Puce de test 3

Table des matières

Le test d'API est un type de test logiciel qui valide une interface de programmation d'application (API) afin de s'assurer qu'elle répond aux attentes en termes de fonctionnalité, de fiabilité, de performances et de sécurité. Il se concentre principalement sur la couche de logique métier et l'échange de données entre les systèmes logiciels, indépendamment d'une interface utilisateur (UI)

Il s'agit de tester les URL entre les textes

https://policycentral.cloudapps.cisco.com/cppc/policy-advisor/policies/view-policy/1624

Le Code de conduite professionnelle de Cisco reflète notre façon de travailler et de prendre des décisions avec intégrité. Il fournit également des ressources pour aider à naviguer dans des questions complexes, comme l'utilisation responsable de l'IA et les conflits d'intérêts.

```
function reverseString(str) {
  return str.split("").reverse().join("");
}
```

https://cisco.account.box.com/login

https://cisco.service-now.com/helpzone?id=sc_cat_item&sys_id=a9860b89dbd9a640cb5772fc0f96191d&u_business_service=

Demandez de l'aide sur un problème que vous rencontrez. Un enregistrement d'incident sera créé et géré jusqu'à sa résolution. Vous serez également averti de la progression.

https://www.geeksforgeeks.org/software-testing/software-testing-manual-testing/

Dans la technique Black Box Testing, le testeur ou l'analyste QA vérifiera seulement la fonctionnalité du module particulier ou de la méthode particulière ou parfois l'application entière en fournissant les différents cas de test manuellement. Ici, le testeur fournit l'entrée de l'application et la teste manuellement.

S'il renvoie le résultat attendu, le testeur procède à un autre jeu d'entrées et signale tous les résultats à l'équipe. Si l'entrée fournie par l'utilisateur manuellement échoue pendant le test, il/elle signalera ce problème à l'équipe de développement.

TEST VIDÉO

Chèque	Tableau
	Vérifier LIEN

TABLE D'ESSAI

https://cisco.service-now.com/now/sow/record/incident/507c393193e672502c66ff60ed03d632

Test de boîte blanche

Dans la technique White Box Testing, la personne vérifiera manuellement la structure interne du système comme les conceptions, le codage, etc. Dans ce cas, l'équipe de développement examinera ligne par ligne l'intégralité du code pour s'assurer de son exactitude.

S'il découvre des différences ou des erreurs dans le code, il corrigera ou corrigera les erreurs dans le codage ou les conceptions. Ici, le procédé est entièrement réalisé manuellement et le procédé est efficace puisque le code ou la conception de contrôle est contrôlé manuellement par des humains.

https://en.wikipedia.org/wiki/Manual_testing

https://www.google.com/

La vérification du « rôle de développeur bdb » a migré de l'API ART vers l'ID d'entrée dans One Access. Lorsque vous demandez l'accès, assurez-vous de sélectionner "Méthode d'intégration : memberOf", car il existe deux droits portant le même nom.

Principaux aspects des tests API ☐

- Communication au niveau de la couche message : Au lieu d'utiliser une interface graphique utilisateur, les tests d'API interagissent directement avec les points d'extrémité (URI) de l'application à l'aide de diverses méthodes HTTP (GET, POST, PUT, DELETE) et de formats de données tels que JSON ou XML.
- Détection précoce des défauts : Les tests d'API peuvent être effectués tôt dans le cycle de développement du logiciel, avant même la création de l'interface utilisateur, ce qui permet aux équipes de trouver et de résoudre les problèmes plus efficacement et à moindre coût.
- Axe automatisation : En raison de la nature de l'interaction directe et de la structure

cohérente, les tests API sont bien adaptés à l'automatisation, qui est essentielle dans les environnements Agile et DevOps modernes pour les tests continus dans les pipelines CI/CD.
- Couverture complète : Il offre une couverture de test plus étendue que le seul test de l'interface utilisateur, y compris le test des cas de périphérie, la gestion des erreurs et les vulnérabilités de sécurité qui peuvent être difficiles d'accès via l'interface utilisateur.

