26.5以降のテストの管理者

内容

概要

Q.

A.

APIテストは、アプリケーションプログラミングインターフェイス(API)を検証するソフトウェアテストの一種で、機能、信頼性、パフォーマンス、およびセキュリティに対する期待を確実に満たすように検証します。主に、ユーザインターフェイス(UI)に関係なく、ソフトウェアシステム間のビジネスロジック層とデータ交換に重点を置いています

これは、テキスト間のURLをテストするためです



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

シスコのビジネス行動規範(COBC)は、私たちが誠実に仕事をし、意思決定を行う方法を反映しています。また、責任あるAIの使用や利害の対立など、複雑な問題の解決に役立つリソースも提供します。

```
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=

発生している問題についてサポートを依頼します。インシデントレコードが作成され、解決に至るまで管理されます。また、進捗状況も通知されます。

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

ブラックボックステストでは、テスターまたはQAアナリストは、特定のモジュールまたは特定の方法、あるいは場合によってはアプリケーション全体の機能をチェックするために、さまざまなテストケースを手動で提供します。ここで、テスターはアプリケーションの入力を与え、それを手動でテストします。

予想される出力が返された場合、テスターは別の入力セットを続行し、すべての結果をチームに報告します。テスト中にユーザが手動で入力した情報が失敗した場合、ユーザはこの問題を開発チームに報告します。



テストビデオ

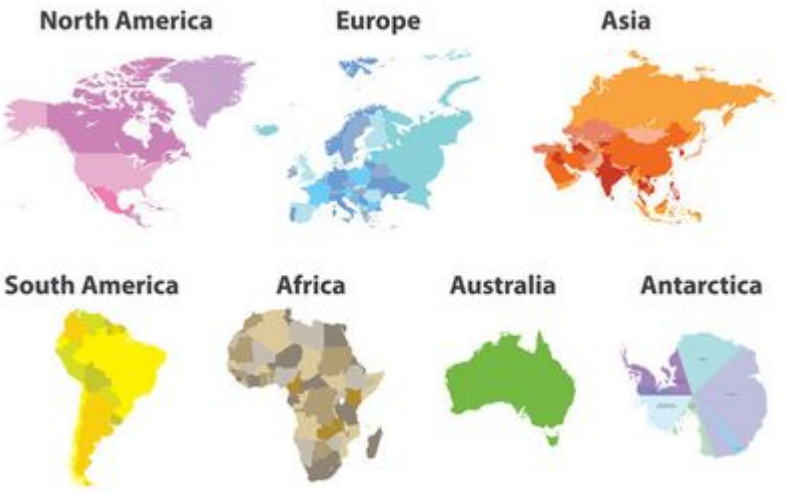
検証	テーブル
	リンクの確認

テスト表

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

ホワイトボックステスト

ホワイトボックステスト技術では、設計、コーディングなど、システムの内部構造を手動で確認します。ここでは、開発チームがコーディング部分全体を行ごとにレビューし、コードの正確性を確認します。



CISCO



HERE IS A SAMPLE



コードに不整合やエラーが見つかった場合、コードまたは設計のエラーを修正または修正します。ここでは、プロセスは完全に手動で実行され、チェックコードまたは設計は人間によって手動でチェックされるため、プロセスは効率的です。

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

https://www.google.com/

「bdb開発者ロール」チェックは、One Access内でART APIからEntra IDに移行しました。アクセス権をリクエストする際は、「Integration Method: memberOf」を選択してください。同じ名前の権限が2つあります。

APIテストの主な側面 ☐

- メッセージ層での通信:APIテストは、GUIを使用する代わりに、さまざまなHTTPメソッド(GET、POST、PUT、DELETE)およびJSONやXMLなどのデータ形式を使用して、アプリケーションのエンドポイント(URI)と直接やり取りします。
- 早期の不具合検出:APIテストは、UIが構築される前でも、ソフトウェア開発の初期段階で実行できるため、より効率的かつ低コストで問題を発見して修正できます。
