Marcador de teste 3

Contents

O teste de API é um tipo de teste de software que valida uma interface de programação de aplicativo (API) para garantir que ela atenda às expectativas de funcionalidade, confiabilidade, desempenho e segurança. Ele se concentra principalmente na camada lógica comercial e no intercâmbio de dados entre sistemas de software, independentemente de uma interface de usuário (UI)

Isto serve para testar URLs entre textos

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

O COBC (Code of Business Conduct, código de conduta comercial) da Cisco reflete como trabalhamos e tomamos decisões com integridade. Ele também fornece recursos para ajudar a navegar em questões complexas, como o uso responsável da IA e conflitos de interesse.

```
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=

Solicite ajuda para resolver um problema que você está enfrentando. Um registro de incidente será criado e gerenciado por meio do para uma resolução bem-sucedida. Você também será notificado sobre o andamento.

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

Na técnica Black Box Testing, o testador ou o analista de controle de qualidade só verificará a funcionalidade do módulo específico ou método específico ou, às vezes, todo o aplicativo fornecendo os diferentes casos de teste manualmente. Aqui, o testador fornecerá a entrada para o aplicativo e o testará manualmente.

Se retornar a saída esperada, o testador prosseguirá com outro conjunto de entradas e relatará todos os resultados à equipe. Se a entrada fornecida manualmente pelo usuário falhar durante o teste, ele/ela relatará esse problema à equipe de desenvolvimento.

TESTAR VÍDEO

Verificar	Tabela
	Verificar LINK

TABELA DE TESTE

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

Teste de white box

Na técnica Teste de caixa branca, a pessoa verificará manualmente a estrutura interna do sistema, como designs, codificação, etc. Aqui, a equipe de desenvolvimento revisará toda a parte de codificação linha por linha para garantir a correção do código.

Se ele/ela encontrar quaisquer dissimilaridades ou erros no código, eles irão corrigir ou corrigir os erros na codificação ou projetos. Aqui, o processo é inteiramente realizado manualmente e o processo é eficiente, uma vez que o código de verificação ou design é verificado manualmente por seres humanos.

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

https://www.google.com/

A verificação da "função de desenvolvedor bdb" migrou da API ART para a ID Entra no One Access. Ao solicitar acesso, certifique-se de selecionar "Método de integração: memberOf", pois há dois direitos com o mesmo nome.

Aspectos principais do teste de API ☐

- Comunicação na camada de mensagens: Em vez de usar uma GUI, os testes de API interagem diretamente com os endpoints do aplicativo (URIs) usando vários métodos HTTP (GET, POST, PUT, DELETE) e formatos de dados como JSON ou XML.
- Detecção antecipada de defeitos: O teste de API pode ser realizado no início do ciclo de vida do desenvolvimento de software, mesmo antes da criação da interface do usuário, permitindo que as equipes localizem e corrijam problemas com mais eficiência e a um custo menor.
- Foco na automação: Devido à natureza da interação direta e da estrutura consistente, os

testes de API são adequados para automação, o que é essencial em ambientes ágeis e DevOps modernos para testes contínuos em pipelines de CI/CD.
- Cobertura abrangente: Ele oferece uma cobertura de teste mais ampla em comparação ao teste de IU sozinho, incluindo casos de borda de teste, manipulação de erros e vulnerabilidades de segurança que podem ser difíceis de acessar através da IU.

Tipos de testes de API
Diferentes tipos de testes de API são usados para cobrir vários aspectos da qualidade de um aplicativo:

Katalon

- Teste funcional: Verifica se a API executa suas operações pretendidas corretamente, manipulando entradas, saídas e códigos de status conforme especificado.
- Teste de desempenho: Avalia a velocidade, a estabilidade e a escalabilidade da API sob várias condições de carga (por exemplo, tráfego de pico, estresse).
- Teste de segurança: Identifica vulnerabilidades, como injeção de SQL, scripts entre sites (XSS) e autenticação/autorização interrompidas para proteger dados confidenciais.