Types de tests API
Différents types de tests API sont utilisés pour couvrir divers aspects de la qualité d'une application :

Katalon

- Tests fonctionnels : Vérifie que l'API exécute correctement les opérations prévues, en gérant les entrées, les sorties et les codes d'état comme spécifié.
- Test des performances : Évalue la vitesse, la stabilité et l'évolutivité de l'API dans diverses conditions de charge (par exemple, trafic de pointe, contrainte).
- Tests de sécurité : Identifie les vulnérabilités telles que l'injection SQL, le script intersite (XSS) et l'authentification/autorisation interrompue pour protéger les données sensibles.
- Tests d'intégration : Confirme que les différentes parties d'un système ou des services externes avec lesquels l'API interagit fonctionnent de manière transparente.
- Test du contrat : Garantit que l'API respecte un contrat convenu (spécification comme OpenAPI/Swaggeror WSDL), empêchant les modifications de rupture entre les mises à jour de service.
- Tests de bout en bout : Valide l'ensemble des parcours utilisateur impliquant plusieurs appels API enchaînés.
- Types de tests API
  Différents types de tests API sont utilisés pour couvrir divers aspects de la qualité d'une application :

  Le test manuel commence par la compréhension de ce que le logiciel est censé faire.

  - Exigences fonctionnelles : Vérifiez les fonctionnalités telles que la connexion correcte de l'utilisateur.
  - Exigences non fonctionnelles : valider les performances, la convivialité et la sécurité (par exemple, temps de chargement des pages inférieur à 2 secondes).
  - Témoignages d'utilisateurs et documents de conception : Comprendre les interactions et les workflows des utilisateurs
  - Commentaires des intervenants : Clarifier les exigences avec les clients, les chefs de produit ou les concepteurs.

  ## Étape 2: Création d'un plan de test

  Un plan de test définit la stratégie et les objectifs de test.

  - Portée : identifie les fonctionnalités à tester et les exclusions.
  - Objectifs : Garantit les fonctionnalités de base et l'expérience utilisateur.
  - Ressources : Spécifie les membres de l'équipe, les outils et les calendriers.

- Techniques de test : inclut les tests fonctionnels, de convivialité et exploratoires.
- Environnements : définit les configurations intermédiaires ou de type production.

Étape 3: Cas de test de conception

Les scénarios de test sont des scripts clairs, étape par étape, qui garantissent un test manuel complet. Les scénarios de test servent de guides détaillés pour les testeurs, en s'assurant que chaque scénario est vérifié. Chaque cas de test inclut :

- ID de test : Un code unique, comme TC_001, pour faciliter le suivi.
- Description: L'objectif, comme la vérification à l'aide d'entrées valides.
- Conditions préalables : Ce dont vous avez besoin avant de commencer, comme être sur la page de recherche.
- Étapes: Actions à entreprendre, par exemple choisir la date de demain et cliquer sur « Rechercher ».
- Résultat prévu : Le résultat souhaité, comme une liste de vols triés par prix.
- Conditions postales : Le système affiche la page de résultats.

En savoir plus: Comment rédiger des scénarios de test ?

Étape 4: Configuration de l'environnement de test

L'environnement de test doit ressembler étroitement à la production.

- Installer les applications requises.
- Configurez les paramètres spécifiques au projet.
- Garantir la disponibilité des données de test.
- Vérifier la configuration matérielle et logicielle requise.

Étape 5: Exécuter des scénarios de test

Exécutez les scénarios de test étape par étape et interagissez avec l'application en tant qu'utilisateur.

- Résultats réels : Ce qui se passe pendant l'exécution.
- État de réussite/échec : Indique si le résultat réel correspond au résultat attendu.
- Observations : Tout comportement inattendu ou problème de convivialité.

Étape 6: Défauts des journaux et des rapports

Lorsqu'un test échoue ou qu'un comportement inattendu se produit, consignez les défauts avec :

- ID de défaut : identifiant unique.
- Résumé : Brève description du défaut réel
- Étapes de reproduction : Étapes détaillées pour déclencher le problème.
- Résultats réels par rapport aux résultats prévus : Ce qui s'est passé par rapport à ce qui aurait dû se passer.