- 自動化の焦点：直接的なインタラクションと一貫した構造の性質により、APIテストは自動化に適しています。これは、CI/CDパイプラインでの継続的なテストを実現するために、最新のアジャイルおよびDevOps環境において重要です。
- 包括的なカバレッジ:UIテスト単独と比較して、幅広いテストカバレッジを提供します。これには、テストエッジケース、エラー処理、UIを介してアクセスすることが困難なセキュリティの脆弱性などが含まれます。

APIテストのタイプ ☐
アプリケーションの品質のさまざまな側面をカバーするために、さまざまなタイプのAPIテストが使用されます。 ☐

カタロン

- 機能テスト:APIが意図した動作を正しく実行していることを確認し、入力、出力、およびステータスコードを指定どおりに処理します。

- パフォーマンステスト：さまざまな負荷条件（ピークトラフィック、負荷など）でAPIの速度、安定性、拡張性を評価します。
- セキュリティテスト:SQLインジェクション、クロスサイトスクリプティング(XSS)、認証/認可の失敗などの脆弱性を特定し、機密データを保護します。
- 統合テスト:APIが相互作用するシステムまたは外部サービスのさまざまな部分がシームレスに連携することを確認します。
- 契約テスト:APIが合意された契約（OpenAPI/SwaggerやWSDLなどの仕様）に準拠していることを確認し、サービス更新間の変更を防ぎます。
- エンドツーエンドテスト：チェーン接続された複数のAPIコールを含むユーザージャーニー全体を検証します。
- APIテストのタイプ □
  アプリケーションの品質のさまざまな側面をカバーするために、さまざまなタイプのAPIテストが使用されます。 □

  カタロン
  - 機能テスト:APIが意図した動作を正しく実行していることを確認し、入力、出力、およびステータスコードを指定どおりに処理します。
  - パフォーマンステスト：さまざまな負荷条件（ピークトラフィック、負荷など）でAPIの速度、安定性、拡張性を評価します。
  - セキュリティテスト:SQLインジェクション、クロスサイトスクリプティング(XSS)、認証/認可の失敗などの脆弱性を特定し、機密データを保護します。
  - 統合テスト:APIが相互作用するシステムまたは外部サービスのさまざまな部分がシームレスに連携することを確認します。
  - 契約テスト:APIが合意された契約（OpenAPI/SwaggerやWSDLなどの仕様）に準拠していることを確認し、サービス更新間の変更を防ぎます。
  - エンドツーエンドテスト：チェーン接続された複数のAPIコールを含むユーザージャーニー全体を検証します。

- 動作の仕組み

  視覚的でコードレスなツールにより、API、Web UI、データベース、ESB、さらにはAIを搭載したシステムで一般的なMCPサーバーでも、テストの作成、拡張、および編成が簡単にできます。高度な技術スキルは必要ありません。SOAtestは120を超えるプロトコルとメッセージ形式をサポートしており、統合フレームワークによってビジネス・ロジックをエンド・ツー・エンドで検証できます。

  SOAtestを使用すると、次のことができます。

  - 実際のビジネストランザクションを模倣したシナリオベースのフローを作成し、特定のシーケンスによって引き起こされる隠れたバグを見つけやすくします。
  - 最小限の技術的専門知識で、テストロジック、複雑なアサーション、ループ、およびデータ駆動型フローを構築します。
  - 個々のテストまたはスイート全体を実行し、予期しない変更をすぐにキャッチするために回帰コントロールをアタッチします。

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.

