CS673 Software Engineering 

Team 4 - Project Portal

Software Test Document

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Brenna Mahn | Team Lead | BM | 5/28/2024 |
| Natanim Eibrahim | Security Lead | NE | 5/28/2024 |
| Savien Love | Design & Implementation Lead | SL | 5/28/2024 |
| Pinwen Mu | Configuration Lead | PM | 5/28/2024 |

Revision history

| Version | Author | Date | Change |
| --- | --- | --- | --- |
| 1.0 | Fehmi Baltaci | 5/28/2024 | First Draft |
| 2.0 | Fehmi Baltaci | 6/9/2024 | Second Draft |

[Project Overview](#_ze66ok89f779)

[Testing Summary](#_sm5odwyvuk3j)

[Test Planning](#_iahjkfrz3b4l)

[Objectives](#_dzuo3jq6eob3)

[Scope](#_2d8248e7ya96)

[Testing Approach](#_6lbq0wnoqko7)

[Unit Testing](#_xis1lh9vd03y)

[Objectives](#_oaxx4mytawwp)

[Testing Approach](#_waee9z8a70oi)

[Smoke Testing](#_gh34815uzcal)

[Objectives](#_rpkd7bszv4d5)

[Testing Approach](#_v3e1xk19cv44)

[Regression Testing](#_4sxhp08bdsog)

[Objectives](#_jkppovxj9o1i)

[Manual Testing Report](#_pqso2mbjyzx4)

[Test Cases](#_clpy2wpan9n5)

[1. User Google Sign-In](#_ljgku4ci1hdn)

[2. Email and Password Sign-In](#_gcsomyxr9b3e)

[3. Display User Projects](#_thzvm01fnnf6)

[3. View Project Information via Side Window](#_k8wypvt5moz0)

[4. Display Project Start Date](#_s2lemz2gfzgh)

[5. Display Progress Comments (?)](#_hqjrsowo6lpt)

[6. Display Project Team Members (?)](#_w540qb193b5d)

[Automated Testing Report](#_mtfbusfb0eq3)

[Testing Metrics](#_rijyjeu2ojqa)

[References](#_15tmymhipvdv)

[Glossary](#_8n34lvocupub)

# Project Overview

This document outlines the software testing strategy for our project management application. The application includes features such as user sign-in, project listing, project details viewing, project progress tracking. The following sections describe the test plan, test cases, and the testing approach to ensure the application meets its requirements and functions correctly.

# Testing Summary

Project management application we’re working on is currently in the midst of building the automated testing framework and testing phase, with a focus on ensuring the application meets its functional and performance requirements. This report will summarize the testing activities that have been planned and those that will be manually and also with automated framework tested.

## Test Planning

### Objectives

* Verify that users can sign in using Google, email, and password.
* Ensure the landing page displays the correct projects accessible to the user.
* Test the functionality of viewing project details via a side window and a detailed project page.
* Validate the project details, including start date, progress comments (?), and team members, are displayed correctly.

### Scope

* Functional testing of user sign-in.
* UI testing of the landing page and project cards.
* Functional and UI testing of project details viewing.
* Validation of charts and detailed information on the project page.

### Testing Approach

* Analysis of requirements and acceptance criteria will be evaluated.
* Definition of Done for all test cases.
* Manual testing will be performed for initial verification.
* Automated tests will be developed using Python with Behave and Selenium for UI automation.
* Automated tests will be developed using Postman for API Automation

## Unit Testing

### Objectives

* Validating the functionality of individual units of code.

### Testing Approach

* Test Driven Development (TDD) will be used with Jest for React.js
* Developers are responsible for writing unit tests for their code.
* The unit tests will be run for every PR in GitHub and before merge they all must pass.

## Smoke Testing

### Objectives

* Evaluate the crucial functionality of the application.
* Critical functionality of the application will be tested using automation framework.

### Testing Approach

* Smoke tests will be automated using Python-Behave-Selenium.
* Smoke tests will run before every deployment.

