Plano de Teste Completo para o E-commerce da Amazon

Este plano de teste abrangente visa garantir a qualidade e a funcionalidade do e-commerce da Amazon, incluindo testes funcionais, de automação, performance, segurança, acessibilidade, regressão e APIs.

1. Plano de Teste Funcional (PDF)

1.1. Casos de Teste:

Cenário Positivo:

* **Navegação:**
    * **Caso 1:** Acessar a página inicial do site e navegar por diferentes categorias.
    * **Caso 2:** Adicionar um item ao carrinho e finalizar a compra.
    * **Caso 3:** Realizar login e acessar a área do cliente.
    * **Caso 4:** Utilizar a função de pesquisa para encontrar um produto específico.
    * **Caso 5:** Verificar detalhes de um produto, incluindo descrição, preço, avaliações e disponibilidade.
* **Busca:**
    * **Caso 1:** Realizar uma pesquisa simples com um termo único.
    * **Caso 2:** Utilizar filtros avançados para refinar a busca.
    * **Caso 3:** Pesquisar por produtos com diferentes variantes (tamanho, cor, etc.).
* **Login/Logout:**
    * **Caso 1:** Realizar login com credenciais válidas.
    * **Caso 2:** Esquecer a senha e realizar a recuperação.
    * **Caso 3:** Realizar logout da conta.
* **Carrinho de Compras:**
    * **Caso 1:** Adicionar e remover itens do carrinho.
    * **Caso 2:** Modificar a quantidade de itens no carrinho.
    * **Caso 3:** Aplicar cupons de desconto.
* **Checkout:**
    * **Caso 1:** Preencher o formulário de checkout com dados válidos.
    * **Caso 2:** Selecionar diferentes métodos de pagamento.
    * **Caso 3:** Escolher opções de entrega.
    * **Caso 4:** Finalizar a compra com sucesso.
* **Histórico de Pedidos:**
    * **Caso 1:** Visualizar o histórico de pedidos.
    * **Caso 2:** Detalhar um pedido específico.
    * **Caso 3:** Fazer o acompanhamento do status de um pedido.
* **Suporte ao Cliente:**
    * **Caso 1:** Acessar a seção de perguntas frequentes (FAQ).
    * **Caso 2:** Entrar em contato com o suporte ao cliente por chat, email ou telefone.

Cenário Negativo:

* **Navegação:**
    * **Caso 1:** Tentar acessar uma página inexistente.
    * **Caso 2:** Acessar uma categoria inválida.
* **Busca:**
    * **Caso 1:** Realizar uma pesquisa com termo inválido.
    * **Caso 2:** Tentar aplicar filtros inválidos.
* **Login/Logout:**
    * **Caso 1:** Tentar realizar login com credenciais inválidas.
    * **Caso 2:** Tentar acessar a área do cliente sem realizar login.
* **Carrinho de Compras:**
    * **Caso 1:** Tentar adicionar um item indisponível ao carrinho.
    * **Caso 2:** Tentar aplicar um cupom inválido.
* **Checkout:**
    * **Caso 1:** Tentar finalizar a compra sem preencher todos os campos obrigatórios.
    * **Caso 2:** Tentar utilizar um método de pagamento inválido.
* **Histórico de Pedidos:**
    * **Caso 1:** Tentar acessar o histórico de pedidos de outro usuário.
* **Suporte ao Cliente:**
    * **Caso 1:** Tentar enviar uma mensagem com conteúdo inválido.

1.2. Critérios de Aceitação:

* Todas as funcionalidades devem funcionar corretamente em todos os cenários.
* O site deve ser responsivo e acessível em diferentes dispositivos e tamanhos de tela.
* As informações devem ser exibidas de forma clara e concisa.
* A navegação deve ser intuitiva e fluida.
* O processo de compra deve ser simples e seguro.
* O suporte ao cliente deve ser eficiente e responsivo.

