Box Testing के लिए आवश्यक समय की मात्रा इस बात पर निर्भर करती है कि फंक्शनल विनिर्देश उपलब्ध हैं या नहीं।	व्यापक कोड के कारण, White Box Testing के लिए Test cases को डिजाइन करने में बहुत समय लगता है।	डिजाइनिंग Test cases को जल्दी से पूरा किया जा सकता है।
8	Tester, डेवलपर्स, एक डी एंड उपयोगकर्ता सभी भाग ले सकते हैं।	Testing में भाग लेने वाले एकमात्र लोग Tester और डेवलपर हैं; अंतिम उपयोगकर्ताओं की अनुमति नहीं है.	Tester, डेवलपर और अंतिम उपयोगकर्ता सभी भाग ले सकते हैं।
9	सभी Testing प्रक्रियाओं में से, इसमें कम से कम समय लगता है।	एक पूरे के रूप में Testing प्रक्रिया सभी Testing प्रक्रियाओं में सबसे लंबी होती है।	White Box Testing की तुलना में कम समय लेना।
10	Black Box Testing वायरल हमलों के खिलाफ लचीलापन और सुरक्षा को कवर करता है।	White Box Testing वायरल हमलों के खिलाफ लचीलापन और सुरक्षा को कवर नहीं करता है।	. Gray Box Testing वायरल हमलों के खिलाफ लचीलापन और सुरक्षा को कवर नहीं करता है।
11	बाहरी अपेक्षाएं और अज्ञात आंतरिक व्यवहार इस Testing का आधार बनते हैं।	कोडिंग, जो आंतरिक संचालन का प्रभारी है, इस Testing की नींव के रूप में कार्य करता है।	Testing के लिए उच्च-स्तरीय डेटाबेस और डेटाफ्लो आरेख का उपयोग किया जाता है।

तालिका 7.1.2 "Black Box" Testing, "White Box" Testing और "Gray Box" Testing की तुलना

सारांश

- सूचना स्टोरेज और पुनर्प्राप्ति डेटा को व्यवस्थित रूप से इकट्ठा करने और सूचीबद्ध करने की प्रक्रिया है ताकि इसे मांग पर स्थित और दिखाया जा सके।
- डेटा निष्कर्षण अंतर्दृष्टिपूर्ण व्यावसायिक जानकारी प्राप्त करने या केंद्रीय डेटा वेयरहाउस में स्टोरेज के लिए बाद में प्रसंस्करण और विश्लेषण के लिए विभिन्न डेटा स्रोतों से डेटा प्राप्त करने की प्रक्रिया है।
- ETL (एक्सट्रैक्ट, ट्रांसफॉर्म और लोड) प्रक्रिया में पहला कदम डेटा निष्कर्षण है।
- डेटा निष्कर्षण तकनीकों की दो मुख्य श्रेणियां तार्किक और भौतिक हैं।
- मैनुअल Testing डेटा पीढ़ी में आपकी क्यूए टीम की सहायता से, या डेवलपर्स द्वारा स्वतंत्र रूप से test डेटा का उत्पादन करना शामिल है।
- एक परिणाम जो प्रत्याशित से भिन्न होता है उसे विसंगति के रूप में जाना जाता है। यह व्यवहार किसी दस्तावेज़ या Tester की धारणाओं और अनुभवों का परिणाम हो सकता है।
- NET का उपयोग डेस्कटॉप ऐप, मोबाइल ऐप, गेम, वेब ऐप और IoT डिवाइस सहित विभिन्न प्रकार के एप्लिकेशन बनाने के लिए किया जा सकता है।
- डोमेन-विशिष्ट भाषा SQL (संरचित क्वेरी भाषा) का उपयोग डेटाबेस में डेटा संग्रहीत, हेरफेर और पुनर्प्राप्त करने के लिए किया जाता है।
- स्टेटिक Testing सॉफ्टवेयर Testing की एक विधि है जिसमें पूरे प्रोग्राम को चलाना या सॉफ़्टवेयर निष्पादित करना शामिल नहीं है।
- एक प्रकार का सॉफ्टवेयर Testing जिसे Dynamic Testing कहा जाता है, सॉफ्टवेयर के कोड के गतिशील व्यवहार की पूरी तरह से जांच करता है।
- आंतरिक संरचना, डिजाइन, कोडिंग और सभी घटनाओं का Test सॉफ्टवेयर Testing तकनीक के हिस्से के रूप में किया जाता है जिसे "White Box Testing" के रूप में जाना जाता है।
- यह Testing की एक विधि है जिसमें एप्लीकेशन के आंतरिक कामकाज ज्ञात नहीं होते हैं।
- White Box + Black Box = Gray Box एक तकनीक है जिसका उपयोग किसी एप्लिकेशन का Test करने के लिए किया जाता है, जबकि यह केवल एक सरसरी समझ रखता है कि यह आंतरिक रूप से कैसे संचालित होता है और सिस्टम की सामान्य समझ है।

एक्टिविटी

Con-video session

- इस सत्र में, ट्रेनर एक वीडियो चलाएगा।
- वीडियो डेटा निष्कर्षण और इसके उपयोग के बारे में सीख रहा होगा।
- वीडियो का यूट्यूब लिंक h ttps://www.youtube.com/watch?v=O6GFM_uGVFY है
- प्रशिक्षु पिन ड्रॉप साइलेंस के साथ वीडियो का अवलोकन करेंगे।
- वे वीडियो से पॉइंटर्स को नोट कर सकते हैं जो उन्हें प्रासंगिक लग सकते हैं।
- प्रशिक्षु कक्षा में शिष्टाचार बनाए रखेंगे और कक्षा में बात नहीं करेंगे, कानाफूसी नहीं करेंगे या चर्चा नहीं करेंगे। किसी भी प्रश्न या भ्रम के मामले में, प्रशिक्षु उन्हें अपनी नोटबुक में लिखेंगे।

चर्चा सत्र

- प्रशिक्षक पूछताछ करेगा कि क्या प्रशिक्षुओं ने अध्याय को समझा है।
- इस गतिविधि में, प्रशिक्षक प्रशिक्षु से विषय से संबंधित कुछ प्रश्न पूछेगा।
- प्रशिक्षु विषय की अपनी समझ के अनुसार प्रश्नों का उत्तर देंगे।
- यदि प्रशिक्षुओं के मन में प्रश्न और भ्रम हैं; वे ट्रेनर के सामने उन लोगों को सामने रख सकते हैं।
- प्रशिक्षक यह सुनिश्चित करेगा कि वह प्रशिक्षुओं द्वारा रखे गए प्रश्नों के उचित उत्तर दे सके।

अभ्यास

A. **निम्नलिखित प्रश्नों के उत्तर दीजिए**

1. Black Box Testing बनाम White Box Testing बनाम Gray Box Testing के बीच अंतर करें।
2. संक्षिप्त Static Testing और Dynamic Testing।

B. **रिक्त स्थानों की पूर्ति कीजिए**

a) _____ETL (एक्सट्रैक्ट, ट्रांसफॉर्म और लोड) प्रक्रिया का पहला चरण है।

b) _____एक सॉफ्टवेयर Testing तकनीक है जिसमें सॉफ्टवेयर का Test पूरे प्रोग्राम के बिना किया जाता है या सॉफ्टवेयर को चलाया या निष्पादित किया जा रहा है।

c) _____एक परिणाम को संदर्भित करता है जो अपेक्षित से अलग है।

d) _____एक डोमेन-विशिष्ट भाषा है जिसका उपयोग डेटाबेस में डेटा को संग्रहीत करने, हेरफेर करने और पुनर्प्राप्त करने के लिए किया जाता है।

नोट्स

संबंधित वीडियो देखने के लिए क्यूआर कोड को स्कैन करें

https://www.youtube.com/watch?v=URmCWIL0IA0

Testing Suite डिजाइन करने के क्षेत्र में नवीनतम परिवर्तन, प्रक्रियाएं और अभ्यास

Skill India
कौशल भारत-कुशल भारत

सत्यमेव जयते
GOVERNMENT OF INDIA
MINISTRY OF SKILL DEVELOPMENT
& ENTREPRENEURSHIP

N·S·D·C
National
Skill Development
Corporation
Transforming the skill landscape

8. सॉफ्टवेयर उत्पादों/ ऍप्लिकेशन्स/मॉड्यूल पर मैनुअल Testing करना

IT - ITeS SSC
nasscom

यूनिट 8.1 - सॉफ्टवेयर उत्पादों/ऍप्लिकेशन्स/मॉड्यूल पर मैनुअल Testing करना

SSC/N1301

प्रमुख सीखने के परिणाम

इस मॉड्यूल के अंत तक, प्रतिभागी निम्न में सक्षम होंगे:

1. किए जाने वाले Testing की प्रकृति और उपयोग किए जाने वाले Testing प्रबंधन उपकरण के उपयोग की समझ प्रदर्शित करें।
2. प्रयोज्य Test cases में प्रयोज्य दिशानिर्देशों के अनुरूप पालन के उपयोग का मूल्यांकन करें।

यूनिट 8.1: सॉफ्टवेयर उत्पादों/ऍप्लिकेशन्स/मॉड्यूलों पर मैनुअल Testing करना

यूनिट के उद्देश्य

इस यूनिट के अंत तक, प्रतिभागी निम्न में सक्षम होंगे:

1. Test cases और स्वचालित स्क्रिप्ट के नवीनतम संस्करणों का चयन करें.
2. सॉफ्टवेयर विनिर्देश दस्तावेज़ से सही वैकल्पिक समाधान का चयन करें।
3. Testing के लिए उपयोग किए जाने वाले एप्लिकेशन और डेटा के सही संस्करणों की पहचान करें।
4. सॉफ्टवेयर विनिर्देश दस्तावेज़ से आवश्यकताओं का विश्लेषण करें।
5. Testing योजना बनाने और Test cases को विकसित करने की प्रक्रिया का प्रदर्शन करें।
6. सहमत Testing प्रबंधन उपकरण का उपयोग करके Testing प्रगति रिपोर्ट, परिणाम और खोजे गए डिफेक्ट्स का विकास करना।
7. परियोजना के दौरान जोखिम ट्रिगर्स की निगरानी के लिए आकस्मिक योजनाओं का प्रदर्शन करें।
8. डिफेक्ट्स की पहचान करने और डिफेक्ट्स ट्रैकिंग सिस्टम में उसी को ट्रैक करने के लिए परिणामों का विश्लेषण करें।

8.1.1 मैनुअल Testing

सॉफ़्टवेयर Testing जो स्वचालित उपकरण के बजाय Test cases के मैन्युअल निष्पादन का उपयोग करता है, मैनुअल Testing के रूप में जाना जाता है। सभी Test cases को Tester द्वारा अंतिम उपयोगकर्ता के दृष्टिकोण से मैन्युअल रूप से चलाया जाता है। आवश्यकता दस्तावेज़ में उल्लिखित आवश्यकताओं के साथ आवेदन का अनुपालन इसके द्वारा निर्धारित किया जाता है। लगभग 100% सॉफ़्टवेयर एप्लिकेशन को समाप्त करने के लिए, Test Case बनाए और कार्यान्वित किए जाते हैं।

सबसे मौलिक Testing तकनीकों में से एक मैनुअल Testing है, जिसमें बाहरी और आंतरिक दोनों सॉफ्टवेयर डिफेक्ट्स को खोजने की क्षमता है। एक गलती तब होती है जब प्रोग्राम द्वारा उत्पादित आउटपुट जो इरादा था उससे भिन्न होता है। Testing के एक और दौर के लिए उन्हें Tester को वापस देने से पहले, डेवलपर ने बग को ठीक कर दिया।

किसी भी नए बनाए गए सॉफ़्टवेयर या उत्पाद को स्वचालित Testing के अधीन होने से पहले मैनुअल Testing से गुजरना होगा। हालांकि इसमें बहुत समय और प्रयास लगता है, यह Testing सुनिश्चित करता है कि अंतिम उत्पाद बग-मुक्त है। जबकि स्वचालित Testing सॉफ़्टवेयर को किसी पूर्व ज्ञान की आवश्यकता नहीं होती है, मैनुअल Testing करता है।

8.1.2 मैनुअल Testing का महत्व क्या है?

हालांकि मैनुअल Testing के अभी भी कई कारण हैं, सॉफ्टवेयर पेशेवर अधिक से अधिक स्वचालित Testing का पक्ष ले रहे हैं। कुछ हैं:

मानव परिप्रेक्ष्य: मनुष्य इसे एक त्वरित नज़र देकर ऐप की मौलिक उपयोगिता और उपस्थिति का त्वरित आकलन कर सकते हैं। एक Tester प्रयोज्य मुद्दों और उपयोगकर्ता इंटरफ़ेस डिफेक्ट्स को देख सकता है जब वे सॉफ़्टवेयर के साथ उसी तरह से बातचीत करते हैं जैसे उपयोगकर्ता करता है। ये समस्याएँ स्वचालित Testing स्क्रिप्ट द्वारा ढूँढा नहीं जा सकता।

सिस्टम वर्कफ़्लोज़ का एक बड़ा दृश्य: ऐप का अधिक व्यापक दृश्य हमेशा मैन्युअल सत्यापन द्वारा प्रदान किया जाता है। प्रक्रियाओं को दोहराने वाले कोडिंग मोड में होने के बजाय, मानव मन लगातार खोज कर रहा है। इसलिए, यह सिस्टम सत्यापन के लिए अधिक से अधिक जमीन को कवर करेगा।

- **मानव परिप्रेक्ष्य:** मनुष्य इसे एक त्वरित नज़र देकर ऐप की मौलिक उपयोगिता और उपस्थिति का त्वरित आकलन कर सकते हैं। एक Tester प्रयोज्य मुद्दों और उपयोगकर्ता इंटरफ़ेस डिफेक्ट्स को देख सकता है जब वे सॉफ़्टवेयर के साथ उसी तरह से बातचीत करते हैं जैसे उपयोगकर्ता करता है। ये समस्याएँ स्वचालित Testing स्क्रिप्ट द्वारा ढूँढा नहीं जा सकता।
- **सिस्टम वर्कफ़्लोज़ का एक बड़ा दृश्य:** ऐप का अधिक व्यापक दृश्य हमेशा मैन्युअल सत्यापन द्वारा प्रदान किया जाता है। प्रक्रियाओं को दोहराने वाले कोडिंग मोड में होने के बजाय, मानव मन लगातार खोज कर रहा है। इसलिए, यह सिस्टम सत्यापन के लिए अधिक से अधिक जमीन को कवर करेगा।
- स्वचालन में पैसा खर्च होता है क्योंकि यह अक्सर Testing के दौरान False Positives और False Negatives परिणाम देता है। Testing प्रक्रिया के दौरान मानवीय स्पर्श को शामिल करके, इन त्रुटियों से बचा जाता है।

ऐसे परिदृश्य जो स्वचालित नहीं हैं या स्वचालन के साथ Testing करते समय उपयोगकर्ता व्यवहार में स्पष्ट विश्वास प्रदान नहीं करते हैं, उदाहरण के लिए, मोबाइल उपकरणों पर " Tap & Pay", टूल बनाम मैन्युअल रूप से सत्यापित का उपयोग करके स्वचालित होने पर अलग-अलग व्यवहार होते हैं। इसके बावजूद, मैनुअल Testing अभी भी सॉफ्टवेयर विकास Life Cycle के त्वरित-चलती सत्यापन चरण में एक प्रमुख स्थान रखता है। इसके अतिरिक्त, ऐसे उदाहरण हैं जहां मैन्युअल सत्यापन सबसे अच्छा विकल्प है।

स्वचालन महंगा है, अक्सर स्वचालित Testing False Positive और False Negative उत्पन्न कर सकता है। Testing प्रक्रिया के दौरान मानवीय स्पर्श को शामिल करके, इन त्रुटियों से बचा जाता है।

8.1.3 Platform और एप्लीकेशन संस्करण

प्रबंधन टीम टीम का कार्य लक्ष्य प्लेटफ़ॉर्म और संगत ऍप्लिकेशन्स का चयन करना है क्योंकि ऑपरेटिंग सिस्टम, वेब ब्राउज़र, संगत संस्करण और अन्य सुविधाएँ ग्राहक के दृष्टिकोण के आधार पर डिज़ाइन की गई हैं। प्लेटफ़ॉर्म और एप्लिकेशन संस्करण विकास और Testing टीमों को यह तय करने में सक्षम बनाते हैं कि क्या किया जाना है।

उदाहरण: अधिकांश सॉफ़्टवेयर पैकेज या स्टार्ट अप स्क्रीन पर आप निम्नलिखित में आ सकते हैं:

- सबसे अच्छा काम करता है
- Windows XP या अधिक की आवश्यकता है
- केवल Unix या Linux 2.6.10 के साथ उपयोग के लिए

संगतता Testing में कई ब्राउज़रों, ऑपरेटिंग सिस्टम और हार्डवेयर के साथ किसी एप्लिकेशन या वेबसाइट की संगतता की जांच करना शामिल है। यह Testing मौजूदा वातावरण पर या तो मैन्युअल रूप से या स्वचालित आधार पर आयोजित किया जाता है।

1. एकाधिक संस्करणों का Test

- प्लेटफार्मों और सॉफ्टवेयर ऍप्लिकेशन्स के विभिन्न संस्करणों का Test करना एक चुनौतीपूर्ण कार्य है। अब हम एक ऐसी स्थिति पर विचार करेंगे जहां एक लोकप्रिय ऑपरेटिंग सिस्टम पर संगतता Testing किया जाना है। प्रोग्रामर ने कई बग तय किए हैं और मौजूदा कोड में नई सुविधाओं को जोड़कर प्रदर्शन में भी सुधार किया है। ऑपरेटिंग सिस्टम के वर्तमान संस्करण के लिए हजारों मौजूदा प्रोग्राम हैं। परियोजना का अंतिम उद्देश्य 100% संगतता सुनिश्चित करना है। Tester की नौकरी को कम करने के लिए तुल्यता विभाजन उचित रूप से लागू किया जाता है।
- संगतता Testing का कार्य सॉफ्टवेयर के सभी संभावित संयोजनों के तुल्यता विभाजन के साथ शुरू होता है। यह सुनिश्चित करने के लिए किया जाता है कि तुल्यता सेट सॉफ्टवेयर के बीच बातचीत की सटीकता को सत्यापित करते हैं। यद्यपि कोई ऑपरेटिंग सिस्टम पर सभी संभावित सॉफ़्टवेयर प्रोग्रामों का Test कर सकता है, केवल सबसे महत्वपूर्ण प्रोग्रामों को अंतिम रूप दिया जाता है और उनका Test किया जाता है।
- आधारभूत संस्करण, मध्यवर्ती संस्करण और संशोधन को तीन प्राथमिक प्रकार के सॉफ़्टवेयर कॉन्फ़िगरेशन रिलीज़ माना जाता है।