## Regression Testing

### Objectives

* To ensure new code has not affected the functionality of the application in a bad way.
* Regression testing will be done both using automation framework and Postman.

# Manual Testing Report

In this section you will find scenarios will be tested for our application.

## Test Cases

### User Google Sign-In

| Test Case ID, Title, Priority (High/Medium/Low) | N/A - Google Sign-In - High |
| --- | --- |
| Functionality | User can sign in with their google account |
| Preconditions | User should have a Google Account |
| Test Steps | 1. Navigate to the sign in page 2. Click on Sign In with Google button. 3. Enter Google credentials 4. Verify that the user can login successfully to the application and the landing page is displayed. |
| Postconditions | Users successfully sign in to the application with their google account. |
| Expected Output / Actual Output | Users successfully sign in to the application with their google account. |
| Pass or Fail (If Failed: Bug ID) | Not Tested |
| Additional notes |  |

### 2. Email and Password Sign-In

| Test Case ID, Title, Priority (High/Medium/Low) | N/A - Email and Password Sign-In - High |
| --- | --- |
| Functionality | User can sign in with their email and passwords |
| Preconditions | User should have registered account with our application |
| Test Steps | 1. Navigate to the sign in page 2. Click on the Sign In button. 3. Enter valid email and password 4. Verify that the user can login successfully to the application and the landing page is displayed. |
| Postconditions | Users successfully sign in to the application with their email and passwords. |
| Expected Output / Actual Output | Users successfully sign in to the application with their email and passwords. |
| Pass or Fail (If Failed: Bug ID) | Not Tested |
| Additional notes |  |

### 3. Display User Projects

| Test Case ID, Title, Priority (High/Medium/Low) | N/A - Display User Projects - High |
| --- | --- |
| Functionality | Users can see the projects in the landing page. |
| Preconditions | User should have signed-in |
| Test Steps | 1. Navigate to the landing page. 2. Verify that the projects are displayed as cards. |
| Postconditions | Projects are correctly displayed as cards. |
| Expected Output / Actual Output | Projects are correctly displayed as cards. |
| Pass or Fail (If Failed: Bug ID) | Not Tested |
| Additional notes | Please note that, there should be a data for the projects. If there are no projects in the system or user has no access to them, there won’t be anything to be displayed. |

### 3. View Project Information via Side Window

| Test Case ID, Title, Priority (High/Medium/Low) | N/A - View Project Information on Side Window - High |
| --- | --- |
| Functionality | Users can see the project's brief information in a small side window that pops-up when clicked on the project card. |
| Preconditions | User is on the landing page with project cards displayed. |
| Test Steps | 1. Click on a project card. 2. Verify that a side window pops up displaying brief project information. |
| Postconditions | Side window with project information is displayed. |
| Expected Output / Actual Output | Side window with project information is displayed. |
| Pass or Fail (If Failed: Bug ID) | Not Tested |
| Additional notes |  |

### 4. Display Project Start Date

| Test Case ID, Title, Priority (High/Medium/Low) | N/A - Display Project Start Date - High |
| --- | --- |
| Functionality | Users can see the projects in the landing page. |
| Preconditions | User is on the detailed project page. |
| Test Steps | 1. User clicks on the title of the project that is displayed on the side window, 2. New page is displayed with all of the information for the clicked project. 3. Verify that the project start date is displayed. |
| Postconditions | Project start date is correctly displayed. |
| Expected Output / Actual Output | Project start date is correctly displayed. |
| Pass or Fail (If Failed: Bug ID) | Not Tested |
| Additional notes |  |

