A logo of a software company

Description automatically generated

**​**

**[AIC xVista Ecosystem – TestVista]**

**​Requirement Specifications**

**​For TestVista**

**​Vietnam, May  2024**

**​Approval Page**

​The endorsement on this document by authorized [Customer Name] representative indicates [Customer Name] and FPT’s agreement on the “[Project Name] Requirement Specifications” document.

​

|  |  |  |  |
| --- | --- | --- | --- |
| **​Prepared  by :** | ​ (FPT)  ​Business Analyst | ​Signature: | ​  ​\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **​** | ​ | ​Date: | ​  ​\_\_\_\_  /\_\_\_\_   / \_\_\_\_ |
| **​** | **​** | ​ | ​ |
| **​** | **​** | ​ | ​ |
| **​Reviewed  by :** | ​(FPT)  ​Project Manager | ​Signature: | ​  ​\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **​** | **​** | ​Date: | ​  ​\_\_\_\_  /\_\_\_\_   / \_\_\_\_ |
| **​** | **​** | ​ | ​ |
| **​Supported by:** | ​ (Customer Name)  ​ | ​Signature: | ​  ​\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **​** | ​ | ​Date: | ​  ​\_\_\_\_  /\_\_\_\_   / \_\_\_\_ |
| **​** | ​ | ​ | ​ |
| **​** | ​ | ​ | ​ |
| **​Approved by :** | ​ (Customer Name)  **​** | ​Signature: | ​  ​\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **​** | **​** | ​Date: | ​  ​\_\_\_\_  /\_\_\_\_   / \_\_\_\_ |

Table of Contents

[1. Introduction 4](#_Toc167098248)

[1.2 Overview 4](#_Toc167098249)

[Software Requirements Document for TestVista 4](#_Toc167098250)

[2. High-level requirements 6](#_Toc167098251)

[2.1 Object Relationship Diagram 6](#_Toc167098252)

[2.2 Object Description: 6](#_Toc167098253)

[**Data Objects:** 6](#_Toc167098254)

[**Actor Objects** 7](#_Toc167098255)

[**External System Objects** 7](#_Toc167098256)

[2.3 State Transition: 8](#_Toc167098257)

[2.4 Workflow 8](#_Toc167098258)

[***Detailed User Workflow for TestVista:*** 8](#_Toc167098259)

[2.5 Use Case 11](#_Toc167098260)

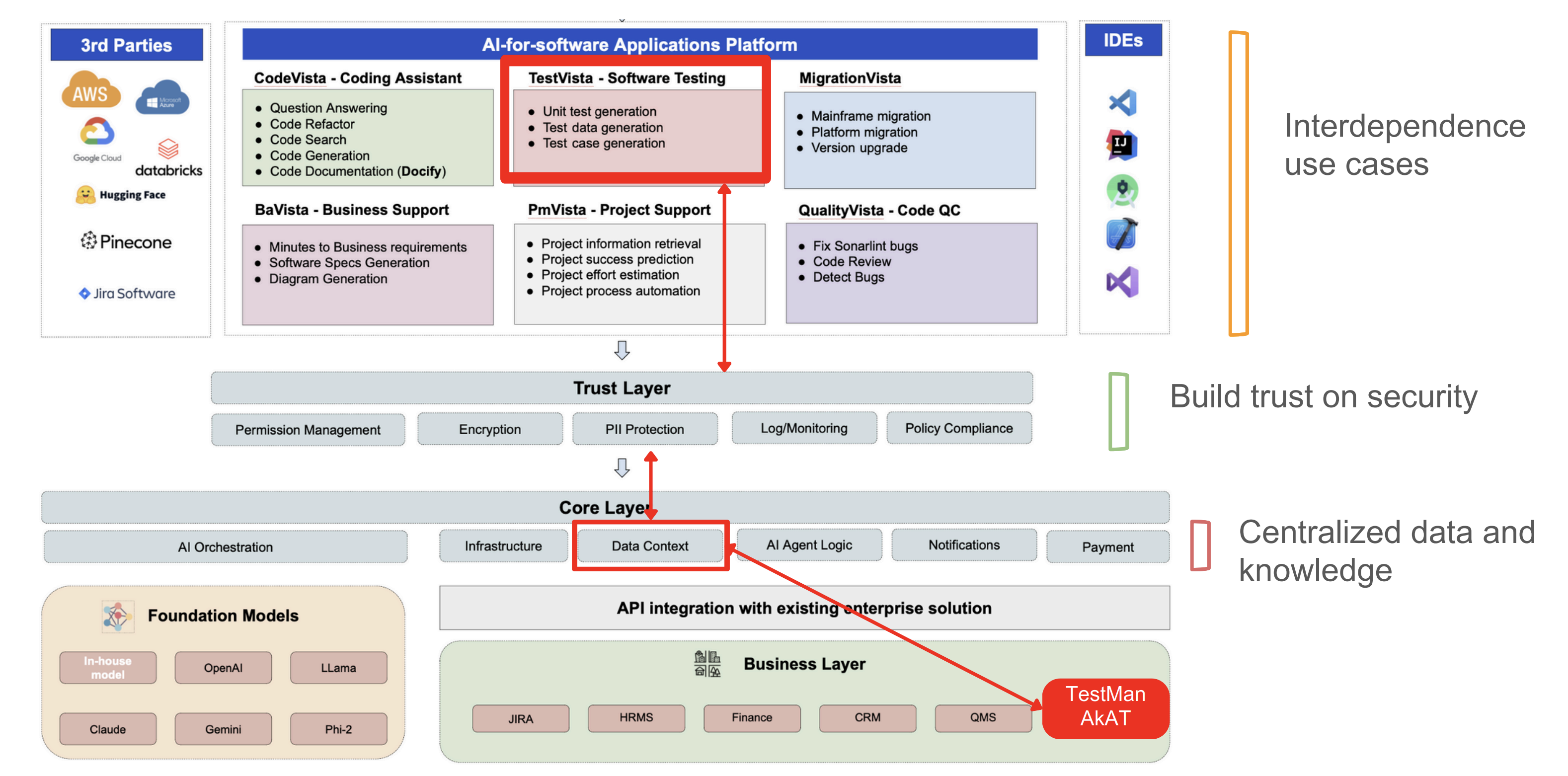
[2.6 Permission Matrix 12](#_Toc167098261)

[3. Use Case Specifications 13](#_Toc167098262)

