Testopsommingsteken 3

Inhoud

API-testen is een type softwaretest dat een API (Application Programming Interface) valideert om ervoor te zorgen dat deze voldoet aan de verwachtingen voor functionaliteit, betrouwbaarheid, prestaties en beveiliging. Het richt zich voornamelijk op de business logic layer en data-uitwisseling tussen softwaresystemen, onafhankelijk van een gebruikersinterface (UI)

Dit is om URL's tussen teksten te testen

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

De Code of Business Conduct (COBC) van Cisco weerspiegelt hoe we werken en beslissingen nemen met integriteit. Het biedt ook middelen om te helpen navigeren door complexe problemen, zoals verantwoord AI-gebruik en belangenconflicten.

```
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=

Hulp vragen bij een probleem dat je hebt. Er wordt een incidentrecord gemaakt en beheerd tot een succesvolle oplossing. U wordt ook op de hoogte gehouden van de voortgang.

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

In de Black Box Testing techniek controleert de tester of de QA-analist alleen de functionaliteit van de specifieke module of bepaalde methode of soms de gehele toepassing door de verschillende testcases handmatig te verstrekken. Hier zal de tester de invoer voor de toepassing geven en deze handmatig testen.

Als het de verwachte uitvoer retourneert, gaat de tester verder met een andere set ingangen en

rapporteert hij alle resultaten aan het team. Als de invoer die de gebruiker handmatig heeft opgegeven tijdens het testen is mislukt, zal hij/zij dit probleem melden aan het ontwikkelteam.

TESTVIDEO

cheque	tafel
	LINK controleren

TESTTABEL

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

White Box-test

In White Box Testing techniek, zal de persoon de interne structuur van het systeem, zoals ontwerpen, codering, etc., handmatig controleren. Hier zal het ontwikkelingsteam de gehele coderingsdeelregel per regel bekijken om de juistheid van de code te garanderen.

Als hij / zij verschillen of fouten in de code vindt, zullen ze de fouten in de codering of ontwerpen corrigeren of repareren. Hier wordt het proces volledig handmatig uitgevoerd en het proces is efficiënt omdat de controlecode of het ontwerp handmatig door mensen wordt gecontroleerd.

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

https://www.google.com/

De "bdb developer role" check is gemigreerd van ART API naar Entra ID binnen One Access. Zorg er bij het aanvragen van toegang voor dat u "Integratiemethode: lid van" selecteert, omdat er twee rechten met dezelfde naam zijn.

Belangrijkste aspecten van API-testen ☐

- Communicatie op de berichtenlaag: in plaats van een GUI te gebruiken, communiceren API-tests rechtstreeks met de eindpunten (URI's) van de toepassing met behulp van verschillende HTTP-methoden (GET, POST, PUT, DELETE) en gegevensformaten zoals JSON of XML.
- Vroege detectie van defecten: API-tests kunnen vroeg in de levenscyclus van de softwareontwikkeling worden uitgevoerd, zelfs voordat de gebruikersinterface is gebouwd, zodat teams problemen efficiënter en tegen lagere kosten kunnen vinden en oplossen.
- Focus op automatisering: vanwege de aard van directe interactie en consistente structuur zijn API-tests zeer geschikt voor automatisering, wat van cruciaal belang is in moderne Agile- en DevOps-omgevingen voor continue testen in CI / CD-pijpleidingen.

- Uitgebreide dekking: het biedt een bredere testdekking in vergelijking met alleen UI-testen, inclusief het testen van edge-cases, foutverwerking en beveiligingsproblemen die mogelijk moeilijk toegankelijk zijn via de UI.

Soorten API-testen
Verschillende soorten API-tests worden gebruikt om verschillende aspecten van de kwaliteit van een toepassing te behandelen:

Katalon

- Functioneel testen: Controleert of de API de beoogde bewerkingen correct uitvoert en de invoer, uitvoer en statuscodes zoals gespecificeerd verwerkt.
- Prestatietests: beoordeelt de snelheid, stabiliteit en schaalbaarheid van de API onder verschillende belastingsomstandigheden (bijv. piekverkeer, stress).
- Beveiligingstests: identificeert kwetsbaarheden zoals SQL-injectie, cross-site scripting (XSS) en kapotte verificatie / autorisatie om gevoelige gegevens te beschermen.
- Integratietesten: Bevestigt dat verschillende onderdelen van een systeem of externe services waarmee de API communiceert, naadloos samenwerken.
