Test Plan Group 3

UMGC CMSC 495 Section 6381

Team Members:
Cook, Sean
Coutros, Peter
Daley, Tasciana
Geisler, Nicholas
Holman, Trevor
Jefferson, Michael
Malixi, Matthew Arvhie
Mussa, Nezifa
Suarez, Karolyn

Table of Contents

1. Introduction

The Calendar/Task Manager Application is a Java-based program that enables multiple users to create, manage, and track their tasks and events through an intuitive calendar interface. Users can add new tasks, set deadlines, categorize tasks by priority or type, and receive notifications for upcoming events or deadlines. The application aims to assist users in organization, improve time management, and enhance overall efficiency.

To ensure the application functions correctly and meets performance standards, a comprehensive test plan has been created to validate its functionality and performance. This document describes the approach and methodology that will be applied to system testing.

2. Test Objectives

The test plan is designed to ensure that the Calendar/Task Management Application operates according to the specified requirements. It serves as a guide for testing activities, outlining how each feature and functionality will be validated to meet expected standards.

Testing Effort:

- **Feature Testing:** Ensure the software provides the desired results such as Create, Read, Update, and Delete (CRUD) operations, and other key features.
- **Defect Identification and Reporting:** Detect bugs or issues in the software and ensure they are addressed before deployment.
- **Risk reduction:** Identify and address potential issues that could impact the testing processes.

Quality Attributes to Be Assessed:

- **Functionality:** Ensure that all features operate correctly according to the specified requirements.
- **Performance:** Assess the application's response times and load handling under different conditions.
- **Reliability:** Confirm that the application runs consistently without failures.
- **Usability:** Provide a user-friendly and accessible platform for all users.

3. Test Strategy

To ensure that each deliverable of the calendar application is tested thoroughly, individual features will be assigned to team members to test separately. This will allow the team to utilize a divide and conquer method, ensuring that no potential bugs or user inconveniences are missed. An alternative method would be for every team member to test every part of the application, but this could lead to more broad feedback which may not be helpful in terms of isolating and fixing issues within the functionality of the calendar. At this stage of the project, fine tuning the application is vital to creating a product that is well rounded and delivers all the goals set during the planning stage.

4. Test Scope

The test plan's scope defines which features and functionalities of the Calendar/Task Management Application will be included or excluded from testing activities. These boundaries are established based on the application's requirements and design specifications.

In scope:

Core functionality testing: Creation, editing, deletion, and viewing of tasks.

Calendar Integration: Displaying tasks/events and navigating calendar views.

Notification system: Sending reminders and configuring notification preferences.

User Authentication: User registration, login, and access control.

Error handling and validation: Testing input validations and error handling.

Different types of testing: Performance, Security, and Compatibility Testing.

Testing under multiple environments: Operating system platforms and environments.

Data storage testing: Assessing the types and volume of data stored in the database.

Out of scope:

Third-party integrations: Excludes any integrations with external applications or services

Artificial Intelligence (AI): AI functionalities will not be included.

Testing beyond defined limits: Stress or load testing beyond anticipated user loads will be excluded.

Test Strategy:

The test strategy outlines the overall approach and methodology for testing the Calendar/Task Management Application. An agile approach will be adopted for flexibility and iterative development. Since the project is not large in scale, testing will primarily rely on manual methods. Automated testing will be utilized for recurring tests when appropriate, resulting in a combination of manual and automated testing.

Resources:

Hardware: Necessary computers and servers to run the application and perform tests across various environments.

Software: Tools such as Selenium for automation and Jira for test management.

Test Data: Prepared test cases to validate application functionality according to requirements.

Testing Types:

Testing will focus on the following quality attributes:

- **Functionality**
- **Performance**
- **Usability**
- **Security**
- **Reliability**
- **Compliance**

Test Prioritization and Execution:

- Tests will be prioritized and scheduled based on key functionalities and features.
- Test cases will be executed using both manual and automated techniques.
- The testing team will report, and track defects using Jira and will retest fixed defects accordingly.

5. Test Cases

Table 1. Calendar App Traceability Matrix - Create Event (Req 2.1.1)

Test Case	Input/Output	Expected Result	Actual Result	Outcome (Pass/Fail)
1.1	Navigate to app **URL** (http://localhost:8080)	The **app homepage** loads in the browser.		
1.2	From **app homepage**, user clicks "Create Event" **hyperlink**	User Interface (UI) transitions **to event creation form**.		
1.3	**Event creation form**/ interface	User enters the following **event** information including: **Title** textbox: "Event 1" **Date** textbox: "09/15/2024" **Time** textbox: "1:00pm" **Recurring** checkbox: selected **Category** textbox: "Cat1" **Location** Textbox: "Adelphi, MD" **Reminder notification**: Current date/ time + 24hrs		
1.4	Completed **event creation form**, user selects "Add Event"	UI indicates **event** is created.		
1.5	In **Calendar App,** User selects to view all events	All entered **events** appear in the UI		
1.6	User selects "Home"	UI returns to **homepage**		