ソフトウェアの手動テストとは何ですか。

手動テストは、さまざまな機能を使用してソフトウェアを検証する手順です。事前に想定された一連のテストに従ってソフトウェアを検証し、最終的な結果レポートを提供します。このタイプのテストは手作業で完全に行われるため、完了までに時間がかかります。したがって、このタイプのテストを実行する際には、常に人的エラーの範囲が存在します。

自動化を採用する前に、新しいソフトウェアはすべて手動でテストされます。ソフトウェア全体を手動で検証するには、さらに時間がかかります。ソフトウェアのすべての機能が安定し、正常に動作したら、手動のテストケースの一部を自動化に変換できます。手動テストケースを最初に評価して、完全に自動化できるかどうかを確認します。このタイプのテストでは、プロセス全体を完了するために自動化ツールを使用する必要はありません。

広告

ソフトウェアの手動テストの特性

ソフトウェアの手動テストの特性は−のとおりです。

- 手動テストは、人間の介入の助けを借りて完全に実行されます。
- 探索的検査は、手動テストの重要な部分です。探索的テストでは、事前に決められた一連のテストを行わずに、テスト担当者がソフトウェアを検証します。予期しない不具合を検出し、顧客満足度を向上させます。
- 手動テストは、要件の変更やその他のテスト条件に基づいてテストケースを変更できるため、柔軟です。
- 手動テストは、ソフトウェア開発ライフサイクル(SDLC)の最初の段階から採用できます。
- 複雑なテストケースの中には、自動化を行わずに手動でのみ実行できるものがあります。
- 手動テストは、ソフトウェアのユーザインターフェイスの検証に役立ちます。これは、ソフトウェアの表示、応答性、および通常の設計を確認するのに役立ちます。

ソフトウェアの手動テストが必要な理由

ソフトウェアの手動テストは、次に示す理由のために必要です−

- 手動テストでは、ソフトウェアに不具合がなく、要件に従って正しく動作し、実稼働環境に導入できる安定性を備えていることが確認されます。
- 手動テストでは、テスト担当者がソフトウェアに慣れ、ソフトウェアが顧客にどのように対応するかを理解できます。これは、効果的なテストケースの開発に役立ちます。
- 手動テストでは、ソフトウェアの不具合を特定し、解決します。

ソフトウェアの手動テストの手順

ソフトウェアの手動テストの各ステップを次に示します−

− ステップ1：最初のステップでは、要件と仕様に関するドキュメント、ガイドなどに目を通して、要件分析フェーズを行います。

ステップ2−2番目のステップでは、すべての要件に関係するテスト計画を作成します。

ステップ3−3番目のステップでは、すべての要件をカバーするテストケースを作成します。

ステップ4 –4番目のステップでは、正しいテスト環境でテストケースを実行します。

ステップ5–5番目のステップでは、テストの実行結果を分析し、その差異を不具合として報告します。

ステップ6 –6番目のステップには、不具合修正と再テストが含まれます。失敗したテストケースの再実行も含まれます。

ソフトウェアの種類の手動テスト

各種のソフトウェアの手動テストを次に示します–

- ブラックボックス試験–これは、テスターがソフトウェアの内部作業に関する知識を持っていない試験技術です。主に、ユーザの要件に従って機能が正しく動作するかどうかを確認します。
- ホワイトボックステスト–ソフトウェアの内部構造およびプログラムソースコードの検証を含むテスト手順です。
- グレーボックス試験–ブラックボックスの原理とホワイトボックスの試験技術の両方を使用する試験技術です。

ソフトウェアの手動テストに使用されるツール

ソフトウェアの手動テストに使用されるさまざまなツールを次に示します–

- テストリンク
- ブジラ
- ジラ
- LoadRunner
- Apache JMeter
- 完了

ソフトウェアマニュアルと自動化テストの違い

ソフトウェアの手動テストと自動化テストの比較–

手動テスト	自動化テスト
手動でソフトウェアを確認する手順です。	自動化ツールを使用してソフトウェアを検証する手順です。
手動によるテストケースの実行を含みます。	自動化スクリプトとツールを使用した

	テストケースの実行が含まれます。
生産性が低く、完了に時間がかかります。	生産性が向上し、完了までの時間が短縮されます。
テストカバレッジを百パーセント保証するものではありません。	手動テストよりも多くのテストカバレッジを保証します。
プログラミングのスキルは必要ありません。ソフトウェアの知識を持っている場合にのみ実行できます。	プログラミングのスキルが必要です。

ソフトウェア手動テストの利点

ソフトウェアの手動テストの利点は–のとおりです