- Contracttesten: zorgt ervoor dat de API zich houdt aan een overeengekomen contract (specificatie zoals OpenAPI / Swagger of WSDL), waardoor wijzigingen tussen service-updates worden voorkomen.
- End-to-End testen: hiermee worden volledige gebruikersritten gevalideerd waarbij meerdere API-aanroepen aan elkaar zijn gekoppeld.
- Soorten API-testen
  Verschillende soorten API-tests worden gebruikt om verschillende aspecten van de kwaliteit van een toepassing te behandelen:

  Handmatig testen begint met het begrijpen van wat de software naar verwachting zal doen.

  - Functionele vereisten: controleer functies zoals de juiste gebruikersaanmelding.
  - Niet-functionele vereisten: prestaties, bruikbaarheid en beveiliging valideren (bijv. laadtijd van pagina's minder dan 2 seconden).
  - Gebruikersverhalen en ontwerpdocumenten: inzicht in gebruikersinteracties en workflows.
  - Input van belanghebbenden: Verduidelijk de vereisten met klanten, productmanagers of ontwerpers.

Stap 2: Het opstellen van een testplan

Een testplan definieert de teststrategie en -doelstellingen.

- Bereik: identificeert te testen functies en uitsluitingen.
- Doelstellingen: Zorgt voor kernfunctionaliteit en gebruikerservaring.
- Bronnen: geeft teamleden, tools en tijdlijnen op.
- Testtechnieken: omvat functionele, bruikbare en verkennende tests.
- Omgevingen: definieert staging of productie-achtige instellingen.

Stap 3: Testcases ontwerpen

Testcases zijn duidelijke, stapsgewijze scripts die een grondige handmatige test garanderen. Testcases fungeren als gedetailleerde handleidingen voor testers, zodat elk scenario wordt gecontroleerd. Elke testcase omvat:

- Test ID: Een unieke code, zoals TC_001, voor eenvoudig volgen.
- Beschrijving: Het doel, zoals controleren met geldige ingangen.
- Voorvoorwaarden: wat je nodig hebt voordat je begint, zoals op de zoekpagina staan.
- Stappen: te ondernemen acties, zoals de datum van morgen kiezen en op 'Zoeken' klikken.
- Verwacht resultaat: Het gewenste resultaat, zoals een lijst met vluchten gesorteerd op prijs.
- Postvoorwaarden: Het systeem toont de resultatenpagina.

Lees meer: Hoe schrijf je testcases?

Stap 4: De testomgeving instellen

De testomgeving moet sterk op de productie lijken.

- Installeer de benodigde toepassingen.
- Projectspecifieke instellingen configureren.
- Zorgen voor beschikbaarheid van testgegevens.
- Controleer de hardware- en softwarevereisten.

Stap 5: Testcases uitvoeren

Voer testcases stap voor stap uit en communiceer met de applicatie als gebruiker.

- Werkelijke resultaten: wat gebeurt er tijdens de uitvoering.
- Pass/Fail Status: of het werkelijke resultaat overeenkomt met het verwachte resultaat.
- Waarnemingen: elk onverwacht gedrag of gebruiksproblemen.

Stap 6: Log- en rapporteer defecten

Wanneer een test mislukt of onverwacht gedrag optreedt, log defecten met:

- Defect-ID: Unieke id.
- Samenvatting: Korte beschrijving van wat het werkelijke defect is
- Stappen om te reproduceren: Gedetailleerde stappen om het probleem te activeren.
- Werkelijke versus verwachte resultaten: wat er is gebeurd versus wat er had moeten gebeuren.
- Ernst: Controleer of de impact kritiek, groot of klein is.
- Bijlagen: screenshots, logs of video's voor het bewijs van het defect.

Stap 7: defecten opsporen en verifiëren

Nadat fixes zijn toegepast:

- Volg de defectstatus in de tool.
- Hertest de vaste problemen.
- Sluit of heropen defecten op basis van de resultaten.

Stap 8: Regressietesten uitvoeren

Regressietests zorgen ervoor dat defectoplossingen of nieuwe wijzigingen de bestaande functionaliteit niet hebben verbroken.

- Getroffen gebieden worden gecontroleerd na het oplossen van bugs.
- Controleer de kritieke functies.
- Controleer integratiepunten om ervoor te zorgen dat ze werken zoals voorheen.

Stap 9: Testafsluitingsrapporten voorbereiden

Als het testen is voltooid, berekent u de resultaten aan de hand van de doelstellingen van het testplan en maakt u een testafsluitingsrapport voor hetzelfde:

- Samenvatting: Overzicht van testactiviteiten.
- Testresultaten: aantal uitgevoerde, goedgekeurde en mislukte testgevallen.
- Gevonden defecten: totale defecten, hun ernst en oplossingsstatus.
- Onopgeloste problemen: onopgeloste gebreken of risico's.