Table 2 – Calendar App Traceability Matrix Delete Event (Req 2.1.2)

Test Case	**Input/Output**	**Expected Result**	**Actual Result**	**Outcome (Pass/Fail)**
2.1	In browser, user navigates to http://localhost:8080/events/list	All calendar **events** appear in the UI as a list/table		
2.2	From app, user clicks "Delete" **hyperlink**	**Alert message** appears asking user to confirm delete action		
2.3	User selects "ok" button	User sees the **event** list without the deleted event.		
2.4.1	Users selects "Create New Event" **hyperlink**	User Interface (UI) transitions to **event creation form**. See Table 1		
2.4.2	User selects "Home"	UI returns to **homepage**. See Table 1		

Table 3 – Calendar App Traceability Matrix Edit Event (Req 2.1.3)

Test Case	Input/Output	Expected Result	Actual Result	Outcome (Pass/Fail)
3.1	In browser, user navigates to http://localhost:8080/events/list	All calendar **events** appear in the UI as a list/table		
3.2	On the first event (top row), user selects the "Edit" **hyperlink**	UI transitions to **Event Update form**. Form displays **event** information		
3.3	**Event Edit form**/ interface	User updates **event** information including: **Title** textbox: UpdatedEvent1 **Date** textbox: "09/16/2024" **Time** textbox: "2:00pm" **Recurring** checkbox: unselected **Category** textbox: Cat2 **Location** textbox: "Silver Spring, MD" **Reminder** Notification date/time: Current date/time + 48hrs		
3.4	Users selects "Update Event" **hyperlink**	User Interface (UI) returns to **event** list showing updated information.		
3.4.1	Users selects "Create New Event" **hyperlink**	User Interface (UI) transitions to **event** creation form. See Table 1		
3.4.2	User selects "Home"	UI returns to **homepage**. See Table 1		

Table 4 – Calendar App Traceability Matrix Create Task (Req 2.1.4)

Test Case	Input/Output	Expected Result	Actual Result	Outcome (Pass/Fail)
4.1	Navigate to app **URL** (e.g. http://localhost:8080)	The app **homepage** loads in the browser.		
4.2	From app homepage, user clicks “Create Task” **hyperlink**	User Interface (UI) transitions to **task creation form**.		
4.3	**Task creation form**/ interface	User can enter **task** information including: **Title** textbox: “Task1’ **Priority** textbox: “1” **Deadline textbox**: “09/16/2024” **Completion Status** checkbox: selected **Category** textbox: “Cat1” **Reminder** Notification date/time: Current date/time + 24hrs		
4.4	Completed **task creation form**, user selects “Add Task”	UI indicates **task** is created.		
4.5	In Calendar App, User selects to **view all tasks**	All entered **tasks** appear in the UI		
4.6.1	User selects “Create Another Task”	User Interface (UI) transitions **to task creation form**. See 4.3		

Table 5 – Calendar App Traceability Matrix Delete Task (Req 2.1.5)

Test Case	**Input/Output**	**Expected Result**	**Actual Result**	**Outcome (Pass/Fail)**
5.1	In browser, user navigates to http://localhost:8080/tasks/list	All **tasks** appear in the UI as a list/ table		
5.2	From app, user clicks "Delete" **hyperlink**	**Alert message** displays asking user to confirm delete action		
5.3	User selects "ok" **button**	User sees the task list without the deleted task.		
5.4	Users selects "Create New Task" **hyperlink**	User Interface (UI) transitions to task creation form. **See Table 4**		
5.4.1	User selects "Home"	UI returns to homepage. **See Table 1**		

Table 6 – Calendar App Traceability Matrix Edit Task (Req 2.1.6)

Test Case	Input/Output	Expected Result	Actual Result	Outcome (Pass/Fail)
6.1	In browser, user navigates to http://localhost:8080/tasks/list	All **tasks** appear in the UI as a list/ table		
6.2	From app, user clicks "Edit" **hyperlink**	UI transitions to **Task Update form**. **Form** displays **task** information		
6.3	**Task Edit form**/ interface	User can edit **task** information including: **Title** textbox: "UpdatedTask" **Priority** textbox: "2" **Deadline textbox**: "09/17/2024" **Completion checkbox**: deselected **Category** textbox: "Cat2" **Reminder Notification** date/time: Current date/time + 48hrs		
6.4	Users selects "Update Task" **hyperlink**	User Interface (UI) returns to **task** list showing updated information.		
6.4.1	Users selects "Create New Task" **hyperlink**	User Interface (UI) transitions to **task** creation form. **See Table 4**		
6.4.2	User selects "Home"	UI returns to homepage. **See Table 1**		

Table 7 – Calendar App Traceability Matrix Track Tasks (Req 2.1.7)