- 手動テストは、画面上で動的に変化する要素を確認するのに役立ちます。
- 手動テストは安価で、熟練したリソースに依存しません。
- 手動テストは、プログラミングの知識を持たないテスターによって実行できます。
- 手動テストは非常に迅速に採用でき、ソフトウェアの予測不可能な変更に対応するのに適しています。

ソフトウェアの手動テストの欠点

ソフトウェアの手動テストの短所は–のとおりです

- 手動テストはあまり信頼できず、人的エラーの範囲を提供します。
- モジュールごとに個別の手動テストケースを開発する必要があるため、再利用の範囲が非常に少なくなります。
- 手動テストは、テストの手動実行に完全に依存しています。ただし、一部のテスト手順は手動の作業では実行できません。
- 手動テストを実行するテスト担当者は、ソフトウェアを操作する経験が必要です。また、手動テストの実行中に、ソフトウェアのすべての機能がカバーされていることを確認することはできません。
- 手動テストは、ほとんどの場合、時間のかかる作業です。

結論

以上で、ソフトウェアの手動テストに関するチュートリアルの全体を終わります。ソフトウェアの手動テストとは何か、ソフトウェアの手動テストの特性は何か、ソフトウェアの手動テストが必要な理由、ソフトウェアの手動テストの異なる手順は何か、ソフトウェアの手動テストの異なるタイプは何か、ソフトウェアの手動テストに使用される異なるツールは何か、ソフトウェアの手動テストと自動化テストの違いは何か、ソフトウェアの手動テストテストの欠点は何か。これにより、ソフトウェアの手動テストに関する詳細な知識が得られます。理解を深め、自分の視野

を広げるために、これまで学んだことを実践し、ソフトウェアテストに関連する他のトピックを探り続けることが賢明です。

アクセシビリティテストとは

アクセシビリティテストは、すべての能力と障害を持つユーザを対象としたユーザビリティテストのサブセットです。このテストの重要性は、使いやすさとアクセシビリティの両方を確認することです。

アクセシビリティは、次のようなさまざまな能力を持つ人々に対応することを目的としています。

- 視覚障害
- 身体的障害
- 聴覚障害
- 認知障害
- 学習障害

優れたWebアプリケーションは、障がい者だけでなく、あらゆる人々に対応する必要があります。802.11 の標準規格には以下があります。

1. 通信インフラストラクチャが不十分なユーザ
2. コンピュータの読み書きが不可能な高齢者や新しいユーザ
3. 古いシステムを使用しているユーザー（最新のソフトウェアを実行できない）
4. NON-Standard Equipmentを使用しているユーザ
5. アクセスが制限されているユーザー

広告

アクセシビリティテストの実行方法

Web Accessibility Initiative(WAI)は、Webサイトの事前レビューと適合性レビューの戦略について説明します。Web Accessibility Initiative(WAI)には、適合性評価を支援するソフトウェアツールのリストが含まれています。これらのツールは、色覚異常などの特定の問題から、自動スパイダーツールを実行するツールまで多岐にわたります。

Webアクセシビリティテストツール

製品	ベンダー	URL
内容確認	Hiソフトウェア	http://www.hisoftware.com
ボビー	ウォッチファイア	http://www.watchfire.com
WebXM	ウォッチファイア	http://www.watchfire.com
ランプ上り	デケ	http://www.deque.com
フォーカス内	SSBテクノロジー	http://www.ssbtechnologies.com/

受け入れテストにおける自動化ツールの役割

上記の自動アクセシビリティテストツールは、アクセシビリティを手動でチェックする必要があるコードのページと行を識別するのに非常に優れています。

1. サイトのコードの構文を確認する
2. 人間がリストした既知のパターンの検索
3. 問題を引き起こす可能性のある要素を含むページを識別する
4. 実際のアクセシビリティの問題を特定する
5. 潜在的な問題を特定する

自動アクセシビリティテストツールの結果を解釈するには、アクセシビリティテクニックの経験を積み、技術的およびユーザビリティの問題を理解している必要があります。

テストは、ソフトウェアの品質を高めるために、公式な方法と非公式な方法の両方で行われます。正式なテストが完了した後、非公式および任意のテストが実施されます。これはアドホックテストと呼ばれます。

アドホックテストとは何ですか。