- Teste de integração: Confirma que diferentes partes de um sistema ou serviços externos com os quais a API interage funcionam perfeitamente.
- Teste de contrato: Garante que a API adere a um contrato acordado (especificação como OpenAPI/Swaggerou WSDL), evitando alterações entre atualizações de serviço.
- Teste completo: Valida jornadas inteiras do usuário que envolvem várias chamadas de API encadeadas.
- Tipos de testes de API
  Diferentes tipos de testes de API são usados para cobrir vários aspectos da qualidade de um aplicativo:

  O teste manual começa com a compreensão do que se espera que o software faça.

  - Requisitos funcionais: Verificar recursos, como login de usuário correto.
  - Requisitos não funcionais: valide o desempenho, a usabilidade e a segurança (por exemplo, o tempo de carregamento da página abaixo de 2 segundos).
  - Histórias de usuário e documentos de design: Entender interações e fluxos de trabalho do usuário.
  - Informações das partes interessadas: Esclareça os requisitos com clientes, gerentes de produtos ou designers.

Passo 2: Criando um Plano de Teste

Um plano de teste define a estratégia e os objetivos de teste.

- Escopo:identifica recursos a serem testados e exclusões.
- Objetivos: Garante a funcionalidade principal e a experiência do usuário.
- Recursos: Especifica membros da equipe, ferramentas e linhas do tempo.
- Técnicas de teste: Inclui testes funcionais, de usabilidade e exploratórios.
- Ambientes: define configurações de preparação ou produção.

Passo 3: Casos de teste de design

Os casos de teste são scripts claros, passo a passo, que garantem testes manuais completos. Os casos de teste atuam como guias detalhados para os testadores, garantindo que todos os cenários sejam verificados. Cada caso de teste inclui:

- ID do teste: Um código exclusivo, como TC_001, para facilitar o rastreamento.
- Descrição: O objetivo, como verificar com entradas válidas.
- Pré-condições: O que é necessário antes de começar, como estar na página de pesquisa.
- Etapas: Ações a serem executadas, como escolher a data de amanhã, e clique em "Pesquisar".
- Resultado esperado: O resultado desejado, como uma lista de voos ordenados por preço.
- Pós-condições: O sistema mostra a página de resultados.

Leia mais: Como escrever casos de teste?

Passo 4: Configurar o ambiente de teste

O ambiente de teste deve se parecer muito com a produção.

- Instale os aplicativos necessários.
- Definir configurações específicas do projeto.
- Garanta a disponibilidade dos dados de teste.
- Verificar os requisitos de hardware e software.

Passo 5: Executar casos de teste

Execute os casos de teste passo a passo e interaja com o aplicativo como um usuário.

- Resultados Reais: O que acontece durante a execução.
- Status Aprovado/Reprovado: Se o resultado real corresponde ao resultado esperado.
- Observações: Qualquer comportamento inesperado ou problemas de usabilidade.

Passo 6: Registrar e relatar defeitos

Quando um teste falhar ou ocorrer um comportamento inesperado, registre defeitos com:

- ID do Defeito: identificador exclusivo.
- Resumo: breve descrição do que é o defeito real
- Etapas para reprodução: Etapas detalhadas para disparar o problema.
- Resultados Reais vs. Esperados:O que aconteceu vs. o que deveria ter acontecido.
- Severidade: verifique se o impacto é crítico, grave ou secundário.
- Anexos:Capturas de tela, logs ou vídeos para comprovar o defeito.

Passo 7: Rastrear e verificar defeitos

Depois que as correções forem aplicadas:

- Rastreie o status do defeito na ferramenta.
- Teste novamente os problemas corrigidos.
- Fechar ou reabrir defeitos com base nos resultados.

Passo 8: Realizar Teste de Regressão

O teste de regressão garante que as correções de defeitos ou novas alterações não interrompam a funcionalidade existente.

- As áreas afetadas são verificadas após a resolução de bugs.
- Verificar recursos críticos.
- Verifique os pontos de integração para garantir que estejam funcionando como antes.

Etapa 9: Preparar Relatórios de Fechamento de Teste

Quando o teste estiver concluído, calcule os resultados em relação aos objetivos do plano de teste e crie um relatório de fechamento de teste para o mesmo:

- Resumo: Visão geral das atividades de teste.
- Resultados do teste: Número de casos de teste executados, aprovados e com falha.
- Defeitos encontrados: Defeitos totais, sua gravidade e status da resolução.