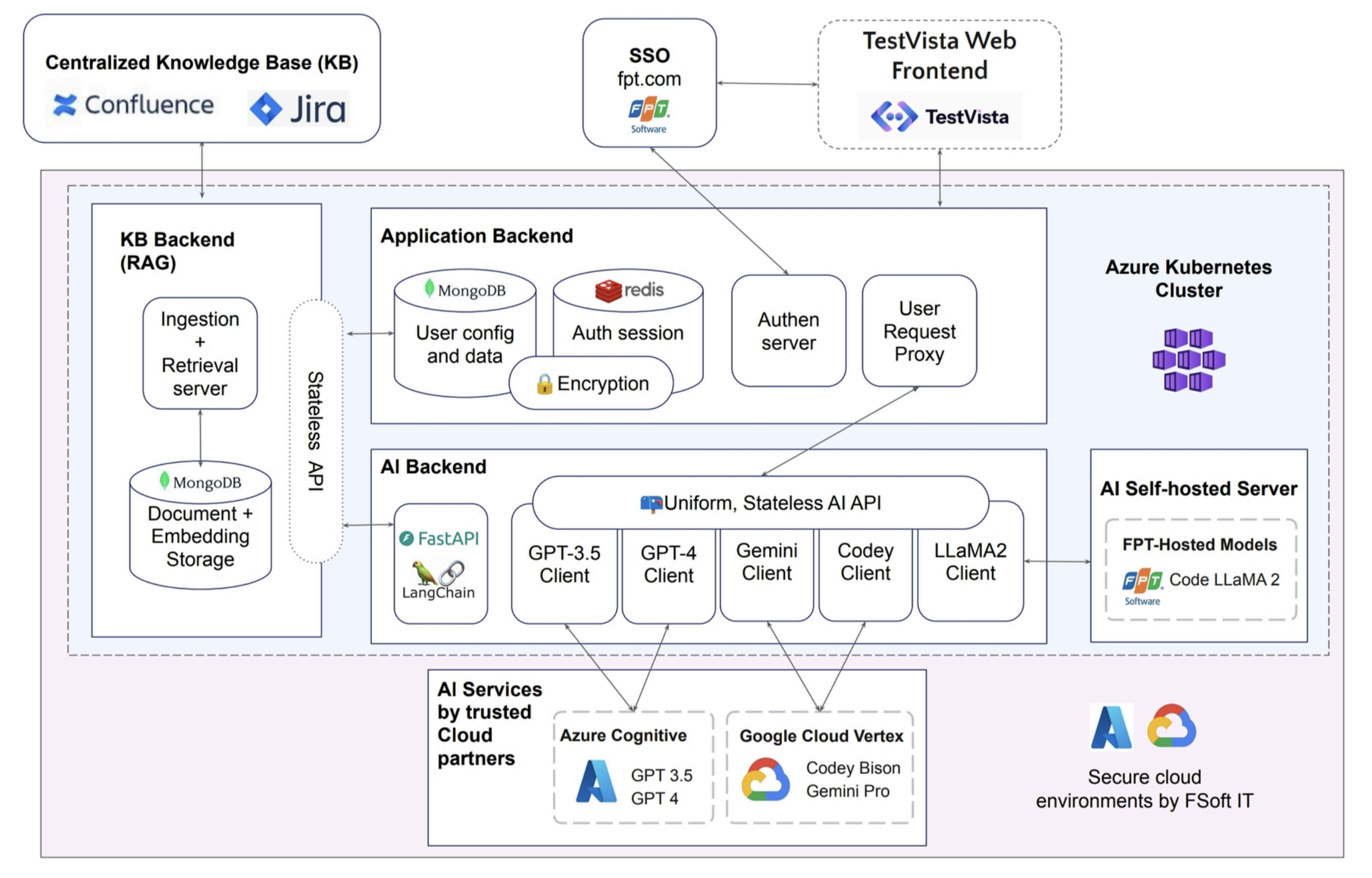
[4. Appendix 21](#_Toc167098263)

# 1. Introduction

## 1.0 xVista Ecosystem Overview



## 1.1 TestVista Architecture Overview



## 1.2 Overview

### Software Requirements Document for TestVista

#### **1. Objective**

Enhance the productivity of software testers and the testing process by leveraging AI to automate the generation, prioritization, and planning of test cases based on software requirement specifications (SRS). The goal is to demonstrate quick wins and prove that AI can significantly improve the quality of work and speed of delivery.

#### **2. System Overview**

TestVista is an AI-powered tool designed to automate key aspects of the software testing process. It focuses on generating, prioritizing, and planning test cases, thereby enabling testers to concentrate on critical testing tasks. The system will integrate seamlessly with existing tools and provide a user-friendly interface for interaction.

#### **3. Functional Requirements**

##### 3.1 Data Input and Integration

* **Objective**: Allow testers to upload key documents for processing.
* **Requirements**:
  + Users must be able to upload requirement documents (SRS), past test cases, and release notes.
  + The system should integrate seamlessly with existing testing tools like TestMan or AkAT.

##### 3.2 Automated Test Case Generation

* **Objective**: Automate the generation of test cases based on provided documents.
* **Requirements**:
  + The system should analyze acceptance criteria from SRS and historical data to suggest relevant and prioritized test cases.
  + Testers should have the ability to approve, reject, or edit the generated test cases.

##### 3.3 Release-Based Test Planning

* **Objective**: Automate test planning based on the latest release notes.
* **Requirements**:
  + The system should trigger test planning upon deployment.
  + It should suggest a comprehensive test plan aligned with the latest release notes.

##### 3.4 Test Plan Export

* **Objective**: Facilitate easy sharing and documentation of test plans.
* **Requirements**:
  + The system should allow export of finalized test cases into Excel or Docs formats.

##### 3.5 Test Data Generation

* **Objective**: Generate relevant test data for each test case.
* **Requirements**:
  + The system should generate test data that is consistent with each test case to enhance test accuracy and relevance.

#### **4. Front-End Requirements**

##### 4.1 Web Application

* **Objective**: Provide a user-friendly interface.
* **Requirements**:
  + The system should be a web application.
  + The interface should be intuitive and easy to navigate.

##### 4.2 Data Upload Capability

* **Objective**: Enable users to upload documents easily.
* **Requirements**:
  + The interface should support easy upload of relevant documents for processing.

##### 4.3 Interactive Feedback System

* **Objective**: Allow users to interact with and modify system outputs.
* **Requirements**:
  + Users should be able to view, interact with, and modify the steps and outputs generated by TestVista.