- Gravité : vérifiez si l'impact est critique, majeur ou mineur.
- Pièces jointes : captures d'écran, journaux ou vidéos pour prouver le défaut.

Étape 7: Suivi et vérification des défauts

Une fois les correctifs appliqués :

- Suivre l'état des défauts dans l'outil.
- Retestez les problèmes résolus.
- Fermer ou rouvrir les défauts en fonction des résultats.

Étape 8: Réaliser des tests de régression

Le test de régression garantit que les correctifs de défauts ou les nouvelles modifications n'ont pas endommagé les fonctionnalités existantes.

- Les zones affectées sont vérifiées après la résolution des bogues.
- Vérifiez les fonctionnalités critiques.
- Vérifiez les points d'intégration pour vous assurer qu'ils fonctionnent comme avant.

Étape 9: Préparer les rapports de fermeture des tests

Une fois le test terminé, calculez les résultats par rapport aux objectifs du plan de test et créez un rapport de clôture de test pour le même :

- Résumé: Présentation des activités de test.
- Résultats des tests : Nombre de cas de test exécutés, réussis et ayant échoué.
- Défauts détectés : Nombre total de défauts, leur gravité et leur état de résolution.
- Problèmes en suspens : Tout défaut ou risque non résolu.
- Leçons apprises : Informations pour les tests futurs.

Étape 10: Formuler des commentaires et des recommandations

Analyser les résultats des tests pour fournir des commentaires exploitables aux parties prenantes, tels que :

- Qualité du logiciel.
- Amélioration des processus.
- Stratégies de test futures.
- Aperçu de l'expérience utilisateur.

Outils utilisés pour les tests manuels

- TestRail : un outil de gestion de tests convivial pour organiser, exécuter et générer des rapports sur les cas de tests manuels avec des intégrations et des tableaux de bord puissants
- Xray (pour Jira) : outil de gestion des tests basé sur Jira qui prend en charge les tests

manuels, automatisés et BDD avec une traçabilité complète et une intégration transparente
- Qase : une plate-forme de gestion de tests moderne basée sur le cloud avec une interface utilisateur simple, la création de cas de test basée sur l'IA et le suivi intégré des défauts
- Zephyr : une solution de gestion de tests évolutive prenant en charge les tests manuels et exploratoires avec une intégration Jira puissante et des fonctionnalités de reporting
- Tuskr : un outil de gestion de tests basé sur le cloud, léger et abordable, doté d'une interface intuitive et de fonctionnalités de collaboration.

Nécessité d'un test manuel

- Sans bogue et stabilité : le principal objectif des tests manuels est de s'assurer que l'application est sans bogue, stable, conforme aux exigences et fournit un produit stable aux clients.
- Connaissance du produit : le test manuel permet aux ingénieurs de test de se familiariser avec le produit et d'avoir une vision de l'utilisateur final. Cela les aide à écrire les scénarios de test corrects pour le logiciel.
- Réparation des défauts : un test manuel permet de s'assurer que les défauts sont réparés par le développeur et que des tests ont été effectués sur les défauts réparés.

Avantages des tests manuels

- Commentaires visuels rapides et précis : Il détecte presque tous les bogues de l'application logicielle et est utilisé pour tester les conceptions dynamiques de l'interface graphique utilisateur comme la disposition, le texte, etc.
- Moins cher : moins cher, car il ne nécessite aucune compétence de haut niveau ni aucun type d'outil spécifique.
- Aucun codage n'est requis : Aucune connaissance en programmation n'est requise lors de l'utilisation de la méthode de test de la boîte noire. Il est facile d'apprendre pour les nouveaux testeurs.
- Efficacité pour les modifications non planifiées : le test manuel est adapté en cas de modifications non planifiées de l'application, car il peut être facilement adopté.