- Problemas pendentes: Quaisquer defeitos ou riscos não resolvidos.
- Lições aprendidas: Informações para testes futuros.

Etapa 10: Fornecer comentários e recomendações

Analise os resultados dos testes para fornecer feedback acionável às partes interessadas, como:

- Qualidade do software.
- Aprimoramentos no processo.
- Estratégias de teste futuras.
- Informações sobre a experiência do usuário.

Ferramentas usadas para teste manual

- TestRail: uma ferramenta de gerenciamento de teste fácil de usar para organizar, executar e relatar casos de teste manuais com integrações fortes e painéis
- Xray (para Jira): uma ferramenta de gerenciamento de testes baseada em Jira que suporta testes manuais, automatizados e BDD com total rastreabilidade e integração transparente
- Qase: uma plataforma moderna de gerenciamento de testes baseada em nuvem com uma interface de usuário simples, criação de casos de teste com IA e rastreamento de defeito integrado
- Zephyr: uma solução escalável de gerenciamento de testes que suporta testes

manuais e exploratórios com recursos fortes de integração e geração de relatórios Jira
- Tuskr: Uma ferramenta de gerenciamento de testes em nuvem leve e acessível com uma interface intuitiva e recursos de colaboração.

Necessidade de testes manuais

- Estabilidade e ausência de bugs:o principal objetivo do teste manual é garantir que o aplicativo esteja livre de bugs, estável, em conformidade com os requisitos e forneça um produto estável aos clientes.
- Familiaridade com o produto: o teste manual ajuda os engenheiros de teste a se familiarizarem mais com o produto e obterem uma perspectiva do usuário final. Isso os ajuda a escrever casos de teste corretos para o software.
- Correção dos defeitos: o teste manual ajuda a garantir que os defeitos sejam corrigidos pelo revelador e que o novo teste tenha sido feito com os defeitos corrigidos.

Vantagens do teste manual

- Feedback visual rápido e preciso:Ele detecta quase todos os bugs na aplicação de software e é usado para testar os projetos de GUI que mudam dinamicamentecomo layout, texto, etc.
- Mais barato:É mais barato, pois não exige nenhuma habilidade de alto nível ou um tipo específico de ferramenta.
- Nenhuma codificação é necessária:Nenhum conhecimento de programação é necessário ao usar o método de teste de caixa preta. É fácil aprender com os novos testadores.
- Eficiente para alterações não planejadas:o teste manual é adequado em caso de alterações não planejadas no aplicativo, pois pode ser adotado facilmente.

Katalon
- Teste funcional: Verifica se a API executa suas operações pretendidas corretamente, manipulando entradas, saídas e códigos de status conforme especificado.

    - Teste de desempenho: Avalia a velocidade, a estabilidade e a escalabilidade da API sob várias condições de carga (por exemplo, tráfego de pico, estresse).
    - Teste de segurança: Identifica vulnerabilidades, como injeção de SQL, scripts entre sites (XSS) e autenticação/autorização interrompidas para proteger dados confidenciais.
    - Teste de integração: Confirma que diferentes partes de um sistema ou serviços externos com os quais a API interage funcionam perfeitamente.
    - Teste de contrato: Garante que a API adere a um contrato acordado (especificação como OpenAPI/Swaggerou WSDL), evitando alterações entre atualizações de serviço.
    - Teste completo: Valida jornadas inteiras do usuário que envolvem várias chamadas de API encadeadas.

- Como funciona

  Ferramentas visuais e sem código permitem criar, estender e organizar facilmente testes em APIs, IUs da Web, bancos de dados, ESBs e até mesmo servidores MCP comuns em sistemas com IA. Não são necessárias habilidades técnicas avançadas. Suportando mais de 120 protocolos e formatos de mensagem, o SOAtest oferece uma estrutura unificada para validar a lógica de negócios de ponta a ponta.

  Usando o SOAtest, você pode:

    - Crie fluxos baseados em cenários que simulem transações comerciais reais, ajudando a encontrar bugs ocultos disparados por sequências específicas.
    - Crie lógica de teste, asserções complexas, loops e fluxos orientados por dados com o mínimo de experiência técnica.
    - Execute testes individuais ou conjuntos completos e anexe controles de regressão para detectar alterações inesperadas imediatamente.

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.
```