#### **5. Non-Functional Requirements**

##### 5.1 Authentication

* **Objective**: Ensure secure user authentication.
* **Requirements**:
  + The system should integrate with Fsoft's SAM/SSO for secure user authentication.

##### 5.2 Data Analytics

* **Objective**: Collect and analyze user interaction data for performance improvement.
* **Requirements**:
  + The system should collect and store data on user interactions, such as clicks, edits, approvals/rejections, and test cases generated.

##### 5.3 UI/UX Requirements

* **Objective**: Align the look and few of the products according to xVista’s brand guideline for consistent and professional experience for end users
* **Requirements**:
  + The system should reflect the UI/UX standard defined by xVista’s PM. Material for design will be provided by the project owner.

#### **6. Business Value**

* **Increased Productivity**: Reduces manual test case generation and planning time.
* **Enhanced Accuracy**: Minimizes human errors by leveraging AI for detailed analysis of SRS and historical data.
* **Improved Test Coverage**: Ensures all critical test scenarios are covered through systematic test plan generation.

# 2. High-level requirements

## 2.1 Object Relationship Diagram A screenshot of a computer screen Description automatically generated

## 

## 2.2 Object Description

### **Data Objects:**

Here's a table listing all data objects within the TestVista software system based on the provided Software Requirements Document:

|  |  |  |
| --- | --- | --- |
| **Item #** | **Object** | **Description** |
| 1 | Requirement Documents (SRS) | Documents containing software requirements specifications uploaded by users. |
| 2 | Past Test Cases | Historical test case data uploaded by users for reference in automated test case generation. |
| 3 | Release Notes | Documents detailing the latest software release notes uploaded by users. |
| 4 | Generated Test Cases | Test cases automatically generated by the system based on SRS and historical data. |
| 5 | Test Plans | Comprehensive test plans generated by the system based on the latest release notes. |
| 6 | Exported Test Plans | Finalized test plans that users can export into Excel or Docs formats. |
| 7 | Test Data | Relevant data generated for each test case to enhance test accuracy and relevance. |
| 8 | User Interaction Data | Data on user interactions, including clicks, edits, approvals/rejections, and test cases generated. |
| 9 | User Authentication Data | Data related to user authentication, integrated with Fsoft's SAM/SSO. |

### **Actor Objects**

The table listing all actors that directly interact with the software, along with a brief description of their interactions:

|  |  |  |
| --- | --- | --- |
| **Item #** | **Object** | **Description** |
| 1 | Testers | Upload requirement documents (SRS), past test cases, and release notes. |
| 2 | Testers | Approve, reject, or edit the generated test cases. |
| 3 | Testers | Trigger test plan generation upon deployment. |
| 4 | Testers | Export finalized test cases into Excel or Docs formats. |
| 5 | Testers | Interact with the web-based interface to manage testing tasks. |
| 6 | Testers | Upload SRS, past test cases, and release notes via the web interface. |
| 7 | Testers | Provide feedback by approving, rejecting, or editing test cases and plans. |
| 8 | Testers | Authenticate using Fsoft's SAM/SSO system. |
| 9 | System Admins | Collect and store data on user interactions for performance analytics. |

These entries capture the primary interactions testers and system administrators have with TestVista based on the functional and non-functional requirements outlined in the document.

### **External System Objects**

Here's a table listing external systems or services based on the requirements plan in the Software Requirements Document for TestVista:

|  |  |  |
| --- | --- | --- |
| **Item #** | **Object** | **Description** |
| 1 | TestMan | Existing system that TestVista must integrate with for integration. |
| 2 | AkAT | Existing system that TestVista must integrate with for integration. |
| 3 | Fsoft's SAM/SSO | A secure authentication service that TestVista must integrate with to ensure user authentication. |

## 2.3 State Transition:

## 2.4 Workflow

This detailed workflow outlines the steps and conditions under which testers interact with TestVista to complete various tasks, from authentication to test data generation and feedback. Each stage ensures that the system's functionalities are efficiently utilized, leading to enhanced productivity and software quality.

### Detailed Workflow for TestVista:

#### Stage 1: User Authentication

**Actors:** Testers, System Admins

1. **Action:** User navigates to the TestVista web application.
   * **Condition:** User is not authenticated.
   * **Next Step:** Display login screen.
2. **Action:** User enters credentials and submits.
   * **Condition:** Credentials are valid.
   * **Next Step:** Redirect to the TestVista dashboard.
   * **Condition:** Credentials are invalid.
   * **Next Step:** Display error message and prompt for re-entry.

#### Stage 2: Data Input and Integration

**Actors:** Testers

1. **Action:** User uploads requirement documents (SRS), past test cases, and release notes.
   * **Condition:** Files are in an acceptable format and size.
   * **Next Step:** System processes the uploaded documents.
   * **Condition:** Files are not in an acceptable format or size.
   * **Next Step:** Display error message and prompt for correct file upload.
2. **Action:** System integrates uploaded documents with existing systems like TestMan or AkAT.
   * **Condition:** Integration is successful.
   * **Next Step:** Notify user of successful integration.
   * **Condition:** Integration fails.
   * **Next Step:** Display error message and provide troubleshooting steps.

#### Stage 3: Automated Test Case Generation

**Actors:** Testers

1. **Action:** System analyzes the SRS and historical test case data to suggest test cases.
   * **Condition:** Analysis is complete.
   * **Next Step:** Display generated test cases to the user.
2. **Action:** User reviews the generated test cases.
   * **Condition:** User approves the test cases.
   * **Next Step:** Mark test cases as approved and store them.
   * **Condition:** User rejects the test cases.
   * **Next Step:** Mark test cases as rejected and prompt for feedback.
   * **Condition:** User edits the test cases.
   * **Next Step:** Save the edited test cases.

#### Stage 4: Release-Based Test Planning

**Actors:** Testers

1. **Action:** User triggers test plan generation upon deployment.
   * **Condition:** Deployment is successful.
   * **Next Step:** System generates a test plan based on the latest release notes.
   * **Condition:** Deployment fails.
   * **Next Step:** Display error message and halt test plan generation.
2. **Action:** System suggests a test plan aligned with the latest release notes.
   * **Condition:** Test plan is generated.
   * **Next Step:** Display the suggested test plan to the user.
3. **Action:** User reviews the suggested test plan.
   * **Condition:** User approves the test plan.
   * **Next Step:** Mark the test plan as approved and store it.
   * **Condition:** User rejects the test plan.
   * **Next Step:** Mark the test plan as rejected and prompt for feedback.
   * **Condition:** User edits the test plan.
   * **Next Step:** Save the edited test plan.