- Lessons Learned: Inzichten voor toekomstige tests.

Stap 10: Feedback en aanbevelingen geven

Analyseer testresultaten om uitvoerbare feedback te geven aan belanghebbenden, zoals:

- Software kwaliteit.
- Procesverbeteringen.
- Toekomstige teststrategieën.
- Inzichten in gebruikerservaring.

Gereedschappen die worden gebruikt voor handmatige tests

- TestRail: een gebruiksvriendelijke testmanagementtool voor het organiseren, uitvoeren en rapporteren van handmatige testcases met sterke integraties en dashboards
- Xray (voor Jira): een op Jira gebaseerde testbeheertool die handmatige, geautomatiseerde en BDD-tests ondersteunt met volledige traceerbaarheid en naadloze integratie
- Qase: een modern cloudgebaseerd testmanagementplatform met een eenvoudige gebruikersinterface, AI-aangedreven testcase-creatie en ingebouwde defecttracking
- Zephyr: een schaalbare testbeheeroplossing die handmatige en verkennende tests ondersteunt met sterke Jira-integratie en rapportagefuncties
- Tuskr: een lichtgewicht en betaalbare cloudgebaseerde testbeheertool met een

intuïtieve interface en samenwerkingsfuncties.

Behoefte aan manueel testen

- Bugvrij en stabiel: Het belangrijkste doel van handmatig testen is ervoor te zorgen dat de toepassing bugvrij, stabiel, in overeenstemming met de vereisten is en een stabiel product aan de klanten levert.
- Bekendheid met het product: handmatige tests helpen de testtechnici om meer vertrouwd te raken met het product en een eindgebruikersperspectief te krijgen. Dit helpt hen om de juiste testcases voor de software te schrijven.
- Het verhelpen van de gebreken: Handmatig testen helpt om ervoor te zorgen dat de gebreken worden verholpen door de ontwikkelaar en dat hertesten is gedaan op de vaste gebreken.

Voordelen van manueel testen

- Snelle en nauwkeurige visuele feedback: het detecteert bijna elke bug in de softwaretoepassing en wordt gebruikt om de dynamisch veranderende GUI-ontwerpen zoals lay-out, tekst, enz.
- Minder duur: het is goedkoper omdat het geen vaardigheden op hoog niveau of een specifiek type gereedschap vereist.
- Er is geen codering vereist: er is geen programmeerkennis vereist bij het gebruik van de black box-testmethode. Het is gemakkelijk te leren voor de nieuwe testers.
- Efficiënt voor ongeplande wijzigingen: handmatige tests zijn geschikt in geval van ongeplande wijzigingen in de toepassing, omdat deze eenvoudig kunnen worden overgenomen.

Katalon

- Functioneel testen: Controleert of de API de beoogde bewerkingen correct uitvoert en de invoer, uitvoer en statuscodes zoals gespecificeerd verwerkt.
- Prestatietests: beoordeelt de snelheid, stabiliteit en schaalbaarheid van de API onder verschillende belastingsomstandigheden (bijv. piekverkeer, stress).

    - Beveiligingstests: identificeert kwetsbaarheden zoals SQL-injectie, cross-site scripting (XSS) en kapotte verificatie / autorisatie om gevoelige gegevens te beschermen.
    - Integratietesten: Bevestigt dat verschillende onderdelen van een systeem of externe services waarmee de API communiceert, naadloos samenwerken.
    - Contracttesten: zorgt ervoor dat de API zich houdt aan een overeengekomen contract (specificatie zoals OpenAPI / Swagger of WSDL), waardoor wijzigingen tussen service-updates worden voorkomen.
    - End-to-End testen: hiermee worden volledige gebruikersritten gevalideerd waarbij meerdere API-aanroepen aan elkaar zijn gekoppeld.

- Hoe het werkt

  Met visuele, codeloze tools kunt u eenvoudig tests maken, uitbreiden en organiseren voor API's, web-UI's, databases, ESB's en zelfs MCP-servers die gebruikelijk zijn in AI-systemen. Er zijn geen diepgaande technische vaardigheden vereist. SOAtest ondersteunt meer dan 120 protocollen en berichtformaten en biedt u een uniform framework om bedrijfslogica end-to-end te valideren.

  Met behulp van SOAtest, kunt u:

    - Creëer scenario-gebaseerde stromen die echte zakelijke transacties nabootsen en u helpen verborgen bugs te vinden die worden veroorzaakt door specifieke sequenties.
    - Bouw testlogica, complexe beweringen, lussen en gegevensgestuurde stromen met minimale technische expertise.
    - Voer individuele tests of volledige suites uit en bevestig regressiecontroles om onverwachte veranderingen onmiddellijk op te vangen.

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.
```