O que é Teste manual de software?

O teste manual é o procedimento para verificar o software com a ajuda de seus vários recursos e funcionalidades. Ele é guiado por um conjunto pré-concebido de testes que validam o software e fornece um relatório final do resultado. Esse tipo de teste leva tempo para ser concluído, já que é realizado completamente através de esforços manuais. Portanto, há sempre uma extensão de erro humano ao realizar esse tipo de teste.

Cada novo software é testado manualmente antes de adotar a automação. Ele consome mais tempo para verificar manualmente um software completo. Quando todos os recursos e funcionalidades do software estiverem estáveis e funcionando bem, alguns dos casos de teste manuais podem ser convertidos em automação. Os casos de teste manual são avaliados primeiro para verificar se podem ser totalmente automatizados. Esse tipo de teste não exige o uso de ferramentas de automação para concluir todo o processo.

Características do teste manual de software

As características do teste manual de software estão listadas abaixo −

- O teste manual é realizado completamente com a ajuda da intervenção humana.
- O teste exploratório é uma parte importante do teste manual. Nos testes exploratórios, os testadores verificam o software sem nenhum conjunto predeterminado de testes. Ele detecta defeitos imprevistos e melhora a satisfação do cliente.

- O teste manual é flexível, pois permite a modificação de casos de teste com base nas alterações nos requisitos e em outras condições de teste.
- Os testes manuais podem ser adotados desde os estágios iniciais do SDLC (Software Development Life Cycle, ciclo de vida de desenvolvimento de software).
- Alguns dos casos de teste complicados podem ser executados apenas manualmente, sem qualquer automação.
- O teste manual é útil para validar a interface de usuário do software. Ele ajuda a verificar a exibição, a capacidade de resposta e o design normal do software.

Por que o teste manual de software é necessário?

O teste manual do software é necessário pelos motivos listados abaixo −

- Os testes manuais confirmam que o software não apresenta defeitos, funciona corretamente conforme os requisitos e é estável o suficiente para ser implantado na produção.
- O teste manual permite que os testadores se acostumem com o software e entendam como o software responde com os clientes. Isso ajuda a desenvolver casos de teste eficazes.
- O teste manual identifica e resolve defeitos no software.

Etapas do teste manual do software

As diferentes etapas do teste manual do software estão listadas abaixo −

Etapa 1− A primeira etapa envolve a fase de análise de requisitos, analisando os documentos de requisitos e especificações, guias, etc.

Etapa 2− A segunda etapa envolve a criação de um plano de teste que englobe todos os requisitos.

Etapa 3− A terceira etapa envolve a criação de casos de teste que cobrem todos os requisitos.

Etapa 4− A quarta etapa envolve a execução de casos de teste no ambiente de teste correto.

Etapa 5− A quinta etapa envolve a análise dos resultados da execução do teste e relata as discrepâncias como defeitos.

Etapa 6− A sexta etapa envolve a correção do defeito e o novo teste. Também inclui a reexecução dos casos de teste que falharam.

Tipos de teste manual de software

Os diferentes tipos de testes manuais de software estão listados abaixo −

- Teste de caixa preta− É a técnica de teste em que o testador não tem nenhum conhecimento do funcionamento interno do software. Trata-se principalmente de verificar se os recursos e as funcionalidades funcionam corretamente de acordo com os requisitos do usuário.
- Teste de caixa branca− É o procedimento de teste que inclui a verificação da estrutura interna e o código-fonte do programa do software.
- Teste de caixa cinza− É a técnica de teste que usa os princípios da caixa preta e técnicas de teste de caixa branca.

Ferramentas usadas para o teste manual do software

As diferentes ferramentas usadas para o teste manual do software estão listadas abaixo −

- Link de teste
- BugzillaName
- Jira
- LoadRunner
- Apache JMeter
- Perfeito

Diferenças entre o manual de software e os testes de automação

Apresentamos aqui uma comparação entre os testes manuais de software e os − de testes de automação

Teste manual	Teste de automação
É o procedimento para verificar o software com esforços manuais.	É o procedimento para verificar o software com a ajuda das ferramentas de automação.
Ela envolve a execução dos casos de teste manualmente.	Ela envolve a execução dos casos de teste por meio de scripts e ferramentas de automação.
Ele é menos produtivo e requer mais tempo para ser concluído.	Ele é mais produtivo e requer menos tempo para ser concluído.
Ele não garante 100% de cobertura de teste.	Ele garante mais cobertura de teste do que o teste manual.
Não exige habilidades de programação. Ele pode	Requer habilidades de programação.

ser executado apenas com o conhecimento do software.	

Vantagens do teste manual de software

As vantagens do teste manual de software estão listadas abaixo −

- O teste manual ajuda a verificar a alteração dinâmica de elementos na tela.
- O teste manual é barato e não depende de recursos qualificados.
- Os testes manuais podem ser realizados por testadores sem conhecimento de programação.
- O teste manual pode ser adotado muito rapidamente e é apropriado para acomodar alterações imprevisíveis no software.