#### Stage 5: Test Plan Export

**Actors:** Testers

1. **Action:** User selects finalized test cases to export.
   * **Condition:** Test cases are selected.
   * **Next Step:** User chooses the export format (Excel or Docs).
2. **Action:** User confirms the export.
   * **Condition:** Export is successful.
   * **Next Step:** Notify user of successful export and provide download link.
   * **Condition:** Export fails.
   * **Next Step:** Display error message and prompt for retry.

#### Stage 6: Test Data Generation

**Actors:** Testers

1. **Action:** System generates relevant test data for each test case.
   * **Condition:** Test data is generated.
   * **Next Step:** Attach test data to corresponding test cases and notify the user.
   * **Condition:** Test data generation fails.
   * **Next Step:** Display error message and provide troubleshooting steps.

#### Stage 7: Interactive Feedback System

**Actors:** Testers

1. **Action:** User interacts with the web-based interface to provide feedback on test cases and plans.
   * **Condition:** User provides feedback (approve/reject/edit).
   * **Next Step:** System logs the feedback and updates the test cases or plans accordingly.

#### Stage 8: Data Analytics

**Actors:** System Admins

1. **Action:** System collects and stores data on user interactions.
   * **Condition:** Data collection is successful.
   * **Next Step:** Store data for performance analytics and system improvement.
   * **Condition:** Data collection fails.
   * **Next Step:** Display error message and prompt for retry.

## 2.5 Use Case Table

This table captures the primary use cases for testers and system administrators, detailing their interactions with TestVista based on the outlined requirements. Everything listed as **Phase 2** is to be delivered in later phases of the project.

|  |  |  |  |
| --- | --- | --- | --- |
| **UC\_ID** | **UC\_Name** | **Description** | **Sprint** |
| UC\_01 | User Login | Testers authenticate using Fsoft's SAM/SSO system to gain access to TestVista. |  |
| UC\_02 | Upload Documents | Testers upload requirement documents (SRS), past test cases, and release notes for automated processing. |  |
| UC\_03 | Document Integration **(Phase 2)** | The system integrates uploaded documents with existing systems like TestMan or AkAT. |  |
| UC\_04 | Generate Test Cases (UT, IT, ST) 🡪 **Thien** Talk with **Tung** to clear | The system analyzes SRS and historical test case data to generate relevant and prioritized test cases. |  |
| UC\_05 | Review Generated Test Cases | Testers review, approve, reject, or edit the generated test cases to ensure accuracy and relevance. |  |
| UC\_06 | Trigger Test Plan Generation | Testers trigger the generation of a comprehensive test plan based on the latest release notes upon deployment. |  |
| UC\_07 | Review Test Plan **(Phase 2)** | Testers review the suggested test plan to ensure it aligns with the latest release notes. |  |
| UC\_08 | Export Test Plan | Testers export finalized test cases and test plans into **Excel** or **Docs** formats for easy sharing and documentation. |  |
| UC\_09 | Generate Test Data **(Phase 2)** | The system generates relevant test data for each test case to enhance test accuracy and relevance. |  |
| UC\_10 | Web Interface Interaction | Testers interact with the web-based interface to manage testing tasks, including viewing, uploading, and providing feedback on test cases and plans. |  |
| UC\_11 | Provide Feedback | Testers provide feedback by approving, rejecting, or editing the generated test cases and plans through the interactive feedback system. |  |
| UC\_12 | Data Collection | System Admins collect and store data on user interactions, including **clicks**, **edits**, **approvals**/**rejections**, and test cases generated, for performance analytics. |  |

## 2.6 Permission Matrix

|  |  |  |
| --- | --- | --- |
| UC Name | Testers | System Admins |
| UC\_01: User Login | O | X |
| UC\_02: Upload Documents | O | X |
| UC\_03: Document Integration | O | X |
| UC\_04: Generate Test Cases | O | X |
| UC\_05: Review Generated Test Cases | O\*\* | X |
| UC\_06: Trigger Test Plan Generation | O | X |
| UC\_07: Review Test Plan | O\*\* | X |
| UC\_08: Export Test Plan | O\*\* | X |
| UC\_09: Generate Test Data | O | X |
| UC\_10: Web Interface Interaction | O | X |
| UC\_11: Provide Feedback | O | X |
| UC\_12: Data Collection | X | O |

**Explanation**:

* "O" means that the actor has permission to perform the corresponding function.
* "O\*\*" means that the actor has permission to perform the corresponding function that the actor created.
* "X" means that the actor does not have permission to perform the corresponding function.

For more detailed information about what each actor can do within each use case, please refer to the corresponding use case descriptions provided.

# 3. Use Case Specifications

## Use Case 1: User login

**Activity Diagram**A diagram of a system response

Description automatically generated

|  |  |
| --- | --- |
| **Specification** | **Details** |
| **Objective** | Authenticate users via Fsoft's SAM/SSO system to grant access to TestVista. |
| **Actor(s)** | Testers, System Admins |
| **Trigger** | User navigates to the TestVista web application. |
| **Pre-condition** | User is not authenticated. |
| **User-Workflow** | 1. User navigates to the TestVista web application.  2. User enters credentials and submits.  3. If credentials are valid, redirect to the TestVista dashboard  4. If credentials are invalid, display error message and prompt for re-entry. |
| **Post-condition** | User is authenticated and redirected to the TestVista dashboard. |
| **Acceptance Criteria** | 1. User is presented with a login screen if not authenticated.  2. Valid credentials grant access to the TestVista dashboard  3. Invalid credentials result in an error message and re-entry prompt. |

## Use Case 2: Upload Documents

A diagram of a file uploading verification process

Description automatically generated

|  |  |
| --- | --- |
| **Specification** | **Details** |
| **Objective** | Allow testers to upload requirement documents (SRS), past test cases, and release notes for automated processing. |
| **Actor(s)** | Testers |
| **Trigger** | Tester initiates the upload process for documents. |
| **Pre-condition** | The tester must be authenticated and have the documents in acceptable formats and sizes. |
| **User Workflow** | 1. Tester navigates to the upload section.  2. Tester selects and uploads requirement documents (SRS), past test cases, and release notes.  3. System checks the format and size of the uploaded files.  4. If files are acceptable, the system processes the documents.  5. If files are not acceptable, an error message is displayed prompting for correct file upload. |
| **Post-condition** | Documents are either successfully processed or an error message is displayed for correction. |
| **Acceptance Criteria** | 1. The system correctly identifies and processes documents in acceptable formats and sizes.  2. Error messages are displayed for files that do not meet the format or size requirements.  3. Successful integration of documents with existing systems like TestMan or AkAT (phase 2) |