1.3. Escopo do Teste:

* Todas as páginas do site, incluindo a página inicial, categorias, produtos, carrinho de compras, checkout, área do cliente, suporte ao cliente.
* Fluxos de usuário principais, como navegação, pesquisa, login, logout, adicionar ao carrinho, finalizar a compra.

1.4. Ambientes de Teste:

* Navegadores: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari.
* Dispositivos móveis: Android, iOS.
* Resoluções de tela: 1024x768, 1280x1024, 1920x1080.

1.5. Riscos e Dependências:

* Dependência de APIs externas.
* Falha no sistema de pagamento.

* Problema de segurança.
* Impacto em performance devido a carga pesada.

1.6. Métricas de Sucesso:

* Taxa de defeitos.
* Cobertura de teste.
* Tempo médio de resolução de defeitos.
* Feedback do usuário.

2. Projeto de Automação em Cypress:

2.1. Estrutura do Projeto:

```
codigo_automacao
% % %  cypress
%    % % %  integration
%    %    % % %  navegacao
%    %    %    % % %  navegar_categorias.spec.js
%    %    %    % % %  finalizar_compra.spec.js
%    %    % % %  busca
%    %    %    % % %  pesquisa_simples.spec.js
%    %    %    % % %  filtros_avancados.spec.js
%    %    % % %  login
%    %       % % %  login_logout.spec.js
%    % % %  support
%    %    % % %  commands.js
%    %    % % %  index.js
%    % % %  fixtures
%    %    % % %  produtos.json
%    %    % % %  usuarios.json
%    % % %  plugins
%    %    % % %  index.js
%    % % %  e2e
%       % % %  navegacao
%          % % %  navegar_categorias.spec.js
% % %  package.json
% % %  cypress.config.js
% % %  allure-report
```

2.2. Código Cypress:

* **Navegação:**
    * `cypress/integration/navegacao/navegar_categorias.spec.js`

    * `cypress/integration/navegacao/finalizar_compra.spec.js`
* **Busca:**
    * `cypress/integration/busca/pesquisa_simples.spec.js`
    * `cypress/integration/busca/filtros_avancados.spec.js`
* **Login/Logout:**
    * `cypress/integration/login/login_logout.spec.js`
* **Outros scripts para testes de carrinho, checkout, histórico de pedidos, etc.**

2.3. Cobertura de Teste:

* Casos de borda, como campos vazios, caracteres inválidos e inputs com valores fora do limite.
* Casos de erro, como mensagens de erro e comportamentos esperados em caso de falha.
* Cenários com dados dinâmicos, como preços e inventário.

2.4. Integração com Allure:

* `cypress/plugins/index.js`: Instalação e configuração do Allure Report.
* `package.json`: Adicionar dependências do Allure (Cypress Allure Plugin).
* Gerar relatórios Allure com o comando `npx cypress run --env environment=staging --reporter allure`

2.5. Configuração de Variáveis de Ambiente:

* `cypress.config.js`: Define variáveis de ambiente para URL do site, credenciais de login, etc.

3. Instruções de Execução (README.md)

3.1. Configuração do Ambiente:

* Instalar Node.js e npm.
* Instalar Cypress: `npm install cypress`.
* Instalar Allure Report: `npm install @shepherd-org/cypress-allure-plugin`.
* Configurar variáveis de ambiente no arquivo `cypress.config.js`.

3.2. Execução dos Testes:

* Executar testes localmente: `npx cypress run --env environment=staging`
* Executar testes em pipeline CI/CD: `npx cypress run --env environment=staging --reporter allure --config-file cypress.config.js`

3.3. Interpretação dos Resultados:

* Resultados no terminal: Verificar mensagens de sucesso, falhas e erros.

* Dashboard do Cypress: Visualizar relatórios de testes com informações detalhadas.
* Relatórios Allure: Analisar resultados de testes com informações gráficas e detalhadas.

3.4. Configuração de CI/CD:

* Integrar os testes Cypress em pipeline CI/CD usando ferramentas como GitHub Actions ou Jenkins.
* Executar testes em cada build ou pull request.
* Gerar relatórios Allure em cada execução.