Desvantagens do teste manual de software

As desvantagens do teste manual de software estão listadas abaixo −

- O teste manual não é muito confiável e dá margem para erros humanos.
- Conjuntos separados de casos de teste manuais precisam ser desenvolvidos para módulos diferentes, permitindo um escopo muito menor de reutilização.
- O teste manual depende totalmente da execução dos testes manualmente. No entanto, algumas das etapas de teste não podem ser executadas com os esforços manuais.
- Os testadores que executam testes manuais devem ter experiência em trabalhar com o software. Além disso, não há garantia de que todos os recursos do software foram cobertos durante a execução dos testes manuais.
- Os testes manuais são, na maioria das vezes, uma atividade demorada.

Conclusão

Isso conclui nossa visão abrangente do tutorial sobre o Teste manual de software. Começamos descrevendo o que é o teste manual de software, quais são as características do teste manual de software, por que o teste manual de software é necessário, quais são as diferentes etapas do teste manual de software, quais são os diferentes tipos de teste manual de software, quais são as diferentes ferramentas usadas para o teste manual de software, quais são as diferenças entre o teste manual de software e o teste de automação, quais são as vantagens do teste manual de software e quais são as desvantagens do teste manual de software. Isso o equipe com conhecimento profundo do Teste manual de software. É sábio continuar praticando o que você aprendeu e explorando outros relevantes para o teste de software para aprofundar seu entendimento e expandir seus horizontes.

O que é Teste de acessibilidade?

O teste de acessibilidade é um subconjunto de testes de usabilidade onde, nos usuários considerados, há pessoas com todas as habilidades e deficiências. O significado deste teste é verificar a usabilidade e a acessibilidade.

A acessibilidade tem como objetivo atender pessoas de diferentes habilidades, como:

- Deficiências Visuais
- Deficiência física
- Deficiência auditiva
- Deficiência cognitiva
- Deficiência de Aprendizagem

Uma boa aplicação web deve atender a todos os conjuntos de pessoas e NÃO apenas às pessoas com deficiência. Eles incluem:

1. Usuários com infraestrutura de comunicação deficiente
2. Pessoas mais velhas e novos usuários, que geralmente são analfabetos da computação
3. Usuários que usam o sistema antigo (NÃO são capazes de executar o software mais recente)
4. Usuários que estão usando equipamentos fora do padrão
5. Usuários com acesso restrito

Como executar o teste de acessibilidade

A Iniciativa de Acessibilidade da Web (WAI) descreve a estratégia para revisões preliminares e de conformidade de sites da Web. A Iniciativa de Acessibilidade da Web (WAI) inclui uma lista de ferramentas de software para auxiliar nas avaliações de conformidade. Essas ferramentas variam de problemas específicos, como daltonismo, a ferramentas que executarão ferramentas de

aranha automatizadas.

Ferramentas de teste de acessibilidade da Web

Produto	Fornecedor	URL
AccVerify	HiSoftware	http://www.hisoftware.com
Bobby	Watchfire	http://www.watchfire.com
WebXM	Watchfire	http://www.watchfire.com
Rampa Ascendente	Exato	http://www.deque.com
EmFoco	Tecnologias SSB	http://www.ssbtechnologies.com/

Função das ferramentas automatizadas no teste de aceitação

As ferramentas de teste de acessibilidade automatizadas acima mencionadas são muito boas para identificar páginas e linhas de código que precisam ser verificadas manualmente quanto à acessibilidade.

1. verificar a sintaxe do código do site
2. Procurar padrões conhecidos que os seres humanos tenham listado
3. identificar páginas que contenham elementos que possam causar problemas
4. identificar alguns problemas reais de acessibilidade
5. identificar alguns problemas em potencial

A interpretação dos resultados das ferramentas automatizadas de ensaio da acessibilidade exige experiência em técnicas de acessibilidade com uma compreensão dos problemas técnicos e de utilização.

Os testes são realizados de forma formal e informal para melhorar a qualidade do software. Após a conclusão do teste formal, é realizada uma série de testes informais e arbitrários. Isso é conhecido como teste ad hoc.

O que é o teste ad hoc?

Um teste ad hoc é uma técnica de teste informal feita no software para localizar defeitos. É realizado em um formato aleatório, e também é conhecido como o teste de macacos. Um teste ad hoc não segue uma abordagem sistemática e é desprovido de casos de teste bem documentados.

Os testes ad hoc não têm nenhuma documentação, cenários de teste, casos etc. Os desenvolvedores acham difícil corrigir defeitos detectados por testes ad hoc devido à ausência desses documentos de teste. Além disso, alguns bugs críticos, raros e não antecipados são identificados apenas pela realização de testes aleatórios e informais no software. Também é um tipo de teste de aceitação e economiza o tempo de criação de novos casos de teste.