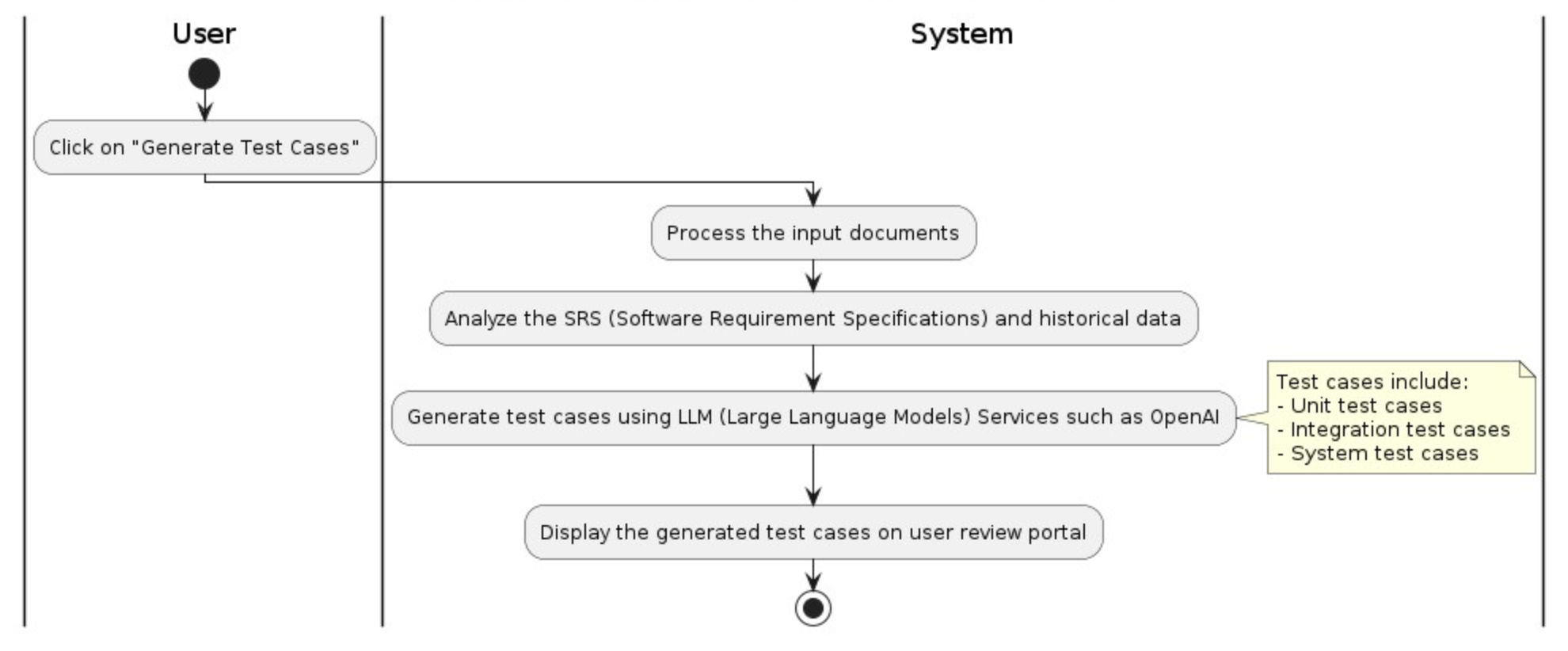
## Use Case 3: Document Integration (Phase 2 – we don’t need TestMan for quick win)