Wat is software manueel testen?

Het handmatig testen is de procedure om de software te verifiëren met behulp van de verschillende functies en functionaliteiten. Het wordt geleid door een vooropgezette reeks tests die de software valideren en een eindrapport oplevert. Dit type testen kost tijd voor voltooiing, omdat het volledig wordt uitgevoerd door de handmatige inspanningen. Er is dus altijd een menselijke fout bij het uitvoeren van dit type testen.

Elke nieuwe software wordt eerst handmatig getest voordat automatisering wordt toegepast. Het kost meer tijd om een complete software handmatig te verifiëren. Zodra alle functies en functionaliteiten van de software stabiel zijn en goed werken, kunnen sommige van de handmatige testcases worden omgezet in automatisering. De handmatige testcases worden eerst geëvalueerd om te controleren of ze volledig geautomatiseerd kunnen worden. Voor dit type testen hoeft geen automatiseringstool te worden gebruikt om het hele proces te voltooien.

Kenmerken van software manual testing

De kenmerken van het testen van de softwarehandleiding worden hieronder opgesomd −

- De handmatige testen worden volledig uitgevoerd met behulp van menselijke tussenkomst.
- Verkennend testen is een belangrijk onderdeel van handmatig testen. In de verkennende testen controleren de testers de software zonder een vooraf bepaalde set tests. Het detecteert onvoorziene gebreken en verbetert de klanttevredenheid.

- Het handmatig testen is flexibel omdat het wijzigingen van testcases mogelijk maakt op basis van de wijzigingen in de vereisten en andere testomstandigheden.
- Het handmatig testen kan worden overgenomen vanaf de allereerste stadia van de levenscyclus van de softwareontwikkeling (SDLC).
- Sommige van de gecompliceerde testcases kunnen alleen handmatig worden uitgevoerd zonder enige automatisering.
- Het handmatig testen is handig om de gebruikersinterface van de software te valideren. Het helpt bij het controleren van het beeldscherm, de responsiviteit en het normale ontwerp van de software.

Waarom is het testen van de softwarehandleiding nodig?

Het testen van de softwarehandleiding is nodig om de hieronder vermelde redenen -

- De handmatige testen bevestigen dat de software zonder gebreken is, correct werkt volgens de vereisten en stabiel genoeg is om in productie te worden ingezet.
- Het handmatig testen stelt de testers in staat om aan de software te wennen en te begrijpen hoe de software reageert met de klanten. Dit helpt bij het ontwikkelen van effectieve testcases.
- Het handmatig testen identificeert en lost defecten in de software op.