2. आधारभूत संस्करण

सिस्टम के विकास या संचालन की शुरुआत में, बेसलाइन सॉफ्टवेयर कॉन्फ़िगरेशन संस्करणों की योजना बनाई जाती है। वे अपने SCI के साथ प्रक्रिया के हिस्से के रूप में समीक्षा, Testing और अनुमोदन से गुजरते हैं। बेसलाइन संस्करण आगे सिस्टम विकास के लिए शुरुआती बिंदु का प्रतिनिधित्व करते हैं और सॉफ्टवेयर सिस्टम के Life Cycle में माइलस्टोन के रूप में कार्य करते हैं।

3. मध्यवर्ती संस्करण

जब ऐसी समस्याएं उत्पन्न होती हैं जिन पर तत्काल ध्यान देने की आवश्यकता होती है - जैसे कि एक महत्वपूर्ण SCI में पहचाने गए डिफेक्ट्स को ठीक करने की आवश्यकता, या एक नए ग्राहक के साथ अनुबंध में परिभाषित तत्काल अनुकूलन करना - सॉफ्टवेयर का एक मध्यवर्ती संस्करण अक्सर तैयार किया जाता है। आमतौर पर, मध्यवर्ती संस्करणों का उपयोग केवल कंपनी के ग्राहकों के एक हिस्से द्वारा किया जाता है और केवल थोड़े समय के लिए एक नए बेसलाइन संस्करण द्वारा प्रतिस्थापित किए जाने से पहले। आमतौर पर, मध्यवर्ती संस्करणों का उपयोग केवल कंपनी के ग्राहकों के एक हिस्से द्वारा किया जाता है और केवल थोड़े समय के लिए एक नए बेसलाइन संस्करण द्वारा प्रतिस्थापित किए जाने से पहले। स्वाभाविक रूप से, हम अनुमान लगा सकते हैं कि इन संस्करणों को आमतौर पर आधारभूत संस्करणों के प्रकाशन में लगाए गए फोकस और प्रयास प्राप्त नहीं होंगे। एक मध्यवर्ती सॉफ्टवेयर कॉन्फ़िगरेशन संस्करण इस प्रकार अगले आधारभूत संस्करण के लिए "पायलट" या स्प्रिंगबोर्ड के रूप में काम कर सकता है।

सॉफ़्टवेयर का एक अंतरिम संस्करण अक्सर तब बनाया जाता है जब ऐसे मुद्दे उत्पन्न होते हैं जो तत्काल ध्यान देने की मांग करते हैं, जैसे कि एक महत्वपूर्ण SCI में पाई गई त्रुटियों को ठीक करने या एक नए ग्राहक के साथ अनुबंध द्वारा आवश्यक त्वरित अनुकूलन करने की आवश्यकता। आमतौर पर, मध्यवर्ती संस्करणों का उपयोग केवल एक नए बेसलाइन संस्करण द्वारा प्रतिस्थापित किए जाने से पहले थोड़े समय के लिए कंपनी के ग्राहकों के एक छोटे प्रतिशत द्वारा किया जाता है। स्वाभाविक रूप से, हम अनुमान लगा सकते हैं कि इन संस्करणों को आमतौर पर आधारभूत संस्करणों के प्रकाशन में लगाए गए फोकस और प्रयास प्राप्त नहीं होंगे। इस प्रकार, एक मध्यवर्ती सॉफ्टवेयर कॉन्फ़िगरेशन संस्करण निम्नलिखित आधारभूत संस्करण के लिए "पायलट" या लॉन्च पैड के रूप में कार्य कर सकता है।

8.1.4 SCI और सॉफ्टवेयर संस्करणों की पहचान के लिए गणना सम्मेलन

SCI की पहचान करने के लिए सबसे व्यापक रूप से इस्तेमाल किया जाने वाला संख्याकरण सम्मेलन दशमलव संख्याकरण है, जो बढ़ते संस्करण और संशोधन संख्याओं को इंगित करता है और तदनुसार पंजीकृत किया जाता है। DD7- Ver.1.0, DD7- Ver.1.1, DD- 7 Ver.2.0, DD7- Ver.3.0, DD7- Ver.3.1, DD7- Ver.3.2, आदि, जहां पहली संख्या संस्करण को दर्शाती है और दूसरी संशोधन, SCI डिज़ाइन दस्तावेज़ों के उदाहरण हैं जिनके कई संस्करण और संशोधन हो सकते हैं।

संस्करण नियंत्रण प्रणाली का उपयोग:

- **Repository:** इसकी तुलना संशोधनों के डेटाबेस से की जा सकती है। इसमें परियोजना के सभी संशोधन और पिछले पुनरावृत्तियों (स्नैपशॉट) शामिल हैं।
- **कार्य की प्रतिलिपि (कभी-कभी चेकआउट कहा जाता है):** यह प्रत्येक प्रोजेक्ट की फाइलों की व्यक्तिगत प्रति है। समाप्त होने पर, आप अपने परिवर्तनों को एक रिपॉजिटरी में प्रतिबद्ध कर सकते हैं और दूसरों के काम को प्रभावित किए बिना इस प्रतिलिपि को संपादित करना जारी रख सकते हैं।
- **एक समूह में काम करना:** कल्पना कीजिए कि आप एक कंपनी द्वारा नियोजित हैं और वर्तमान परियोजना पर काम करने के लिए आवश्यक हैं। चूंकि मुख्य कोड पहले से ही उपयोग में है और उपयोगकर्ता को कोई असुविधा पैदा किए बिना बदला नहीं जा सकता है, इसलिए आपको उनके परिवर्तनों को सहयोग करने और शामिल करने के लिए अपनी टीम के साथ काम करना चाहिए। संस्करण नियंत्रण का उपयोग करने से आपको बिना किसी अनपेक्षित परिवर्तन के विभिन्न अनुरोधों को मुख्य Repository में विलय करने में सहायता मिलेगी। कार्यक्षमता का Test करने के लिए आपको हर बार डाउनलोड और सेट करने की आवश्यकता नहीं है; आप बस परिवर्तनों को खींच सकते हैं, परिवर्तन कर सकते हैं, उनका Test कर सकते हैं और फिर उन्हें वापस मर्ज कर सकते हैं। इसे चित्रित किया जा सकता है।

कल्पना कीजिए कि आप एक कंपनी द्वारा नियोजित हैं और आपको वर्तमान परियोजना पर काम करने की आवश्यकता है। चूंकि मुख्य कोड पहले से ही उपयोग में है और उपयोगकर्ता को कोई असुविधा पैदा किए बिना बदला नहीं जा सकता है, इसलिए आपको उनके परिवर्तनों को सहयोग करने और शामिल करने के लिए अपनी टीम के साथ काम करना चाहिए। आप बिना किसी अनपेक्षित परिवर्तन के संस्करण नियंत्रण की सहायता से विभिन्न अनुरोधों को मुख्य Repository में मर्ज कर सकते हैं। कार्यक्षमता का Test करने के लिए आपको हर बार डाउनलोड और सेट करने की आवश्यकता नहीं है; आप बस परिवर्तनों को खींच सकते हैं, परिवर्तन कर सकते हैं, उनका Test कर सकते हैं और फिर उन्हें वापस मर्ज कर सकते हैं। इसे इस प्रकार देखा जा सकता है।

8.1.5 संस्करण हिस्ट्री

आपके एंटरप्राइज़ testr रिपॉजिटरी में प्रत्येक घटक एक संस्करण हिस्ट्री के साथ आता है। वर्जनिंग एंटरप्राइज़ testr की एक विशेषता है जो मानक आती है और, बेसलाइन की तरह, संस्करण नियंत्रण Repository पर निर्भर नहीं है। जब भी कोई नया तत्व, जैसे कि आवश्यकता या Testing स्क्रिप्ट जोड़ा जाता है, तो एक प्रारंभिक संस्करण का उत्पादन किया जाता है।

जैसे ही आप परिवर्तन करते हैं, तत्व का एक नया वृद्धिशील संस्करण स्वचालित रूप से बन जाता है. अपनी आवश्यकता या Testing स्क्रिप्ट देखते समय संस्करण टैब का चयन करके, आप सभी संस्करणों को देख सकते हैं और वर्तमान संस्करण के साथ चुने गए संस्करण के घटकों की तुलना कर सकते हैं। आप किसी भी समय किसी तत्व के पिछले संस्करण पर वापस जाने का चयन करके किसी भी समय किसी पुराने संस्करण को जल्दी से पुनर्स्थापित कर सकते हैं। जब आप किसी पुराने संस्करण पर वापस रोल करते हैं, तो एक नया संस्करण सबसे हाल के संस्करण के रूप में बनाया जाता है, और पिछले सभी संस्करण—जिनमें वे संस्करण भी शामिल हैं जिन्हें आप वापस रोल कर रहे हैं—के बाद बनाए गए हैं—भी रखे जाते हैं। यह आपको Repository परिवर्तनों को विनियमित करने के लिए एक बहुत मजबूत तंत्र देता है।

8.1.6 मैनुअल Test case लिखना

Test cases Tester को चरणों के अनुक्रम के माध्यम से मार्गदर्शन करने में मदद करते हैं ताकि यह सत्यापित किया जा सके कि क्या कोई सॉफ़्टवेयर एप्लिकेशन बग से मुक्त है, और एंड-यूज़र द्वारा आवश्यकतानुसार काम कर रहा है। स्पष्ट रूप से लिखने की क्षमता, विस्तार पर ध्यान देना, और Testing (AUT) के तहत आवेदन की ठोस समझ रखना सॉफ्टवेयर के लिए Test cases को लिखना सीखते समय सभी आवश्यक हैं। सॉफ्टवेयर के लिए Test cases को लिखना सीखने के लिए बुनियादी लेखन कौशल, विस्तार पर ध्यान देने और Testing (एयूटी) के तहत आवेदन की अच्छी समझ की आवश्यकता होती है।

एक Testing Suite आमतौर पर किसी विशेष मॉड्यूल या एप्लिकेशन के अनुभाग के लिए Test cases से बना होता है। आमतौर पर Testing किए जाने के लिए कई विशिष्ट परिदृश्य होते हैं, इसलिए एक Testing सत्र में आमतौर पर कई Test Case शामिल होंगे।

किसी भी Tester को एक अच्छी तरह से लिखित Test Case का उपयोग करके Testing को समझने और करने में सक्षम होना चाहिए।

Test cases को लिखते समय उपयोगकर्ता के दृष्टिकोण पर विचार करना और सभी आवश्यक जानकारी शामिल करना महत्वपूर्ण है। आप गुणवत्ता Test cases को सामने लिखने के लिए अतिरिक्त प्रयास करके लंबे समय में समय और प्रयास बचाएंगे। एक अच्छी तरह से Testing किए गए आवेदन को एक से अलग किया जा सकता है जिसे अच्छी तरह से लिखित Test cases द्वारा पूरी तरह से Testing नहीं किया गया है।

Testing के मामलों को लिखना - विशेष रूप से उनमें से एक बार में उच्च मात्रा - एक समय लेने वाला कार्य हो सकता है। TestLodge के साथ संगठित, उच्च-गुणवत्ता वाले Test cases को लिखना और बनाए रखना सरल है। प्रभावी Test cases को लिखना और उन्हें व्यवस्थित रखना testलॉज के साथ सरल है। यहां, हमने Test cases को लिखने के लिए कुछ दिशानिर्देश प्रदान किए हैं। इस लेख के अंत में, हमने एक नमूना Test Case भी शामिल किया है।

8.1.7 सॉफ्टवेयर के लिए Test case कैसे लिखें

- **एक मजबूत शीर्षक का प्रयोग करें**

  मजबूत शीर्षक एक अच्छे Test Case की नींव हैं। सर्वोत्तम अभ्यास के रूप में, आपके द्वारा Testing किए जा रहे मॉड्यूल के बाद Test Case का नाम देना उचित है। उदाहरण के लिए, अगर आप लॉग इन पेज का Test कर रहे हैं, तो Test case का शीर्षक "लॉग इन पेज" होना चाहिए. यदि आप जिस उपकरण का उपयोग कर रहे हैं वह पहले से ऐसा नहीं करता है, तो कुछ परिस्थितियों में Test Case के शीर्षक में एक विशिष्ट पहचानकर्ता को शामिल करना समझ में आ सकता है ताकि इसे लंबे शीर्षक के बजाय संदर्भित किया जा सके।

- **एक मजबूत विवरण शामिल करें**

  Tester को इस बात से अवगत कराया जाना चाहिए कि वे विवरण में क्या Testing करेंगे। अन्य प्रासंगिक जानकारी, जैसे Testing वातावरण, test डेटा, और पूर्व शर्तें और धारणाएं, कभी-कभी इस खंड में शामिल की जा सकती हैं। एक विवरण को तुरंत पढ़ने और Testing के मुख्य उद्देश्य को व्यक्त करने के लिए सरल होना चाहिए।

- **मान्यताओं और पूर्व शर्तों को शामिल करें**

  Testing निष्पादित करने से पहले पूरी की जाने वाली किसी भी पूर्व शर्त को शामिल करें, साथ ही Testing पर लागू होने वाली कोई भी धारणा। इस जानकारी में वह पृष्ठ शामिल हो सकता है जिस पर उपयोगकर्ता को Testing लॉन्च करना चाहिए, पर्यावरण निर्भरताएं, और कोई भी अद्वितीय सेटअप आवश्यकताएं जिन्हें Testing लॉन्च करने से पहले पूरा किया जाना चाहिए। ये विवरण Testing प्रक्रियाओं को संक्षिप्त और बिंदु तक रखने में सहायता करते हैं।

- **Testing चरणों को स्पष्ट और संक्षिप्त रखें**
  सरल Test Case आदर्श हैं। याद रखें कि जरूरी नहीं कि Test case लिखने वाला व्यक्ति भी test चलाएगा। Testing करने के तरीके के बारे में आवश्यक डेटा और निर्देश Testing चरणों में शामिल किए जाने चाहिए। यह संभवतः एक Test Case का सबसे महत्वपूर्ण घटक है। किसी भी महत्वपूर्ण जानकारी को न छोड़ें, लेकिन इस खंड को संक्षिप्त और स्पष्ट रखें। Test Case को इस तरह से बनाएं जिससे कोई भी Testing निष्पादित कर सके।
- **अपेक्षित परिणाम शामिल करें**
  अपेक्षित परिणाम बताता है कि Testing प्रक्रियाओं के परिणामस्वरूप Tester को क्या सामना करना चाहिए। Tester इस पद्धति का उपयोग यह तय करने के लिए करता है कि Test Case "pass" या "fail" है या नहीं।
- **इसे पुन: प्रयोज्य बनाएं**
  एक अच्छा Test Case पुन: प्रयोज्य है और सॉफ्टवेयर Testing टीम को दीर्घकालिक मूल्य प्रदान करता है। Test Case का मसौदा तैयार करते समय इस पर विचार करें। स्क्रैच से लिखने के विरोध में Testing केस का पुन: उपयोग करने से भविष्य में आपका समय बचेगा।
- **नमूना Test Case**

यहां एक Test Case का एक उदाहरण है:

- शीर्षक: लॉगिन पृष्ठ - gmail.com पर सफलतापूर्वक प्रमाणित करें
- विवरण: एक पंजीकृत उपयोगकर्ता gmail.com पर सफलतापूर्वक लॉग इन करने में सक्षम होना चाहिए।
- पूर्व शर्त: उपयोगकर्ता को पहले से ही एक ईमेल पते और पासवर्ड के साथ पंजीकृत होना चाहिए।
- धारणा: एक समर्थित ब्राउज़र का उपयोग किया जा रहा है।

Testing चरण:

- gmail.com पर नेविगेट करें
- 'ईमेल' फ़ील्ड में, पंजीकृत उपयोगकर्ता का ईमेल पता दर्ज करें।
- 'next' बटन पर क्लिक करें।
- पंजीकृत उपयोगकर्ता का पासवर्ड दर्ज करें
- 'साइन इन' पर क्लिक करें

8.1.8 Testing प्रगति रिपोर्ट

Test Report, एक दस्तावेज होता है, जिसमें Testing परियोजना की सभी Testing गतिविधियों और अंतिम Testing परिणामों का सारांश निहित होता है। एक Test Report मूल्यांकन करती है कि Testing कितनी अच्छी तरह किया जाता है। एक Test Report मूल्यांकन करती है कि Testing कितनी अच्छी तरह किया जाता है। Test Report के आधार पर, हितधारक Testing किए गए उत्पाद की गुणवत्ता का मूल्यांकन कर सकते हैं और सॉफ्टवेयर रिलीज पर निर्णय ले सकते हैं।

उदाहरण के लिए, यदि Test Report से पता चलता है कि उत्पाद में अभी भी कई खामियां हैं, तो हितधारक तब तक रिलीज को रोकने का निर्णय ले सकते हैं जब तक कि हर डिफेक्ट्स को संबोधित नहीं किया जाता है।

Test Report हितधारक और Testing प्रबंधक के बीच संचार के एक चैनल के रूप में कार्य करती है। हितधारक Test Report के माध्यम से परियोजना की स्थिति, उत्पाद की गुणवत्ता और अन्य चीजों को समझ सकता है।

इसके बाद का परिदृश्य गुणवत्ता Test Report की आवश्यकता को प्रदर्शित करता है

एक अच्छी Test Report में निम्नलिखित शामिल होना चाहिए:

- **विस्तार:** आपको Testing गतिविधि का संपूर्ण विवरण देना चाहिए और आपके द्वारा किए गए Testings को सूचीबद्ध करना चाहिए। अपनी रिपोर्ट में abstract जानकारी शामिल करने से बचें क्योंकि पाठक को यह भ्रमित करने वाला लगेगा।
- **स्पष्टता:** Test Report में केवल संक्षिप्त, आसानी से समझी जाने वाली जानकारी होनी चाहिए।
- **स्टैंडर्ड:** Test Report निर्धारित प्रारूप का पालन करना चाहिए। कई परियोजनाओं में, हितधारकों के लिए Test Report के बीच स्थिरता की समीक्षा और पुष्टि करना आसान है।
- **निश्चित:** परियोजना गतिविधि के बारे में एक निबंध लिखने से बचें। परीक्षा परिणाम विनिर्देश का वर्णन और सारांश करते समय मुख्य बिंदु पर ध्यान दें।

8.1.9 Test Report के प्रकार

Testing योजना, Testing विनिर्देश और Testing रिपोर्टिंग सभी सॉफ्टवेयर Testing प्रलेखन के IEEE मानक में उल्लिखित दस्तावेजों द्वारा कवर किए गए हैं।

test रिपोर्टिंग में चार दस्तावेज़ प्रकार शामिल हैं:

- यदि Testing निष्पादन के लिए एक औपचारिक शुरुआत वांछित है, तो एक test आइटम ट्रांसमिटल रिपोर्ट Testing के लिए Testing समूह को विकास से भेजे जा रहे Testing आइटम की पहचान करती है।
  - रिपोर्ट में शामिल किए जाने वाले विवरण - उद्देश्य, रूपरेखा, प्रेषण-रिपोर्ट पहचानकर्ता, प्रेषित आइटम, स्थान, स्थिति और अनुमोदन।
- Testing टीम द्वारा Testing निष्पादन के दौरान क्या हुआ यह रिकॉर्ड करने के लिए एक test लॉग का उपयोग किया जाता है
  - रिपोर्ट में शामिल किए जाने वाले विवरण - उद्देश्य, रूपरेखा, Testing-लॉग पहचानकर्ता, विवरण, गतिविधि और घटना प्रविष्टियाँ, निष्पादन विवरण, प्रक्रिया परिणाम, पर्यावरण जानकारी, विषम घटनाएँ, घटना-रिपोर्ट पहचानकर्ता।
  - **रिपोर्ट में शामिल किए जाने वाले विवरण -** उद्देश्य, रूपरेखा, Testing-घटना-रिपोर्ट पहचानकर्ता, सारांश, प्रभाव।

- एक Testing सारांश रिपोर्ट एक या अधिक Testing-डिज़ाइन विनिर्देशों से जुड़ी Testing गतिविधियों को सारांशित करती है।
  - o **रिपोर्ट में शामिल किए जाने वाले विवरण -** उद्देश्य, रूपरेखा, Testing-सारांश-रिपोर्ट, पहचानकर्ता, सारांश, भिन्नता, व्यापकता मूल्यांकन, परिणामों का सारांश, गतिविधियों का सारांश और अनुमोदन।

8.1.10 डिफेक्ट समाधान

सॉफ्टवेयर Testing में डिफेक्ट्स समाधान डिफेक्ट्स को ठीक करने की एक चरण दर चरण प्रक्रिया है। सॉफ्टवेयर Testing में डिफेक्ट्स समाधान धीरे-धीरे खामियों को ठीक करने की प्रक्रिया है। डिफेक्ट्स समाधान प्रक्रिया में पहला कदम डेवलपर को डिफेक्ट्स निर्दिष्ट कर रहा है। प्राथमिकता के अनुसार डिफेक्ट्स फिक्स शेड्यूल करने के बाद, डिफेक्ट्स ठीक किया गया है, और उसके बाद डेवलपर Testing प्रबंधक को समाधान की एक रिपोर्ट भेजता है। यह प्रक्रिया डिफेक्ट्स को आसानी से ठीक करने और ट्रैक करने में मदद करती है।

डिफेक्ट्स को ठीक करने के लिए आप निम्न चरणों का पालन कर सकते हैं

- **असाइनमेंट:** स्थिति को प्रतिसाद में परिवर्तित किया और समस्या को डेवलपर या अन्य तकनीशियन को असाइन किया.
- **शेड्यूल के अनुसार:** इस चरण को डेवलपर पक्ष द्वारा नियंत्रित किया जाता है। डिफेक्ट्स की प्राथमिकता के आधार पर, वे उन्हें ठीक करने के लिए एक कार्यक्रम विकसित करेंगे।
- **डिफेक्ट्स को ठीक करें:** Testing प्रबंधक उपरोक्त अनुसूची की तुलना में डिफेक्ट्स को ठीक करने की प्रक्रिया की निगरानी करता है जबकि विकास दल डिफेक्ट्स को ठीक कर रहा है।
- **समाधान की रिपोर्ट करें:** जब बग ठीक हो जाते हैं, तो डेवलपर्स से समाधान की रिपोर्ट का अनुरोध करें।
- **सत्यापन:** Testing टीम पुष्टि करती है कि विकास टीम द्वारा तय किए जाने और उन्हें रिपोर्ट करने के बाद डिफेक्ट्स को ठीक कर दिया गया है।
- **समापन:** एक डिफेक्ट्स को ठीक करने और सत्यापित करने के बाद इसकी बंद स्थिति में बदल दिया जाता है।

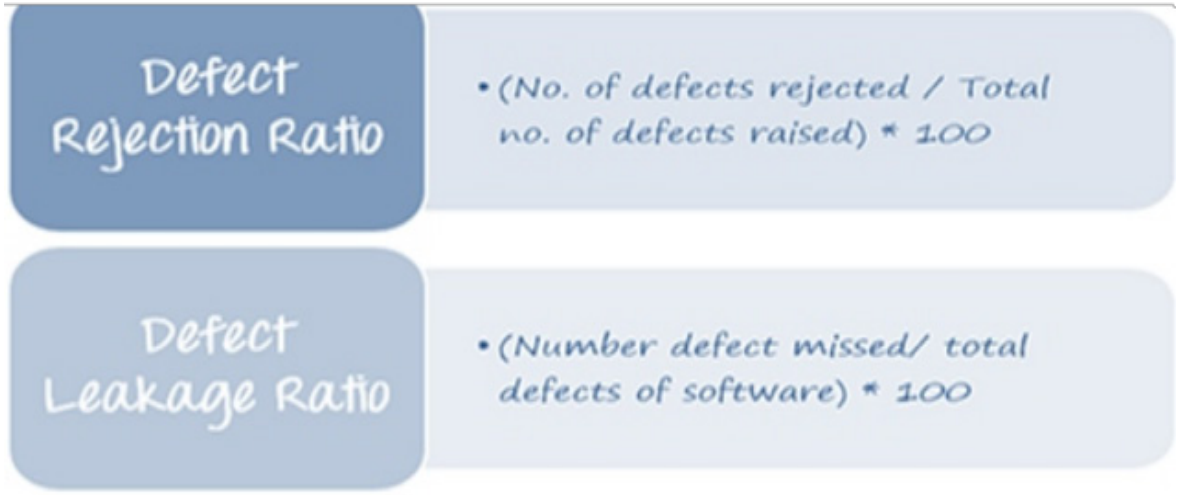
8.1.11 Testing निष्पादन की गुणवत्ता को कैसे मापें और मूल्यांकन करें?

Testing प्रबंधक डिफेक्ट्स प्रबंधन प्रक्रिया और डिफेक्ट्स की स्थिति पर प्रतिक्रिया प्राप्त करने के लिए सॉफ्टवेयर Testing के लिए डिफेक्ट्स रिपोर्टिंग प्रक्रिया के दौरान प्रबंधन टीम को डिफेक्ट्स रिपोर्ट तैयार करते हैं और भेजते हैं। प्रबंधन टीम तब डिफेक्ट्स रिपोर्ट की समीक्षा करती है और आवश्यकतानुसार प्रतिक्रिया या अतिरिक्त सहायता भेजती है। डिफेक्ट्स रिपोर्टिंग स्पष्ट संचार, विस्तृत ट्रैकिंग और डिफेक्ट्स की व्याख्या की सुविधा प्रदान करती है।

प्रबंधन बोर्ड को डिफेक्ट्स की स्थिति के बारे में सूचित करने का अधिकार है। इस परियोजना में आपकी सहायता करने के लिए, उन्हें डिफेक्ट्स प्रबंधन प्रक्रिया से परिचित होना चाहिए। इसलिए, वर्तमान डिफेक्ट्स स्थिति के बारे में उनसे प्रतिक्रिया प्राप्त करने के लिए, आपको उन्हें सूचित करना होगा।

8.1.12 Testing निष्पादन की गुणवत्ता को कैसे मापें और मूल्यांकन करें?

प्रत्येक परीक्षा प्रबंधक इस प्रश्न का उत्तर जानने में रुचि रखता है। निम्नलिखित 2 पैरामीटर हैं जिन्हें आप ध्यान में रख सकते हैं:



चित्र 8.1.1 डिफेक्ट्स के प्रलेखन की रिपोर्ट करें

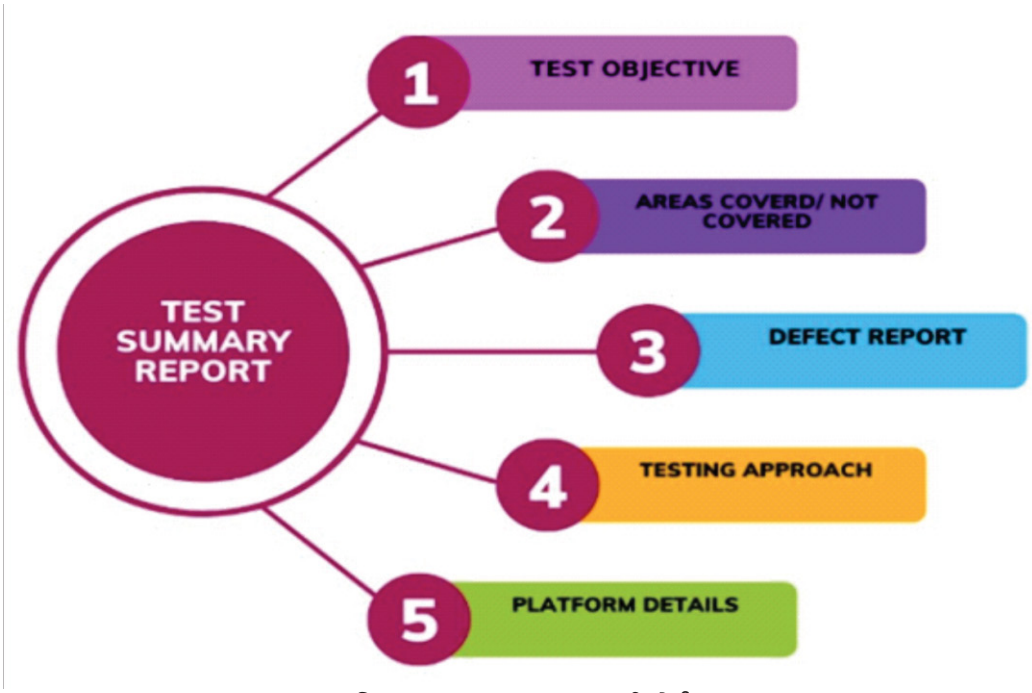
8.1.13 Test Report क्या है?

एक Test Report Testing के लक्ष्यों, प्रक्रियाओं और परिणामों का एक सुव्यवस्थित सारांश है। इसे विकसित किया गया था और इसका उपयोग हितधारकों (उत्पाद प्रबंधक, विश्लेषकों, Testing टीम और डेवलपर्स) की सहायता के लिए किया जाता है, यह निर्धारित करने में कि कोई सुविधा, उत्पाद या डिफेक्ट्स समाधान रिलीज के लिए तैयार है या नहीं।

Testing सारांश रिपोर्ट नामक एक महत्वपूर्ण दस्तावेज एक Testing परियोजना के समापन पर बनाया जाता है, या बल्कि, Testing चक्र पूरा होने के बाद। इस दस्तावेज़ का मुख्य लक्ष्य परियोजना के लिए किए गए Testing गतिविधियों के बारे में विस्तार से संबंधित हितधारकों, जैसे वरिष्ठ प्रबंधन, ग्राहकों, आदि को सूचित करना है। इसके अतिरिक्त, यह दर्शाता है कि एप्लिकेशन की गुणवत्ता कितनी अच्छी तरह गोल है।

Testing सारांश रिपोर्ट में क्या शामिल करना है?

एक Testing सारांश रिपोर्ट जो शिक्षाप्रद है, संक्षिप्त और प्रासंगिक होनी चाहिए। ऑनलाइन कई Testing सारांश रिपोर्ट टेम्पलेट उदाहरण हैं, लेकिन उनमें से सभी आपकी स्थिति में लागू नहीं हो सकते हैं। इसलिए, हमारी Testing परियोजना की विशेषताओं के अनुसार गहन विश्लेषण करने के बाद हमारी रिपोर्ट को समायोजित करना महत्वपूर्ण है।



चित्र 8.1.2 Testing सारांश रिपोर्ट

1. **Testing उद्देश्य** - यह प्रदर्शित करने के लिए Testing का उद्देश्य शामिल करें कि Testing वस्तु और आवश्यकता के बारे में टीम की समझ पूरी हो गई थी।
2. **कवर किए गए क्षेत्र-** Testing के दौरान जांच किए गए उत्पाद के घटकों की सूची बनाएं। इसमें प्रत्येक Testing परिदृश्य को बहुत विस्तार से शामिल करने की आवश्यकता नहीं है, लेकिन इसमें सभी प्रमुख आधारों को शामिल किया जाना चाहिए।
3. **कवर नहीं किए गए क्षेत्र-** उत्पाद के उन क्षेत्रों को कैप्चर करना बहुत आवश्यक है जो Testing में शामिल नहीं थे। Testing नहीं किए गए किसी भी क्षेत्र में ग्राहक के अंत में चिंता का कारण बन सकता है, इसलिए हमें यह सुनिश्चित करने की आवश्यकता है कि कुछ भी अप्रयुक्त नोट किया गया है और अपेक्षाओं को उचित रूप से निर्धारित किया गया है। Testing नहीं किए गए किसी भी क्षेत्र में ग्राहक के अंत में चिंता का कारण बन सकता है, इसलिए हमें यह सुनिश्चित करने की आवश्यकता है कि कुछ भी अप्रयुक्त नोट किया गया है और अपेक्षाओं को उचित रूप से निर्धारित किया गया है। इसके अलावा, सुनिश्चित करें कि प्रत्येक बिंदु का एक संबद्ध कारण है, उदाहरण के लिए, उपकरणों की उपलब्धता तक पहुंच की सीमा।
4. **Testing दृष्टिकोण**- इस खंड को कवर करने की आवश्यकता है क्योंकि यह Testing की प्रकृति और तरीकों का वर्णन करता है। किए गए कार्यों और कार्य को पूरा करने के लिए उपयोग की जाने वाली विभिन्न Testing रणनीतियों का विस्तार से वर्णन करना सुनिश्चित करें।
5. **डिफेक्ट्स रिपोर्ट**- हालांकि एक डिफेक्ट्स रिपोर्ट आमतौर पर एक बग रिपोर्ट में शामिल होती है, Testing सारांश रिपोर्ट में एक सहित आपकी रिपोर्ट को बढ़त मिल सकती है।
6. **Platform की बारीकियां**- वर्तमान में विभिन्न प्लेटफार्मों पर उत्पादों का Test किया जाता है। बढ़ती मांग के कारण, Testing में अब न केवल विभिन्न हार्डवेयर कॉन्फ़िगरेशन और ब्राउज़र बल्कि विभिन्न सॉफ़्टवेयर रिलीज़ भी शामिल हैं। आइए प्रत्येक प्लेटफ़ॉर्म और वातावरण के बारे में जानकारी शामिल करना सुनिश्चित करें, जिसमें उत्पाद का Test किया गया था।

8.1.14 डिफेक्ट्स ट्रैकिंग सिस्टम

एक सॉफ्टवेयर प्रोग्राम जो रिपोर्ट किए गए सॉफ़्टवेयर मुद्दों को ट्रैक करता है, बग ट्रैकिंग सिस्टम या डिफेक्ट्स ट्रैकिंग सिस्टम के रूप में जाना जाता है। समग्र महत्व और संभावना के आधार पर कि भविष्य में बग की पुनरावृत्ति होगी, सिस्टम में प्रत्येक बग को इसकी तात्कालिकता के लिए एक मूल्य दिया जा सकता है। कम-तात्कालिकता कीड़े मामूली हैं और जैसे ही समय अनुमति देता है, इसे ठीक किया जाना चाहिए। बग के बारे में अन्य जानकारी में क्लाइंट शामिल है जिसने इसकी सूचना दी, जमा करने की तारीख, समस्या का गहन विवरण, प्रयास किए गए फिक्स और अन्य प्रासंगिक डेटा। तकनीशियन सिस्टम को ताजा डेटा के साथ अपडेट करता है क्योंकि वे उस बग को ठीक करते हैं। समस्या को हल करने के किसी भी प्रयास को बग सिस्टम में रिकॉर्ड किया जाना चाहिए क्योंकि प्रत्येक बग सभी परिवर्तनों का हिस्ट्री रखता है।

बग ट्रैकिंग सिस्टम के महत्वपूर्ण तत्व इस प्रकार हैं

पहली और सबसे महत्वपूर्ण चीज जो हमें चाहिए वह है प्रोग्राम के बग के सटीक स्थान का ज्ञान। स्थान प्रोग्राम की विधि, कक्षा और लाइन नंबर जैसे विवरण प्रदान करता है जहां बग स्थित है। इससे डेवलपर्स के लिए समस्याग्रस्त कोड ढूंढना आसान हो जाता है। यह जानकारी सॉफ़्टवेयर डेवलपमेंट परिवेशों (IDEs) का उपयोग करके प्राप्त किया जा सकता है।

एक डेटाबेस जो ज्ञात बग के बारे में जानकारी का ट्रैक रखता है, बग ट्रैकिंग सिस्टम का एक महत्वपूर्ण हिस्सा है। जिस समय बग की सूचना दी गई थी, उसकी गंभीरता, गलत प्रोग्राम व्यवहार, बग को पुन: उत्पन्न करने के तरीके के बारे में जानकारी, और किसी भी प्रोग्रामर के नाम जो इसे ठीक करने पर काम कर रहे हैं, सभी प्रासंगिक तथ्यों के उदाहरण हैं। उपयोगकर्ता जानकारी, लॉगिन जानकारी, डेटा कोड की जानकारी और डेटाबेस में संग्रहीत रिपोर्ट सभी को बग ट्रैकिंग सिस्टम द्वारा नियंत्रित किया जाएगा। ये सेवाएं हमारी परियोजना द्वारा JSP और HTML भाषाओं का उपयोग करके प्रदान की जाती हैं। लॉगिन, साइनअप आदि के लिए एक यूजर इंटरफेस HTML द्वारा प्रदान किया जाता है। दूसरी ओर, जेएसपी का उपयोग HTML को सर्वर से कनेक्ट करने के लिए किया जाता है। इसमें यूजर की जानकारी को स्टोर करने के लिए, हमने एक MySQL डेटाबेस का उपयोग किया। दिन के लिए आपके शुरुआती बिंदु के रूप में, हमारे उपयोगकर्ता के अनुकूल इंटरफेस में एक डैशबोर्ड दृश्य शामिल होता है जो "मुझे असाइन किया गया," "अनअसाइन," " Reported by Me," और कई अन्य सामान्य फ़िल्टर प्रदर्शित करता है।