Test Case	**Input/Output**	**Expected Result**	**Actual Result**	**Outcome (Pass/Fail)**
7.1	In browser, user navigates to http://localhost:8080/tasks/list	All **tasks** appear in the UI as a list/ table		
7.2.1	From **app**, user selects **hyperlink** to initiate **"tracking" function**	All **tasks** appear in the list/table, sorted by **priority**		

Table 8 – Calendar App Traceability Matrix Reminder Notifications (Req 2.1.8)

Test Case	**Input/Output**	**Expected Result**	**Actual Result**	**Outcome (Pass/Fail)**
8.1	Navigate to app **URL** (e.g. http://localhost:8080)	All **tasks** appear in the UI as a list/table		
8.2.1	From app **homepage**, user clicks "Create Event" **hyperlink**	User Interface (UI) transitions to **event creation form**.		
8.2.2	**Event creation form**/ interface	User is able to enter event information including: **Title** textbox: "Notify Event" **Date** textbox: "09/18/2024" **Time** textbox: "3:00pm" **Recurring** checkbox: deselected **Category** textbox: "Cat3" **Location** textbox: "Bethesda, MD" User sets **reminder notification** for date/time of the next testing day.		
8.2.3	Completed **event** creation form, user selects "Add Event"	UI indicates **event** is created.		
8.3.1	From app **homepage**, user clicks "Create Task" **hyperlink**	User Interface (UI) transitions to **task creation form**.		

8.3.2	**Task creation form**/ interface	User can enter **task** information including: **Title** textbox: “Notify Task” **Priority** textbox: “3” **Deadline** textbox: “09/19/2024” **Completion** checkbox: deselected **Category** textbox: “Cat4” User sets **reminder** notification for date/time of the next testing day.		
8.3.3	Completed **task** creation form, user selects “Add Task”	UI indicates **task** is created.		
8.4.1	At the appointed **event** reminder day/time.	App sends user notification reminding user of the **event**		
8.4.2	At the appointed **task** reminder day/time.	App sends user notification reminding user of the **task**		

Table 9 – Calendar App Traceability Matrix Filter by Category (Req 2.1.9)

Test Case	Input/Output	Expected Result	Actual Result	Outcome (Pass/Fail)
9.1	In **browser**, user navigates to http://localhost:8080/tasks/list	All **tasks** appear as a list/table		
9.2	From **app**, user selects the **filtering** option	User selects/enters **categories**. Only **tasks** with the given categories are displayed.		
9.3	In **browser**, user navigates to http://localhost:8080/events/list	All **events** appear as a list/table		
9.4	From **app**, user selects the **filtering** option	User selects/enters **categories**. Only **events** with the given **categories** are displayed.		

6. Testing Procedures

The first step to testing the program is to configure the required environment. The IDE that will be used during testing is Eclipse IDE, version 4.29.0, with JavaSE-17. Spring Tools 4 version 4.24.0 is also required and can be downloaded from the Eclipse Marketplace (Step-by-step instructions for configuring the environment can be found in the Phase 1 Source document). Once the code is downloaded from GitHub and the project is configured in Eclipse, the tester can begin.

Testers will be assigned individual features, highlighted in the various tables found under section 5 of this document, to test. The results of these tests will be documented, and any potential issues will be highlighted, so that programmers can attempt to resolve said issues during the second phase of coding. Features that do not work as intended will then be tested again after programmers believe the issue has been resolved. Any feature that is easy to use and works without issue will only be tested again if it is believed that adjustments made in the code might have affected their functionality.

7. Testing Schedule

1. **Unit testing:** September 18-24, 2024
   Each team member will focus on testing individual components and modules to ensure they function correctly

2. **Integration testing:** September 25-29, 2024
   Testing how different modules work together in the application.

3. **System testing:** September 30-October 4, 2024
   Conduct full application testing to verify the end-to-end functionality.

4. **User Acceptance Testing (UAT):** October 5-8, 2024
   The final phase, where a sample group of users will test the application to ensure it meets the project requirements.

8. Testing Resources

For our testing process, the core team will be responsible for conducting unit and integration testing, while a select group of stakeholders will participate in user acceptance testing (UAT). We will use existing equipment, including team laptops/desktops and cloud-based testing environments, to run tests efficiently. Tools like JUnit will be utilized for automated testing, keeping costs minimal since we plan to leverage open-source software and existing resources. Overall, no significant additional budget is expected, allowing us to maintain the current resource allocation.

9. Defect Management

Our approach to defect management involves a clear process for identifying, reporting, and tracking bugs. Team members will thoroughly document any issues encountered during testing, ensuring detailed descriptions for clarity. All bugs will be reported and tracked using GitHub Issues, where each defect will be assigned a priority level—Critical, High, Medium, or Low. We will hold regular bug review meetings to assess progress, prioritize fixes, and ensure defects are addressed in a timely manner. This structured process will help maintain the quality and functionality of our code as we move through each testing phase.