Testpost 26,5 admin

Inhoud

Inleiding

Q.

A.

API-testen is een type softwaretests dat een API valideert om ervoor te zorgen dat deze voldoet aan de verwachtingen voor functionaliteit, betrouwbaarheid, prestaties en beveiliging. Het richt zich vooral op de business logic laag en data-uitwisseling tussen softwaresystemen, onafhankelijk van een user interface (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 complexe problemen te helpen navigeren, 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=

Vraag hulp bij een probleem dat je hebt. Er wordt een incidentrecord gemaakt en beheerd tot een succesvolle oplossing. Ook wordt u op de hoogte gebracht van de voortgang.

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

Bij 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 leveren. Hier geeft de tester de invoer voor de toepassing en test deze handmatig.

Als het de verwachte uitvoer retourneert, zal de tester doorgaan met een andere set inputs en alle resultaten aan het team rapporteren. Als de invoer die de gebruiker handmatig heeft gegeven, tijdens het testen mislukt, zal hij / zij dit probleem aan het ontwikkelingsteam melden.



TESTVIDEO

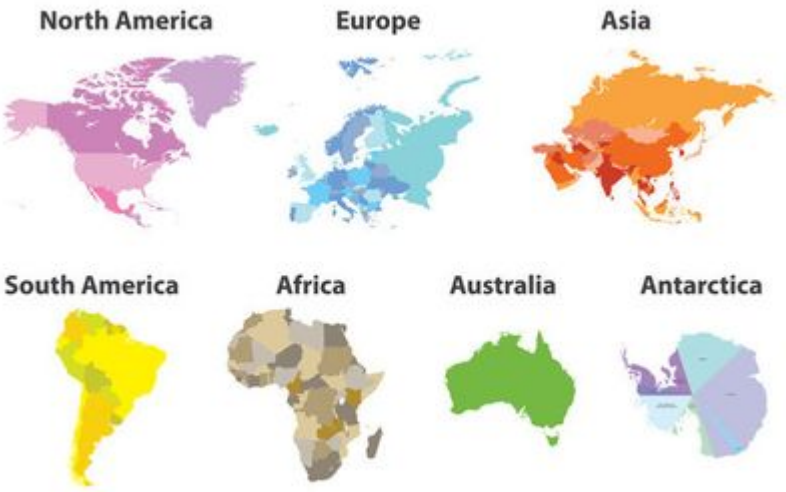
controleren	tafel
	LINK controleren

TESTTAFEL

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

Witte doos testen

In de White Box-testtechniek controleert de persoon de interne structuur van het systeem, zoals ontwerpen, codering, enz., Handmatig. Hier zal het ontwikkelingsteam de gehele codeerdeelregel voor regel beoordelen om de juistheid van de code te garanderen.



CISCO



HERE IS A SAMPLE



Als hij / zij eventuele 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 is het proces 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, aangezien er twee rechten met dezelfde naam zijn.

Belangrijkste aspecten van API-testen ☐

- Communicatie op de berichtenlaag: In plaats van een GUI te gebruiken, interacteren 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.
- Early Defect Detection: API-tests kunnen vroeg in de ontwikkelcyclus van de software worden uitgevoerd, zelfs voordat de gebruikersinterface is gebouwd, zodat teams problemen efficiënter en tegen lagere kosten kunnen vinden en oplossen.
- Automatiseringsfocus: 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, foutafhandeling en beveiligingslekken 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 dekken: ☐

Katalon

- Functioneel testen: controleert of de API de beoogde bewerkingen correct uitvoert, waarbij de opgegeven ingangen, uitgangen en statuscodes worden verwerkt.
- Prestatietest: Beoordeelt de snelheid, stabiliteit en schaalbaarheid van de API onder verschillende belastingsomstandigheden (bijv. piekverkeer, stress).
- Beveiligingstests: hiermee worden kwetsbaarheden geïdentificeerd zoals SQL-injectie, cross-site scripting (XSS) en gebroken verificatie/autorisatie om gevoelige gegevens te beschermen.
- Integratietests: Bevestigt dat verschillende delen van een systeem of externe services waarmee de API interageert, naadloos samenwerken.
- Contracttesten: zorgt ervoor dat de API zich houdt aan een overeengekomen contract (specificatie zoals OpenAPI / Swagger of WSDL), waardoor het doorbreken van wijzigingen tussen serviceupdates wordt voorkomen.
- End-to-end testen: hiermee worden volledige gebruikerstrajecten gevalideerd waarbij meerdere API-oproepen aan elkaar zijn gekoppeld.
- Soorten API-testen
  Verschillende soorten API-tests worden gebruikt om verschillende aspecten van de kwaliteit van een toepassing te dekken:

  Katalon
  - Functioneel testen: controleert of de API de beoogde bewerkingen correct uitvoert, waarbij de opgegeven ingangen, uitgangen en statuscodes worden verwerkt.
  - Prestatietest: Beoordeelt de snelheid, stabiliteit en schaalbaarheid van de API onder verschillende belastingsomstandigheden (bijv. piekverkeer, stress).
  - Beveiligingstests: hiermee worden kwetsbaarheden geïdentificeerd zoals SQL-injectie, cross-site scripting (XSS) en gebroken verificatie/autorisatie om gevoelige gegevens te beschermen.
  - Integratietests: Bevestigt dat verschillende delen van een systeem of externe services waarmee de API interageert, naadloos samenwerken.
  - Contracttesten: zorgt ervoor dat de API zich houdt aan een overeengekomen contract (specificatie zoals OpenAPI / Swagger of WSDL), waardoor het doorbreken van wijzigingen tussen serviceupdates wordt voorkomen.
  - End-to-end testen: hiermee worden volledige gebruikerstrajecten gevalideerd waarbij meerdere API-oproepen 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-gebaseerde systemen. Er zijn geen diepgaande technische vaardigheden nodig. SOAtest ondersteunt meer dan 120 protocollen en berichtformaten en biedt u een uniform framework om bedrijfslogica end-to-end te valideren.

  Met SOAtest kunt u:

  - Maak op scenario's gebaseerde stromen die echte zakelijke transacties nabootsen, zodat u verborgen bugs kunt vinden die worden veroorzaakt door specifieke

sequenties.
- Bouw testlogica, complexe beweringen, loops en gegevensgestuurde stromen met minimale technische expertise.
- Voer individuele tests of volledige suites uit en bevestig regressiecontroles om onverwachte veranderingen onmiddellijk op te vangen.

```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Statements</h1>

<p>Multiple statements on one line are allowed.</p>

<p id="demo1"></p>

<script>
let a, b, c;
a = 5; b = 6; c = a + b;

document.getElementById("demo1").innerHTML = c;
</script>

</body>
</html>
```


```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Statements</h1>

<p>Multiple statements on one line are allowed.</p>

<p id="demo1"></p>

<script>
let a, b, c;
a = 5; b = 6; c = a + b;

document.getElementById("demo1").innerHTML = c;
</script>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Statements</h1>

<p>Multiple statements on one line are allowed.</p>

<p id="demo1"></p>

<script>
let a, b, c;
a = 5; b = 6; c = a + b;

document.getElementById("demo1").innerHTML = c;
```

```
</script>

</body>
</html>

<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Statements</h1>

<p>Multiple statements on one line are allowed.</p>

<p id="demo1"></p>

<script>
let a, b, c;
a = 5; b = 6; c = a + b;

document.getElementById("demo1").innerHTML = c;
</script>

</body>
</html>

<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Statements</h1>

<p>Multiple statements on one line are allowed.</p>

<p id="demo1"></p>

<script>
let a, b, c;
a = 5; b = 6; c = a + b;

document.getElementById("demo1").innerHTML = c;
</script>

</body>
</html>
```

Wat is een software manual testing?

De handmatige testen is de procedure om de software te verifiëren met behulp van de verschillende functies en functionaliteiten. Het wordt geleid door een vooraf bedachte set tests die de software valideren en biedt een eindrapport. Dit type testen kost tijd voor voltooiing, omdat het volledig wordt uitgevoerd door de handmatige inspanningen. Daarom is er altijd een omvang van menselijke fouten bij het uitvoeren van dit type testen.