डिफेक्ट्स ट्रैकिंग एक उत्पाद में खामियों की पहचान करने की प्रक्रिया है। यह असतत प्रणालियों का रूप ले सकता है जहां डिफेक्ट्स के बारे में जानकारी और इसे ठीक करने के लिए काम की प्रगति संग्रहीत की जाती है, या यह कॉन्फ़िगरेशन प्रबंधन उपकरणों के एक एकीकृत सेट का एक घटक हो सकता है जहां डिफेक्ट्स की स्थिति अन्य सिस्टम घटनाओं के लिए कुंजी या ट्रिगर के रूप में कार्य कर सकती है।

8.1.15 प्रभावी डिफेक्ट्स रिपोर्टिंग

डिफेक्ट्स को समझने और पुन: उत्पन्न करने की कोशिश में खर्च किए जा रहे अनावश्यक समय और प्रयास को रोकने के लिए प्रभावी डिफेक्ट्स रिपोर्टिंग महत्वपूर्ण है। यहां कुछ सिफारिशें दी गई हैं:

विशिष्ट रहें:

अधिक विवरण दें, कम नहीं। दूसरे शब्दों में, सुस्त होने से बचें। आपके द्वारा डेवलपर्स को दी जाने वाली जानकारी का उपयोग किया जा सकता है या नहीं, लेकिन वे आपके पास नहीं आएंगे और आपके द्वारा याद की गई किसी भी चीज़ के लिए पूछेंगे।

वस्तुनिष्ठ बनें:

एप्लिकेशन की गुणवत्ता के बारे में कोई निर्णय न लें, जैसे "यह एक घटिया एप्लिकेशन है" या "आपने इसे वास्तव में खराब किया है।

तथ्यों को रखें और अपनी भावनाओं से दूर रहें।

डिफेक्ट्स को पुन: उत्पन्न करें:

यदि आप जल्दबाजी में एक डिफेक्ट्स पाते हैं तो तुरंत रिपोर्ट न करें। कम से कम, निश्चित होने के लिए इसे एक बार फिर दोहराएं। (यदि आपको इसे पुन: पेश करने में परेशानी हो रही है, तो सटीक Testing स्थिति को याद रखने का प्रयास करें और प्रयास करते रहें। रिपोर्ट को आगे की जांच के लिए प्रस्तुत किया जाना चाहिए यदि आप अंततः कई प्रयासों के बाद डिफेक्ट्स को दोहराने में असमर्थ हैं, जबकि आपके दावे का समर्थन करने के लिए आपके द्वारा एकत्र किए गए किसी भी सबूत को भी शामिल किया गया है।

रिपोर्ट की समीक्षा करें: रिपोर्ट समाप्त करने के तुरंत बाद "सबमिट" न दबाएं। कम से कम एक बार इसकी समीक्षा करें। सभी टाइपो को हटा दें।

उदाहरण: कर्मचारी लॉगिन पृष्ठ डिफेक्ट्स रिपोर्ट:

- डिफेक्ट्स आईडी: D001
- परियोजना का नाम: MyASP
- मॉड्यूल का नाम: लॉगिन
- उप मॉड्यूल नाम: कर्मचारी लॉगिन
- डिफेक्ट्स का प्रकार: गुम
- स्थिति: नया
- गंभीरता: उच्च
- प्राथमिकता:
- कर्मचारी लॉगिन पृष्ठ खोलने में विफलता
- या क़िस्म:
- URL दर्ज करें
- मेनू से " Employee लॉगिन" चुनें।
- कर्मचारी लॉगिन पृष्ठ अपेक्षा के अनुरूप लोड होना चाहिए।
- वास्तविक परिणाम: Employee लॉगिन पृष्ठ खुला नहीं मिलता है
- द्वारा रिपोर्ट: ABC Tester
- को असाइन करें: XYZ डेवलपर
- दिनांक & समय: 09/11/2019

8.1.16 मैनुअल test पर test डेटा के प्रभाव

सॉफ़्टवेयर के एक विशिष्ट टुकड़े का Test करने के लिए उपयोग किए जाने वाले डेटा को test डेटा के रूप में जाना जाता है। जबकि कुछ डेटा का उपयोग पुष्टि किए गए परिणाम प्राप्त करने के लिए किया जाता है, अन्य डेटा का उपयोग सॉफ़्टवेयर की क्षमताओं पर सवाल उठाने के लिए किया जा सकता है। विभिन्न तरीकों से सिस्टम Testing के लिए उपयुक्त test डेटा प्राप्त किया जा सकता है। एक विशिष्ट प्रणाली के Testing के लिए test डेटा एक Tester या एक कार्यक्रम द्वारा उत्पन्न किया जा सकता है।

उदाहरण के लिए, Testing टीम यह जांचना चाहती है कि सॉफ़्टवेयर वांछित परिणाम देता है या नहीं। सिस्टम को डेटा फीड किया जाएगा, और यह चलेगा। परिणाम का विश्लेषण करने के बाद, यह तय करेगा कि वांछित परिणाम प्राप्त हुए हैं या नहीं। सॉफ्टवेयर को कम से कम त्रुटि के बिना काम करना चाहिए और वांछित परिणाम उत्पन्न करना चाहिए। इसे पूरा करना चाहिए क्योंकि इसे पहले स्थान पर बनाने का मुख्य कारण था।

test डेटा कितने प्रकार के होते हैं?

नीचे कुछ सामान्य प्रकार के test डेटा की सूची दी गई है:

boundary test डेटा

इस तरह की जानकारी boundary मूल्यों के प्रसंस्करण के दौरान उत्पन्न होने वाले जुड़े डिफेक्ट्स के उन्मूलन में सहायता करती है। boundary मानों का एक संयोजन जो एप्लिकेशन को संभालने के लिए पर्याप्त है, इस डेटा प्रकार में डेटा बनाता है। इसके अतिरिक्त, यदि Tester आगे जाता है, तो एप्लिकेशन टूट सकता है।

मान्य test डेटा

ये डेटा प्रकार वैध हैं और एप्लिकेशन उन्हें स्वीकार करता है। ये यह पुष्टि करने में सहायता करते हैं कि सिस्टम कैसे काम करता है, और जब कोई इनपुट दिया जाता है, तो वे वांछित परिणाम प्राप्त करने में सहायता करते हैं।

अमान्य test डेटा

इन डेटा प्रकारों में असमर्थित डेटा स्वरूप शामिल हैं. टीमें डेटा का उपयोग यह आकलन करने के लिए करती हैं कि एप्लिकेशन ठीक से काम कर रहा है या नहीं। अमान्य मान दर्ज करके, ऐप को उपयुक्त त्रुटि संदेश प्रदर्शित करना चाहिए और उपयोगकर्ता को यह बताना चाहिए कि डेटा उपयोग के लिए अयोग्य है।

अनुपस्थित डेटा

बिना किसी डेटा वाली फ़ाइलों को रिक्त फ़ाइलों या बिना डेटा फ़ाइलों के रूप में संदर्भित किया जाता है। सॉफ़्टवेयर में रिक्त या कोई डेटा दर्ज करके, ऐप की प्रतिक्रिया का Test यह देखने के लिए किया जा सकता है कि यह कैसे प्रतिक्रिया करता है।

सारांश

- सॉफ़्टवेयर Testing जो स्वचालित उपकरण के बजाय Test cases के मैन्युअल निष्पादन का उपयोग करता है, मैनुअल Testing के रूप में जाना जाता है।
- किसी भी Tester को एक अच्छी तरह से लिखित Test Case का उपयोग करके Testing को समझने और करने में सक्षम होना चाहिए।
- Test Case Tester को कार्यों की एक श्रृंखला के माध्यम से निर्देशित करने में सहायता करते हैं ताकि यह सत्यापित किया जा सके कि कोई सॉफ़्टवेयर एप्लिकेशन बग से मुक्त है और अंतिम-उपयोगकर्ता अपेक्षाओं के अनुसार प्रदर्शन कर रहा है।
- सॉफ्टवेयर Testing में, डिफेक्ट्स संकल्प चरण दर चरण डिफेक्ट्स को ठीक करने की प्रक्रिया है।
- Testing प्रबंधक डिफेक्ट्स प्रबंधन प्रक्रिया और डिफेक्ट्स की स्थिति पर प्रतिक्रिया प्राप्त करने के लिए सॉफ्टवेयर Testing के लिए डिफेक्ट्स रिपोर्टिंग प्रक्रिया के दौरान प्रबंधन टीम को डिफेक्ट्स रिपोर्ट तैयार करते हैं और भेजते हैं।
- एक Test Report Testing के लक्ष्यों, प्रक्रियाओं और परिणामों का एक सुव्यवस्थित सारांश है। यह विकसित किया गया था और उत्पाद की गुणवत्ता को समझने और यह निर्धारित करने में हितधारकों (उत्पाद प्रबंधक, विश्लेषकों, Testing टीम और डेवलपर्स) की सहायता के लिए नियोजित किया गया है कि कोई उत्पाद, सुविधा या डिफेक्ट्स समाधान रिलीज के लिए निर्धारित समय पर है या नहीं।
- एक डेटाबेस जो ज्ञात बग के बारे में जानकारी का ट्रैक रखता है, बग ट्रैकिंग सिस्टम का एक महत्वपूर्ण हिस्सा है।
- यह आवश्यक है कि आप डिफेक्ट्स को प्रभावी ढंग से रिपोर्ट करें ताकि डिफेक्ट्स को समझने और पुन: उत्पन्न करने की कोशिश में समय और प्रयास अनावश्यक रूप से बर्बाद न हो।

एक्टिविटी

चार्ट पेपर

- इस गतिविधि में, प्रशिक्षक Test Report बनाने पर एक सत्र आयोजित करेगा
- प्रत्येक उम्मीदवार को एक सामान्य Test Report टेम्पलेट प्रदान किया जाएगा।
- प्रत्येक उम्मीदवार को इसे प्री-रिलीज़ Testing के लिए भरना होगा।
- यह महत्वपूर्ण है कि प्रशिक्षु अपने उत्तर न केवल जानकारी से समृद्ध हों, बल्कि फ्लो चार्ट द्वारा समर्थित भी हों।
- समूह, जो 30 मिनट के भीतर अपने उत्तरों को सर्वोत्तम तरीके से प्रस्तुत कर सकता है, उसे प्रशंसा और प्रशंसा से सम्मानित किया जाएगा।

चर्चा सत्र

- प्रशिक्षक पूछताछ करेगा कि क्या प्रशिक्षुओं ने अध्याय को समझा है।
- इस गतिविधि में, प्रशिक्षक प्रशिक्षु से विषय से संबंधित कुछ प्रश्न पूछेगा।
- प्रशिक्षु विषय की अपनी समझ के अनुसार प्रश्नों का उत्तर देंगे।
- यदि प्रशिक्षुओं के मन में प्रश्न और भ्रम हैं; वे ट्रेनर के सामने उन लोगों को सामने रख सकते हैं।
- प्रशिक्षक यह सुनिश्चित करेगा कि वह प्रशिक्षुओं द्वारा रखे गए प्रश्नों के उचित उत्तर दे सके।

अभ्यास

A. निम्नलिखित प्रश्नों के उत्तर दीजिए:

1. Test Report क्या है?
2. Testing सारांश रिपोर्ट में क्या होना चाहिए?
3. मैनुअल Testing के महत्व की गणना करें?

B. रिक्त स्थानों की पूर्ति कीजिए

संस्करण, डिफेक्ट्स, मैनुअल Testing, Testing

1. एक _____रिपोर्ट Testing उद्देश्यों, गतिविधियों और परिणामों का एक संगठित सारांश है।
2. सॉफ्टवेयर Testing जो एक स्वचालित उपकरण के बजाय Test cases के मैन्युअल निष्पादन का उपयोग करता है, किस रूप में जाना जाता है?_____.
3. _____हिस्ट्री आपके एंटरप्राइज़ Tester Repository के भीतर सभी तत्वों पर प्रदान किया जाता है।
4. एक सॉफ्टवेयर प्रोग्राम जो रिपोर्ट किए गए सॉफ़्टवेयर बग को ट्रैक करता है, बग के रूप में जाना जाता है _____ ट्रैकिंग सिस्टम।

नोट्स

Skill India
कौशल भारत-कुशल भारत

सत्यमेव जयते
GOVERNMENT OF INDIA
MINISTRY OF SKILL DEVELOPMENT
& ENTREPRENEURSHIP

N·S·D·C
National
Skill Development
Corporation
Transforming the skill landscape

9. एक बनाए रखें समावेशी, पर्यावरणीय रूप से स्थायी कार्यस्थल

IT - ITeS SSC
nasscom

यूनिट 9.1 - धारणीय व्यवहार

यूनिट 9.2 - विविधता का सम्मान करें और समानता को बढ़ावा देने के लिए प्रथाओं को मजबूत करें

SSC/N9014

प्रमुख सीखने के परिणाम

इस मॉड्यूल के अंत तक, प्रतिभागी निम्न में सक्षम होंगे:

1. ऊर्जा उपयोग को अनुकूलित करने के लिए कार्यस्थल में स्थायी प्रथाओं पर चर्चा करें
2. विविधता का सम्मान करके समानता को बढ़ावा देने के लिए उचित कार्यस्थल शिष्टाचार का प्रदर्शन करें

यूनिट 9.1: संधारणीय व्यवहार

यूनिट उद्देश्य

इस यूनिट के अंत तक, प्रतिभागी निम्न कार्य करने में सक्षम होंगे:

1. विभिन्न कार्यों में बिजली/ऊर्जा, कंटेंट और पानी के उपयोग को अनुकूलित करने का तरीका प्रदर्शित करें
2. ऊर्जा दक्ष प्रणालियों के चरणबद्ध तरीके से कार्यान्वयन की प्रक्रिया को समझाइए
3. निपटान या कुशल अपशिष्ट प्रबंधन के लिए उत्पन्न पुनर्नवीनीकरण, गैर-पुनर्चक्रण योग्य और खतरनाक कचरे को पहचानें और अलग करें

9.1.1 बिजली/ऊर्जा, कंटेंट के उपयोग का अनुकूलन, और पानी

कार्यालय परिसर और अन्य कॉर्पोरेट वातावरण के भीतर और आसपास हरियाली न केवल कार्यस्थल की सजावट को बढ़ाने में मदद करती है, बल्कि कर्मचारियों की उत्पादकता पर भी सकारात्मक प्रभाव डालती है। हरियाली लोगों को काम पर ध्यान केंद्रित करने में मदद करती है, श्रमिकों और आगंतुकों के बीच सकारात्मक वाइब्स बनाती है।

हरियाली की शुरूआत के अलावा, ऊर्जा का संरक्षण और उपयोग का अनुकूलन समान रूप से महत्वपूर्ण है। कुछ आवश्यक उपकरण और उपकरण हैं जो हर कार्यस्थल में उपयोग किए जाते हैं, जिनके लिए बिजली की आवश्यकता होती है। उदाहरण के लिए, एयर कंडीशनर, लाइट, पंखा, कंप्यूटर, कॉफी वेंडिंग मशीन ऐसे विद्युत गैजेट या उपकरण हैं जो कार्यालयों में बड़े पैमाने पर उपयोग किए जाते हैं। इसी तरह, वॉशरूम में स्थिर पानी की आपूर्ति एक और महत्वपूर्ण आवश्यकता है। इन सभी आवश्यक ऊर्जा या वस्तुओं का अनुकूलित उपयोग ऊर्जा संरक्षण और पर्यावरण के अनुकूल कार्य वातावरण बनाने के लिए बिल्कुल महत्वपूर्ण है।

हरियाली क्या करती है?

- कार्यस्थलों में पौधे हवा को शुद्ध करते हैं; वे सीओ 2 (कार्बन डाइऑक्साइड गैस) और अन्य वाष्पशील कार्बनिक यौगिकों की एकाग्रता को कम करते हैं, हवा को ताजा और स्वस्थ रखते हैं
- बाहरी वनस्पति गर्मियों में कार्यालय ब्लॉक में और उसके आसपास गर्मी को नियंत्रित करती है, गर्मी के तनाव को कम करती है और एयर कंडीशनिंग की आवश्यकता को कम करती है
- हरी छतें और facades इन्सुलेशन या गर्मी की अवशोषण क्षमता का प्रसार करते हैं, हीटिंग और शीतलन खर्चों को कम करते हैं
- कार्यालय भवनों में और उसके आसपास के पौधे जल वाष्प छोड़ते हैं जो हवा को नम करता है, सिरदर्द को कम करता है
- 'हरे रंग के दृश्य' भी फोकस को बढ़ावा देते हैं, और तनाव से जल्दी वसूली में सहायता करते हैं
- हरित वातावरण लोगों को दोपहर के भोजन के समय चलने, कर्मचारियों को सतर्क और स्वस्थ रखने जैसी गतिविधियों को करने के लिए प्रोत्साहित करता है। लंबे समय तक बैठने से स्वास्थ्य पर बुरा असर पड़ता है

ऊर्जा कुशल प्रणालियों के कार्यान्वयन की योजना बनाएं

यहां कुछ सरल ऊर्जा प्रबंधन विचार दिए गए हैं जिन्हें कार्य स्टेशन में लागू किया जा सकता है।

- कार्यालयों में कृत्रिम प्रकाश का उपयोग न करें जब प्राकृतिक प्रकाश पर्याप्त हो
- जब भी खिड़कियों से पर्याप्त रोशनी उपलब्ध हो तो ड्रेपरियां खोलें और रंगों को बढ़ाएं
- ऊर्जा की बचत फ्लोरोसेंट रोशनी और लैंप का उपयोग करें
- खाली कार्यालय स्थानों या सम्मेलन कक्ष जैसे अप्रयुक्त कमरों में रोशनी और उपकरणों को बंद कर दें। सम्मेलन के दौरान रोशनी और एसी/पंखे चालू करें
- जब भी बाथरूम में न हो तो पंखा और लाइट बंद कर दें
- कार्यालय उपयोगकर्ताओं को बर्बाद प्रकाश के बारे में याद दिलाने और शिक्षित करने के लिए प्रकाश सेंसर स्थापित करें
- कैलकुलेटर और अन्य कार्यालय उपकरणों के लिए रिचार्जेबल बैटरी का उपयोग करें
- उपयोग नहीं किए गए कंप्यूटरों को बंद करें, और कंप्यूटर के ऊर्जा / पावर प्रबंधन उपकरण (यानी स्लीप मोड, हाइबरनेट मोड, स्क्रीन सेवर) का उपयोग करें
- रात की सफाई के दौरान प्रकाश व्यवस्था का उपयोग कम करें
- यदि हीटिंग और एयर कंडीशनिंग चालू है तो कार्यालय के दरवाजे और खिड़कियां बंद रखें
- कार्यालयों में एचवीएसी सिस्टम को बंद कर दें जब वे उपयोग में न हों
- सुनिश्चित करें कि थर्मोस्टैट्स सही ढंग से समायोजित हैं
- उच्च दक्षता वाले कार्यालय उपकरण और उपकरणों की खरीद और उपयोग करें
- कार्यालय ऊर्जा खपत के लिए एक स्व-वयस्क प्रणाली स्थापित करें