Stappen van het testen van de softwarehandleiding

De verschillende stappen van het testen van de softwarehandleiding worden hieronder opgesomd –

Stap 1− De eerste stap bestaat uit de fase van de analyse van de vereisten door de documenten met vereisten en specificaties, handleidingen, enz. door te nemen.

Stap 2− De tweede stap bestaat uit het opstellen van een testplan dat aan alle vereisten voldoet.

Stap 3− De derde stap bestaat uit het maken van testcases die elke vereiste dekken.

Stap 4− De vierde stap bestaat uit het uitvoeren van testcases in de juiste testomgeving.

Stap 5− De vijfde stap bestaat uit de analyse van de resultaten van de testuitvoering en het rapporteren van de afwijkingen als defecten.

Stap 6− De zesde stap omvat het verhelpen van het defect en het opnieuw testen. Het omvat ook

het opnieuw uitvoeren van de mislukte testgevallen.

Soorten software handmatig testen

De verschillende types software manueel testen worden hieronder opgesomd −

- Black Box Testing− Het is de testtechniek waarbij de tester geen kennis heeft van de interne werking van de software. Het gaat voornamelijk om het controleren of functies en functionaliteiten correct werken volgens de gebruikersvereisten.
- White Box Testing− De testprocedure omvat de verificatie van de interne structuur en de broncode van het programma.
- Grijze doos testen− Het is de testtechniek die zowel de principes van de zwarte doos, en de witte doos testtechnieken gebruikt.

Gereedschappen die worden gebruikt voor het handmatig testen van software

De verschillende tools die worden gebruikt voor het testen van de softwarehandleiding worden hieronder opgesomd −

- testverbinding
- Bugzilla
- Jira
- LoadRunner
- Apache JMeter
- perfecto

Verschillen tussen softwarehandleiding en automatiseringstests

Hier een vergelijking van software handmatig testen en automatisering testen −

manueel testen	Automatiseringstests
Het is de procedure om de software met handmatige inspanningen te verifiëren.	Het is de procedure om de software te verifiëren met behulp van de automatiseringstools.
Het gaat om het handmatig uitvoeren van de testcases.	Het gaat om de uitvoering van de testcases via automatiseringsscripts en tools.
Het is minder productief en vereist meer tijd voor voltooiing.	Het is productiever en vereist minder tijd voor voltooiing.

Het garandeert geen honderd procent testdekking.	Het zorgt voor meer testdekking dan de handmatige tests.
Er zijn geen programmeervaardigheden voor nodig. Dit kan alleen met kennis van de software.	Het vereist programmeervaardigheden.

Voordelen van Software Manual Testing

De voordelen van het handmatig testen van software worden hieronder opgesomd.

- Het handmatig testen helpt om de dynamisch veranderende elementen op het scherm te verifiëren.
- Het handmatig testen is goedkoop en vertrouwt niet op geschoolde middelen.
- De handmatige testen kunnen worden uitgevoerd door testers die geen programmeerkennis hebben.
- Het handmatig testen kan zeer snel worden aangenomen en is geschikt om onvoorspelbare veranderingen in de software aan te kunnen.

Nadelen van Software Manual Testing

De nadelen van het handmatig testen van software worden hieronder opgesomd −

- Het handmatig testen is niet erg betrouwbaar en geeft ruimte voor menselijke fouten.
- Voor verschillende modules moeten afzonderlijke reeksen handmatige testcases worden ontwikkeld, waardoor de herbruikbaarheid zeer beperkt is.
- Het handmatig testen is volledig afhankelijk van het handmatig uitvoeren van tests. Sommige teststappen kunnen echter niet met de handmatige inspanningen worden uitgevoerd.
- De testers die handmatig testen moeten de ervaring hebben met het werken met de software. Bovendien is er geen garantie dat alle functies van de software zijn gedekt tijdens het uitvoeren van de handmatige tests.
- Het handmatig testen is meestal een tijdrovende bezigheid.