アドホックテストは、不具合を見つけるためにソフトウェア上で行われる非公式のテストテクニックです。これはランダムな形式で行われ、モンキーテストとも呼ばれます。アドホックテストは体系的なアプローチに従わず、十分に文書化されたテストケースはありません。

アドホックテストには、ドキュメント、テストシナリオ、ケースなどはありません。アドホックテストで検出された不具合は、これらのテスト文書が存在しないため、修正が困難です。また、重要なバグ、まれなバグ、および予期しないバグの一部は、ソフトウェアでランダムかつ非公式のテストを実施することによってのみ特定できます。受け入れテストの一種でもあり、新しいテストケースを作成する時間を節約します。

アドホックテストの実例としては、ソフトウェアを1日でクライアントに出荷する必要があり、その1日前に開発が完了し、この時点でテストケースを作成して実行する時間がないため、テストチームは製品全体の知識と経験に基づいてソフトウェア全体のアドホックテストを実施します。

広告

アドホックテストのタイプ

アドホックテストのタイプの違いは–のとおりです。

バディテスト

バディテストでは、テストプロセス中に少なくとも2人のメンバー（1人の開発者と1人のテスター）が関与します。開発者がコンポーネントの実装を完了すると、そのコンポーネントに対して単体テストを行います。テスターが任意のランダムなデータを同じコンポーネントに供給し、結果を調べることを投稿します。不具合が発生した場合、開発者はエラーを修正します。

ペアテスト

ペアテストでは、2人のテスターが関与します。一方はソフトウェアの非公式かつランダムな検証を行い、もう一方はテスト結果の記録を保持します。したがって、両者はペアで動作し、テストが適切に行われるようにアイデア、知識を交換します。

アドホックテストの機能

アドホックテストの機能は次のとおりです–

- これはテストに対する無作為かつ非公式なアプローチです。
- ドキュメント、テストシナリオ、ケースなどによってサポートされていません。

- これは、正式なテストが完了した後で実行されます。
- 整然としたアプローチや構造化されたアプローチは採用されません。
- アドホックテストの実行に要する時間が短縮されます。
- テストケースを利用できないソフトウェアのバグを検出します。

アドホックテストはいつ行われますか。

アドホックテストは、次に示すシナリオで実行されます。&minu;

- ソフトウェアのテストに使用できる時間は限られています。
- 正式なテストが完了しました。
- テストケースは使用できません。

アドホックテストが実行されないのはなぜですか。

次に示すシナリオでは、アドホックテストは行われません−

- テストケースの実行によってバグが検出された場合は実行されません。
- ベータテストの時点では、これは行われません。

アドホックテストの利点

アドホックテストの利点は−のとおりです。

- どのプロセスにも従わないため、ソフトウェア開発のライフサイクルのどの時点でもアドホックテストを実行できます。
- テストチームは、テストケースのみに依存することなく、新しいテスト技術を適用することで、ソフトウェアを検証し、エラーを見つけることができます。
- 開発者は、開発中の同じモジュールでアドホックテストを実行し、コード品質を向上させることができます。
- 正式なテストプロセスには多くの時間がかかりますが、アドホックテストは短時間で実行できます。
- ドキュメントは必要ありません。

アドホックテストの欠点

アドホックテストの短所は−のとおりです

- アドホックテストは、テストの経験があり、製品に関する十分な知識を持つチームメンバーによって実施される必要があります。経験の浅いチームメンバーはアドホックテストを実行できません。
- バグが発生した場合、アドホックテストは計画によって実行されないため、バグを再現することは困難です。

アドホックテストで従うべきベストプラクティス

アドホックテストで従うべきベストプラクティスを–に示します。

- 製品に関するすべての知識を収集します。
- 不具合が発生しやすいソフトウェアコンポーネントを特定し、優先順位を付ける。
- 適切なテストツールの使用

結論

以上で、ソフトウェアのアドホックテストに関するチュートリアルの概要を終わります。アドホックテストとは何か、アドホックテストの種類、機能、手法、利点、欠点、時間、およびベストプラクティスについて説明することから始めました。

これにより、ソフトウェアのアドホックテストに関する詳細な知識が得られます。理解を深め、自分の視野を広げるために、これまで学んだことを実践し、ソフトウェアテストに関連する他のトピックを探り続けることが賢明です。

翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。