**TEST PLAN/STRATEGY FOR REPOSITORY MANAGEMENT WEB APPLICATION**

**TABLE OF CONTENTS**

1. Introduction

* Purpose
* Project Overview
* Key Functionality

1. Test Strategy

* Scope
* Test Types
* Test Environment

1. Test Case Design

* Repository Creation
* Issue Management

1. Automation Approach

* Automated Test Selection
* Sample Automated Test Script

1. Bug Reporting & Tracking
2. Additional Considerations

* CI/CD Integration
* Regression Testing
* Performance & Security Testing

**INTRODUCTION**

**Purpose**

This test plan defines the testing approach and overall framework that will drive the Repository Management web application, ensuring its core functionalities are thoroughly tested to meet business and user requirements. This has been created to facilitate communication within the team members.

**Project Overview**

The Repository Management web application allows users to create and manage repositories, track issues, handle pull requests, and manage user roles and permissions.

**Key Functionality**

* Repository Creation, Editing, and Deletion
* Issue Tracking and Commenting
* Pull Request Workflow
* User Role Management

**TEST STRATEGY**

**Scope**

The test will cover both functional and non functional tests:

* Functional Testing (Repository actions, Issue tracking, Pull requests, User roles)
* Regression Testing
* Integration Testing
* UI/UX Testing/Non functional
* Security and Performance Testing

**Test Types**

|  |  |
| --- | --- |
| **Test Type** | **Description** |
| Functional | Validate core features like repository creation and issue tracking. |
| Regression | Ensure new updates do not break existing functionality. |
| Integration | Validate interactions between modules (e.g., repository and pull request) |
| UI/UX | Verify interface usability and responsiveness. |
| Security | Test user authentication, authorization, and data protection. |
| Performance | Assess application speed and load capacity. |

**Test Environment**

* **Development:** Unit and integration tests.
* **Staging:** Full test execution with real-like test data.
* **Production:** Post-release monitoring.

**Test Acceptance Criteria**

1. Approved Functional Specification document, Use case documents must be available prior to the start of the test phase.
2. Development completed, unit tested with pass status and results shared to Testing team to avoid duplicate defects
3. Test application installed, configured and ready to use state

## **TEST CASE DESIGN**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Case | Test Scenario | Pre Conditions | Steps | Expected Results | Post Conditions |
| TC01 |  | Log in to the application. | User already has an account | - Launch [https://repository.com/](https://repository.com/user)  - Click on Login | User successfully logs in. | User should view user dashboard |
| TC02 |  | Navigate to "Create Repository" page. | User is already logged in | - Click on “+” button at the top right of page  - Click on New Repository | Repository creation form is displayed. | User can successfully fill out all required fields |
| TC03 |  | Enter valid repository name and description. | User is on Create Repository page | - Input new repo name | Name is accepted. | Name is shown with a green check-mark signifying acceptance |
| TC04 |  | Click "Create Repository". | User has filled in new repo name | - Click create repository | Repository is successfully created. | User can see repository |
| TC05 |  | Verify repository appears in list. | User has created a new repository | - Navigate to repositories | Repository is listed correctly. | User can see repository in repository list |
| TC06 |  | Navigate to repository issues tab. | User has an existing repository | - Launch <https://repository.com/user>  - Click on repositories  - Selec a repository from repo list  - Clcik on issues tab | Issues list is displayed. | User should view all available issues from list |
| TC07 |  | Click "Create Issue". | User is on issues page | - Click on create issue | Issue creation form appears. | User can fill issue creation form |
| TC08 |  | Enter issue title and description. | User is on create issue form | - Fill out create issue form | Data is accepted. | User can fill form correctly |
| TC09 |  | Click "Submit". | User has filled issue form | - Click on submit | Issue is successfully created. | User can view issue in issue list |
| TC10 |  | Add a comment to the issue. | User has created an issue | - Click on add comment | Comment is added successfully. | User can view added comment |

**AUTOMATION APPROACH**

Automated Test Selection

* Critical User Flows (e.g., Repository creation, Issue management)
* Regression Tests to cover frequent changes

**Sample Automated Test Script (Cypress - JavaScript)**

|  |
| --- |
| import {createRepository} from "../../fixtures/selectors.js";  describe('Repository Management Tests', () => {  beforeEach(() => {  //Launch Webapp          cy.visit('/');  //Login  cy.get(username).type('testuser');          cy.get(password).type('securepassword');          cy.get(submitBtn).click();          });  //User Logs in to the repository account      it('should log in successfully', () => {          cy.url().should('include', '/dashboard');          cy.contains('Welcome, testuser').should('be.visible');      });  //User creates a new repository      it('should create a new repository successfully', () => {          cy.contains('New Repository').click();          cy.get(inputRepoNameFld).type('NewRepo');          cy.get(createRepoBtn).click();            cy.contains('NewRepo').should('exist');          cy.get(repoList).should('contain', 'NewRepo');  });  //User creates new issue   it('should create a new issue successfully', () => {          cy.contains('NewRepo').click();          cy.contains('Issues').click();          cy.contains('New Issue').click();            cy.get(newIssueFld).type('Bug in login functionality');          cy.get(issueDescFld).type('Login button does not work when clicked.');          cy.get(submitIssue).click();            cy.contains('Bug in login functionality').should('exist');      });  }); |
|  |
|  |

## **BUG REPORTING & TRACKING**

### ****Bug Report Template****

|  |  |
| --- | --- |
| **Field** | **Description** |
| ID | Unique bug identifier |
| Title | Short description of the bug |
| Description | Detailed steps to reproduce |
| Expected Result | What should have happened? |
| Actual Result | What actually happened? |
| Severity | Critical / High / Medium / Low |
| Status | Open / In Progress / Fixed |
| Reporter | Name of person reporting |
| Assigned To | Developer handling the fix |

**ADDITIONAL CONSIDERATIONS**

**CI/CD Integration**

* Automated tests will be triggered on new commits (GitHub Actions, Jenkins)
* Reports generated and integrated with bug tracking on Jira.

**Regression Testing Strategy**

* Automated regression suite will runs before every major deployment
* Focus will be on core functionalities and recent bug fixes

**Performance & Security Testing**

* Load testing with JMeter to check repository handling under high usage
* Security checks for user authentication and data privacy vulnerabilities

**Conclusion**

This test plan ensures that the Repository Management application undergoes rigorous testing for reliability, security, and performance. The combination of manual and automated tests will help maintain high-quality standards and seamless CI/CD integration.