4. Plano de Testes de Performance (PDF)

4.1. Testes de Carga:

* Utilizar ferramentas como JMeter ou k6.
* Simular cenários de carga real com diferentes números de usuários simultâneos.
* Monitorar métricas de performance como tempo de resposta, throughput, taxa de erros e utilização de recursos.

4.2. Testes de Estresse:

* Simular carga extrema para identificar o ponto de falha do site.
* Monitorar comportamento do site sob carga extrema, como tempo de resposta, erros e degradação de performance.

4.3. Testes de Capacidade:

* Determinar a capacidade máxima do site em termos de usuários simultâneos sem degradação significativa da performance.

4.4. Métricas de Performance:

* Tempo de resposta: Tempo que o site leva para responder a uma requisição.
* Throughput: Número de requisições processadas por segundo.
* Taxa de erros: Porcentagem de requisições que resultam em erros.

5. Plano de Testes de Segurança (PDF)

5.1. Testes de Vulnerabilidade:

* Utilizar ferramentas de análise de vulnerabilidades como Burp Suite ou OWASP ZAP.
* Identificar vulnerabilidades comuns como injeção de SQL, XSS, CSRF, etc.
* Verificar se as medidas de segurança são eficazes.

5.2. Teste de Autenticação/Autorização:

* Verificar se as políticas de autenticação e autorização estão implementadas corretamente.
* Tentar acessar recursos restritos sem autenticação e com permissões inválidas.

5.3. Teste de Penetração:

* Simular ataques maliciosos para avaliar a resistência do site.
* Utilizar ferramentas e técnicas de penetração para testar a segurança do site.

6. Plano de Testes de Acessibilidade (PDF)

6.1. Conformidade com WCAG:

* Testar o site contra as diretrizes WCAG (Web Content Accessibility Guidelines).
* Garantir que o site seja acessível para todos os usuários, incluindo pessoas com deficiências.

6.2. Ferramentas de Acessibilidade:

* Utilizar ferramentas como Axe, Lighthouse e NVDA.
* Realizar testes de acessibilidade para verificar a conformidade com WCAG.

6.3. Casos de Teste Acessíveis:

* Validar a navegação com teclado, uso de leitores de tela, acessibilidade para pessoas com deficiências visuais e motoras.

7. Estratégia de Teste de Regressão:

7.1. Seleção de Casos de Teste:

* Selecionar casos de teste críticos que abrangem as funcionalidades principais do site.
* Priorizar casos de teste que foram modificados ou afetados por novas implementações.

7.2. Automação de Regressão:

* Criar scripts de teste automatizados para executar o conjunto de testes de regressão.
* Utilizar ferramentas como Cypress para automatizar os testes.

7.3. Planejamento de Execução:

* Executar testes de regressão em cada build ou pull request.
* Definir a frequência de execução dos testes de regressão, como diariamente ou semanalmente.

8. Documentação de Integração de APIs:

8.1. Testes de API Automatizados:

* Criar casos de teste para validar as respostas de APIs.
* Utilizar ferramentas como Postman ou Newman para automatizar os testes de API.

8.2. Testes de Contrato:

* Garantir que as APIs respeitam os contratos esperados.
* Utilizar ferramentas como Pact para definir e validar contratos de API.

8.3. Testes de Carga em APIs:

* Realizar testes de carga especificamente para as APIs.
* Utilizar ferramentas como k6 para simular carga e monitorar performance das APIs.

Organização e Acesso aos Artefatos:

* Organizar todos os arquivos e pastas de forma clara e acessível.
* Armazenar todos os relatórios e artefatos em locais facilmente acessíveis para revisão posterior.

Este plano de teste abrangente fornecerá uma base sólida para garantir a qualidade e a funcionalidade do e-commerce da Amazon. Ele também ajudará a identificar e mitigar riscos, melhorar a experiência do usuário e promover a confiança nos serviços da Amazon.