Katalon

- Tests fonctionnels : Vérifie que l'API exécute correctement les opérations prévues, en gérant les entrées, les sorties et les codes d'état comme spécifié.
- Test des performances : Évalue la vitesse, la stabilité et l'évolutivité de l'API dans diverses conditions de charge (par exemple, trafic de pointe, contrainte).
- Tests de sécurité : Identifie les vulnérabilités telles que l'injection SQL, le script intersite (XSS) et l'authentification/autorisation interrompue pour protéger les données sensibles.
- Tests d'intégration : Confirme que les différentes parties d'un système ou des services externes avec lesquels l'API interagit fonctionnent de manière transparente.
- Test du contrat : Garantit que l'API respecte un contrat convenu (spécification comme OpenAPI/Swaggeror WSDL), empêchant les modifications de rupture entre les mises à jour de service.
- Tests de bout en bout : Valide l'ensemble des parcours utilisateur impliquant plusieurs appels API enchaînés.

- Comment ça fonctionne

  Des outils visuels et sans code vous permettent de créer, d'étendre et d'organiser facilement des tests sur des API, des interfaces utilisateur Web, des bases de données, des ESB et même des serveurs MCP courants dans les systèmes infusés par l'IA. Aucune compétence technique approfondie n'est requise. Prenant en charge plus de 120 protocoles et formats de messages, SOAtest vous offre un cadre unifié pour valider la logique métier de bout en bout.

  Grâce à SOAtest, vous pouvez :

  - Créez des flux basés sur des scénarios qui imitent les transactions commerciales réelles, vous aidant à trouver les bogues cachés déclenchés par des séquences spécifiques.
  - Créer une logique de test, des assertions complexes, des boucles et des flux pilotés par les données avec un minimum d'expertise technique.
  - Exécutez des tests individuels ou des suites complètes et associez des contrôles de régression pour détecter immédiatement les modifications inattendues.

JavaScript Statements

Multiple statements on one line are allowed.

JavaScript Statements

Multiple statements on one line are allowed.

JavaScript Statements

Multiple statements on one line are allowed.

JavaScript Statements

Multiple statements on one line are allowed.

JavaScript Statements

```
Multiple statements on one line are allowed.
```

Qu'est-ce que le test manuel logiciel ?

Le test manuel est la procédure permettant de vérifier le logiciel à l'aide de ses différentes fonctionnalités. Il est guidé par un ensemble de tests préconçus qui valident le logiciel et fournit un rapport final sur les résultats. Ce type de test prend du temps car il est entièrement effectué par le biais d'efforts manuels. Par conséquent, il y a toujours une marge d'erreur humaine lors de l'exécution de ce type de test.

Chaque nouveau logiciel est d'abord testé manuellement avant d'adopter l'automatisation. La vérification manuelle d'un logiciel complet prend plus de temps. Une fois que toutes les fonctionnalités du logiciel sont stables et fonctionnent correctement, certains des cas de tests manuels peuvent être convertis en automatisation. Les scénarios de test manuels sont évalués en premier pour vérifier s'ils peuvent être entièrement automatisés. Ce type de test ne nécessite pas l'utilisation d'outils d'automatisation pour effectuer l'ensemble du processus.

Caractéristiques du test manuel du logiciel

Les caractéristiques des tests manuels du logiciel sont répertoriées ci-dessous −

- Le test manuel est effectué entièrement à l'aide d'une intervention humaine.
- Les tests exploratoires constituent une partie importante des tests manuels. Lors des tests exploratoires, les testeurs vérifient le logiciel sans aucun ensemble prédéterminé de tests. Il détecte les défauts imprévus et améliore la satisfaction du client.
- Le test manuel est flexible, car il permet de modifier les scénarios de test en fonction des modifications des exigences et d'autres conditions de test.
- Le test manuel peut être adopté dès les premières étapes du cycle de développement du logiciel (SDLC).
- Certains des cas de tests compliqués peuvent être exécutés uniquement manuellement, sans aucune automatisation.
- Le test manuel est utile pour valider l'interface utilisateur du logiciel. Il permet de vérifier l'affichage, la réactivité et la conception normale du logiciel.

Pourquoi le test manuel du logiciel est-il nécessaire ?

