테스트 글머리 기호 3

목차

API 테스트는 API(Application Programming Interface)를 검증하여 기능, 안정성, 성능 및 보안에 대한 기대치를 충족하는지 확인하는 소프트웨어 테스트입니다. 주로 UI(사용자 인터페이스)에 관계없이 소프트웨어 시스템 간의 비즈니스 로직 레이어 및 데이터 교환에 중점을 둡니다

텍스트 간 URL을 테스트하기 위한 것입니다.

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

Cisco의 COBC(Code of Business Conduct)는 Cisco가 성실하게 업무를 수행하고 의사 결정을 내리는 방식을 반영합니다. 또한 책임 있는 AI 사용 및 이해상충과 같은 복잡한 문제를 탐색할 수 있도록 리소스를 제공합니다.

```
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=

현재 발생한 문제에 대한 지원을 요청합니다. 인시던트 레코드가 생성되고 관리되어 문제를 성공적으로 해결합니다. 또한 진행 상황에 대한 알림이 제공됩니다.

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

Black Box Testing 기술에서 테스터나 QA 분석가는 수동으로 서로 다른 테스트 사례를 제공하여 특정 모듈이나 특정 방법 또는 전체 애플리케이션의 기능만 확인합니다. 여기서 테스터는 애플리케이션에 대한 입력을 주고 수동으로 테스트합니다.

예상 출력을 반환하면 테스터가 다른 입력 집합을 진행하고 모든 결과를 팀에 보고합니다. 테스트 중에 사용자가 수동으로 입력한 내용이 실패하면 개발팀에 이 문제를 보고합니다.

테스트 비디오

수표	표
	링크 확인

검사대

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

화이트박스 테스트

화이트박스 테스트 기법은 설계, 코딩 등 시스템 내부 구조를 사람이 수동으로 체크합니다. 여기서 개발팀은 코딩부분 전체를 한 줄씩 검토해 코드의 정확성이 보장되도록 할 예정이다.

코드에서 유사점이나 오류가 발견되면 해당 코딩이나 디자인의 오류를 수정하거나 수정하게 된다. 여기서 체크 코드나 도안은 인간에 의해 수동으로 체크 하므로 전적으로 수동으로 프로세스가 진행되고 있으며 프로세스가 효율적이다.

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

https://www.google.com/

"bdb 개발자 역할" 검사가 ART API에서 One Access 내의 Entra ID로 마이그레이션되었습니다. 액세스를 요청할 때 "통합 방법: memberOf"입니다. 같은 이름의 두 가지 권한이 있습니다.

API 테스트의 주요 특징 ☐

- 메시지 레이어에서의 통신: API 테스트는 GUI를 사용하는 대신 다양한 HTTP 메서드(GET, POST, PUT, DELETE) 및 JSON 또는 XML 같은 데이터 형식을 사용하여 애플리케이션의 엔드포인트(URI)와 직접 상호 작용합니다.
- 조기 결함 감지: API 테스트는 UI가 구축되기 전이라도 소프트웨어 개발 라이프사이클의 초기에 수행할 수 있으므로, 팀이 더 적은 비용으로 더 효율적으로 문제를 찾아 해결할 수 있습니다.
- 자동화 주안점: 직접 상호 작용의 특성과 일관된 구조로 인해 API 테스트는 자동화에 매우 적합하며, 이는 CI/CD 파이프라인에서 지속적인 테스트를 수행하기 위한 최신 Agile 및 DevOps 환경에서 매우 중요합니다.
- 포괄적 지원 범위: 테스트 에지 사례, 오류 처리, UI를 통해 액세스하기 어려울 수 있는 보안 취약성 등 UI 테스트에만 비해 폭넓은 테스트 범위를 제공합니다.

API 테스트 유형 ☐
다양한 유형의 API 테스트는 애플리케이션 품질의 다양한 측면을 다루는 데 사용됩니다. ☐

카탈론

- 기능 테스트: API가 의도된 작업을 올바르게 수행하고, 지정된 대로 입력, 출력 및 상태 코드를 처리하는지 확인합니다.