प्राकृतिक संसाधनों और ऊर्जा के कुशल उपयोग, प्रदूषण में कमी और रोकथाम की दिशा में पहल

ये कुछ माप हैं जो कार्यस्थल में ऊर्जा के उपयोग को अनुकूलित करने में मदद करते हैं। हालांकि, ऊर्जा और अन्य सामग्रियों के उपयोग को अनुकूलित करने का एक अन्य महत्वपूर्ण पहलू उचित रखरखाव है। संगठनों को ऊर्जा और कंटेंट संरक्षण को मापने और बनाए रखने के लिए एक चेकलिस्ट तैयार करनी चाहिए। कार्यस्थलों पर ऊर्जा और कंटेंट संरक्षण मॉड्यूल के लिए एक नमूना चेकलिस्ट निम्नलिखित है।

कोटि	चेकलिस्ट आइटम
ऊर्जा प्रबंधन	ऊर्जा प्रबंधन संगठन की स्थापना, और कर्मचारी शिक्षा
	ऊर्जा संरक्षण लक्ष्य और निवेश बजट निर्धारण
	ऊर्जा संरक्षण के कार्यान्वयन की स्थिति को समझें
	मासिक उपयोग की माप और रिकॉर्डिंग (बिजली, गैस, तेल और पानी)
	आंकड़ों की तैयारी, जिसमें अंतर दिखाने वाले रेखांकन शामिल हैं पिछला महीना या वर्ष
	ऊर्जा तीव्रता की समझ (एमजे/एम2/वर्ष)
	प्रबंधन मानकों की स्थापना
गर्मी स्रोत और गर्मी संदेश देने वाले उपकरण	ठंडा पानी, ठंडा पानी और गर्म पानी के लिए तापमान नियंत्रण
	प्रवाह दर और पंपों और प्रशंसकों के दबाव का समायोजन
	भाप रिसाव और इन्सुलेशन प्रबंधन
	दहन उपकरण के वायु अनुपात और निकास गैस का प्रबंधन
	भाप के दबाव और झटका-डाउन का नियंत्रण
	ठंडा पानी की गुणवत्ता नियंत्रण (विद्युत चालकता)
	वाल्व और डैम्पर्स के उद्घाटन का नियंत्रण (जैसे स्वचालित वाल्व)
एयर कंडीशनिंग और वेंटिलेशन उपकरण	उचित तापमान सेटिंग
	उपयोग में नहीं आने वाले या खाली कमरों के लिए एयर कंडीशनिंग बंद करना
	उपयुक्त बाहरी हवा के सेवन की मात्रा का समायोजन
	परिचालन घंटों की समीक्षा
	कुल हीट एक्सचेंजर का प्रभावी संचालन (जैसे रोसुनाई)
	स्थानीय शीतलन और स्थानीय निकास
	इनडोर वायु गुणवत्ता नियंत्रण (जैसे CO2)
	वेंटिलेशन प्रशंसकों के लिए (मैनुअल या स्वचालित) इन्वर्टर डिवाइस की स्थापना
	4-पाइप एयर कंडीशनिंग सिस्टम के संचालन को निलंबित करना, यदि उपयोग किया जाता है
	कार पार्किंग स्थान में वेंटिलेशन का नियंत्रण (सीओ एकाग्रता नियंत्रण)

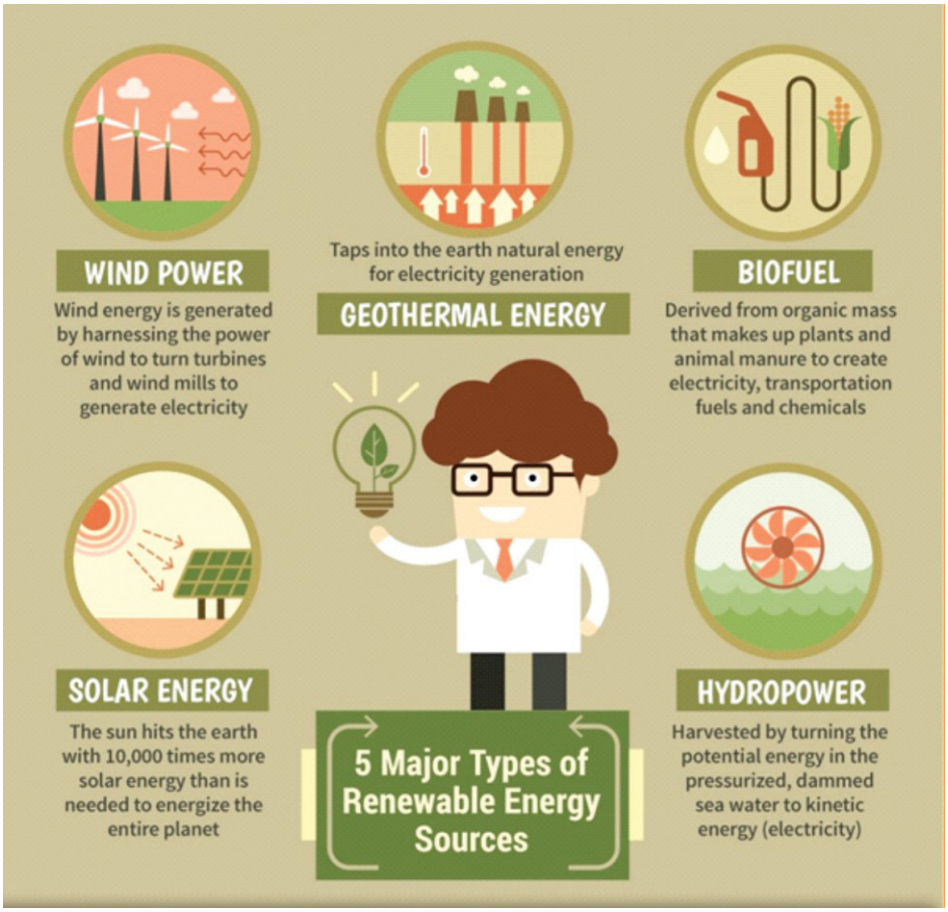
पाणीपुरवठा/ड्रेनेज आणि स्वच्छता उपकरणे	आपूर्ति किए गए जल प्रवाह और दबाव का नियंत्रण
	पानी की बचत के उपाय (जैसे पानी की बचत शीर्ष और स्वचालित चमकती)
	गर्मी स्रोत पर तापमान और दबाव सेटिंग बदलें मौसम के आधार पर उपकरण
	गर्म पानी की आपूर्ति परिसंचरण पंप में अंतराल के साथ संचालन
	वर्षा जल और कुएं के पानी का उपयोग
	रसोई उपकरण का प्रबंधन (जैसे खाना पकाने और वाशिंग मशीन)
विद्युत शक्ति प्राप्त करने और बदलने की सुविधाओं का प्रबंधन	मांग का अनुकूलन
	उपयोग नियंत्रण
	वोल्टेज समायोजन
	पावर फैक्टर प्रबंधन
प्रकाश उपकरणों का संचालन प्रबंधन	इष्टतम रोशनी नियंत्रण
	आवश्यक न होने पर रोशनी बंद करना (दिन के उजाले का उपयोग)
	प्रकाश जुड़नार की सफाई और अधिक ऊर्जा-बचत जुड़नार में परिवर्तन
	गरमागरम लैंप को फ्लोरोसेंट लैंप में बदलें
	ऊर्जा-बचत FFE (फर्नीचर, स्थिरता और उपकरण) को अपनाना
उन्नत मशीनों का संचालन और प्रबंधन	परिचालन
	इन्वर्टर नियंत्रण को अपनाना
	एस्केलेटर के लिए मानव गति सेंसर को अपनाना
इमारतों	खिड़कियों पर सौर विकिरण को अवरुद्ध करना (जैसे छायांकन पर्दे और प्रकाश-परिरक्षण फिल्में)
	छत पर सौर विकिरण को अवरुद्ध करना (गर्मी परावर्तन कोटिंग)
दूसरों	एयर कंडीशनिंग के लिए संघनक इकाइयों के आसपास जगह बनाए रखें और चिलर
	गर्म पानी के झरने से गर्मी का उपयोग
	ईंधन के रूप में अपशिष्ट पदार्थों का उपयोग करके बॉयलरों की स्थापना
	सौर ताप का उपयोग
	पवन, सौर और लघु जल विद्युत उत्पादन
	देर रात बिजली का उपयोग करें
	सह-पीढ़ी

तालिका 9.1.1: ऊर्जा और कंटेंट संरक्षण चेकलिस्ट

नवीकरणीय और गैर-नवीकरणीय सहित विभिन्न ऊर्जा विकल्प

नवीकरणीय ऊर्जा एक शाश्वत ऊर्जा स्रोत है जो दोहन पर समाप्त नहीं होता है और शून्य या न्यूनतम अपशिष्ट उत्पाद प्राप्त करता है। ऊर्जा के ऐसे स्रोत मानव समय-सीमा पर स्वाभाविक रूप से भर जाते हैं। अंतर्राष्ट्रीय ऊर्जा एजेंसी (IEA), पर्यावरण और सतत विकास पर पेरिस स्थित स्वायत्त प्राधिकरण, बताते हैं:

- "अक्षय ऊर्जा प्राकृतिक प्रक्रियाओं से ली गई है जो लगातार फिर से भर जाती हैं। अपने विभिन्न रूपों में, यह सीधे सूर्य से, या पृथ्वी के भीतर गहराई से उत्पन्न गर्मी से प्राप्त होता है। परिभाषा में शामिल सौर, पवन, महासागर, जल विद्त, बायोमास, भूतापीय संसाधन, जैव ईंधन और नवीकरणीय संसाधनों से प्राप्त हाइड्रोजन से उत्पन्न बिजली और गर्मी है।
- पवन ऊर्जा: पवन ऊर्जा सौर ऊर्जा का एक रूप है। पवन ऊर्जा (या पवन ऊर्जा) उस प्रक्रिया का वर्णन करती है जिसके द्वारा बिजली उत्पन्न करने के लिए हवा का उपयोग किया जाता है। पवन टरबाइन हवा में गतिज ऊर्जा को यांत्रिक शक्ति में परिवर्तित करते हैं। एक जनरेटर यांत्रिक शक्ति को बिजली में बदल सकता है।
- भूतापीय ऊर्जा: भूतापीय ऊर्जा (ग्रीक मूल भू से, जिसका अर्थ है पृथ्वी, और थर्मस से, जिसका अर्थ है गर्मी) पृथ्वी की पपड़ी के अंदर गर्मी द्वारा बनाई गई ऊर्जा है। हालांकि सूर्य पृथ्वी की सतह को गर्म करता है, पृथ्वी के अंदर से गर्मी सूर्य के कारण नहीं होती है।
- सौर ऊर्जा: सौर ऊर्जा सूर्य से ऊर्जा है जिसे थर्मल या विद्युत ऊर्जा में परिवर्तित किया जाता है। सौर ऊर्जा उपलब्ध सबसे स्वच्छ और सबसे प्रचुर मात्रा में अक्षय ऊर्जा स्रोत है।
- जैव ऊर्जा: बायोएनेर्जी प्राकृतिक, जैविक स्रोतों से बनाई गई अक्षय ऊर्जा है। आधुनिक तकनीक यहां तक कि लैंडफिल या अपशिष्ट क्षेत्रों को संभावित जैव ऊर्जा संसाधन बनाती है। इसका उपयोग एक स्थायी ऊर्जा स्रोत के रूप में किया जा सकता है, जो गर्मी, गैस और ईंधन प्रदान करता है।
- जल विद्युत: पनबिजली, या जल-ऊर्जा, अक्षय ऊर्जा का एक रूप है जो बांधों में संग्रहीत पानी का उपयोग करता है, साथ ही जल विद्त संयंत्रों में बिजली बनाने के लिए नदियों में बहता है। घूर्णन ब्लेड एक जनरेटर को स्पिन करते हैं जो कताई टरबाइन की यांत्रिक ऊर्जा को विद्त ऊर्जा में परिवर्तित करता है।



चित्र 9.1.1: नवीकरणीय ऊर्जा स्रोत

नवीकरणीय ऊर्जा ऊर्जा उत्पादन प्रक्रियाओं में ग्रीनहाउस गैसों का उत्सर्जन नहीं करते हैं, जिससे वे पर्यावरणीय क्षरण को रोकने के लिए सबसे स्वच्छ, सबसे व्यवहार्य समाधान बन जाते हैं। कोयला, गैस, तेल और परमाणु जैसे पारंपरिक ऊर्जा स्रोतों की तुलना में - जिनमें से भंडार परिमित हैं - स्वच्छ ऊर्जा उत्पन्न होती है और प्राकृतिक चक्रों के अनुकूल होती है। यह उन्हें एक स्थायी ऊर्जा प्रणाली में एक आवश्यक तत्व बनाता है जो भविष्य की पीढ़ियों को जोखिम में डाले बिना आज विकास की अनुमति देता है।

बिजली प्राथमिक चिकित्सा आपातकालीन प्रक्रियाएं

प्राथमिक चिकित्सा किट को चोटों के लिए प्राथमिक चिकित्सा प्रदान करने के लिए बुनियादी उपकरण प्रदान करना चाहिए, जिनमें शामिल हैं:

- कटौती, खरोंच, पंचर, चराई और किरचें
- मांसपेशियों में मोच और खिंचाव
- मामूली जलन
- विच्छेदन और/या प्रमुख रक्तस्राव घाव
- टूटी हुई हड्डियां
- आंख की चोट
- सदमा

यह सुनिश्चित करने के लिए प्राथमिक चिकित्सा प्रक्रियाओं को विकसित और कार्यान्वित करना चाहिए कि श्रमिकों को अपने कार्यस्थल में प्राथमिक चिकित्सा की स्पष्ट समझ है। प्रक्रिया को कवर करना चाहिए:

- प्राथमिक चिकित्सा किट का प्रकार और वे कहाँ स्थित हैं
- प्राथमिक चिकित्सा सुविधाओं जैसे प्राथमिक चिकित्सा कक्ष का स्थान
- प्राथमिक चिकित्सा किट और सुविधाओं के लिए कौन जिम्मेदार है और उन्हें कितनी बार जांचा और बनाए रखा जाना चाहिए
- प्राथमिक चिकित्सा के साथ तेजी से आपातकालीन कम्युनिकेशन सुनिश्चित करने के लिए उपयुक्त कम्युनिकेशन प्रणाली (उपकरण और प्रक्रियाओं सहित) को कैसे स्थापित और बनाए रखें
- प्राथमिक चिकित्सा की आवश्यकता होने पर उपयोग किए जाने वाले कम्युनिकेशन उपकरण और सिस्टम (विशेषकर दूरस्थ और पृथक श्रमिकों के लिए)। इन प्रक्रियाओं में कम्युनिकेशन उपकरण का पता लगाने के तरीके के बारे में जानकारी होनी चाहिए, जो उपकरण के लिए जिम्मेदार है और इसे कैसे बनाए रखा जाना चाहिए
- कार्य क्षेत्र और बदलाव जो प्रत्येक प्राथमिक उपचारकर्ता को आवंटित किए गए हैं। इन प्रक्रियाओं में प्रत्येक प्राथमिक उपचारकर्ता के नाम और संपर्क विवरण शामिल होने चाहिए
- प्राथमिक चिकित्साकर्मियों को उचित प्रशिक्षण मिले, इसके लिए व्यवस्था
- यह सुनिश्चित करने के लिए व्यवस्था कि श्रमिकों को प्राथमिक चिकित्सा के संबंध में उचित जानकारी, निर्देश और प्रशिक्षण प्राप्त हो
- जानकारी प्राप्त करना जब कोई कार्यकर्ता किसी भी प्राथमिक चिकित्सा आवश्यकताओं के बारे में काम करना शुरू करता है, जिसे चिकित्सा आपातकाल में विशिष्ट उपचार की आवश्यकता हो सकती है, जैसे कि गंभीर एलर्जी। एक कार्यकर्ता के स्वास्थ्य के बारे में जानकारी गोपनीय रखी जानी चाहिए और केवल कार्यकर्ता की सहमति से प्राथमिक उपचारकर्ताओं को प्रदान की जानी चाहिए
- कार्यस्थल में होने वाली चोटों और बीमारियों की रिपोर्ट कैसे करें
- रक्त और शरीर के पदार्थों के संपर्क से बचने के लिए अभ्यास
- क्या करना है जब कोई कार्यकर्ता या अन्य व्यक्ति काम पर रहने के लिए बहुत घायल या बीमार है, उदाहरण के लिए यदि उन्हें चिकित्सा सेवा, घर या कहीं और परिवहन के साथ सहायता की आवश्यकता होती है जहां वे आराम कर सकते हैं और ठीक हो सकते हैं
- एक गंभीर कार्यस्थल घटना के बाद प्राथमिक उपचारकर्ताओं और श्रमिकों का समर्थन करने के लिए डीब्रीफिंग या परामर्श सेवाओं तक पहुंच मुख्य शक्ति बंद करें।

यहां किसी व्यक्ति को करंट लगने से मुक्त करने के चरण बताए गए हैं।

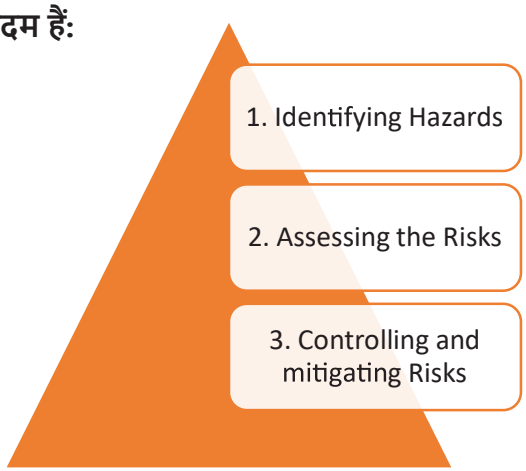
मुख्य शक्ति बंद करें।	
जिस व्यक्ति को करंट लगा है उसे न छुएं।	
व्यक्ति को विद्युत स्रोत से अचालक वस्तुओं जैसे छड़ी, गत्ता, बांस आदि की सहायता से निकालने का प्रयास कीजिए।	
व्यक्ति को इस स्थिति में रखें।	