Le test manuel du logiciel est nécessaire pour les raisons énumérées ci-dessous −

- Le test manuel confirme que le logiciel est exempt de défauts, qu'il fonctionne correctement selon les exigences et qu'il est suffisamment stable pour être déployé en production.
- Le test manuel permet aux testeurs de s'habituer au logiciel et de comprendre comment le logiciel répond aux clients. Cela permet de développer des scénarios de test efficaces.
- Le test manuel identifie et résout les défauts du logiciel.

Étapes du test manuel du logiciel

Les différentes étapes du test manuel du logiciel sont répertoriées ci-dessous −

Étape 1− La première étape comprend la phase d'analyse des exigences en parcourant les documents relatifs aux exigences et aux spécifications, les guides, etc.

Étape 2− La deuxième étape consiste en la création d’un plan de test portant sur l’ensemble des exigences.

Étape 3− La troisième étape consiste à créer des scénarios de test couvrant chaque exigence.

Étape 4− La quatrième étape consiste à exécuter des scénarios de test dans l'environnement de test approprié.

Étape 5− La cinquième étape consiste à analyser les résultats de l'exécution du test et à signaler les écarts comme des défauts.

Étape 6− La sixième étape consiste à corriger les défauts et à les tester de nouveau. Elle inclut également la réexécution des cas de test ayant échoué.

Types de tests manuels de logiciels

Les différents types de tests manuels de logiciels sont répertoriés ci-dessous −

- Black Box Testing− Il s'agit de la technique de test dans laquelle le testeur n'a aucune connaissance du fonctionnement interne du logiciel. Il s'agit principalement de vérifier si les fonctionnalités fonctionnent correctement en fonction des besoins de l'utilisateur.
- White Box Testing− C'est la procédure de test qui inclut la vérification de la structure interne, et le code source du programme du logiciel.
- Test de boîte grise− C'est la technique de test qui utilise à la fois les principes de la boîte noire et des techniques de test de boîte blanche.

Outils utilisés pour le test manuel des logiciels

Les différents outils utilisés pour le test manuel du logiciel sont répertoriés ci-dessous −

- Lien de test
- Bugzilla
- Jira
- LoadRunner
- jaugeur Apache
- Parfait

Différences entre les tests manuels et les tests d'automatisation

Voici une comparaison des tests manuels et des tests d'automatisation du logiciel −

Test manuel	Test d'automatisation
Il s'agit de la procédure permettant de vérifier le logiciel à l'aide d'efforts manuels.	Il s'agit de la procédure permettant de vérifier le logiciel à l'aide des outils

	d'automatisation.
Elle implique l'exécution manuelle des scénarios de test.	Elle implique l'exécution des scénarios de test via des scripts et des outils d'automatisation.
Il est moins productif et nécessite plus de temps pour être terminé.	Il est plus productif et nécessite moins de temps pour être terminé.
Il ne garantit pas une couverture de test à cent pour cent.	Il assure une couverture de test supérieure à celle des tests manuels.
Il ne nécessite pas de compétences en programmation. Elle ne peut être effectuée qu'à la connaissance du logiciel.	Cela nécessite des compétences en programmation.

Avantages du test manuel des logiciels

Les avantages du test manuel du logiciel sont répertoriés ci-dessous −

- Le test manuel permet de vérifier les éléments changeant dynamiquement sur l'écran.
- Les tests manuels sont bon marché et ne font pas appel à des ressources qualifiées.
- Le test manuel peut être effectué par des testeurs n'ayant aucune connaissance en programmation.
- Le test manuel peut être adopté très rapidement et est approprié pour s'adapter à des changements imprévisibles dans le logiciel.

Inconvénients du test manuel du logiciel

Les inconvénients du test manuel du logiciel sont répertoriés ci-dessous −

- Le test manuel n'est pas très fiable et laisse place à des erreurs humaines.
- Des jeux distincts de scénarios de test manuel doivent être développés pour différents modules, ce qui réduit considérablement la possibilité de réutilisation.
- Le test manuel est totalement dépendant de l'exécution manuelle des tests. Cependant, certaines étapes du test ne peuvent pas être effectuées avec les efforts manuels.
- Les testeurs qui effectuent des tests manuels doivent avoir l'expérience de l'utilisation du logiciel. En outre, il n'y a aucune garantie que toutes les fonctionnalités du logiciel ont été couvertes lors de l'exécution des tests manuels.
- Le test manuel est généralement une activité fastidieuse.