### 5. Display Progress Comments (?)

| Test Case ID, Title, Priority (High/Medium/Low) | N/A - Display Progress Comments - Medium |
| --- | --- |
| Functionality | Users can see the comments in the detailed project page. |
| Preconditions | User is on the detailed project page. |
| Test Steps | 1. User clicks on the title of the project that is displayed on the side window, 2. New page is displayed with all of the information for the clicked project. 3. Verify that there are comments in the comments section displayed successfully. |
| Postconditions | Project comments are displayed. |
| Expected Output / Actual Output | Project comments are displayed. |
| Pass or Fail (If Failed: Bug ID) | Not Tested |
| Additional notes | Please note that, there should be a comment posted before for the project, otherwise there won’t be any comment displayed to the user. |

### 

### 6. Display Project Team Members (?)

| Test Case ID, Title, Priority (High/Medium/Low) | N/A - Display Project Team Members - Medium |
| --- | --- |
| Functionality | Users can see the team members that are working on the project. |
| Preconditions | User is on the detailed project page. |
| Test Steps | 1. User clicks on the title of the project that is displayed on the side window. 2. New page is displayed with all of the information for the clicked project. 3. Verify that there are comments in the comments section displayed successfully. |
| Postconditions | Project members are displayed. |
| Expected Output / Actual Output | Project members are displayed. |
| Pass or Fail (If Failed: Bug ID) | Not Tested |
| Additional notes |  |

# 

# Automated Testing Report

We will provide an overview of our automated testing efforts, including the location of test code in our repository, the test frameworks used, and any generated testing reports or screenshots.

1. Everyone should test their code before creating a PR for merge.
2. Testing methods including but not limited to Python-Behave Framework, Postman, and Jest.
3. Everyone is a manual tester in our team. They will test the front end by opening pages in the browser and checking error codes in the browser console, or checking the error code provided by IDE.
4. Test codes will be located under ./project-thisisteam4/automated-testing/
5. With this location we made it clear where test cases are and ready to do a double check at any time to our developers and testers.
6. We will have as many automated tests as we can in this short time frame for our application and also the manual test cases for both UI and API will be documented within this document.

# Testing Metrics

We'll report relevant metrics used for evaluation, such as the number of test cases, test coverage, defect rate, etc. These metrics will help us assess the effectiveness of our testing process and identify areas for improvement.

| Metric Name | Description |
| --- | --- |
| Number of Test Cases | The total number of test cases written for the application including manual and automated tests. |
| Test Coverage | The percentage of functionality covered by testing. |
| Defect Rate | Number of Defects / Number of Total Test Cases |
| Pass/Fail Rate | Passed Test Cases / Failed Test Cases |
| Number of Automated Test Cases | Number of Automated Test Cases |
| Automation Rate | Number of Automated Test Cases / Number of Total Test Cases |

# 

# 

# 

# References

* [Behave Documentation](https://behave.readthedocs.io/en/latest/)
* [Selenium Documentation](https://www.selenium.dev/documentation/)
* [Jest Documentation](https://archive.jestjs.io/docs/en/22.x/getting-started)
* [Postman Documentation](https://learning.postman.com/docs/introduction/overview/)
* [PEP 8 - Python Style Guide](https://www.python.org/dev/peps/pep-0008/)
* [JavaScript Style Guide](https://developer.mozilla.org/en-US/docs/MDN/Writing_guidelines/Writing_style_guide/Code_style_guide)

# Glossary

We'll include a glossary of terms to ensure clarity and consistency in communication throughout the document. This glossary will define key terms and abbreviations used in the testing process.

| Term | Definition |
| --- | --- |
| Acceptance Criteria | The conditions that a software product must satisfy to be accepted by a user or customer. |
| Behavior-Driven Development (BDD) | A software development process that encourages collaboration among developers, QA, and non-technical participants. |
| Test Driven Development (TDD) | A software development process where test cases are developed to specify and validate what the code will do. |
| Unit Testing | Testing individual units or components of a software. |
| User Acceptance Testing (UAT) | A type of testing performed by the end user or client to verify that the software meets the business requirements. |
| Definition of Done (DoD) | A shared understanding of what it means for work to be complete, ensuring all criteria are met before a product increment is considered "done". |