- 성능 테스트: 다양한 로드 조건(예: 피크 트래픽, 스트레스)에서 API의 속도, 안정성 및 확장성을 평가합니다.
- 보안 테스트: 민감한 데이터를 보호하기 위해 SQL 주입, XSS(Cross-Site Scripting), 손상된 인증/권한 부여와 같은 취약점을 식별합니다.
- 통합 테스트: API가 상호 작용하는 시스템 또는 외부 서비스의 서로 다른 부분이 원활하게 함께 작동하는지 확인합니다.
- 계약 테스트: API가 합의된 계약(OpenAPI/Swaggeror WSDL과 같은 사양)을 준수하도록 하여 서비스 업데이트 간의 변경 사항을 중단하지 않도록 합니다.
- 완벽한 테스트: 여러 API 호출이 함께 연결된 전체 사용자 여정을 검증합니다.
- API 테스트 유형 □
  다양한 유형의 API 테스트는 애플리케이션 품질의 다양한 측면을 다루는 데 사용됩니다. □

  수동 테스트는 소프트웨어가 수행해야 하는 작업을 이해하는 것에서부터 시작됩니다.

  - 기능 요구 사항: 올바른 사용자 로그인과 같은 기능을 확인합니다.
  - 기능 외 요구 사항: 성능, 사용 편의성 및 보안(예: 2초 미만의 페이지 로드 시간) 검증
  - 사용자 사례 및 설계 문서: 사용자 상호 작용 및 워크플로를 이해합니다.
  - 이해 관계자 입력: 고객, 제품 관리자 또는 디자이너와 함께 요구 사항을 명확히 합니다.

2단계: 테스트 계획 생성

테스트 계획은 테스트 전략 및 목표를 정의합니다.

- 범위: 테스트할 기능 및 제외 항목을 식별합니다.
- 목표: 핵심 기능 및 사용자 경험 보장
- 리소스: 팀 구성원, 도구 및 일정을 지정합니다.
- 테스트 기술: 기능, 사용성 및 탐색 테스트를 포함합니다.
- 환경: 스테이징 또는 프로덕션 유사 설정을 정의합니다.

3단계: 설계 테스트 사례

테스트 사례는 철저한 수동 테스트를 보장하는 명확한 단계별 스크립트입니다. 테스트 케이스는 테스트 담당자를 위한 세부 가이드 역할을 하며 모든 시나리오를 점검합니다. 각 테스트 사례에는 다음이 포함됩니다.

- 테스트 ID: 쉬운 추적을 위한 TC_001과 같은 고유한 코드입니다.
- 설명: 목표(예: 유효한 입력으로 확인).
- 전제 조건: 시작하기 전에 필요한 사항(예: 검색 페이지에 있는 것)
- 단계: 수행할 작업(예: 내일 날짜를 선택하고 "검색"을 클릭합니다.
- 예상 결과: 가격 기준으로 분류된 항공편의 목록과 같은 바람직한 결과입니다.
- 사후 조건: 결과 페이지가 표시됩니다.

자세히 보기: [테스트 사례를 작성하는 방법]

4단계: 테스트 환경 설정

테스트 환경은 프로덕션과 비슷해야 합니다.

- 필요한 응용 프로그램을 설치합니다.
- 프로젝트별 설정을 구성합니다.
- 테스트 데이터의 가용성을 보장합니다.
- 하드웨어 및 소프트웨어 요구 사항을 확인합니다.

5단계: 테스트 사례 실행

테스트 사례를 단계별로 실행하고 애플리케이션과의 사용자 상호작용을 수행합니다.

- 실제 결과: 실행 중 발생하는 작업
- 합격/불합격 상태: 실제 결과가 예상 결과와 일치하는지 여부.