Conclusion

Nous voici à la fin de notre présentation complète du didacticiel sur le test manuel des logiciels.

Nous avons commencé par décrire ce qu'est le test manuel du logiciel, quelles sont les caractéristiques du test manuel du logiciel, pourquoi le test manuel du logiciel est nécessaire, quelles sont les différentes étapes du test manuel du logiciel, quels sont les différents types de test manuel du logiciel, quels sont les différents outils utilisés pour le test manuel du logiciel, quelles sont les différences entre le manuel du logiciel et le test d'automatisation, quels sont les avantages du test manuel du logiciel, et quels sont les inconvénients du test manuel du logiciel. Vous disposez ainsi d'une connaissance approfondie du test manuel des logiciels. Il est sage de continuer à mettre en pratique ce que vous avez appris et à explorer d'autres aspects pertinents pour le test de logiciels afin d'approfondir vos connaissances et d'élargir vos horizons.

Qu'est-ce que le test d'accessibilité ?

Le test d'accessibilité est un sous-ensemble du test d'utilisabilité où les utilisateurs considérés sont des personnes avec toutes les capacités et tous les handicaps. L'importance de ce test est de vérifier à la fois la convivialité et l'accessibilité.

L'accessibilité vise à répondre aux besoins des personnes de différentes capacités telles que :

- Troubles Visuels
- Déficience Physique
- Déficience Auditive
- Déficience Cognitive
- Trouble D'Apprentissage

Une bonne application web devrait s'adresser à tous les groupes de personnes et PAS seulement aux personnes handicapées. Ceux-ci incluent :

1. Utilisateurs avec une infrastructure de communication médiocre
2. Les personnes âgées et les nouveaux utilisateurs, qui sont souvent analphabètes en informatique
3. Utilisateurs utilisant l'ancien système (NON capable d'exécuter le dernier logiciel)
4. Utilisateurs qui utilisent un équipement NON standard
5. Utilisateurs, qui ont un accès restreint

Comment effectuer des tests d'accessibilité

La Web Accessibility Initiative (WAI) décrit la stratégie d'examen préliminaire et d'examen de la conformité des sites Web. La Web Accessibility Initiative (WAI) inclut une liste d'outils logiciels pour faciliter les évaluations de conformité. Ces outils vont de problèmes spécifiques tels que le daltonisme à des outils qui exécuteront des outils automatisés de spidering.

