Test Plan for CheckMate

Prepared by: Johnathon Hall, Helen Bueti, Jordan Bailey, Jared Ball,
Daniel Maslowski, Hsu Thawdar San

Table of Contents

1. Introduction

1.1 General Information

This document provides a comprehensive overview of the testing activities for the project, including the scope of testing, submission procedures, test strategies, and defect reporting methods. It's designed to guide the quality assurance by detailing the approach for verifying and validating the product's functionality, performance, and overall quality.

1.2 Purpose

The purpose of this testing plan is to ensure that the product fully meets user requirements and performs reliably under various conditions. This plan outlines the strategies for various types of testing to assess functionality, identify defects, and ensure that all the components work together seamlessly. By evaluating the product against criteria, this plan aims to deliver a quality product that satisfies the user's expectations and standards.

2. Scope

The following components and functions will be tested:

1. **The server**
    - Ensure the server starts successfully and remains operational.
    - Verify the correct response to a curl request for the homepage.

2. **User accounts**
    - **Account registration:**
        - Verify that users can create a new account and that account data persists in the system.
    - **Login:**
        - Standard Login:
            - Ensure users can login using valid credentials and view accessible to-do lists and not see inaccessible ones.
        - OAuth Login via Github;

        - Ensure that users can log in via GitHub OAuth using GitHub credentials.
        - Verify that the user is redirected to their task page upon successful authentication.

- **Account deletion:**
    - Verify users can delete their account, ensuring all associated data, such as lists, tasks, and events, are also deleted.
    - Ensure re-login with the same account is prevented.

3. **Managing Tasks**
    - **Task:**
        - Validate that users create a task, even with no description, and that they persist in the system.
        - Attempt to create a task with an excessively long description (e.g., 1 MB) and verify that the operation fails with appropriate error messages.

    - **Task Completion:**
        - Mark a task as complete and 'commit' the change.
        - Verify that completed tasks are moved to their designated location (e.g., a “Completed” section).
        - Ensure that deleted tasks are moved to their designated location. (e.g., “Deleted” section).
        - Confirm that completing a task triggers the appropriate animation.

    - **Project:**
        - Verify that tasks can be assigned to specific projects and that the assignment persists.
        - Ensure that users can create new projects, and these are reflected in the system.

3. Submission Process

We designed a template for submitting code for review in order to standardize the code review process. This template includes guidelines for preparing code before submission by providing necessary details such as a brief description of the feature, an example submission format and an overview of the review process. For more information, please refer to the link below.

Process for Submitting Code for Review:

4. Test Plan and Strategies

4.1 Testing Methods

We utilized the following testing techniques to ensure the product's quality and reliability.

- **Smoke Testing:** Conduct initial tests to ensure that critical functionalities are working.
- **System Testing:** Perform a comprehensive evaluation of the entire system to verify that all components work together as intended.
- **Compatibility Testing:** Access the product's performance across different browsers and devices.
- **Performance Testing:** Analyze the product's responsiveness stability under various conditions.
- **Functional Verification Test (FVT):** Validate that the product fulfills all functional requirements.
- **Regression Testing:** Re-test the product after changes or updates to confirm that existing functionality remains unaffected.
- **Manual Testing:** Manually interact with the application to identify bugs and defects by testing its functionality and performance from the end-user perspective.
- **Automated Testing:** Use JUnit to automate the execution of test cases for efficiency and consistency.

4.2 Test Log

To ensure that all critical aspects of the products are systematically tested, a test plan log has been created. The test log has the following information.

- **Test features:** Records the features/functions being tested to ensure comprehensive coverage.
- **Categories:** Organizes features into categories for clarity, such as Login, Home Page, Main Task Page, Sign Up, Projects Pages, and Native Login.
- **Status:** Indicates the outcome of each test case, determining whether the feature passed or failed during testing. If a feature fails, it is labelled as a "Defect", and corresponding issues are reported.
- **Priority:** Classifies features as high, medium, or low priority, allowing the team to address issues based on their urgency.

- **Testers:** Records the names of individuals responsible for testing each feature, ensuring traceability.
- **Dates:** Logs the specific dates when each feature was tested, providing the timeline for the testing process.
- **Defect notes:** Provides the description of the defect for any feature marked as a defect.
- **Defect status:** Reports the current state of each defect, specifying whether it is open (unresolved) or closed (resolved).

4.3 Defect Reporting

We utilized GitHub and Miro to effectively report and manage issues or defects, assigning them to the respective teams for resolution.

GitHub Issues

Defects are reported through GitHub issues and assigned to appropriate teams. To ensure comprehensive defect documentation, we created an issue template that provides the following details:

- **Problem:** A clear description of the issue.
- **Replication steps:** Detailed steps to reproduce the defect.
- **Testing environment:** Information about the environment (e.g., browser, operating system) where the issue was observed.
- **Location:** URLs where the defect occurs.
- **Expected behavior:** Description of what should happen.
- **Observed behavior:** Description of what actually happens.
- **Console output:** Any relevant console logs or errors.
- **Screenshots:** Visual evidence of the defect.
- **Suggested fix:** Suggestions for resolving the issue.
- **Other notes:** Additional information to assist in understanding or fixing the issue.

Please see below for the Defect Report Template.
Defect Report Template

Miro Issues

Issues are also tracked and documented on the Miro board, providing a visual representation for better collaboration. Each issue is categorized under one of three statuses: **To Do, In Progress,** or **Done,** and includes the following details:

- **Title:** A description of the defect found.
- **Status:** The current progress of the issue (e.g., To Do, In Progress, Done)
- **Assignee:** The person responsible for addressing the issue.
- **Dates:** Date the issue was found

- **Tags:** Labels to team, such as engine or GUI.
- **Link to GitHub issue:** A direct link to the corresponding GitHub issue for additional information.

Please refer to the link below for the Miro Board.
Miro Board

5. Resources

5.1 Tools

- Development: VSCode, OpenLiberty, Node, Vite, Maven, Java, JUnit, Selenium
- Collaboration: Google Sheets, Miro, Github, Discord

5.2 Browsers

- Chrome
- Firefox
- Safari

5.3 Operating System

- macOS
- Windows
- Linux

6. Unimplemented Features

Some of the features initially included within the scope were not implemented in the final product due to time constraints and resource limitations. Additionally, some advanced features required infrastructure that could not be developed with the given timeframe. These include:

- Sharing to-do lists with other accounts or via shareable links
- Account deletion with associated data removal
- Receiving in-app notifications

While not implemented in this phase of development, the features listed remain strong candidates for future development. Future additions of these features would expand the application's capabilities and improve the user experience.Despite these omissions, the final product successfully delivered core functionalities, including user registration, logging in via Github, task management, project management.