Conclusie

Dit besluit onze uitgebreide kijk op de tutorial over Software Manual Testing. We begonnen met het beschrijven van wat software handmatig testen is, wat de kenmerken zijn van de software handmatig testen, waarom is de software handmatig testen nodig, wat zijn de verschillende stappen van de software handleiding testen, wat zijn de verschillende soorten software handmatig testen, wat zijn de verschillende tools die worden gebruikt voor software handmatig testen, wat zijn de verschillen tussen de software handleiding en automatisering testen, wat zijn de voordelen van software handmatig testen, en wat zijn de nadelen van software handmatig testen. Dit geeft u

een grondige kennis van het testen van de softwarehandleiding. Het is verstandig om te blijven oefenen wat je hebt geleerd en anderen te verkennen die relevant zijn voor Software Testing om je begrip te verdiepen en je horizon te verbreden.

Wat is een toegankelijkheidstest?

Toegankelijkheidstests zijn een subset van bruikbaarheidstests waarbij de gebruikers in kwestie mensen met alle capaciteiten en handicaps zijn. Het belang van deze test is om zowel de bruikbaarheid als de toegankelijkheid te verifiëren.

Toegankelijkheid is bedoeld voor mensen met verschillende capaciteiten, zoals:

- Gezichtsstoornissen
- Lichamelijke beperking
- Gehoorstoornis
- Cognitieve stoornis
- Leerstoornis

Een goede webapplicatie moet geschikt zijn voor alle groepen mensen en niet alleen beperkt tot mensen met een handicap. Dit zijn onder meer:

1. Gebruikers met een slechte communicatie-infrastructuur
2. Ouderen en nieuwe gebruikers, die vaak computeranalfabeet zijn
3. Gebruikers die het oude systeem gebruiken (NIET in staat om de nieuwste software uit te voeren)
4. Gebruikers die niet-standaardapparatuur gebruiken
5. Gebruikers die beperkte toegang hebben

Hoe toegankelijkheidstests uit te voeren

Het Web Accessibility Initiative (WAI) beschrijft de strategie voor voorlopige en conformiteitsbeoordelingen van websites. Het Web Accessibility Initiative (WAI) bevat een lijst met softwaretools om te helpen bij conformiteitsevaluaties. Deze hulpmiddelen variëren van specifieke problemen zoals kleurenblindheid tot hulpmiddelen die geautomatiseerde spideringstools uitvoeren.

Tools voor het testen van webtoegankelijkheid

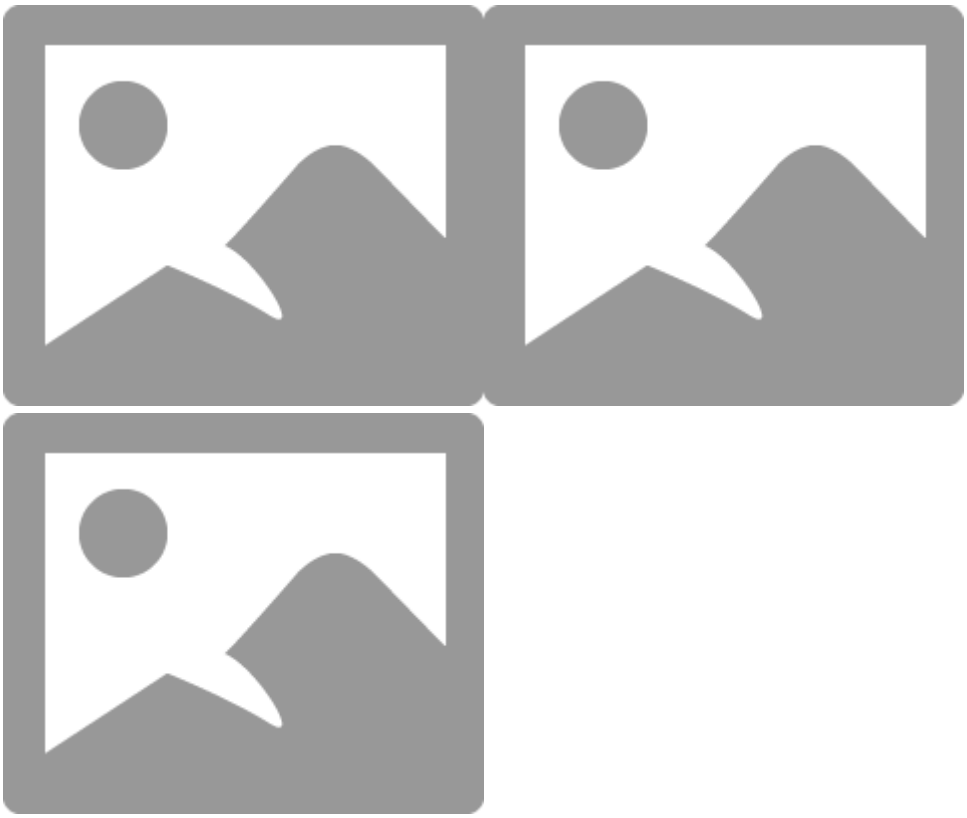
Product	verkoper	URL
AccVerify	HiSoftware	http://www.hisoftware.com
Bobby	waakvuur	http://www.watchfire.com
WebXM	waakvuur	http://www.watchfire.com
hellingshoek	Deque	http://www.deque.com
in focus	SSB Technologies	http://www.ssbtechnologies.com/