- 관찰: 예기치 않은 동작 또는 사용 편이성 문제

6단계: 로그 및 보고서 결함

테스트가 실패하거나 예기치 않은 동작이 발생하면 다음과 같이 결함을 기록합니다.

- 결함 ID: 고유 식별자.
- 요약: 실제 결함이 무엇인지에 대한 간략한 설명
- 재현할 단계: 문제를 발생시키는 자세한 단계.
- 실제 결과 대 예상 결과:발생한 것과 발생했어야 하는 것 비교.
- 심각도: 중대한 영향, 중대한 영향 또는 경미한 영향을 확인합니다.
- 첨부 파일:결함 입증을 위한 스크린샷, 로그 또는 비디오

7단계: 결함 추적 및 확인

수정 적용 후:

- 도구의 결함 상태를 추적합니다.
- 해결된 문제를 다시 테스트합니다.
- 결과에 따라 결함을 닫거나 다시 엽니다.

8단계: 회귀 테스트 실시

회귀 테스트는 결함 수정 또는 새로운 변경이 기존 기능을 손상시키지 않았음을 보장합니다.

- 영향을 받는 부분은 버그를 해결한 후 점검합니다.
- 주요 기능을 확인합니다.
- 통합 지점을 확인하여 이전처럼 작동하는지 확인합니다.

9단계: 테스트 종료 보고서 준비

테스트가 완료되면 테스트 계획의 목표에 대한 결과를 계산하고 동일한 목표에 대한 테스트 완료 보고서를 생성합니다.

- 요약: 테스트 활동 개요

- 테스트 결과: 실행, 통과 및 실패한 테스트 사례 수입니다.
- 발견된 결함: 총 결함, 심각도 및 해결 상태.
- 해결되지 않은 문제: 해결되지 않은 결함 또는 위험.
- 얻은 교훈: 향후 테스트를 위한 통찰력

10단계: 피드백 및 권장 사항 제공

테스트 결과를 분석하여 이해 관계자에게 다음과 같은 실행 가능한 피드백을 제공합니다.

- 소프트웨어 품질.
- 프로세스 개선.
- 향후 테스트 전략
- 사용자 경험 통찰력.

수동 테스트에 사용되는 툴

- TestRail: 강력한 통합 및 대시보드를 통해 수동 테스트 사례를 구성, 실행 및 보고하는 사용자 친화적인 테스트 관리 도구입니다.
- Xray(Jira용): 완벽한 추적 및 원활한 통합을 통해 수동, 자동화 및 BDD 테스트를 지원하는 Jira 기반 테스트 관리 툴입니다.
- Qase: 간단한 UI, AI 기반 테스트 사례 생성, 내장된 결함 추적을 갖춘 현대적인 클라우드 기반 테스트 관리 플랫폼
- Zephyr: 강력한 Jira 통합 및 보고 기능으로 수동 및 탐색 테스트를 지원하는 확장 가능한 테스트 관리 솔루션
- Tuskr: 직관적인 인터페이스와 협업 기능을 갖춘 경량 경제적인 클라우드 기반 테스트 관리 도구입니다.

수동 테스트 필요

- 버그 프리 및 안정성: 수동 테스트의 주요 목표는 애플리케이션이 버그 프리(bug-free), 안정성(stable), 요구 사항을 준수하며, 고객에게 안정적인 제품을 제공하도록 보장하는 것입니다.
- 제품에 익숙함: 수동 테스트를 통해 테스트 엔지니어는 제품에 대해 더 잘 알고 최종 사용자의 관점을 파악할 수 있습니다. 이를 통해 소프트웨어에 대한 올바른 테스트 사례를 작성할 수 있습니다.
- 결함 수정: 수동 테스트는 개발자가 결함을 고정하고 고정된 결함에 대해 재테스트를 수행했는지 확인하는 데 도움이 됩니다.

수동 테스트의 장점

- 빠르고 정확한 시각적 피드백: 소프트웨어 애플리케이션에서 거의 모든 버그를 탐지하며 레이아웃, 텍스트 등과 같이 동적으로 변화하는 GUI 설계를 테스트하는 데 사용됩니다.