Um exemplo prático de teste ad hoc é supor que um software precisa ser enviado ao cliente em um dia e seu desenvolvimento é concluído apenas um dia antes disso. Neste ponto, não há tempo restante para criar e executar casos de teste para que a equipe de teste realize testes ad hoc em todo o software com base no conhecimento e na experiência geral do produto.

Tipos de testes ad hoc

Os diferentes tipos de testes ad hoc estão listados abaixo −

Teste do amigo

No teste de parceiro, há envolvimento de pelo menos dois membros durante o processo de teste - um desenvolvedor e um testador. Quando o desenvolvedor conclui a implementação de um componente, ele realiza testes de unidade nele. Apresente que o testador alimenta alguns dados aleatórios e arbitrários para o mesmo componente e examina os resultados. Em caso de erros, o desenvolvedor corrige esses defeitos.

Teste de pares

No teste em par, há envolvimento de dois testadores. Um deles faz a verificação informal e aleatória do software e o outro mantém registro dos resultados dos testes. Assim, ambos trabalham em duplas e trocam ideias, conhecimento para que os testes sejam feitos de forma adequada.

Recursos dos testes ad hoc

Os recursos dos testes ad hoc estão listados abaixo −

- Trata-se de uma abordagem aleatória e informal dos testes.
- Ele não é suportado por nenhuma documentação, cenários de teste, casos etc.
- Ele é executado após a conclusão do teste formal.
- Não segue uma abordagem metódica ou estruturada.
- Leva menos tempo para realizar testes ad hoc.
- Ele detecta bugs no software onde os casos de teste não estão disponíveis.

Quando o teste ad hoc é concluído?

O teste ad hoc é feito nos cenários listados abaixo &minu;

- Há tempo limitado disponível para testar o software.
- Testes formais concluídos.
- Os casos de teste não estão disponíveis.

Quando os testes ad hoc não são concluídos?

O teste ad hoc não é feito nos cenários listados abaixo −

- Isso não é feito se os bugs forem detectados pela execução dos casos de teste.
- No momento do teste beta, isso não está feito.

Vantagens dos testes ad hoc

As vantagens dos testes ad hoc estão listadas abaixo −

- Ele não adere a nenhum processo, portanto testes ad hoc podem ser feitos em qualquer ponto do ciclo de vida do desenvolvimento de software.
- A equipe de teste pode verificar o software e encontrar erros aplicando novas técnicas de teste sem depender apenas dos casos de teste.
- Um desenvolvedor pode executar testes ad hoc no mesmo módulo que está desenvolvendo e aumentar a qualidade do código.
- Embora o processo formal de teste demore muito tempo, testes ad hoc podem ser executados em pouco tempo.
- Ele não requer nenhuma documentação.

Desvantagens dos testes ad hoc

As desvantagens dos testes ad hoc estão listadas abaixo −

- Os testes ad hoc precisam ser realizados por membros da equipe com experiência em testes e conhecimento sólido sobre o produto. Qualquer membro inexperiente da equipe não pode executar um teste ad hoc.
- No caso de um bug, é difícil reproduzir o mesmo, já que o teste ad hoc não é orientado por nenhum planejamento.

Práticas recomendadas a serem seguidas nos testes ad hoc

As melhores práticas a serem seguidas nos testes ad hoc estão listadas abaixo −

- Reúna todo o conhecimento sobre o produto.
- Identificar os componentes do software que apresentam falhas e priorizá-los.
- Utilização de ferramentas de ensaio adequadas.

Conclusão

Isso conclui nossa visão abrangente sobre o tutorial sobre Testes Ad Hoc de Software. Começamos descrevendo o que é teste ad hoc, quais são os tipos, recursos, técnicas, vantagens, desvantagens, tempo e práticas recomendadas de teste ad hoc.

Isso o capacita com conhecimento aprofundado de Teste Ad Hoc de Software. É sábio continuar praticando o que você aprendeu e explorando outros relevantes para o teste de software para aprofundar seu entendimento e expandir seus horizontes.

Sobre esta tradução
A Cisco traduziu este documento com a ajuda de tecnologias de tradução automática e humana para oferecer conteúdo de suporte aos seus usuários no seu próprio idioma, independentemente da localização.

Observe que mesmo a melhor tradução automática não será tão precisa quanto as realizadas por um tradutor profissional.

A Cisco Systems, Inc. não se responsabiliza pela precisão destas traduções e recomenda que o documento original em inglês (link fornecido) seja sempre consultado.