Elke nieuwe software wordt eerst handmatig getest voordat automatisering wordt toegepast. Het kost meer tijd om een volledige software handmatig te verifiëren. Zodra alle functies en functionaliteiten van de software stabiel zijn en goed werken, kunnen sommige handmatige

testcases worden omgezet in automatisering. De handmatige testcases worden eerst geëvalueerd om te controleren of ze volledig geautomatiseerd kunnen worden. Dit type testen vereist geen gebruik van automatiseringstools om het hele proces te voltooien.

reclame

Kenmerken van het handmatig testen van software

De kenmerken van het handmatig testen van software worden hieronder vermeld -

- De handmatige testen worden volledig uitgevoerd met behulp van menselijke interventie.
- Exploratieve tests zijn een belangrijk onderdeel van handmatige tests. In de verkennende tests verifiëren de testers de software zonder vooraf bepaalde set tests. Het detecteert onvoorspelbare defecten 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.
- De handmatige tests kunnen worden uitgevoerd vanaf de allereerste fasen van de levenscyclus van de softwareontwikkeling (SDLC).
- Sommige van de ingewikkelde 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 de weergave, het reactievermogen en het normale ontwerp van de software.

Waarom is de Software Manual Testing nodig?

Het handmatig testen van de software is nodig om de hieronder vermelde redenen -

- De handmatige tests bevestigen dat de software geen defecten vertoont, correct werkt volgens de vereisten en stabiel genoeg is om in productie te worden geïmplementeerd.
- Met de handmatige tests kunnen de testers wennen aan de software en 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 Software Manual Testing

De verschillende stappen van het handmatig testen van de software worden hieronder vermeld -

Stap 1 − De eerste stap omvat de vereiste-analysefase door de vereisten en specificaties van documenten, gidsen, enz. te doorlopen.

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

Stap 3 − De derde stap omvat het creëren van testcases die alle vereisten dekken.

Stap 4 − De vierde stap omvat de uitvoering van testcases in de juiste testomgeving.

Stap 5 − De vijfde stap omvat de analyse van de resultaten van de testuitvoering en rapporteert de discrepanties als defecten.

Stap 6 − De zesde stap omvat de defectoplossing en het opnieuw testen. Het omvat ook het opnieuw uitvoeren van de mislukte testgevallen.

Soorten software handmatig testen

De verschillende soorten software handmatige testen worden hieronder vermeld -

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

Gereedschappen die worden gebruikt voor het handmatig testen van software

De verschillende tools die worden gebruikt voor het handmatig testen van de software worden hieronder vermeld -

- testverbinding
- Bugzilla
- Jira

- LoadRunner
- Apache JMeter
- volmaakt

Verschillen tussen software handmatig en automatisering testen

Hier is een vergelijking van software handmatig testen en automatisering testen −

Handmatig testen	Automatiseringstests
Het is de procedure om de software te verifiëren met handmatige inspanningen.	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.
Het vereist geen programmeervaardigheden. Dit kan alleen met kennis van de software.	Het vereist programmeervaardigheden.

Voordelen van Software Manual Testing

De voordelen van handmatig testen van software worden hieronder vermeld -

- De handmatige tests helpen om de dynamisch veranderende elementen op het scherm te verifiëren.
- Het handmatig testen is goedkoop en is niet afhankelijk van bekwame middelen.
- De handmatige tests kunnen worden uitgevoerd door testers zonder programmeerkennis.
- De handmatige tests kunnen zeer snel worden uitgevoerd en zijn geschikt voor onvoorspelbare veranderingen in de software.

Nadelen van software handmatig testen

De nadelen van handmatig testen van software worden hieronder vermeld -

- Het handmatig testen is niet erg betrouwbaar en geeft ruimte voor menselijke fouten.
- Voor verschillende modules moeten afzonderlijke reeksen handmatige testcases worden ontwikkeld, waardoor er minder ruimte is voor herbruikbaarheid.
- 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 handmatige tests uitvoeren, moeten de ervaring hebben met het werken met de software. Bovendien is er geen zekerheid 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 concludeert onze uitgebreide kijk op de tutorial over het testen van software-handleidingen. We begonnen met het beschrijven van wat handmatig testen van software is, wat de kenmerken zijn van het handmatig testen van de software, waarom is het handmatig testen van de software nodig, wat zijn de verschillende stappen van het handmatig testen van de software, wat zijn de verschillende soorten handmatig testen van de software, wat zijn de verschillende tools die worden gebruikt voor handmatig testen van de software, wat zijn de verschillen tussen het handmatig testen van de software en automatiseringstests, wat zijn de voordelen van handmatig testen van de software en wat zijn de nadelen van handmatig testen van de software. Dit voorziet u van diepgaande kennis van Software Manual 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.