- 비용 절감: 고급 기술이나 특정 유형의 툴을 필요로 하지 않으므로 비용이 절감됩니다.
- 코딩이 필요 없음: 블랙박스 테스트 방법을 사용할 때 프로그래밍 지식이 필요하지 않습

니다. 새로운 테스터들은 배우기 쉽습니다.
    - 계획되지 않은 변경에 대한 효율성: 수동 테스트는 애플리케이션에 대한 계획되지 않은 변경 시 적합하며, 이는 쉽게 채택할 수 있기 때문입니다.

카탈론
    - 기능 테스트: API가 의도된 작업을 올바르게 수행하고, 지정된 대로 입력, 출력 및 상태 코드를 처리하는지 확인합니다.
    - 성능 테스트: 다양한 로드 조건(예: 피크 트래픽, 스트레스)에서 API의 속도, 안정성 및 확장성을 평가합니다.
    - 보안 테스트: 민감한 데이터를 보호하기 위해 SQL 주입, XSS(Cross-Site Scripting), 손상된 인증/권한 부여와 같은 취약점을 식별합니다.
    - 통합 테스트: API가 상호 작용하는 시스템 또는 외부 서비스의 서로 다른 부분이 원활하게 함께 작동하는지 확인합니다.
    - 계약 테스트: API가 합의된 계약(OpenAPI/Swaggeror WSDL과 같은 사양)을 준수하도록 하여 서비스 업데이트 간의 변경 사항을 중단하지 않도록 합니다.
    - 완벽한 테스트: 여러 API 호출이 함께 연결된 전체 사용자 여정을 검증합니다.

- 운영 방식

AI 기반 시스템에서 일반적인 API, 웹 UI, 데이터베이스, ESB, 심지어 MCP 서버 전반에 걸쳐 테스트를 쉽게 생성, 확장 및 구성할 수 있는 시각적, 무삭제 툴. 심층적인 기술력이 필요하지 않습니다. 120개 이상의 프로토콜과 메시지 형식을 지원하는 SOAtest는 비즈니스 논리를 엔드 투 엔드로 검증할 수 있는 통합 프레임워크를 제공합니다.

SOAtest를 사용하여 다음을 수행할 수 있습니다.

    - 실제 비즈니스 트랜잭션을 모방하는 시나리오 기반 플로우를 생성하여 특정 시퀀스에 의해 트리거되는 숨겨진 버그를 찾을 수 있습니다.
    - 최소한의 기술적 전문성으로 테스트 로직, 복잡한 어설션, 루프, 데이터 중심 플로우를 구축합니다.
    - 개별 테스트 또는 전체 테스트를 실행하고 회귀 제어를 추가하여 예기치 않은 변경 사항을 즉시 포착합니다.

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.

소프트웨어 수동 테스트란 무엇입니까?

수동 테스트는 다양한 기능 및 기능을 사용하여 소프트웨어를 확인하는 절차입니다. 이 설명서는 소프트웨어를 검증하고 최종 결과 보고서를 제공하는 미리 고안된 일련의 테스트를 통해 제공됩니다. 이 유형의 테스트는 수동 작업을 통해 완전히 수행되므로 완료하는 데 시간이 걸립니다. 따라서, 이러한 유형의 테스트를 수행하는 동안 항상 사람의 실수의 범위가 있다.

모든 새 소프트웨어는 자동화를 채택하기 전에 먼저 수동으로 테스트합니다. 전체 소프트웨어를 수동으로 확인하는 데 더 많은 시간이 소요됩니다. 소프트웨어의 모든 기능과 기능이 안정적이고 잘 작동하면 수동 테스트 사례 중 일부를 자동화로 변환할 수 있습니다. 수동 테스트 사례를 먼저 평가하여 완전 자동화 가능 여부를 확인한다. 이러한 유형의 테스트는 전체 프로세스를 완료하는 데 자동화 툴을 사용할 필요가 없습니다.