तालिका 9.1.2: किसी व्यक्ति को करंट लगने से बचाने के चरण

9.1.2 पुनर्नवीनीकरण, गैर-पुनर्नवीनीकरण और खतरनाक अपशिष्ट

खतरे को एक कारक के रूप में परिभाषित किया गया है, जो लोगों और संपत्तियों को समान रूप से नुकसान पहुंचा सकता है, जैसे बिजली, ज्वलनशील उत्पाद, विस्फोटक कंटेंट, संक्षारक रसायन, कार्यस्थल पर भारी सीढ़ी का उपयोग आदि। सीधे शब्दों में कहें, एक खतरा केवल एक स्थिति या परिस्थितियों का एक समूह है जो नुकसान की संभावना पेश करता है। जोखिम को संभावना या संभावना के रूप में परिभाषित किया गया है कि एक खतरा वास्तव में किसी को नुकसान पहुंचा सकता है। उदाहरण के लिए, सिगरेट पीने वालों को कैंसर होने का खतरा होता है। संभावित या आसन्न खतरा जो जोखिम और खतरे संबंधित परिसर को उजागर करते हैं, उसे खतरे के रूप में जाना जाता है। उदाहरण के लिए, एक व्यक्ति, जिसके पास एक इमारत को उड़ाने की क्षमता है, उस इमारत और उसके निवासियों के लिए खतरा है।

जोखिम प्रबंधन में शामिल कदम हैं:



चित्र 9.1.2.: जोखिम प्रबंधन मैट्रिक्स

कार्यस्थल में खरीदे जाने वाले सबसे आम अपशिष्ट पदार्थों को निम्नलिखित में वर्गीकृत किया जा सकता है:

तरल अपशिष्ट

- कीचड़, गंदा पानी, कार्बनिक तरल पदार्थ, धोने के बाद अपशिष्ट जल

ठोस अपशिष्ट

- औद्योगिक लावा, प्लास्टिक अपशिष्ट, लकड़ी का कचरा, कागज का कचरा, धातु और कांच

जैविक अपशिष्ट

- बायोडिग्रेडेबल खाद्य अपशिष्ट, पशु अपशिष्ट, सब्जी अपशिष्ट, बगीचे के कचरे, जानवरों के सड़े हुए मांस को लैंडफिल में जमा किया जा सकता है या खाद और बायोगैस में परिवर्तित किया जा सकता है

पुन: प्रयोज्य अपशिष्ट

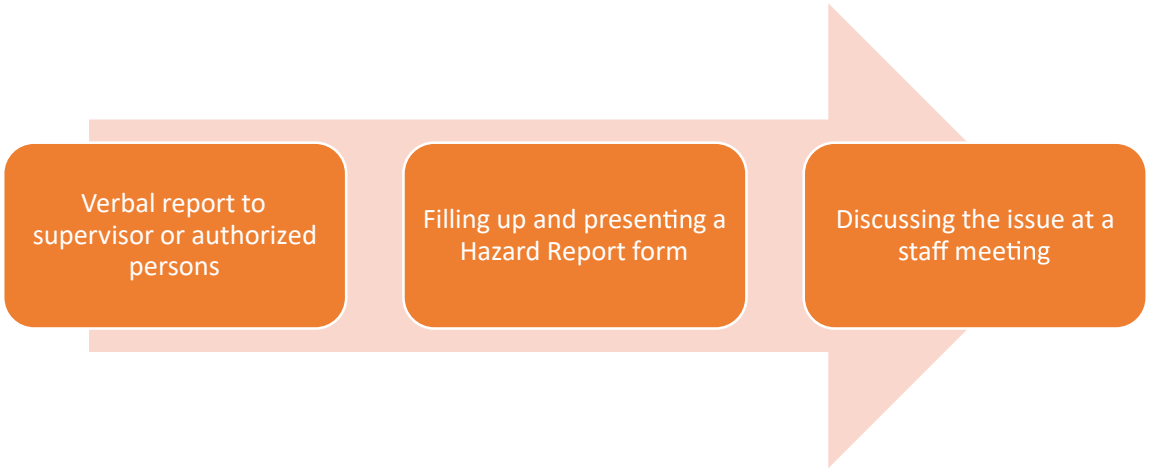
- कागज, धातु, लकड़ी, जैविक अपशिष्ट आदि को पुनर्नवीनीकरण किया जा सकता है
- उपयुक्त पुनर्चक्रण बिन में रखा जाना चाहिए और कचरे की प्रकृति के अनुसार इलाज किया जाना चाहिए
- उदाहरण के लिए, जैविक कचरे को खाद और बायोगैस में परिवर्तित किया जा सकता है

खतरनाक अपशिष्ट

- ऐसा अपशिष्ट ज्वलनशील, संक्षारक, रेडियोधर्मी, विषैला आदि हो सकता है।
- ये संभावित रूप से पर्यावरण को नुकसान पहुंचा सकते हैं और उचित उपचार और निपटान के लिए स्पष्ट रूप से और स्पष्ट रूप से लेबल किए गए डिब्बे में रखा जाना चाहिए

चित्र 9.1.3: अपशिष्ट पृथक्करण और निपटान डिब्बे

खतरों और संभावित जोखिमों / खतरों की पहचान की जा सकती है और फिर पर्यवेक्षकों या अन्य अधिकृत व्यक्तियों को निम्नलिखित तरीकों से सूचित किया जा सकता है:



चित्र 9.1.4: विभव धुंध की सूचना का फ़्लोचार्ट

खतरे की पहचान का मतलब है कि काम आधा हो गया है। खतरों के खिलाफ पर्याप्त एहतियाती उपाय करने के लिए, कार्यस्थल में आमतौर पर पाए जाने वाले खतरों की पहचान करने की आवश्यकता होती है। खतरे की पहचान के सामान्य तरीके हैं:

जॉब हैज़र्ड एनालिसिस (JHA)

- कर्मचारियों को चोटों के जोखिम को कम करने के लिए, नौकरी की भूमिका में विशिष्ट कार्यों से जुड़े खतरों की पहचान करने के लिए यह एक लोकप्रिय तकनीक है।
- जेएचए को सफलतापूर्वक आयोजित करने में शामिल कदम हैं:

संपूर्ण कार्य भूमिका को छोटे कार्यों या चरणों में विभाजित करें

आइए हम एक उदाहरण की मदद से अवधारणा को समझते हैं, जहां जेएचए को ग्राहक देखभाल कार्यकारी जैसे कॉर्पोरेट काम पर आयोजित किया जा रहा है।

	खतरे संबद्ध	सिफारिशों
1. व्यापार के उपकरण और उपकरण संभालना		
2. कार्यक्षेत्र के सामान्य विद्युत उपकरणों के साथ काम करना		
3. नौकरी की भूमिका का तनाव कारक		

तालिका 9.1.3: खतरे की पहचान के लिए जेएचए चेकलिस्ट

A. जैसे प्रश्न पूछकर प्रत्येक चरण से जुड़े खतरों का पता लगाएं:

- इस कार्य में क्या गलत हो सकता है?
- यदि कार्य गलत हो गया तो परिणाम क्या होंगे?
- कार्य गलत कैसे हो सकता है?
- अन्य योगदान कारक क्या हैं?
- इस खतरे के होने की क्या संभावना है?

B. कर्मचारियों के साथ खतरों के दायरे की समीक्षा और चर्चा करें, जो वास्तव में हाथ पर कार्य करेंगे

C. खतरों को कम करने या उनसे बचने के लिए रणनीतियों और तरीकों का पता लगाएं

D. समय-समय पर जेएचए की समीक्षा और संशोधन करें

खतरा और संचालन (HAZOP) अध्ययन

- इस तकनीक में मौजूदा विधि/प्रक्रिया की एक संरचित और व्यवस्थित परीक्षा शामिल है, इस प्रकार, बदले में, संबंधित खतरों की पहचान और आकलन करना।
- इन खतरों को प्रक्रिया मापदंडों (भौतिक स्थितियों और तत्वों जैसे प्रवाह, दबाव, तापमान, आर्द्रता, आदि) में विचलन के रूप में आसानी से पहचाना जा सकता है।
- विचलन की गंभीरता को विशिष्ट और पूर्व निर्धारित गाइड वर्ड की मदद से चित्रित किया जा सकता है।
- एक विचलन एक ऐसा तरीका है जिसमें प्रक्रिया की स्थिति अपेक्षित मूल्यों से दूर हो जाती है।

HAZOP के संचालन में शामिल कदम हैं:

- संपूर्ण सिस्टम या प्रक्रिया को अनुभागों या घटकों में अलग करना
- एक अध्ययन नोड या बिंदु का चयन करें
- अपेक्षित परिणाम या परिणाम को परिभाषित करें

- अपेक्षित परिणाम के आधार पर एक प्रक्रिया पैरामीटर चुनें
- एक उपयुक्त गाइड शब्द लागू करें
- विचलन के पीछे का कारण निर्धारित करें
- उस कारण से शुरू करें जो सबसे खराब संभावित परिणाम का कारण बन सकता है
- इस प्रकार पता लगाए गए विचलनों का आकलन करें
- कार्रवाई तैयार करें और निर्धारित करें
- रिकॉर्ड और दस्तावेज़ जानकारी
- बी से प्रक्रिया को दोहराएं

गाइड शब्द + प्रक्रिया की स्थिति / पैरामीटर = विचलन।

उदाहरण के लिए, नहीं + सिग्नल = कोई सिग्नल नहीं

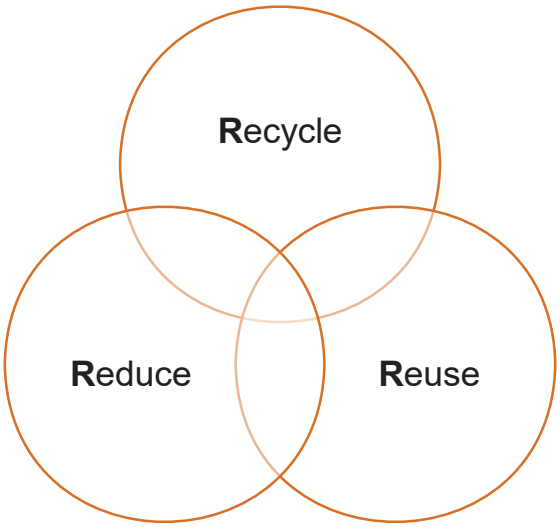
गाइड शब्दों और उनके अर्थ के सामान्य उदाहरण हैं:

गाइड वर्ड	मतलब	
नहीं (नहीं, कोई नहीं)	वांछित परिणामों में से कोई भी प्राप्त नहीं होता है	संचित गंदगी के कारण गैस काटने वाले नोजल के माध्यम से गैस का प्रवाह नहीं
अधिक (से अधिक, अधिक)	एक में मात्रात्मक वृद्धि कुछ प्रक्रिया पैरामीटर	आरा संचालन के दौरान अधिक गर्मी उत्पन्न और अपेक्षा से अधिक तापमान प्राप्त किया गया
कम (से कम, कम)	एक में मात्रात्मक कमी कुछ प्रक्रिया पैरामीटर	उम्मीद से कम दबाव
साथ ही (इसके अलावा)	सभी डिजाइन इरादे हासिल किए जाते हैं और एक अतिरिक्त गतिविधि होती है	सभी वाल्व एक ही समय में बंद हो गए
उल्टा	के तार्किक विपरीत डिजाइन इरादा जगह लेता है	बिजली आपूर्ति बंद होने के बाद भी पावर ड्रिल ड्रिलिंग जारी है
के सिवाय	एक अप्रत्याशित गतिविधि होती है स्थान	गैस सिलेंडर में तरल ईंधन की उपस्थिति

तालिका 9.1.4: गाइड शब्द और उनकी व्याख्या

अपशिष्ट अनुकूलन के 3 रुपये

- संसाधन अनुकूलन: कच्चे माल का पूरा उपयोग किया जाना चाहिये, ताकि कच्चे माल को तैयार उत्पादों में परिवर्तित करते समय न्यूनतम अपशिष्ट प्राप्त किया जा सके।
- स्क्रैप कंटेंट का पुनर्चक्रण: स्क्रैप, जब बनाया जाता है, तो उसे तुरंत विनिर्माण प्रक्रिया में शामिल किया जाना चाहिए, ताकि वे कच्चे माल के रूप में पूरी तरह से पुन: उपयोग किए जा सकें।
- उन्नत गुणवत्ता नियंत्रण: इसे प्रति बैच रिजेक्ट की संख्या को कम करके लागू किया जा सकता है। यह सावधानीपूर्वक निरीक्षण की उच्च आवृत्ति के साथ आसानी से प्राप्त किया जा सकता है, साथ ही निरंतर निगरानी के साथ।
- कचरे का आदान-प्रदान: कुछ अपशिष्टों को विनिर्माण प्रक्रिया से पूरी तरह से समाप्त नहीं किया जा सकता है। इस तरह के कचरे को अपशिष्ट विनिमय तकनीकों के माध्यम से प्रभावी ढंग से प्रबंधित किया जा सकता है, जहां एक निश्चित प्रक्रिया में खरीदा गया कचरा दूसरे का कच्चा माल बन जाता है, और इसके विपरीत।



चित्र 9.1.5: अपशिष्ट अनुकूलन के 3R

लैंडफिल

- अपशिष्ट, जिसे पुनर्नवीनीकरण नहीं किया जा सकता है, जमा किया जाता है और इसके ऊपर मिट्टी की एक परत डाली जाती है

भस्मीकरण

- कचरे का नियंत्रित दहन शामिल है
- कचरे की 90% मात्रा कम हो जाती है और राख, गैसों और गर्मी जैसी अज्वलनशील, हल्के वजन वाली कंटेंट में परिवर्तित हो जाती है
- गैसों को पर्यावरण में छोड़ा जाता है जबकि गर्मी का उपयोग बिजली उत्पादन में किया जाता है

बायोगैस उत्पादन

- जैविक अपशिष्ट बायोडिग्रेडेबल होते हैं और कुछ कवक और बैक्टीरिया की मदद से बायोगैस संयंत्रों में बायोगैस में परिवर्तित किए जा सकते हैं
- बायोगैस के उत्पादन के बाद अवशेषों का उपयोग खाद के रूप में किया जाता है

खाद उत्पादन और खाद बनाना

- जैविक कचरे को अक्सर मिट्टी के बेड के नीचे दबा दिया जाता है
- वे पोषक तत्वों और खनिजों से भरपूर समृद्ध खाद में विघटित हो जाते हैं

कृमि खाद

- कीड़े की मदद से खाद में जैविक कचरे के क्षरण को शामिल करता है
- कीड़े जैविक कचरे पर फ़ीड करते हैं और उन्हें खाद में परिवर्तित करते हैं

नोट्स

Scan the QR Code to watch the related videos

https://www.youtube.com/watch?v=wQ7zJYBuY74

Demonstrate how to optimize usage of electricity, energy, materials, and water in various tasks

यूनटि 9.2 - ववििधता का सम्मान करें और समानता को बढ़ावा देने के लिए

यूनिट के उद्देश्य

इस यूनिट के अंत तक, प्रतिभागी निम्न कार्य करने में सक्षम होंगे:

1. संगठन की विविधता नीति की व्याख्या करें
2. एक अनुकूलनीय और न्यायसंगत कार्य वातावरण के लिए पीडब्ल्यूडी समावेशी नीतियों का पालन करें

9.2.1 लिंग की अवधारणा, लैंगिक समानता और लैंगिक भेदभाव

सहकर्मियों के साथ काम करते समय लैंगिक समावेशिता, समानता और स्थिरता के बारे में नीतियां और प्रक्रियाएं

भारत का संविधान रोजगार या स्वस्थ या विकलांग से संबंधित मामलों में सभी नागरिकों (भारत के प्रत्येक कानूनी नागरिक सहित, चाहे वे विकलांग हों) के लिए अवसर की समानता पर समान रूप से लागू होता है। संविधान के तहत राज्य के तहत किसी भी कार्यालय में नियुक्ति। तथ्य की बात के रूप में, एक संगठन के कर्मचारी प्रमुख विविधता का गठन करते हैं। वे विभिन्न सांस्कृतिक और धार्मिक मान्यताओं के साथ विभिन्न क्षेत्र से आते हैं। हालांकि, नियोक्ता को लिंग, संस्कृति, धर्म के बावजूद प्रत्येक कर्मचारी को समान अवसर प्रदान करना चाहिए। विशेष रूप से, भारत सरकार ने कार्यस्थल में लैंगिक समानता सुनिश्चित करने के लिए कई उपाय किए हैं। कार्यस्थल में महिलाओं के अधिकार को स्थापित करने के लिए, सरकार ने बिल पारित किए हैं। कार्यस्थल पर महिलाओं का यौन उत्पीड़न (रोकथाम, निषेध और निवारण) अधिनियम, 2013 भारत में एक विधायी अधिनियम है जो महिलाओं को उनके कार्यस्थल पर यौन उत्पीड़न से बचाने का प्रयास करता है। इसे 3 सितंबर 2012 को लोकसभा (भारतीय संसद का निचला सदन) द्वारा पारित किया गया था। इसे 26 फरवरी 2013 को राज्यसभा (भारतीय संसद का ऊपरी सदन) द्वारा पारित किया गया था।

नीति की प्रमुख विशेषताओं में शामिल हैं:

- अधिनियम कार्यस्थल पर यौन उत्पीड़न को परिभाषित करता है और शिकायतों के निवारण के लिए एक तंत्र बनाता है। यह झूठे या दुर्भावनापूर्ण आरोपों के खिलाफ सुरक्षा उपाय भी प्रदान करता है।
- इस अधिनियम में यौन उत्पीड़न के रूपों के रूप में क्विड प्रो को उत्पीड़न और शत्रुतापूर्ण कार्य वातावरण की अवधारणाओं को भी शामिल किया गया है यदि यह यौन उत्पीड़न के किसी कार्य या व्यवहार के संबंध में होता है।
- "पीड़ित महिला" की परिभाषा, जिसे अधिनियम के तहत सुरक्षा मिलेगी, सभी महिलाओं को कवर करने के लिए अत्यंत व्यापक है, चाहे उसकी उम्र या रोजगार की स्थिति कुछ भी हो, चाहे वह संगठित या असंगठित क्षेत्रों में हो, सार्वजनिक या निजी और ग्राहकों, ग्राहकों और घरेलू कामगारों को भी कवर करती है।