Outils de test d'accessibilité Web

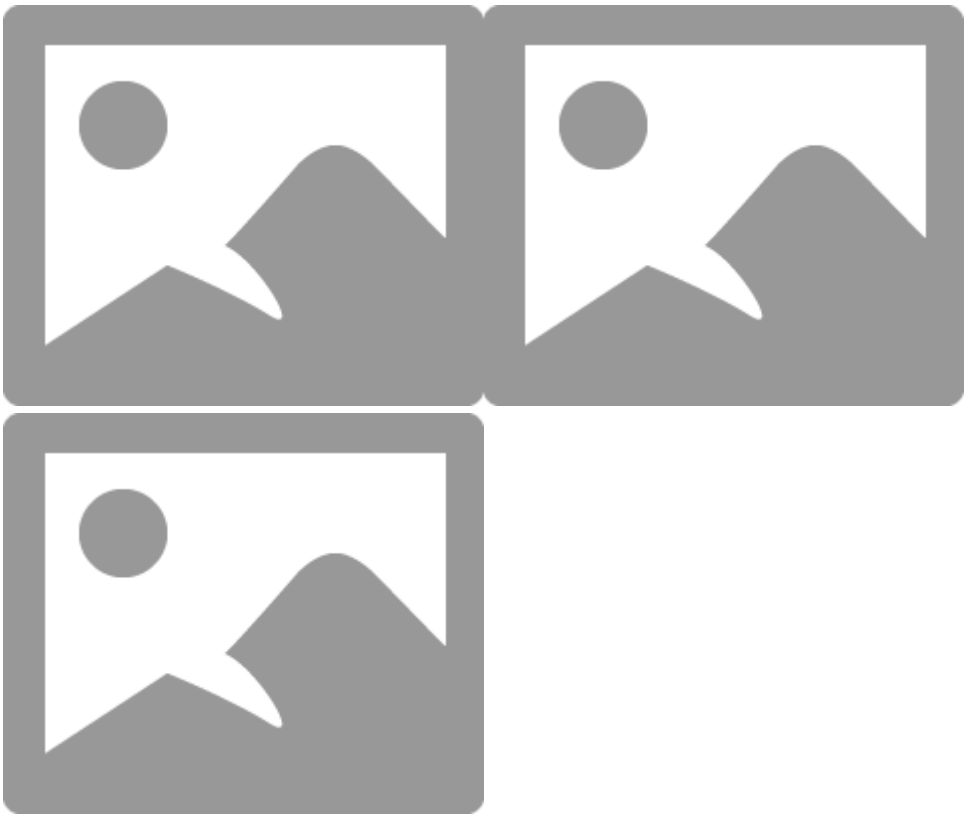
Product (produit)	Fournisseur	URL
VérificationAcc	HiSoftware	http://www.hisoftware.com
Bobby	Feu De Veille	http://www.watchfire.com
WebXM	Feu De Veille	http://www.watchfire.com
Montée En Rampe	Deque	http://www.deque.com
InFocus	Technologies SSB	http://www.ssbtechnologies.com/

Rôle des outils automatisés dans les tests d'acceptation

Les outils de test d'accessibilité automatisés mentionnés ci-dessus sont très efficaces pour identifier les pages et les lignes de code qui doivent être vérifiées manuellement pour l'accessibilité.

1. vérifier la syntaxe du code du site
2. Rechercher les modèles connus que les humains ont répertoriés
3. identifier les pages contenant des éléments susceptibles de poser des problèmes
4. identifier certains problèmes réels d'accessibilité
5. identifier certains problèmes potentiels

L'interprétation des résultats des outils de test d'accessibilité automatisés nécessite une expérience des techniques d'accessibilité avec une compréhension des problèmes techniques et de convivialité.



Les tests sont effectués de manière formelle et informelle afin d'améliorer la qualité des logiciels. Une fois les tests formels terminés, une série de tests informels et arbitraires est effectuée. On parle alors de test ad hoc.

Qu'est-ce que le test ad hoc ?

Un test ad hoc est une technique de test informelle effectuée sur le logiciel pour détecter les défauts. Il est effectué dans un format aléatoire, et est également connu sous le nom de test de singe. Un test ad hoc ne suit pas une approche systématique et est dépourvu de cas de test bien documentés.

Les tests ad hoc ne comportent aucune documentation, aucun scénario de test, aucun cas, etc. Les développeurs ont du mal à corriger les défauts détectés par les tests ad hoc en raison de l'absence de ces documents de test. En outre, certains bogues critiques, rares et imprévus ne sont identifiés qu'en effectuant des tests aléatoires et informels sur le logiciel. Il s'agit également d'une sorte de test d'acceptation qui permet de gagner du temps lors de la création de nouveaux scénarios de test.

Un exemple pratique de test ad hoc est supposé qu'un logiciel doit être expédié au client dans une journée et que son développement est terminé juste un jour avant cela. À ce stade, il n'y a plus de temps pour créer et exécuter des scénarios de test, de sorte que l'équipe de test effectue des tests ad hoc sur l'ensemble du logiciel sur la base de la connaissance et de l'expérience globales du produit.

Types de tests ad hoc

Les différents types de tests ad hoc sont répertoriés ci-dessous −

Test D'Équipage

Dans les tests de copains, au moins deux membres participent au processus de test : un développeur et un testeur. Une fois que le développeur a terminé l'implémentation d'un composant, il effectue des tests unitaires sur celui-ci. Publier que le testeur fournit des données aléatoires et arbitraires au même composant et examine les résultats. En cas d'erreurs, le développeur corrige ces défauts.