소프트웨어 수동 테스트의 특성

소프트웨어 수동 테스트의 특징은 다음과 같습니다−

- 수동 테스트는 사람의 개입으로 완벽하게 수행됩니다.
- 탐색 테스트는 수동 테스트의 중요한 부분입니다. 예비 테스트에서 테스터는 사전 설정된 테스트 없이 소프트웨어를 확인합니다. 예측하지 못한 결함을 찾아내 고객 만족도를 향상시킨다.
- 수동 테스트는 요구 사항 및 기타 테스트 조건의 변화에 따라 테스트 사례를 수정할 수 있으므로 유연합니다.
- 수동 테스트는 SDLC(Software Development Life Cycle)의 초기 단계부터 적용할 수 있습니다.
- 복잡한 일부 테스트 사례는 자동화 없이 수동으로만 실행할 수 있습니다.
- 수동 테스트는 소프트웨어의 사용자 인터페이스를 검증하는 데 유용합니다. 소프트웨어의 디스플레이, 응답성 및 정상적인 설계를 검증하는 데 도움이 됩니다.

소프트웨어 수동 테스트가 필요한 이유는?

아래 나열된 이유로 소프트웨어 수동 테스트가 필요합니다−

- 수동 테스트를 통해 소프트웨어에 결함이 없으며, 요구 사항에 따라 올바르게 작동하며, 프로덕션 환경에 구축할 수 있을 만큼 안정적임을 확인합니다.
- 수동 테스트를 통해 테스트 담당자는 소프트웨어에 익숙해지고 소프트웨어가 고객에게 어떻게 반응하는지 이해할 수 있습니다. 이는 효과적인 테스트 사례를 개발하는 데 도움이 됩니다.
- 수동 테스트를 통해 소프트웨어의 결함을 식별하고 해결합니다.

소프트웨어 수동 테스트 단계

소프트웨어 수동 테스트의 여러 단계가 아래에 나와 있습니다−

1단계− 첫 번째 단계는 요구사항 및 사양 문서, 가이드 등을 거치면서 요구사항 분석 단계를 포함한다.

2단계− 두 번째 단계에서는 모든 요구 사항에 부합하는 테스트 계획을 생성합니다.

3단계- 세 번째 단계에서는 모든 요구 사항을 다루는 테스트 사례를 생성합니다.

4단계- 4단계에서는 올바른 테스트 환경에서 테스트 사례를 실행합니다.

5단계- 5단계는 테스트 실행 결과를 분석하고 불일치를 결함으로 보고한다.

6단계- 6단계는 결함 수정 및 재테스트를 포함합니다. 실패한 테스트 사례를 다시 실행하는 것도 포함됩니다.

소프트웨어 수동 테스트 유형

다양한 유형의 소프트웨어 수동 테스트가 아래에 나와 있습니다-

- 블랙박스 테스트- 테스터가 소프트웨어의 내부 작업에 대한 지식이 없는 테스트 기술입니다. 주로 사용자 요구사항에 따라 기능 및 기능이 올바르게 작동하는지 확인하는 것을 다룬다.
- White Box Testing- 내부 구조 및 소프트웨어의 프로그램 소스 코드 확인을 포함하는 테스트 절차입니다.
- 그레이 박스 테스트- 블랙 박스의 원리와 화이트 박스 테스트 기법을 모두 사용하는 테스트 기법입니다.

소프트웨어 수동 테스트에 사용되는 툴

소프트웨어 수동 테스트에 사용되는 여러 툴이 아래에 나와 있습니다-

- 테스트 링크
- 부가질라
- 지라
- 로드러너
- Apache JMeter
- 완벽해

소프트웨어 수동 테스트와 자동화 테스트의 차이점

다음은 소프트웨어 수동 테스트 및 자동화 테스트 - 비교입니다

수동 테스트	자동화 테스트
소프트웨어를 수동으로 확인하는 절차입니다.	자동화 도구의 도움으로 소프트웨어를 확인하는 절차입니다.