- एक नियोक्ता को किसी भी व्यक्ति के रूप में परिभाषित किया गया है जो कार्यस्थल के प्रबंधन, पर्यवेक्षण और नियंत्रण के लिए जिम्मेदार है और इसमें ऐसे व्यक्ति शामिल हैं जो धारा 2 (जी) के तहत ऐसे संगठन की नीतियों को तैयार और प्रशासित करते हैं।
- जबकि विशाखा दिशानिर्देशों में "कार्यस्थल" पारंपरिक कार्यालय सेट-अप तक ही सीमित है, जहां एक स्पष्ट नियोक्ता-कर्मचारी संबंध है, अधिनियम सार्वजनिक और निजी क्षेत्र में संगठनों, विभाग, कार्यालय, शाखा यूनिट आदि को शामिल करने के लिए बहुत आगे जाता है, संगठित और असंगठित, अस्पताल, नर्सिंग होम, शैक्षणिक संस्थान, खेल संस्थान, स्टेडियम, खेल परिसर और परिवहन सहित रोजगार के दौरान कर्मचारी द्वारा दौरा किए गए किसी भी स्थान पर। यहां तक कि गैर-पारंपरिक कार्यस्थल जिनमें टेली-कम्यूटिंग शामिल है, इस कानून के तहत कवर किए जाएंगे।
- समिति को 90 दिनों की समयावधि के भीतर जांच पूरी करनी होती है। जांच पूरी होने पर, रिपोर्ट नियोक्ता या जिला अधिकारी को भेजी जाएगी, जैसा भी मामला हो, उन्हें 60 दिनों के भीतर रिपोर्ट पर कार्रवाई करने के लिए अनिवार्य किया गया है।
- प्रत्येक नियोक्ता को 10 या अधिक कर्मचारियों के साथ प्रत्येक कार्यालय या शाखा में एक आंतरिक शिकायत समिति का गठन करना आवश्यक है। जिला अधिकारी को प्रत्येक जिले में एक स्थानीय शिकायत समिति का गठन करना आवश्यक है, और यदि आवश्यक हो तो ब्लॉक स्तर पर।
- शिकायत निवारण समितियों को साक्ष्य एकत्र करने के लिए सिविल न्यायालयों की शक्तियां प्राप्त हैं।
- शिकायत समितियों से अपेक्षा की जाती है कि वे जांच शुरू करने से पहले, यदि शिकायतकर्ता अनुरोध करे, सुलह की व्यवस्था करें।
- अधिनियम के तहत जांच प्रक्रिया गोपनीय होनी चाहिए और अधिनियम में 5000 रुपये का जुर्माना लगाया गया है उस व्यक्ति पर जिसने गोपनीयता भंग की है।
- अधिनियम में नियोक्ताओं को अन्य दायित्वों के बीच शिक्षा और संवेदीकरण कार्यक्रम आयोजित करने और यौन उत्पीड़न के खिलाफ नीतियां विकसित करने की आवश्यकता होती है। जागरूकता निर्माण का उद्देश्य परिसर में प्रदर्शित बैनर और पोस्टर, कर्मचारियों, प्रबंधकों और आंतरिक समिति के सदस्यों के लिए ई-लर्निंग पाठ्यक्रम, कक्षा प्रशिक्षण सत्र, ईमेल के माध्यम से संगठनात्मक यौन उत्पीड़न नीति के कम्युनिकेशन, ई-लर्निंग या कक्षा प्रशिक्षण के माध्यम से प्राप्त किया जा सकता है। यह अनुशंसा की जाती है कि ई-लर्निंग या कक्षा प्रशिक्षण कर्मचारी की प्राथमिक कम्युनिकेशन भाषा में दिया जाए।
- नियोक्ताओं के लिए दंड निर्धारित किया गया है। अधिनियम के प्रावधानों का अनुपालन न करने पर 50,000/- रुपये तक के जुर्माने से दंडनीय होगा। बार-बार उल्लंघन करने पर अधिक दंड लग सकता है और व्यापार करने के लिए लाइसेंस रद्द या पंजीकरण रद्द किया जा सकता है।
- सरकार किसी भी संगठन में यौन उत्पीड़न से संबंधित कार्यस्थल और रिकॉर्ड का निरीक्षण करने के लिए एक अधिकारी को आदेश दे सकती है।
- अधिनियम के तहत, जो स्कूलों और कॉलेजों में छात्रों के साथ-साथ अस्पतालों में रोगियों को भी शामिल करता है, नियोक्ताओं और स्थानीय अधिकारियों को सभी शिकायतों की जांच के लिए शिकायत समितियों का गठन करना होगा। अनुपालन करने में विफल रहने वाले नियोक्ताओं को 50,000 रुपये तक का जुर्माना लगाया जाएगा।

9.2.2 संगठन के निवारण तंत्र

कार्यस्थल पर लैंगिक समानता के कारण को स्वीकार करने/मान्य करने, साझा करने और बढ़ावा देने के लिए कम्युनिकेशन के समावेशी उपकरण और प्रथाएं

महिलाओं की सुरक्षा और इसके मुद्दों पर दुनिया भर में चर्चा और बहस की जाती है। फिर भी हर साल यौन उत्पीड़न पर रिपोर्टों की संख्या खतरनाक दर से बढ़ रही है। इसलिए, एक निश्चित कंपनी को अपनी सुरक्षा सुनिश्चित करने के लिए महिला कर्मचारियों की अच्छी देखभाल करनी चाहिए।

इसलिए, एक कंपनी को महिलाओं को उन विभिन्न सुविधाओं के बारे में सूचित करना चाहिए जो वे उन्हें प्रदान करने जा रही हैं। कुछ बुनियादी सुविधाओं में निम्नलिखित शामिल हैं।

1. परिवहन सुविधाएं:
   महिलाओं की सुरक्षा सुनिश्चित करने में परिवहन बहुत बड़ी भूमिका निभाता है। यह सुनिश्चित करने से कि महिलाओं के साथ विश्वसनीय ड्राइवर होंगे, महिलाओं की सुरक्षा को बढ़ाने में मदद करेगा। उस सुरक्षा के बारे में पारदर्शी रहें जो आप रात की यात्राओं के दौरान प्रदान कर सकते हैं। प्रत्येक महिला को कंपनी द्वारा प्रदान किए जाने वाले विभिन्न सुरक्षा उपायों के बारे में पता होना चाहिए।
2. दुरुपयोग की रिपोर्टिंग:
   जब भी कोई दुर्घटना होती है तो प्रबंधन को अपने निर्णय लेने में तत्पर होना चाहिए। दुर्व्यवहार की रिपोर्ट करने के तरीके महिला को स्पष्ट किए जाने चाहिए ताकि शीघ्र उपचार सुनिश्चित किया जा सके।
3. मातृत्व से संबंधित शिकायत:

नियोक्ताओं को महिलाओं को कार्यबल में शामिल होने पर मातृत्व लाभ अधिनियम के तहत उपलब्ध मातृत्व लाभों के बारे में इलेक्ट्रॉनिक रूप से लिखित रूप में सूचित करना आवश्यक है

   कानून महिला कर्मचारियों को मातृत्व लाभ अवधि के अलावा घर से काम करने की अनुमति देता है यदि काम की प्रकृति इसकी अनुमति देती है

4. सीसीटीवी कैमरे
   सुनिश्चित करें कि प्रत्येक स्टेशन सीसीटीवी कैमरों से लैस है जो आजकल यौन उत्पीड़न के मामलों की जांच के लिए सबसे महत्वपूर्ण घटक हैं।
5. सुरक्षा गार्ड
   सुनिश्चित करें कि सामरिक स्थानों पर पर्याप्त मात्रा में सुरक्षा गार्ड तैनात किए जाएं ताकि महिलाओं की सुरक्षा के लिए किसी भी खतरे को कम किया जा सके। सुनिश्चित करें कि महिलाओं को उन विभिन्न स्थानों के बारे में सूचित किया जाता है जहां सुरक्षा गार्ड मौजूद हैं।
6. महिला हेल्पलाइन:
   महिला हेल्पलाइन नंबर और अन्य महत्वपूर्ण संपर्कों वाला एक पत्रक साझा करें।
7. चेन लॉक/लैच
   महिलाओं को चेन लॉक और कुंडी प्रदान करें ताकि उनका सामान ठीक से और सुरक्षित रूप से रखा जा सके और किसी भी प्रकार की चोरी से बचा जा सके।
8. स्मोक डिटेक्टर:
   महिलाओं को परिसर के अंदर स्मोक डिटेक्टर के स्थान के बारे में सूचित करें।

   इन बुनियादी सुविधाओं को प्रदान करने से यह सुनिश्चित होगा कि महिलाएं बिना किसी डर के आरामदायक आवास का आनंद लें।

लिंग भेदभाव, हिंसा और असमानता के सभी रूप

कार्यस्थल पर महिलाओं का यौन उत्पीड़न (रोकथाम, निषेध और प्रतितोष) अधिनियम, 2013 कार्यस्थल पर महिलाओं के यौन उत्पीड़न के खिलाफ शिकायतों की जांच और निवारण के लिए एक प्रणाली निर्धारित करता है। यह झूठे या दुर्भावनापूर्ण आरोपों के खिलाफ सुरक्षा उपाय भी प्रदान करता है।

अधिनियम के प्रमुख प्रावधान नियोक्ताओं के लिए निम्नलिखित जिम्मेदारियां निर्धारित करते हैं, ताकि महिलाओं के लिए एक सुरक्षित कार्य वातावरण सुनिश्चित किया जा सके:

- यौन उत्पीड़न के दंडात्मक परिणाम प्रदर्शित करें
- कार्यशालाओं और संवेदीकरण कार्यक्रमों का आयोजन करें
- एक आंतरिक नीति, चार्टर, संकल्प, घोषणा तैयार करना
- एक 'आंतरिक शिकायत समिति' (आईसीसी) बनाएं जहां कर्मचारियों की संख्या दस से अधिक हो
- समितियों को आवश्यक सुविधाएं प्रदान करें
- गवाहों/प्रतिवादी की सुरक्षित उपस्थिति
- समिति की रिपोर्ट समय पर प्रस्तुत करने की निगरानी करें
- आपराधिक मामले को आगे बढ़ाने में महिला की सहायता करें यदि वह ऐसा चाहती है
- जांच प्रक्रिया की गोपनीयता बनाए रखें। अधिनियम में 5,000 रुपये (यूएस $ 68) का जुर्माना लगाया गया है वह व्यक्ति जिसने गोपनीयता भंग की है
- यौन उत्पीड़न एक अपराध होने के साथ, नियोक्ता अपराधों की रिपोर्ट करने के लिए बाध्य हैं

कार्यस्थल पर यौन उत्पीड़न की समस्या से निपटने के लिए, कॉर्पोरेट मामलों के मंत्रालय ने 31 जुलाई, 2018 को एक अधिसूचना के माध्यम से कंपनी (लेखा) नियम 2014 में संशोधन किया। अधिसूचना में निजी कंपनियों के लिए अपने निदेशकों की वार्षिक रिपोर्ट में अधिनियम के अनुपालन का खुलासा करना अनिवार्य है।

इसके अलावा, अधिनियम उपयुक्त राज्य सरकार को स्थानीय शिकायत समिति (एलसीसी) की स्थापना के लिए जिला अधिकारी को अधिसूचित करने की जिम्मेदारी देता है।

मानव संसाधन प्रबंधक अग्रिम पंक्ति में हैं जब यौन उत्पीड़न के बारे में सांस्कृतिक दृष्टिकोण बदलने की बात आती है।

नीचे कुछ सर्वोत्तम प्रथाएं दी गई हैं जो एचआर महिलाओं के लिए सुरक्षित कार्य वातावरण सुनिश्चित करने के लिए विकसित कर सकती हैं:

- आधिकारिक कर्मचारी पुस्तिका को अपडेट करें जो उस प्रक्रिया को रेखांकित करती है जो काम पर यौन उत्पीड़न का अनुभव होने पर होगी। एक स्पष्ट बयान शामिल करें कि यौन उत्पीड़न बर्दाश्त नहीं किया जाएगा
- उत्पीड़न करने वाले व्यवहार या आचरण का एक स्पष्ट, सरल और आसानी से समझ में आने वाला विवरण दें, जिसमें कार्यस्थल पर उत्पीड़क करने वाले व्यवहार के उदाहरण शामिल हैं

- लिंग पहचान और यौन अभिविन्यास पर अधिक ध्यान केंद्रित करने के लिए सभी के लिए प्रशिक्षण लागू करें, और यौन उत्पीड़न का अनुभव करने वाले लिंग तटस्थता पर जोर दें
- पुरुष कर्मचारियों को संवेदनशील बनाना और महिलाओं के बीच आगे आने और शिकायत दर्ज करने के लिए आत्मविश्वास को मजबूत करना
- रोजगार कानून में बदलाव के बारे में अपडेट रहें जहां उनके कर्मचारी रहते हैं या काम करते हैं। एचआर को पेशेवर संघों, कानूनी सलाह और ऑनलाइन संसाधनों का भी उपयोग करना चाहिए ताकि यह सुनिश्चित किया जा सके कि कंपनी कर्मचारी अधिकारों से संबंधित मौजूदा और आगामी विधायी परिवर्तनों के अनुरूप और जागरूक है

सहकर्मियों के लिए आंतरिक और बाहरी कम्युनिकेशन का उपयोग करें

यह अक्सर कहा जाता है कि किसी का व्यवहार किसी के चरित्र का दर्पण है। वास्तव में, आपका व्यवहार बहुत कुछ कहता है कि आप किस तरह के व्यक्ति हैं। यदि आप एक अच्छे व्यवहार वाले व्यक्ति नहीं हैं तो आपकी शैक्षिक डिग्री बहुत कम महत्व रखती है। आपको लगभग हर स्थिति में अच्छा आचरण करने की आवश्यकता होती है चाहे आप नौकरी के लिए इंटरव्यू दें या स्नातकोत्तर डिग्री का पीछा करें, अपने कार्यस्थल पर या अपने ग्राहकों के साथ व्यवहार करते समय, अपने स्कूल / कॉलेज में या पार्टियों में भाग लेते समय। यहां तक कि अपने रिश्तेदारों के सामने अपने घर पर भी, यह आपका अच्छा व्यवहार है जो सबसे ज्यादा मायने रखता है। लेकिन व्यवहार शिष्टाचार एक ऐसी चीज है जिसे किसी पर थोपा नहीं जा सकता है, इसे अपने भीतर विकसित और पोषित करना होगा।

महिलाओं के प्रति आज्ञाकारी व्यवहार शिष्टाचार दिखाना बहुत महत्वपूर्ण है।

ऐसे कौन से विभिन्न उदाहरण हैं जहां कोई इस तरह के शिष्टाचार दिखा सकता है? चलो एक नज़र डालते हैं:

- कमरे में प्रवेश करने से पहले: प्रवेश करने से पहले आपको हमेशा दस्तक देनी चाहिए और अनुमति मांगनी चाहिए। यह शायद सबसे बुनियादी शिष्टाचार है। आपको यह सुनिश्चित करना चाहिए कि महिला की गोपनीयता को कोई नुकसान न पहुंचे। इसलिए, एक कमरे में प्रवेश करने से पहले दस्तक दें और मौखिक अनुमति लें
- स्पर्श संपर्क से बचना: आपको हमेशा यह सुनिश्चित करना चाहिए कि आप ग्राहक के व्यक्तिगत स्थान पर घुसपैठ न करें। यह न केवल अनप्रोफेशनल है, बल्कि अनहेल्दी भी है। इसलिए टच कॉन्टैक्ट से बचने की पूरी कोशिश करें। यदि बिल्कुल आवश्यक हो, तो अनुमति मांगें और फिर ग्राहक की सहायता करें।
- अपमानजनक भाषाओं या इशारों का उपयोग करना: यह आखिरी चीज है जो एक महिला/ग्राहक आपसे उम्मीद करती है। सुनिश्चित करें कि आप ग्राहक के सामने कभी भी किसी भी अभद्र भाषा का प्रयोग न करें। सुनिश्चित करें कि आप मेहमानों के सामने अपने सहयोगियों को गाली न दें

महिलाओं को समाज के साथ-साथ कानूनी रूप से भी सशक्त बनाया जाता है, बस कुछ बुनियादी अधिकार जो सार्वभौमिक हैं जो दोनों लिंगों पर लागू होते हैं लेकिन विशेष रूप से महिलाओं के लिए निम्नलिखित शामिल हैं: -

- गरिमा और सम्मान के लिए एक महिला के रूप में अधिकार जिसका अर्थ है कि किसी भी उम्र के किसी भी पुरुष को किसी महिला को आगे बढ़ाने, उसे चिढ़ाने या यौन उत्पीड़न करने का अधिकार नहीं है
- सभी परिस्थितियों में सम्मान करने का विशेषाधिकार: किसी को भी महिलाओं को असहज करने का अधिकार नहीं है, चाहे वह कार्यस्थल पर, घर पर या सड़कों पर, चाहे स्कूल, कॉलेज या सामाजिक सभा में

- शारीरिक और मानसिक सुरक्षा के अधिकार: किसी को भी शारीरिक बल का उपयोग करने, शारीरिक या मानसिक रूप से यातना देने या किसी भी तरह से महिलाओं को मजबूर करने का अधिकार नहीं है, चाहे उस व्यक्ति के साथ कोई भी संबंध क्यों न हो
- शिकायत करने का विशेषाधिकार: महिलाओं को छोटे से छोटे तरीके से भी उल्लंघन होने पर शिकायत करने का पूरा अधिकार है। सलाह लें और ऐसी परिस्थितियों में सही रास्ता अपनाएं, चाहे व्यक्ति की स्थिति कुछ भी हो, चाहे वह बॉस, रिश्तेदार या पड़ोस का बदमाश हो
- कार्यस्थल पर यौन उत्पीड़न की रोकथाम के लिए विशाखा दिशानिर्देशों के अनुसार संगठन द्वारा सुरक्षा के लिए महिला कर्मचारी के रूप में अधिकार
- महिलाओं के खिलाफ हिंसा, शारीरिक या मानसिक उसकी नियति नहीं है जैसा कि कुछ मामलों में बताया जाता है। प्रभुत्व वाला व्यवहार किसी का अधिकार नहीं है और न ही महिलाओं की नियति है, इसलिए इसके खिलाफ शिकायत करना सही क्रम में है

एक सुरक्षा प्रक्रिया आवश्यक गतिविधियों का एक सेट अनुक्रम है जो एक विशिष्ट सुरक्षा कार्य या कार्य करता है। प्रक्रियाओं को आम तौर पर अंतिम परिणाम प्राप्त करने के लिए एक सुसंगत और दोहराव वाले दृष्टिकोण या चक्र के रूप में पालन किए जाने वाले चरणों की एक श्रृंखला के रूप में डिज़ाइन किया जाता है।