Wat is toegankelijkheidstests?

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

Toegankelijkheid is gericht op mensen met verschillende vaardigheden, zoals:

- Gezichtsstoornissen
- lichamelijke beperking
- slechthorendheid
- cognitieve stoornis
- Leerstoornis

Een goede webapplicatie moet geschikt zijn voor alle groepen mensen en NIET alleen beperkt zijn tot gehandicapten. Dit zijn onder meer:

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

reclame

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 tools variëren van specifieke problemen zoals kleurenblindheid tot tools die geautomatiseerde spideringtools uitvoeren.

Webtoegankelijkheidstesttools

Product	verkoper	URL
AccVerify	HiSoftware	http://www.hisoftware.com
Bobby	horlogebrand	http://www.watchfire.com
WebXM	horlogebrand	http://www.watchfire.com
hellingbaan	Deque	http://www.deque.com
in focus	SSB-technologieën	http://www.ssbtechnologies.com/

De 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. Zoek naar bekende patronen die mensen hebben vermeld

3. Pagina's identificeren die elementen bevatten die problemen kunnen veroorzaken
4. Identificeer een aantal werkelijke toegankelijkheidsproblemen
5. Identificeer een aantal potentiële problemen

De interpretatie van de resultaten van de geautomatiseerde toegankelijkheidstesttools vereist ervaring in toegankelijkheidstechnieken met een goed begrip van technische en bruikbaarheidsproblemen.

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 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 staat ook bekend als het testen van apen. Een ad-hoctest volgt geen systematische aanpak en heeft geen goed gedocumenteerde testgevallen.

Ad-hoctests bevatten geen documentatie, testscenario's, cases, enz. De ontwikkelaars vinden het moeilijk om defecten gedetecteerd door ad-hoctests te repareren vanwege de afwezigheid van deze testdocumenten. Ook worden sommige kritieke, zeldzame en onverwachte bugs alleen geïdentificeerd door een willekeurige en informele test op de software uit te voeren. Het is ook een soort acceptatietest en bespaart de tijd om nieuwe testcases te maken.

Een praktisch voorbeeld van ad-hoctesten is de veronderstelling dat een software in één 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.

reclame

Soorten ad hoc testen

De verschillende soorten ad-hoctests worden hieronder vermeld -

Buddy-test

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

paartesten

Bij het testen van paren zijn twee testers betrokken. Een van hen voert informele en willekeurige verificatie van de software uit en de andere houdt een record bij van de testresultaten. Zo werken ze allebei in een paar en wisselen ze ideeën uit, kennis zodat het testen goed wordt gedaan.

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, cases enz.
- Het wordt uitgevoerd nadat de formele test is voltooid.
- Het volgt geen methodische of gestructureerde aanpak.
- Het kost minder tijd om ad hoc testen uit te voeren.
- Het detecteert bugs op de software waar testcases niet beschikbaar zijn.

Wanneer is een ad hoc test gedaan?

De ad-hoctests worden uitgevoerd in de scenario's die hieronder worden vermeld &min;

- Er is beperkte tijd beschikbaar voor het testen van de software.
- De formele test is 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 de testcases uit te voeren.
- Op het moment van beta-testen 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 gedaan.
- Het testteam kan de software verifiëren en fouten vinden door nieuwe testtechnieken toe te passen zonder alleen op de testcases te vertrouwen.
- Een ontwikkelaar kan ad-hoctesten 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.
- Er is geen documentatie voor 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 van het product hebben. 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-hoctests niet worden aangestuurd door een planning.

Best practices voor ad-hoctesten

De beste praktijken die bij ad-hoctests moeten worden gevolgd, worden hieronder vermeld.

- Verzamel alle kennis over het product.
- Identificeer de defectgevoelige componenten van de software en geef er prioriteit aan.
- Gebruik van geschikte testtools.

Conclusie

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

Dit voorziet u van diepgaande 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.