테스트 사례를 수동으로 실행하는 작업이 포함됩니다.	자동화 스크립트 및 툴을 통한 테스트 사례 실행이 포함됩니다.
생산성이 떨어지고 완료하는 데 시간이 더 걸립니다.	생산성이 향상되고 완료하는 데 시간이 단축됩니다.
100% 테스트 커버리지를 보장하지는 않습니다.	수동 테스트보다 더 많은 테스트 범위를 보장합니다.
프로그래밍 기술이 필요하지 않습니다. 소프트웨어에 대한 지식이 있어야만 수행할 수 있다.	프로그래밍 기술이 필요합니다.

소프트웨어 수동 테스트의 장점

소프트웨어 수동 테스트의 장점은 다음과 같습니다−

- 수동 테스트를 통해 동적으로 변화하는 요소를 확인할 수 있습니다.
- 수동 테스트는 비용이 저렴하며 숙련된 리소스에 의존하지 않습니다.
- 수동 테스트는 프로그래밍 지식이 없는 테스터에 의해 수행될 수 있다.
- 수동 테스트는 매우 신속하게 채택할 수 있으며, 소프트웨어의 예측할 수 없는 변경을 수용하기에 적합합니다.

소프트웨어 수동 테스트의 단점

소프트웨어 수동 테스트의 단점은 다음과 같습니다−

- 수동 테스트는 매우 신뢰할 수 없으며 사람의 실수에 대한 범위를 제공합니다.
- 각 모듈에 대해 별도의 수동 테스트 사례 세트를 개발해야 하므로 재사용성의 범위가 매우 적습니다.
- 수동 테스트는 전적으로 수동 테스트 실행에 따라 달라집니다. 그러나 일부 테스트 단계는 수동 작업으로 수행할 수 없습니다.
- 수동 테스트를 수행하는 테스트 담당자는 소프트웨어를 사용한 경험이 있어야 합니다. 또한 수동 테스트를 실행하는 동안 소프트웨어의 모든 기능이 포함되었다는 보장이 없습니다.
- 수동 테스트는 시간이 많이 걸리는 작업입니다.

결론

이것으로 Software Manual Testing(소프트웨어 수동 테스트)에 대한 포괄적인 자습서를 마치겠습니다. 소프트웨어 수동 테스트란 무엇이며, 소프트웨어 수동 테스트의 특징이 무엇인지, 소프트웨어 수동 테스트가 필요한 이유는 무엇인지, 소프트웨어 수동 테스트의 여러 단계는 무엇인지, 소프트웨어 수동 테스트의 여러 유형은 무엇인지, 소프트웨어 수동 테스트에 사용되는 여러 툴은 무엇인지, 소프트웨어 수동 테스트와 자동화 테스트의 차이점은 무엇인지, 소프트웨어 수동 테스트의 장점은 무엇인지, 소프트웨어 수동 테스트의 단점은 무엇인지 설명하면서 시작했습니다. 이를 통해

소프트웨어 수동 테스트에 대한 심층적인 지식을 얻을 수 있습니다. 배운 내용을 꾸준히 실천하고 소프트웨어 테스트와 관련된 다른 사람을 탐색하여 이해를 심화하고 시야를 넓히는 것이 현명합니다.

접근성 테스트란 무엇입니까?

접근성 테스트는 사용성 테스트의 하위 집합으로, 고려 중인 사용자는 모든 능력과 장애를 가진 사용자입니다. 이 테스트의 의의는 사용성과 접근성을 모두 검증하는 것이다.

접근성은 다음과 같은 다양한 능력을 가진 사람들을 만족시키는 것을 목표로 합니다.

- 시각 장애
- 신체 장애
- 청각 장애
- 인지 장애
- 학습 장애

좋은 웹 애플리케이션은 장애인에 국한되지 않고 모든 사람들에게 적합해야 합니다. 그러한 구성 요소는 다음과 같습니다.