A diagram of a system

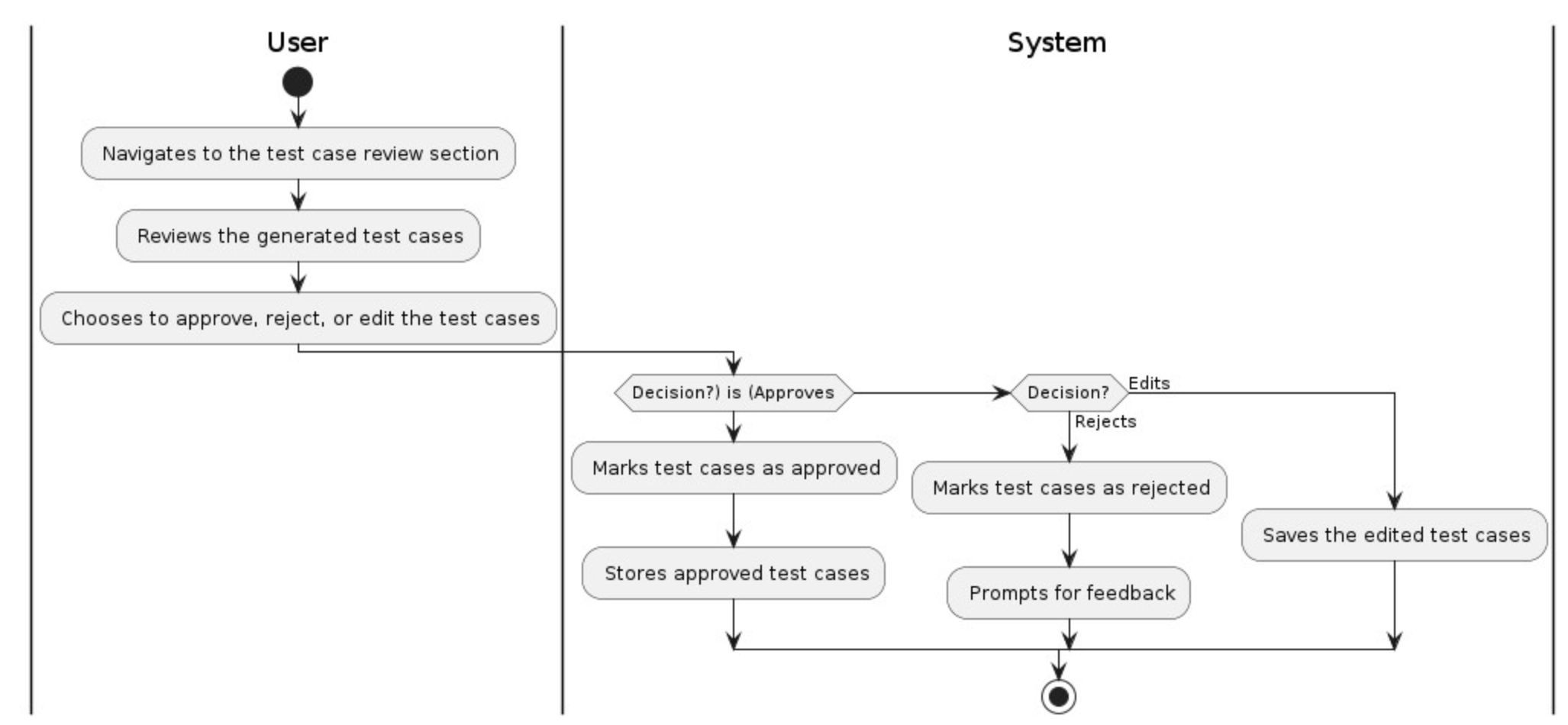
Description automatically generated

|  |  |
| --- | --- |
| **Specification** | **Details** |
| **Objective** | To integrate uploaded documents with existing systems like TestMan |
| **Actor(s)** | Testers |
| **Trigger** | User select TestMan section to retrieve requirement documents (SRS), past test cases, and release notes. |
| **Pre-condition** | User is authenticated and has uploaded documents in acceptable format and size. |
| **User-Workflow** | 1. User navigates to the TestMan section  2. System pull documents from TestMan including:  requirement documents (SRS), past test cases, and release notes.  3. System notifies the user of the integration status (success or failure).  4. System display the retrieved document for user to review |
| **Post-condition** | The documents are successfully integrated with existing systems, or an error message is displayed if the integration fails. |
| **Acceptance Criteria** | 1. The system should successfully integrate the documents with TestMan or AkAT  2. The system should notify the user of the successful document retrieval  3. If retrieval fails, the system should display an error message and provide troubleshooting steps.  4. If retrieval success, system successfully display the documents content to the user |

## Use Case 4: Generate Test Case

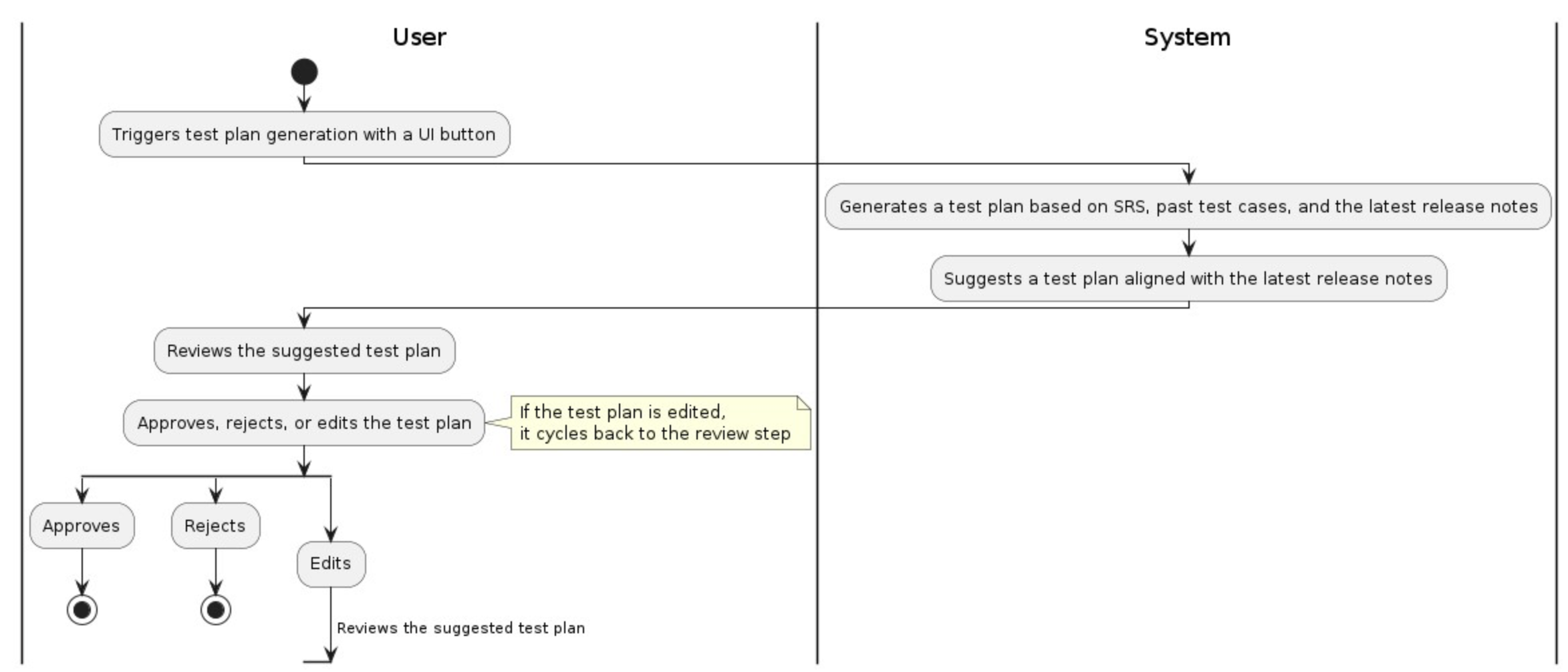
****

|  |  |
| --- | --- |
| **Specification** | **Details** |
| **Objective** | Automatically generate relevant and prioritized test cases based on SRS and historical test case data. |
| **Actor(s)** | Testers |
| **Trigger** | User uploads SRS and past test cases, and the system initiates the analysis. |
| **Pre-condition** | User is authenticated and has uploaded valid SRS and historical test case data. |
| **User-Workflow** | 1. User click on “Generate Test Cases”  2. System processes the input documents  3. System analyzes the SRS and historical data.  4. System generate test cases using LLM Services such as OpenAI:  - Unit test cases  - Integration test cases  - System test cases  5. System display the generated test cases to user review portal |
| **Post-condition** | Generated test cases are displayed to the user for review. |
| **Acceptance Criteria** | 1. The UI must show the “Generate Test Cases” clearly  2. System successfully processes and ingest the uploaded documents  3. System generates relevant test cases, including Unit test cases, Integration test cases, and System test cases  4. Generated test cases are displayed on review portal |