Rol van geautomatiseerde tools bij acceptatietests

De bovengenoemde geautomatiseerde hulpmiddelen voor toegankelijkheidstests zijn zeer goed in het identificeren van pagina's en coderegels die handmatig moeten worden gecontroleerd op toegankelijkheid.

1. Controleer de syntaxis van de code van de site
2. Zoeken naar bekende patronen die mensen hebben vermeld
3. Pagina's identificeren met elementen die problemen kunnen veroorzaken
4. Identificeer enkele daadwerkelijke toegankelijkheidsproblemen
5. Identificeer enkele potentiële problemen

De interpretatie van de resultaten van de geautomatiseerde hulpmiddelen voor toegankelijkheidstests vereist ervaring in toegankelijkheidstechnieken met een goed begrip van technische en bruikbaarheidskwesties.



Testen gebeurt op zowel formele als informele manieren om de kwaliteit van de software te verbeteren. Nadat de formele test is voltooid, wordt een ronde van informele en willekeurige tests uitgevoerd. Dit wordt ook wel ad hoc testen genoemd.

Wat is een ad hoc test?

Een ad-hoctest is een informele testtechniek die op de software wordt uitgevoerd om defecten te vinden. Het wordt uitgevoerd in een willekeurig formaat, en is ook bekend als de aap testen. Bij een ad-hoctest wordt geen systematische aanpak gevolgd en zijn er geen goed gedocumenteerde testgevallen.

Ad hoc testen heeft geen documentatie, testscenario's, cases etc. De ontwikkelaars vinden het moeilijk om fouten te herstellen die door ad hoc testen zijn gedetecteerd vanwege het ontbreken van deze testdocumenten. Sommige kritieke, zeldzame en onverwachte bugs worden alleen geïdentificeerd door een willekeurige en informele test op de software uit te voeren. Het is ook een soort acceptatietest en bespaart tijd bij het maken van nieuwe testcases.

Een praktisch voorbeeld van ad-hoctesten is dat een software binnen een dag naar de klant moet worden verzonden en dat de ontwikkeling ervan slechts een dag daarvoor is voltooid, op dit moment is er geen tijd meer om testcases te maken en uit te voeren, dus het testteam voert ad-hoctests uit op volledige software op basis van algemene productkennis en -ervaring.