1. 커뮤니케이션 인프라가 열악한 사용자
2. 컴퓨터 문맹인 노인과 신규 사용자
3. 이전 시스템을 사용하는 사용자(최신 소프트웨어를 실행할 수 없음)
4. NON-Standard 장비를 사용하는 사용자
5. 액세스가 제한된 사용자

접근성 테스트를 수행하는 방법

WAI(Web Accessibility Initiative)에서는 웹 사이트의 사전 검토 및 적합성 검토 전략에 대해 설명합

니다. WAI(Web Accessibility Initiative)에는 적합성 평가를 지원하는 소프트웨어 도구 목록이 포함되어 있습니다. 이러한 도구는 색맹에서부터 자동 스파이더 도구를 수행할 도구에 이르기까지 특수한 문제가 있습니다.

웹 접근성 테스트 도구

제품	공급업체	URL
AccVerify	하이소프트웨어	http://www.hisoftware.com
보비	망루 불	http://www.watchfire.com
웹XM	망루 불	http://www.watchfire.com
램프 어센드	데케	http://www.deque.com
인포커스	SSB 기술	http://www.ssbtechnologies.com/

수락 테스트에서 자동화된 툴의 역할

위의 자동 접근성 테스트 도구는 접근성을 수동으로 확인해야 하는 페이지와 코드 줄을 식별하는 데 매우 우수합니다.

1. 사이트 코드 구문 확인
2. 인간이 나열한 알려진 패턴 검색
3. 문제를 일으킬 수 있는 요소가 포함된 페이지 식별
4. 몇 가지 실제 액세스 가능성 문제 파악
5. 몇 가지 잠재적 문제 파악

자동화된 접근성 검사 도구에서 나온 결과의 해석은 기술적, 사용성 문제에 대한 이해와 함께 접근성 기법에 대한 경험이 필요하다.

테스트는 소프트웨어 품질을 높이기 위해 공식적 및 비공식적 방식으로 수행됩니다. 공식적인 테스트가 완료된 후 비공식적이고 임의적인 테스트의 라운드를 실시한다. 이를 임시 테스트라고 합니다.

임시 테스트란 무엇입니까?

임시 테스트는 결함을 찾기 위해 소프트웨어에서 수행되는 비공식적인 테스트 기법입니다. 무작위 형식으로 진행되며, 원숭이 검사라고도 한다. 임시 테스트는 체계적인 접근 방식을 따르지 않으며 잘 문서화된 테스트 사례가 없습니다.

임시 테스트에는 기록, 테스트 시나리오, 사례 등이 없습니다. 이러한 테스트 문서가 없기 때문에 개발자는 임시 테스트로 탐지된 결함을 수정하기가 어렵습니다. 또한 일부 중요하고 드물며 예상치 못한 버그는 소프트웨어에 대해 무작위 비공식 테스트를 수행해야만 식별됩니다. 일종의 허용 테스트이기도 하며 새로운 테스트 사례를 작성하는 시간을 절약할 수 있습니다.

임시 테스트의 실제 예는 하루 만에 소프트웨어를 클라이언트에 배송해야 하고 그 하루 전에 개발이 완료되어 테스트 사례를 만들고 실행할 시간이 없다고 가정했을 때 테스트 팀이 전반적인 제품 지식과 경험을 바탕으로 전체 소프트웨어에 대한 임시 테스트를 실시합니다.

임시 테스트 유형

임시 테스트의 차이 유형은 아래에 나열되어 있습니다-

버디 테스트

친구 테스트에서는 테스트 프로세스 중에 적어도 2명의 멤버가 참여합니다. 즉, 개발자 1명과 테스터 1명이 참여합니다. 개발자는 구성 요소 구현을 완료하면 그에 대한 단위 테스트를 수행합니다. 테스터가 임의의 데이터를 같은 구성 요소에 공급하고 그 결과를 검토하도록 게시합니다. 오류가 있을 경우 개발자는 해당 결함을 수정합니다.