Use Case 5: Review Generated Test Cases  


|  |  |
| --- | --- |
| **Specification** | **Details** |
| **Objective** | To allow testers to review, approve, reject, or edit generated test cases to ensure they meet the required standards and are relevant. |
| **Actor(s)** | Testers |
| **Trigger** | System generates and displays test cases based on the SRS and historical data. |
| **Pre-condition** | Test cases have been generated by the system and are ready for review. |
| **User-Workflow** | 1. User navigates to the test case review section.  2. User reviews the generated test cases.  3. User either approves, rejects, or edits the test cases  4. If approved, test cases are marked as approved and stored.  5. If rejected, test cases are marked as rejected and feedback is prompted.  6. If edited, the edited test cases are saved. |
| **Post-condition** | Test cases are either approved, rejected with feedback, or edited and saved. |
| **Acceptance Criteria** | 1. Test cases are displayed for review.  2. User can approve, reject, or edit test cases.  3. System correctly logs the user's actions (approval, rejection with feedback, or edits).  4. Approved test cases are stored.  5. Rejected test cases prompt for feedback and are logged as rejected.  6. Edited test cases are saved correctly. |

## Use Case 6: Trigger Test Plan Generation



|  |  |
| --- | --- |
| **Specification** | **Details** |
| **Objective** | Generate a comprehensive test plan based on the latest release notes upon deployment. |
| **Actor(s)** | Testers |
| **Trigger** | User triggers test plan generation upon deployment. |
| **Pre-condition** | User is authenticated and deployment is successful. |
| **User-Workflow** | 1. User triggers test plan generation with an UI Button  2. System generates a test plan based on SRS, past test cases, the latest release notes.  3. System suggests a test plan aligned with the latest release notes.  4. User reviews the suggested test plan.  5. User approves, rejects, or edits the test plan. |
| **Post-condition** | Test plan is generated, reviewed, and either approved, rejected, or edited and saved. |
| **Acceptance Criteria** | 1. System generates a test plan based on button event.  2. Suggested test plan is displayed to the user.  3. User can review and either approve, reject, or edit the test plan.  4. Approved or edited test plan is stored in the system storage. |

## Use Case 7: Review Test Plan (phase 2)

|  |  |
| --- | --- |
| **Specification** | **Details** |
| **Objective** | Ensure the suggested test plan aligns with the latest release notes. |
| **Actor(s)** | Testers |
| **Trigger** | User navigates to the suggested test plan review section. |
| **Pre-condition** | User is authenticated and a suggested test plan has been generated by the system. |
| **User-Workflow** | 1. User reviews the suggested test plan.  2. User approves, rejects, or edits the test plan.  - If approved, the system marks the test plan as approved and stores it.  - If rejected, the system marks the test plan as rejected and prompts for feedback.  - If edited, the system saves the edited test plan. |
| **Post-condition** | The test plan is either approved, rejected with feedback, or edited and saved. |
| **Acceptance Criteria** | 1. User can successfully review the suggested test plan.  2. User can approve, reject, or edit the test plan.  3. System correctly updates the status of the test plan based on user action.  4. Feedback is logged if the test plan is rejected. |

Use Case 8: Export Test Plan (include UT, IT & ST)   
A screenshot of a computer program

Description automatically generated

|  |  |
| --- | --- |
| **Specification** | **Details** |
| **Objective** | Allow testers to export finalized test cases and test plans into Excel or Docs formats for sharing and documentation. |
| **Actor(s)** | Testers |
| **Trigger** | Tester selects the option to export finalized test cases and test plans. |
| **Pre-condition** | Test cases and test plans must be finalized and saved in the system. |
| **User-Workflow** | 1. User selects finalized test cases to export with an UI Button  2. User chooses the export format (only .xlsx is supported for now)  3. User confirms the export  4. System generates the export file  5. User downloads the export file with an UI Button |
| **Post-condition** | Exported test cases and test plans are available in the chosen format for download. |
| **Acceptance Criteria** | 1. User can successfully select and export test cases and test plans  2. Exported file is in the correct format (Excel or .docx).  3. User receives a download link for the exported file  4. System handles errors gracefully and provides appropriate feedback for retry. |

**See example outputs of an test plan:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S/N** | **Test Summary** | **Data/Pre-condition** | **Steps** | **Expected Result** |
| **Unit Test Case** | | | | |
| 1 | Perform End to End Testing using payment method = 3DS1 Credit Card - MM | Campaign: Automated Testing - Amazing Hokkaido  Fare Class: Economy  Passenger Composition: 1 Adult Insider Account: ks\_xxxxx@xxx.com ; xxxxxx@1 Card Holder name: 3DS\_V1\_CHALLENGE\_IDENTIFIED.AUTHORISED Card Number: 5149000000000001 Date of Expiry: 2 months from today CVV: 123" | "1. Launch a web browser 2. Go to https://uat-xxxxxxxxxx.com/ 3. try username/ password valid and invalid " | verify username login successfully and invalid data not successfully |
| **Integration Test Case** | | | | |
| 1 | Perform End to End Testing using payment method = 3DS1 Credit Card - MM | "Campaign: Automated Testing - Amazing Hokkaido Fare Class: Economy Passenger Composition: 1 Adult Insider Account: ks\_xxxxx@xxx.com ; xxxxxx@1 Card Holder name: 3DS\_V1\_CHALLENGE\_IDENTIFIED.AUTHORISED Card Number: 5149000000000001 Date of Expiry: 2 months from today CVV: 123" | "1. Launch a web browser 2. Go to https://uat-xxxxxxxxxx.com/ 3. Modify campain, add one item new caimpain faire class 1 4. Navigate to https://uat-xxxxxxxxxx.com/ 5. Locate the Current Flights section 6. Select campaign 7. Click View Campaign 8. Locate the desired fare class 1 9. Add the passenger composition 10. Click the Make Pledge button at the footer 11. Login to Scoot Insider account " | 8. Fair class 1 enable and be able to applied |
| **System Test Cases: Booking Flow Feature** | | | | |
| 1 | Perform End to End Testing using payment method = 3DS1 Credit Card - MM | "Campaign: Automated Testing - Amazing Hokkaido Fare Class: Economy Passenger Composition: 1 Adult Insider Account: ks\_xxxxx@xxx.com ; xxxxxx@1 Card Holder name: 3DS\_V1\_CHALLENGE\_IDENTIFIED.AUTHORISED Card Number: 5149000000000001 Date of Expiry: 2 months from today CVV: 123" | "1. Launch a web browser 2. Navigate to https://uat-xxxxxxxxxx.com/ 3. Locate the Current Flights section 4. Select campaign 5. Click View Campaign 6. Locate the desired fare class 7. Add the passenger composition 8. Click the Make Pledge button at the footer 9. Login to account 10. On passenger details page or Registration page, tick the ""<Scoot Insider member name> is travelling on this fare class"" checkbox 11. If the form is incomplete, fill-up the missing details. 12. Click the Make Pledge button at the footer 13. Fill-up the Card Holder name field 14. Fill-up the Card Number field 15. Select the Date of Expiry 16. Fill-up the CVV field 17. If there's any other field left empty, fill it up. 18. Tick the terms and condition checkbox 19. Click the Make Pledge button at the footer 20. Click 'OK' in the challenge validator / OTP page" | "1. Pledge is successful. 2. User is redirected to Confirmation Page." |
| 2 | Perform End to End Testing using payment method = 3DS1 Credit Card - VV | "Campaign: Automated Testing - Amazing Hokkaido Fare Class: Early Bird Passenger Composition: 1 Adult Insider Account: any Card Holder name: 3DS\_V1\_CHALLENGE\_IDENTIFIED.AUTHORISED Card Number: 4006090000000015 Date of Expiry: 2 months from today CVV: 123" | "1. Launch a web browser 2. Navigate to https://uat-xxxxxxxxxx.com/ 3. Locate the Current Flights section 4. Select campaign 5. Click View Campaign 6. Locate the desired fare class 7. Add the passenger composition 8. Click the Make Pledge button at the footer 9. Login to Scoot Insider account 10. On passenger details page or Registration page, tick the ""<Scoot Insider member name> is travelling on this fare class"" checkbox 11. If the form is incomplete, fill-up the missing details. 12. Click the Make Pledge button at the footer 13. Fill-up the Card Holder name field 14. Fill-up the Card Number field 15. Select the Date of Expiry 16. Fill-up the CVV field 17. If there's any other field left empty, fill it up. 18. Tick the terms and condition checkbox 19. Click the Make Pledge button at the footer 20. Click 'OK' in the challenge validator / OTP page" | "1. Pledge is successful. 2. User is redirected to Confirmation Page." |