Typen ad hoc testen

De verschillende soorten ad hoc testen worden hieronder opgesomd −

Buddy Testing

Bij buddy-testen zijn ten minste twee leden betrokken tijdens het testproces - één ontwikkelaar en één tester. Zodra de ontwikkelaar de implementatie van een onderdeel heeft voltooid, voert hij unit-tests uit. Post dat de tester enkele willekeurige gegevens aan dezelfde component voedt en de resultaten onderzoekt. In geval van fouten, de ontwikkelaar lost deze gebreken.

paartest

Bij paaronderzoek zijn twee testers betrokken. Een van hen voert informele en willekeurige verificatie van de software uit en de andere houdt een register bij van de testresultaten. Beiden werken dus in een paar en wisselen ideeën uit, zodat het testen goed wordt uitgevoerd.

Kenmerken van ad hoc testen

De kenmerken van ad-hoctests worden hieronder opgesomd −

- Het is een willekeurige en informele benadering van testen.
- Het wordt niet ondersteund door documentatie, testscenario's, gevallen enz.
- Het wordt uitgevoerd nadat de formele test is voltooid.
- Er wordt geen methodische of gestructureerde aanpak gevolgd.
- Het kost minder tijd om ad-hoctests uit te voeren.
- Het detecteert bugs op de software waar testcases niet beschikbaar zijn.

Wanneer wordt er ad hoc getest?

De ad-hoctests worden uitgevoerd in de onderstaande scenario's &min.;

- Er is beperkte tijd beschikbaar om de software te testen.
- De formele tests zijn afgerond.

- Testcases zijn niet beschikbaar.

Wanneer is een ad hoc test niet gedaan?

De ad-hoctests worden niet uitgevoerd in de onderstaande scenario's

- Het wordt niet gedaan als bugs worden gedetecteerd door het uitvoeren van de testgevallen.
- Op het moment van de beta-test is dit nog niet gebeurd.

Voordelen van ad hoc testen

De voordelen van ad hoc testen worden hieronder opgesomd.

- Het houdt zich niet aan een proces, dus ad-hoctesten kunnen op elk moment in de levenscyclus van de softwareontwikkeling worden uitgevoerd.
- Het testteam kan de software verifiëren en fouten opsporen door nieuwe testtechnieken toe te passen zonder alleen op de testcases te vertrouwen.
- Een ontwikkelaar kan ad hoc testen uitvoeren op dezelfde module die hij ontwikkelt en zijn codekwaliteit verhogen.
- Hoewel het formele testproces veel tijd kost, kunnen ad-hoctests in korte tijd worden uitgevoerd.
- Hiervoor is geen documentatie nodig.

Nadelen van ad hoc testen

De nadelen van ad hoc testen worden hieronder opgesomd.

- Ad-hoctesten moeten worden uitgevoerd door teamleden die ervaring hebben met testen en gedegen kennis hebben van het product. Een onervaren lid van het team kan geen ad-hoctests uitvoeren.
- In het geval van een bug is het moeilijk om hetzelfde te reproduceren, omdat de ad hoc testen niet wordt aangedreven door enige planning.

Best practices voor ad-hoctests

De beste praktijken die bij ad-hoctests moeten worden gevolgd, worden hieronder opgesomd −

- Verzamel alle kennis over het product.
- Identificeer de defectgevoelige componenten van de software en geef ze prioriteit.
- Gebruik van geschikte testmiddelen.

Conclusie

Dit besluit onze uitgebreide kijk op de tutorial over Software Ad Hoc Testing. We begonnen met het beschrijven van wat ad-hoctesten is, wat de typen, functies, technieken, voordelen, nadelen, tijd en best practices van ad-hoctesten zijn.

Dit geeft u een grondige kennis van Software Ad Hoc Testing. Het is verstandig om te blijven oefenen wat je hebt geleerd en anderen te verkennen die relevant zijn voor Software Testing om je begrip te verdiepen en je horizon te verbreden.

Over deze vertaling

Cisco heeft dit document vertaald via een combinatie van machine- en menselijke technologie om onze gebruikers wereldwijd ondersteuningscontent te bieden in hun eigen taal. Houd er rekening mee dat zelfs de beste machinevertaling niet net zo nauwkeurig is als die van een professionele vertaler. Cisco Systems, Inc. is niet aansprakelijk voor de nauwkeurigheid van deze vertalingen en raadt aan altijd het oorspronkelijke Engelstalige document (link) te raadplegen.