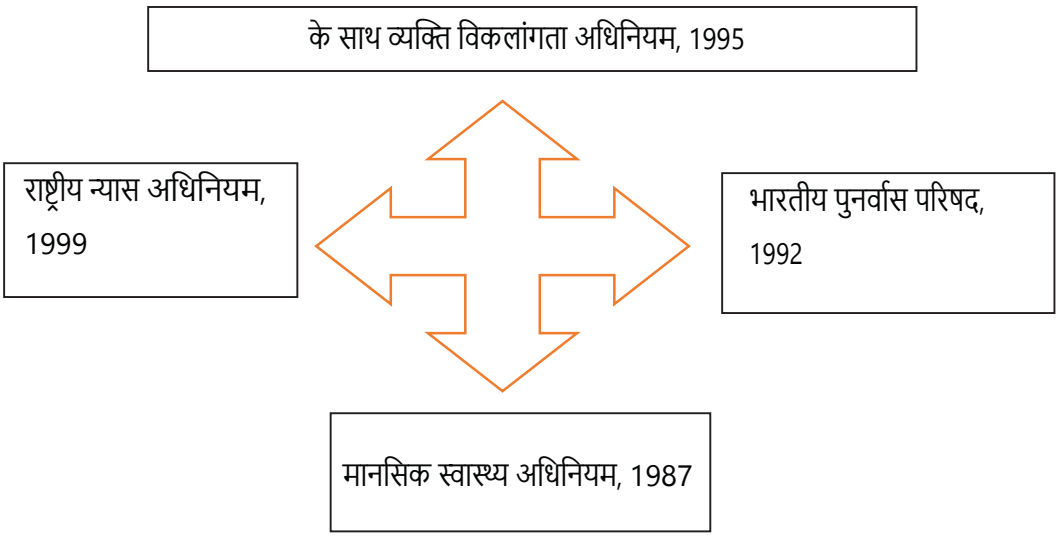
एक बार लागू होने के बाद, सुरक्षा प्रक्रियाएं संगठन के सुरक्षा मामलों के संचालन के लिए स्थापित कार्यों का एक सेट प्रदान करती हैं, जो प्रशिक्षण, प्रक्रिया लेखा परीक्षा और प्रक्रिया में सुधार की सुविधा प्रदान करेगी। प्रक्रियाएं सुरक्षा प्रक्रियाओं में भिन्नता को कम करने के लिए आवश्यक स्थिरता को लागू करने के लिए एक प्रारंभिक बिंदु प्रदान करती हैं, जो संगठन के भीतर सुरक्षा के नियंत्रण को बढ़ाती है।

एक नियोक्ता को यह सुनिश्चित करना चाहिए कि कर्मचारी सुरक्षा प्रक्रियाओं और संबंधित वातावरण से अधिक खतरे में पड़े बिना हर समय सुरक्षित महसूस करें।

9.2.3 पीडब्ल्यूडी समावेशी नीतियों का अनुपालन करें

किसी भी उत्पीड़न से मुक्त अनुकूल कार्य वातावरण को कैसे बनाए रखें और प्रदान करें; पीडब्ल्यूडी को सुविधाएं और सुविधाएं

भारत सरकार समानता का सम्मान करती है और इसलिए विकलांगता के आधार पर कोई भेदभाव नहीं किया जाना चाहिए। संविधान विकलांगों सहित नागरिकों को न्याय का अधिकार, विचार, अभिव्यक्ति, विश्वास, विश्वास और पूजा की स्वतंत्रता, स्थिति और अवसर की समानता और भाईचारे को बढ़ावा देने के लिए सुरक्षित करता है। किसी भी विकलांग व्यक्ति को किसी विशेष धर्म या धार्मिक समूह के प्रचार और रखरखाव के लिए कोई कर देने के लिए मजबूर नहीं किया जा सकता है। इसे लागू करने के लिए, सरकार ने विकलांगों और समानता के उनके अधिकार की रक्षा के लिए कानून पारित किए हैं। विकलांगों से संबंधित कानून इस प्रकार हैं:



चित्र 9.2.1: विकलांगों से संबंधित अधिनियम

विशेष रूप से डिजाइन की गई भर्ती पद्धतियों, पीडब्ल्यूडी के अनुकूल बुनियादी ढांचे, नौकरी की भूमिकाओं आदि के माध्यम से सुधार करना।

2016 एक्ट के तहत 'विकलांग व्यक्ति' की परिभाषा को व्यापक बनाया गया है: इसमें विकलांग व्यक्तियों, बेंचमार्क विकलांगता वाले व्यक्तियों और उच्च समर्थन आवश्यकताओं वाले विकलांग व्यक्तियों को शामिल किया गया है। यह परिभाषा समावेशी है और 21 प्रकार की विकलांगताओं को 'विशिष्ट विकलांगता' के रूप में वर्गीकृत करती है। यह अधिनियम सरकारी प्रतिष्ठानों के साथ-साथ निजी प्रतिष्ठानों पर भी लागू होता है। कानून के तहत, निजी प्रतिष्ठान एक कंपनी, फर्म, सहकारी या अन्य समाज, संघों, ट्रस्ट, एजेंसी, संस्था, संगठन, संघ, कारखाने, या सरकार द्वारा निर्दिष्ट ऐसे अन्य प्रतिष्ठान को संदर्भित करते हैं।

अधिनियम में सभी प्रतिष्ठानों को एक समान अवसर नीति तैयार करने और प्रकाशित करने की आवश्यकता है। विकलांग व्यक्तियों के खिलाफ भेदभाव के सभी रूप निषिद्ध हैं, जब तक कि यह साबित नहीं किया जा सकता है कि इस तरह का भेदभाव प्रकृति में आनुपातिक है और वैध उद्देश्यों को प्राप्त करने का एक आवश्यक साधन है। अधिनियम बेंचमार्क विकलांग व्यक्तियों के लिए अतिरिक्त लाभ प्रदान करता है, जैसे सरकारी प्रतिष्ठानों में रोजगार की रिक्तियां, शिक्षा के अवसर, भूमि आवंटन और गरीबी उन्मूलन योजनाएं, अन्य।

त्वरित न्याय सुनिश्चित करने के लिए, विकलांग व्यक्तियों के अधिकारों के उल्लंघन से संबंधित मामलों से निपटने के लिए प्रत्येक जिले में विशेष अदालतें स्थापित की जाती हैं। विकलांग व्यक्तियों के अधिकारों के उल्लंघन के लिए दंड 7,750 अमेरिकी डॉलर (500,000 रुपये) का मौद्रिक जुर्माना और पांच साल तक के कारावास तक बढ़ सकता है।

पीडब्ल्यूडी के उचित मौखिक/अशाब्दिक कम्युनिकेशन, योजनाओं और लाभों के लिए उपयोग और वकालत करें

हालांकि अधिनियम के तहत अधिकांश अनुपालन विशेष रूप से सरकारी प्रतिष्ठानों पर लागू होते हैं, निजी प्रतिष्ठान भी अधिनियम के दायरे में आते हैं और उन्हें निम्नलिखित आवश्यकताओं का पालन करना चाहिए:

- प्रतिष्ठान की वेबसाइट पर या प्रतिष्ठान परिसर के भीतर एक विशिष्ट स्थान पर एक समान अवसर नीति तैयार करें और प्रकाशित करें। नीति में कार्यस्थल पर विकलांग व्यक्तियों को प्रदान किए गए लाभों और सुविधाओं का विवरण होना चाहिए। नीति की एक प्रति राज्य आयुक्त के पास भी पंजीकृत होनी चाहिए
- 20 से अधिक कर्मचारियों वाले प्रतिष्ठानों को विकलांग व्यक्तियों की भर्ती और उनके लिए प्रदान की गई विशेष सुविधाओं की देखरेख के लिए एक संपर्क अधिकारी नियुक्त करना चाहिए
- प्रतिष्ठानों को नौकरी की रिक्तियों की पहचान करने की आवश्यकता होती है, जो विकलांग व्यक्तियों के लिए उपयुक्त होगी। सरकार से प्रोत्साहन प्राप्त करने वाले प्रतिष्ठानों के मामले में, नौकरी की रिक्तियों का न्यूनतम पांच प्रतिशत अनिवार्य रूप से विकलांग व्यक्तियों के लिए आरक्षित होना चाहिए
- नियोक्ता को कार्यस्थल के भीतर विकलांग व्यक्तियों के खिलाफ नाजायज भेदभाव का निषेध सुनिश्चित करना चाहिए
- नियोक्ता को विकलांग कर्मचारियों को उनकी पहुंच बढ़ाने के लिए अतिरिक्त सुविधाएं या विशेष लाभ प्रदान करना चाहिए, जैसे कि विशेष छुट्टी और प्रशिक्षण कार्यक्रम
- सभी प्रतिष्ठानों को विकलांग व्यक्तियों के संबंध में सरकार द्वारा जारी पहुंच मानदंडों के अनुरूप होना चाहिए। अभिगम्यता मानदंड कार्यस्थल अवसंरचना और कम्युनिकेशन प्रौद्योगिकियों से संबंधित हैं, जो विकलांग व्यक्तियों के लिए सुलभ होने चाहिए
- प्रत्येक कवर किए गए प्रतिष्ठान को अपने विकलांग कर्मचारियों का रिकॉर्ड रखना चाहिए

सारांश

- कार्यालय परिसर और अन्य कॉर्पोरेट वातावरण के भीतर और आसपास हरियाली न केवल कार्यस्थल की सजावट को बढ़ाने में मदद करती है, बल्कि कर्मचारियों की उत्पादकता पर भी सकारात्मक प्रभाव डालती है
- कार्यस्थलों में पौधे हवा को शुद्ध करते हैं; वे सीओ 2 (कार्बन डाइऑक्साइड गैस) और अन्य वाष्पशील कार्बनिक यौगिकों की एकाग्रता को कम करते हैं, हवा को ताजा और स्वस्थ रखते हैं
- बाहरी वनस्पति गर्मियों में कार्यालय ब्लॉक में और उसके आसपास गर्मी को नियंत्रित करती है, गर्मी के तनाव को कम करती है और एयर कंडीशनिंग की आवश्यकता को कम करती है
- हरी छतें और facades इन्सुलेशन या गर्मी की अवशोषण क्षमता का प्रसार करते हैं, हीटिंग और शीतलन खर्चों को कम करते हैं
- कार्यालय भवनों में और उसके आसपास के पौधे जल वाष्प छोड़ते हैं जो हवा को नम करता है, सिरदर्द को कम करता है
- खतरे को एक कारक के रूप में परिभाषित किया गया है, जो लोगों और संपत्तियों को समान रूप से नुकसान पहुंचा सकता है, जैसे बिजली, ज्वलनशील उत्पाद, विस्फोटक कंटेंट, संक्षारक रसायन, कार्यस्थल पर भारी सीढ़ी का उपयोग आदि।

- खतरों के खिलाफ पर्याप्त एहतियाती उपाय करने के लिए, कार्यस्थल में आमतौर पर पाए जाने वाले खतरों की पहचान करने की आवश्यकता होती है
- भारत का संविधान रोजगार या स्वस्थ या विकलांग से संबंधित मामलों में सभी नागरिकों (भारत के प्रत्येक कानूनी नागरिक सहित, चाहे वे विकलांग हों) के लिए अवसर की समानता पर समान रूप से लागू होता है
- कार्यस्थल पर महिलाओं का यौन उत्पीड़न (निवारण, प्रतिषेध और प्रतितोष) अधिनियम, 2013 कार्यस्थल पर महिलाओं के यौन उत्पीड़न के विरुद्ध शिकायतों की जांच और निवारण के लिए एक प्रणाली निर्धारित करता है
- 2016 एक्ट के तहत 'विकलांग व्यक्ति' की परिभाषा को व्यापक बनाया गया है: इसमें विकलांग व्यक्तियों, बेंचमार्क विकलांगता वाले व्यक्तियों और उच्च समर्थन आवश्यकताओं वाले विकलांग व्यक्तियों को शामिल किया गया है।

एक्टिविटी

गतिविधि 1

ऊर्जा संरक्षण - एक नमूना चेकलिस्ट तैयार करें और निगरानी करें

- यह गतिविधि "एक नमूना चेकलिस्ट तैयार करें और ऊर्जा उपयोग की निगरानी करें" के रूप में है
- यह गतिविधि प्रशिक्षुओं को कार्यस्थल में ऊर्जा के अनुकूलन को समझने का लक्ष्य रखती है
- प्रशिक्षक कक्षा को तीन समूहों में विभाजित करेगा
- ट्रेनर केस स्टडी के लिए एक विशेष कमरे को अलग करेगा
- प्रत्येक समूह को निम्नलिखित कार्य सौंपे जाएंगे
  - केस स्टडी रूम में लाइट, पंखे और एसी की संख्या गिनें
  - उनके उपयोग की अवधि नोट करें
  - उचित उपयोग और अपव्यय का आकलन करें
  - ऊर्जा उपयोग को अनुकूलित करने के तरीके का मूल्यांकन करने के लिए एक चेकलिस्ट तैयार करें
  - टिप्पणियों को प्रस्तुत करने वाला एक दस्तावेज जमा करें
- ट्रेनर दस्तावेजों की जांच करेगा और सर्वश्रेष्ठ समूह घोषित करेगा।

गतिविधि 2

अपशिष्ट प्रबंधन

- यह गतिविधि "अपशिष्ट प्रबंधन" के रूप में है
- प्रशिक्षक प्रत्येक प्रशिक्षु को नमूना खतरा माप चेकलिस्ट तैयार करने के लिए कहेगा
- प्रशिक्षुओं को भवन की अपशिष्ट प्रबंधन प्रणाली का आकलन करना चाहिए
- उन्हें मौजूदा अपशिष्ट प्रबंधन प्रणाली पर एक दस्तावेज तैयार करना चाहिए और इसे बढ़ाने के लिए सिस्टम का प्रस्ताव करना चाहिए
- उन्हें विभिन्न प्रकार के कचरे और उनके उपचार के बीच अलग करने में सक्षम होना चाहिए
- प्रशिक्षुओं द्वारा प्रस्तुत दस्तावेज की योग्यता के आधार पर, प्रशिक्षक सर्वश्रेष्ठ रिपोर्ट की घोषणा करेगा
- सर्वश्रेष्ठ रिपोर्ट प्रस्तुत करने वाले प्रशिक्षुओं को कक्षा द्वारा सराहा जाएगा।

अभ्यास

A. निम्नलिखित का मिलान करें:

कॉलम ए	कॉलम बी
कार्यस्थल पर महिलाओं का यौन उत्पीड़न (रोकथाम, निषेध और निवारण) अधिनियम	1995
विकलांग व्यक्ति अधिनियम	1992
मानसिक स्वास्थ्य अधिनियम	1999
भारतीय पुनर्वास परिषद	2013
राष्ट्रीय न्यास अधिनियम	1987

B. दिए गए उत्तरों में से सही उत्तर चुनें:

1. निम्नलिखित में से कौन सा विकल्प गलत है?
   a) हरियाली गर्मी को अवशोषित करती है और कार्यालय भवन को ठंडा रखती है
   b) सरकार द्वारा लागू कानून के अनुसार हरियाली अनिवार्य है
   c) हरियाली उत्पादकता बढ़ाती है
2. IEA का अर्थ है -
   a) भारतीय ऊर्जा एजेंसी
   b) भारतीय ऊर्जा प्राधिकरण
   c) अंतर्राष्ट्रीय ऊर्जा एजेंसी
3. जो नियोक्ता अनुपालन में विफल रहेंगे, उन्हें निम्न तक के जुर्माने से दंडित किया जाएगा -
   a) 50,000 रुपये
   b) INR 5,00,000
   c) अभी तक निर्धारित नहीं किया गया है

C. निम्नलिखित प्रश्नों के उत्तर दें

1. जोखिम प्रबंधन के बुनियादी चरण क्या हैं?
2. महिला सुरक्षा के संबंध में संगठन के निवारण तंत्र की प्रमुख विशेषताएं लिखिए।
3. नवीकरणीय ऊर्जा के सामान्य स्रोत क्या हैं?
4. PWD नीतियों के उल्लंघन के संभावित परिणाम क्या हो सकते हैं?
5. किसी व्यक्ति को बिजली के झटके से बचाने के उपाय लिखिए।

नोट्स

Skill India
कौशल भारत-कुशल भारत

सत्यमेव जयते
GOVERNMENT OF INDIA
MINISTRY OF SKILL DEVELOPMENT
& ENTREPRENEURSHIP

N·S·D·C
National
Skill Development
Corporation
Transforming the skill landscape

IT - ITeS SSC
nasscom

10. रोजगार कौशल

DGT/VSQ/N0102

Employability Skills is available at the following location

https://www.skillindiadigital.gov.in/content/list

Employability Skills

IT - ITeS SSC
nasscom

11. अनुलग्नक

Annexure of QR Codes for Test Engineers

Chapter No.	Unit No.	Topic	Page No.	QR Code Links	QR Code (s)	Video Duration
Chapter 2: Concept and Principle of Quality Testing	Unit 2.1 - Concept and Principle of Quality Testing	QA Testing and Version Control	41	https://www.youtube.com/watch?v=LKUtzEHYK2I	QATesting and Version Control	0:02:15
Chapter 3: Design Tests for Software Products/Applications/Modules	Unit 3.1 - Design Tests for Software Products/ Applications/ Modules	Design Tests for Software Products/ Applications/ Modules	68	https://www.youtube.com/watch?v=RDBzZ-9XnDE	Design Tests for Software Products/Applications/ Modules	00:02:06
Chapter 5: Contribute to Quality Assurance of Projects	Unit 5.1 - Contribute to Quality Assurance of Projects	Discuss the scope of quality assurance for a Test Engineer	108	https://www.youtube.com/watch?v=TAjwPW2iULY	Discuss the scope of quality assurance for a Test Engineer	00:02:12
Chapter 6: Key Indicators for Software Applications	Unit 6.1 - Key Indicators for Software Applications	The purpose of impact indicator, efficiency indicator in testing	126	https://www.youtube.com/watch?v=I1uT7mhalio	The purpose of impact indicator, efficiency indicator in testing	00:02:17
Chapter 7: Technical Skills for Manual Tests	Unit 7.1 - Technical Skills for Handling Incidents	Latest Changes, procedures and practice in the field of designing test suite	153	https://www.youtube.com/watch?v=URmCWIL0IA0	How to store and retrive information	00:02:42
Chapter 9: Maintain an Inclusive, Environmentally Sustainable Workplace	Unit 9.1 - Sustainable Practices	Demonstrate how to optimize usage of electricity, energy, materials, and water in various tasks	191	https://www.youtube.com/watch?v=wQ7zJYBuY74	Demonstrate how to optimize usage of electricity, energy, materials, and water in various tasks	0:02:27

नोट्स

Skill India
कौशल भारत-कुशल भारत

सत्यमेव जयते
GOVERNMENT OF INDIA
MINISTRY OF SKILL DEVELOPMENT
& ENTREPRENEURSHIP

N·S·D·C
National
Skill Development
Corporation
Transforming the skill landscape

IT - ITeS SSC
nasscom

आईटी – आईटीईएस सेक्टर स्किल काउंसिल nasscom
पता: प्लॉट नंबर – 7, 8, 9 और 10 सेक्टर – 126, नोएडा, उत्तर प्रदेश – 201303 नई दिल्ली – 110049
वेबसाइट: www.sscnasscom.com
ईमेल: ssc@nasscom.com
फ़ोन: 0120 4990111 – 0120 4990172

दाम: ₹