## Use Case 9: Generate Test Data

A screenshot of a chat

Description automatically generated

|  |  |
| --- | --- |
| **Specification** | **Details** |
| **Objective** | Generate relevant test data for each test case to enhance test accuracy and relevance. |
| **Actor(s)** | Testers |
| **Trigger** | User initiates the test data generation process. |
| **Pre-condition** | Test cases have been finalized and approved. |
| **User-Workflow** | 1. User selects a specifc test case to generate test data.  2. System processes the test cases to generate relevant test data.  3. If data generation is successful, the system attaches the test data to the corresponding test cases and notifies the user.  4. If data generation fails, the system displays an error message and provides troubleshooting steps. |
| **Post-condition** | Test data is attached to the corresponding test cases, or an error message is displayed. |
| **Acceptance Criteria** | 1. Test data is accurately generated and attached to the corresponding test cases.  2. User is notified of the successful generation and attachment of test data.  3. In case of failure, an appropriate error message is displayed with troubleshooting steps. |

## Use Case 10: Web Interface Interaction

**A diagram with orange arrows

Description automatically generated**

|  |  |
| --- | --- |
| **Specification** | **Details** |
| **Objective** | To enable testers to manage testing tasks through a web-based interface, including viewing, uploading, and providing feedback on test cases and plans. |
| **Actor(s)** | Testers, System Admins |
| **Trigger** | User navigates to the TestVista web application. |
| **Pre-condition** | User is not authenticated. |
| **User-Workflow** | 1. User navigates to the TestVista web application.  2. User enters credentials and submits (validation is handled by FPT SSO)  3. User uploads requirement documents (SRS), past test cases, and release notes.  4. System integrates uploaded documents with existing systems.  5. System analyzes the SRS and historical test case data to suggest test cases.  6. User reviews the generated test cases.  7. User triggers test plan generation upon deployment.  8. System suggests a test plan aligned with the latest release notes  9. User reviews the suggested test plan.  10. User selects finalized test cases to export  11. User confirms the export.  12. System generates relevant test data for each test case.  13. User interacts with the web-based interface to provide feedback on test cases and plans.  14. System collects and stores data on user interactions. |
| **Post-condition** | Test cases, plans, and feedback are stored and managed within the system. |
| **Acceptance Criteria** | 1. User can successfully log in and access the dashboard.  2. Users can upload documents in acceptable formats and sizes.  3. System successfully integrates uploaded documents  4. System generates and displays test cases and plans accurately  5. Users can provide feedback and the system updates accordingly.  6. Test data is generated and attached to test cases  7. Data on user interactions is collected and stored successfully. |

## Use Case 11: User Provide Feedback (Phase 2)

|  |  |
| --- | --- |
| **Specification** | **Details** |
| **Objective** | Allow testers to provide feedback on generated test cases and plans by approving, rejecting, or editing them. |
| **Actor(s)** | Testers |
| **Trigger** | Tester decides to provide feedback on the generated test cases or test plans. |
| **Pre-condition** | Test cases or test plans have been generated and are available for review. |
| **User Workflow** | 1. User navigates to the feedback section of the TestVista web application.  2. User reviews the generated test cases or test plans.  3. User provides feedback by approving, rejecting, or editing the test cases or plans.  4. System logs the feedback and updates the test cases or plans accordingly. |
| **Post-condition** | Feedback is logged, and the test cases or plans are updated based on the user's input. |
| **Acceptance Criteria** | 1. The system must allow users to approve, reject, or edit test cases and plans  2. Feedback must be accurately logged by the system  3. Test cases and plans must be updated according to the feedback provided  4. Users must receive confirmation that their feedback has been successfully logged and applied. |

## Use Case 12: Data Collection

|  |  |
| --- | --- |
| **Specification** | **Details** |
| **Objective** | Collect and store data on user interactions for performance analytics and system improvement. |
| **Actor(s)** | System Admins |
| **Trigger** | Initiation of data collection process by the system. |
| **Pre-condition** | System is operational and user interactions are occurring. |
| **User-Workflow** | 1. System Admins configure data collection settings  2. System monitors and logs user interactions  3. System Admins review collected data for analysis. |
| **Post-condition** | Data on user interactions is successfully stored and available for performance analytics. |
| **Acceptance Criteria** | 1. Data collection settings can be configured by System Admins  2. User interactions are accurately logged  3. Collected data is accessible for analysis  4. System displays error messages and prompts for retry if data collection fails. |

Data requirements details (TBD)

# 4. Non-functional requirements