***Software Test Plan (STP) for Gita***

***Test plan identifier***

* ***Document Title: Gita Test Plan.***
* ***Version: 1.0***
* ***Date of Issue*: January 19, 2025**
* **Status: Draft**

***Approval Authority***

* **Reviewed By**: maias omar

***Change History***

|  |  |  |  |
| --- | --- | --- | --- |
| ***Version*** | ***Date*** | ***Description*** | ***Author*** |
| 1.0 | January 19,2025 | Initial version of test plan | Maias omar |

***Introduction***

**Introduction:** This test plan outlines the strategy, scope, objectives, and approach for automated testing of the Gita web application. The testing will focus on the following key features:

1. **Sign-In Page**: Automate the testing of the login functionality to ensure users can sign in successfully with valid credentials and are redirected to the appropriate page.
2. **Add New Project in Gita**: Automate the testing of the process of adding a new project in Gita, ensuring that the project creation works as expected when all required fields are entered correctly.

The primary goal of this testing effort is to ensure that the Gita web application’s sign-in functionality and the new project creation process work correctly and consistently under automated testing conditions.

.

***References***

* Gita source code
* Gita official documentation (API endpoints, UI guidelines
* GitHub new project creation workflow for comparison.

***Analysis of testing (what to test)***

1. **Test Items:**

* **Sign-In Page**: The functionality of the sign-in page, including username/password validation and successful login.
* **Add New Project**: The ability to create a new project by entering valid project details and ensuring the project is saved correctly.
* **UI Consistency**: The consistency and responsiveness of the user interface across different browsers (Chrome and Firefox).
* **Error Handling**: Proper error messages when invalid credentials are entered or required fields are missing during project creation.

1. **Test Scope:**

**In-Scope:**

* Verification of the sign-in functionality (valid and invalid logins).
* Adding a new project in Gitea, including filling out the form and confirming that the project is created.
* Cross-browser testing on Chrome and Firefox.
* UI validation and error handling for both the sign-in page and new project creation form.

.

**Out-of-Scope**:

* Testing of Gita integrations with third-party services.
* Performance under high-load conditions..

.

1. **Assumptions and constraints**

***Assumptions:***

* **User Will Have Correct Login Credentials**: We assume that the user has valid login information for Gita to test the sign-in page.
* **Correct Data for New Project**: We assume that when creating a new project, the user will provide all the necessary details such as the project title and description.

***Constraints:***

* **Limited Browser Support**: The tests will only be run on Chrome and Firefox, so it may not work on other browsers..
* **Network Connection**: The tests depend on having a stable network connection for accessing the Gita application.

***Features to Be Tested:***

1. ***Sign-In Functionality***

* ***Why this feature?***

*The sign-in feature is very important because it allows users to access their accounts securely. If this feature doesn't work, users can't use the rest of the application.*

* ***How it fits the application****:*

*The sign-in ensures that only the right people can access their accounts and projects, keeping their data safe.*

* ***What could go wrong?***

1. *The page might not load properly.*
2. *After logging in, users might not be taken to the dashboard.*

***2.*** ***Project Creation***

* ***Why this feature?***

*Creating a new project is one of the main things users do in Gitea. If it doesn’t work, users can’t manage their work effectively.*

* ***How it fits the application:***

*Gitea is designed for managing projects and repositories. This feature is a big part of how users interact with the application****.***

* ***What could go wrong?***

1. *The form might not work if required fields, like the project title, are left empty****.***
2. *The project might not appear in the project list after being created.*

**Test Approach**

***Testing Methodology:***

1. **Black-box Testing**: We will focus on the functional aspects of the UI and API without looking at the internal code**.**
2. **Regression Testing**: Ensure that changes made to the sign-in or project creation process do not break other parts of the application.
3. **Smoke Testing**: Perform quick tests to ensure that the core functionality (sign-in, project creation) is working.

***Test Strategy:***

* **UI Testing**: Selenium WebDriver will be used to automate UI tests for the sign-in and project creation workflows.
* **API Testing**: REST-assured will be used to test the RESTful API endpoints for login and project creation.

***Test Tools:***

* **Selenium WebDriver**: For automating the browser interaction for UI testing.
* **REST-assured**: For API testing
* **GitHub Actions**: For Continuous Integration (CI) to automate the test execution process on every change.

***Test Environment:***

The tests will be executed in a local environment where Gita will be running on the local machine or in a Docker container. Automated tests will run on Chrome and Firefox browsers via Selenium Grid in GitHub actions.

***Pass/Fail Criteria:***

***Sign-In:***

* *Pass: User is able to log in with valid credentials and redirected to the dashboard.*
* *Fail: Login page does not load, or user cannot log in with valid credentials.*

***Project Creation:***

* ***Pass****: A new project is created successfully when the title field is filled, and the project appears in the project list.*
* ***Fail****: The project creation fails when the title field is not filled, and the project is not created or saved in the project list.*

***Test Objectives:***

1. **Verify User Sign-In**: Ensure users can log in with valid credentials and are redirected to the dashboard; ensure login fails with invalid credentials.
2. **Project Creation Validation:** Confirm that a project can be created when the "Title" field is filled, and creation fails when the field is empty.
3. **UI Responsiveness:** Ensure all UI elements (buttons, input fields, navigation) are functional and responsive.
4. **API Functionality:** Test key API endpoints to ensure correct responses and functionality.
5. **Regression Testing:** Ensure new updates do not break existing functionality
6. **Cross-Browser Compatibility:** Confirm the platform works properly on both Chrome and Firefox.

.

* ***Metrics to be Collected:***
  + - * **Requirements Coverage:** Ensuring that all the features are adequately covered by test cases.

**Test Data Requirements:**

* A set of test users with varying roles (e.g., admin, member, guest).
* Sample project data, including project titles, descriptions, and repositories.
* Data for creating new projects, ensuring the title field is filled, and handling error cases when the title is empty.
* Test data for sign-in attempts with valid and invalid credentials.

**Suspension and Resumption Criteria**

* + - * **Suspension Criteria:** Testing will be paused if a critical issue is discovered, such as a major failure in core functionality (e.g., user login, repository creation, or access control) that prevents the system from being used or tested further.
      * **Resumption Criteria:** Testing will resume once the critical defect is fixed and the system is stable, ensuring core functionality is working properly and allowing further testing of additional features.

Risk Register

* Updates to the Gitea application may introduce changes in the UI or API, potentially breaking automated tests.
* Test data dependencies might lead to failures if the database is altered or reset.
* Network or server downtime could disrupt test execution in a local or CI/CD environment.

**Tasks**

**Task 1: Write test cases for validating the creation of a new project, including title requirements and special character handling (1 hour).  
Task 2: Execute automated tests on the web interface for user sign-in and project creation functionalities (2 hours).  
Task 3: Perform API testing for project-related operations using Postman, such as creating, updating, and fetching projects (1.5 hours).  
Task 4: Log and track identified defects in Jira, specifying details about failure points and expected outcomes (1 hour).  
Task 5: Compile a comprehensive test summary report detailing test results, defects, and coverage (1 hour).**

**Staffing:**

* **Test Lead:** maias omar

Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Start Date** | **End Date** | **Duration** |
| Execute Tests (UI) | 2024-12-20 | 2024-12-20 | 2 hours |
| API Testing (Postman) | 2024-12-20 | 2024-12-20 | 2 hours |
| Write Test Cases | 2024-12-21 | 2024-12-21 | 3 hours |
| Prepare Test Summary Report | 2024-12-22 | 2024-12-22 | 2 hours |