쌍 테스트

두 번의 시험에서는 두 명의 시험자의 참여가 있다. 그 중 하나는 소프트웨어에 대한 비공식적, 임의적 검증을 수행하고, 다른 하나는 테스트 결과를 기록한다. 그래서 둘 다 짝을 이뤄 일하며, 테스트가 제대로 이뤄지도록 아이디어, 지식을 교환한다.

임시 테스트의 기능

임시 테스트의 기능은 다음과 같습니다−

- 이것은 무작위적이고 비공식적인 테스트입니다.
- 문서, 테스트 시나리오, 사례 등에서는 지원되지 않습니다.
- 공식적인 테스트가 완료된 후 수행됩니다.
- 방법적이거나 구조화된 접근 방식을 따르지 않습니다.
- 임시 테스트를 수행하는 데 걸리는 시간이 줄어듭니다.
- 테스트 사례를 사용할 수 없는 소프트웨어에서 버그를 탐지합니다.

임시 테스트는 언제 완료됩니까?

임시 테스트는 &minu; 아래에 나열된 시나리오에서 수행됩니다

- 소프트웨어를 테스트할 수 있는 시간이 제한되어 있습니다.
- 공식 테스트가 완료되었습니다.
- 테스트 사례를 사용할 수 없습니다.

임시 테스트는 언제 완료되지 않습니까?

임시 테스트는 아래 나열된 시나리오에서 수행되지 않습니다−

- 테스트 사례를 실행하여 버그가 탐지되면 수행되지 않습니다.
- 베타 테스트 시에는 완료되지 않습니다.

임시 테스트의 장점

임시 테스트의 장점은 다음과 같습니다−

- 어떤 프로세스도 준수하지 않으므로 소프트웨어 개발 라이프사이클의 어느 시점에서든 임시 테스트를 수행할 수 있습니다.
- 테스트 팀은 테스트 사례에만 의존하지 않고 새로운 테스트 기법을 적용하여 소프트웨어를 검증하고 오류를 발견할 수 있습니다.
- 개발자는 개발 중인 모듈에 대해 임시 테스트를 수행하고 코드 품질을 높일 수 있습니다.
- 공식적인 테스트 프로세스는 많은 시간이 걸리지만, 임시 테스트는 짧은 시간 내에 수행될 수 있습니다.
- 어떠한 문서화도 필요하지 않습니다.

임시 테스트의 단점

임시 테스트의 단점은 다음과 같습니다−

- 임시 테스트는 제품에 대한 테스트 경험과 건전한 지식이 있는 팀 구성원이 수행해야 합니다. 팀의 경험이 없는 구성원은 임시 테스트를 수행할 수 없습니다.
- 버그의 경우 임시 테스트는 계획에 의해 실행되지 않으므로 재현하기가 어렵습니다.

임시 테스트에서 따라야 할 모범 사례

임시 테스트에서 따라야 할 모범 사례는 다음과 같습니다−

- 제품에 대한 모든 지식을 수집합니다.
- 소프트웨어의 결함이 발생하기 쉬운 구성 요소를 식별하고 우선 순위를 지정합니다.
- 적합한 테스트 도구 사용.

결론

이것으로 소프트웨어 임시 테스트에 대한 종합적인 자습서를 마치겠습니다. 임시 테스트란 무엇이며, 임시 테스트의 유형, 기능, 기법, 장점, 단점, 시간 및 모범 사례는 무엇인지에 대한 설명으로 시작했습니다.

이를 통해 소프트웨어 Ad Hoc 테스트에 대한 심층적인 지식을 얻을 수 있습니다. 배운 내용을 꾸준히 실천하고 소프트웨어 테스트와 관련된 다른 사람을 탐색하여 이해를 심화하고 시야를 넓히는 것

이 현명합니다.

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.