Essai Par Paires

Dans le test de paires, deux testeurs sont impliqués. L'un d'eux effectue une vérification informelle et aléatoire du logiciel, et l'autre conserve un enregistrement des résultats des tests. Ainsi, les deux travaillent en couple et échangent des idées, des connaissances afin que le test soit fait correctement.

Caractéristiques des tests ad hoc

Les fonctionnalités des tests ad hoc sont répertoriées ci-dessous −

- Il s'agit d'une approche aléatoire et informelle des tests.
- Il n'est pris en charge par aucune documentation, scénarios de test, cas, etc.
- Il est effectué une fois les tests formels terminés.
- Elle ne suit pas une approche méthodique ou structurée.
- La réalisation de tests ad hoc prend moins de temps.
- Il détecte les bogues sur le logiciel lorsque les cas de test ne sont pas disponibles.

Quand les tests ad hoc sont-ils effectués ?

Le test ad hoc est effectué dans les scénarios répertoriés ci-dessous &mini ;

- Le temps disponible pour tester le logiciel est limité.
- Les tests formels ont été effectués.
- Les scénarios de test ne sont pas disponibles.

Quand le test ad hoc n'est-il pas terminé ?

Le test ad hoc n'est pas effectué dans les scénarios répertoriés ci-dessous −

- Cela n'est pas fait si des bogues sont détectés en exécutant les cas de test.
- Au moment des tests bêta, ce n'est pas fait.

Avantages des tests ad hoc

Les avantages des tests ad hoc sont répertoriés ci-dessous −

- Il n'adhère à aucun processus, de sorte que les tests ad hoc peuvent être effectués à tout moment du cycle de développement du logiciel.
- L'équipe de test peut vérifier le logiciel et détecter les erreurs en appliquant de nouvelles techniques de test sans se fier uniquement aux cas de test.
- Un développeur peut effectuer des tests ad hoc sur le même module qu'il développe et améliorer la qualité de son code.
- Bien que le processus de test formel prenne beaucoup de temps, les tests ad hoc peuvent être effectués en peu de temps.
- Il ne nécessite aucune documentation.

Inconvénients des tests ad hoc

Les inconvénients des tests ad hoc sont répertoriés ci-dessous −

- Les tests ad hoc doivent être effectués par des membres de l'équipe qui ont une expérience des tests et une connaissance approfondie du produit. Un membre inexpérimenté de l'équipe ne peut pas effectuer de test ad hoc.
- En cas de bug, il est difficile de reproduire le même puisque le test ad hoc n'est pas piloté par une planification.

Meilleures pratiques à suivre dans les tests ad hoc

Les meilleures pratiques à suivre dans les tests ad hoc sont répertoriées ci-dessous −

- Recueillir toutes les connaissances sur le produit.
- Identifier les composants du logiciel susceptibles de présenter des défauts et les hiérarchiser.
- Utilisation d'outils de test appropriés.

Conclusion

Nous voici à la fin de notre présentation complète du didacticiel sur le test ad hoc des logiciels. Nous avons commencé par décrire ce qu'est un test ad hoc, quels sont les types, les caractéristiques, les techniques, les avantages, les inconvénients, le temps et les meilleures pratiques de test ad hoc.

Vous disposez ainsi d'une connaissance approfondie des tests logiciels ad hoc. Il est sage de continuer à mettre en pratique ce que vous avez appris et à explorer d'autres aspects pertinents pour le test de logiciels afin d'approfondir vos connaissances et d'élargir vos horizons.

À propos de cette traduction
Cisco a traduit ce document en traduction automatisée vérifiée par une personne dans le cadre d'un service mondial permettant à nos utilisateurs d'obtenir le contenu d'assistance dans leur propre langue.

Il convient cependant de noter que même la meilleure traduction automatisée ne sera pas aussi précise que celle fournie par un traducteur professionnel.