Table of Contents

[1. Project Proposal 2](#_Toc174272978)

[1.1 Project background and Objectives 2](#_Toc174272979)

[1.2 Proposal Outline 2](#_Toc174272980)

[1.2.1 Three-Tier Architecture 3](#_Toc174272981)

[1.2.2 Software Development Process and Assessment Plan 4](#_Toc174272982)

[1.2.3 Resources and Logistics 6](#_Toc174272983)

[1.2.4 Scheduling 7](#_Toc174272984)

[2. Project Plan 8](#_Toc174272985)

[2.1 Work Break-down Structure (WBS) 8](#_Toc174272986)

[2.2 Project Gantt chart 9](#_Toc174272987)

[3. System Requirement Specification (SRS) 10](#_Toc174272988)

[3.1 User Needs 10](#_Toc174272989)

[❖ User Needs Identification 10](#_Toc174272990)

[❖ Implementation 10](#_Toc174272991)

[❖ Prioritisation 10](#_Toc174272992)

[3.2 Assumptions and Dependencies 11](#_Toc174272993)

[3.3 Functional Requirements 12](#_Toc174272994)

[3.4 Nonfunctional Requirements 12](#_Toc174272995)

[4. System design/Architecture 13](#_Toc174272996)

[4.1 Context Model 13](#_Toc174272997)

[4.2 Interaction Model 14](#_Toc174272998)

[4.2.1 Use case diagram 14](#_Toc174272999)

[4.2.2 Use case study 15](#_Toc174273000)

[4.2.3 ERD Diagram 21](#_Toc174273001)

[5. Implementation and Testing 23](#_Toc174273002)

[5.1 Development testing 23](#_Toc174273003)

[5.2 Manual test cases 26](#_Toc174273004)

[6. References 37](#_Toc174273005)

# 1. Project Proposal

## 1.1 Project background and Objectives

In today’s fast-paced business environment, effective project management is crucial for the success of any organisation. Project managers and product managers are often overwhelmed with numerous tasks, tight deadlines, and the need to constantly monitor progress. To address these challenges, our team (group 5) aims to develop a web-based project management application designed to streamline the workflow and enhance productivity.

Our application aims to simplify the task assignment process, allowing project managers to efficiently allocate tasks with specific due dates and track the progress in real-time. This will not only help in maintaining an organised work environment but also ensure that all team members are aligned with the project’s goals and timelines. The application will provide normal users with a clear view of their responsibilities and enable them to update the status of their tasks, fostering better communication and collaboration within the team.

* **Task Assignment Simplification**: Create an intuitive interface for project managers to assign tasks with due dates quickly and accurately.
* **Real-time Progress Tracking**: Develop a real-time dashboard for project managers to monitor the progress of each task, view completion percentages, and identify any bottlenecks promptly.
* **Enhanced User Experience for Team Members**: Provide normal users with a straightforward platform to view their assigned tasks, update progress and communicate any issues or delays effectively.
* **Improved Collaboration and Communication**: Foster better collaboration within the team by offering features that enhance communication, such as task comments, notifications, and updates.
* **Data-Driven Insights**: Implement analytics and reporting tools to give project managers insights into project performance, helping them make informed decisions and improve future project planning.
* **Scalability and Flexibility**: Ensure the application can scale to accommodate growing teams and projects and offer customisation options to meet the specific needs of different organisations.
* **User-Friendly interface**: Design a user-friendly and visually appealing interface that minimises the learning curve and maximises productivity for all users.

## 1.2 Proposal Outline

Our web-based project management application - “**Alpha-PM**” aims to become an indispensable tool for project managers and team members alike, enabling them to work efficiently and achieve their project goals with greater efficiency.

In “Alpha-PM” application, our team will be using three-layer architecture

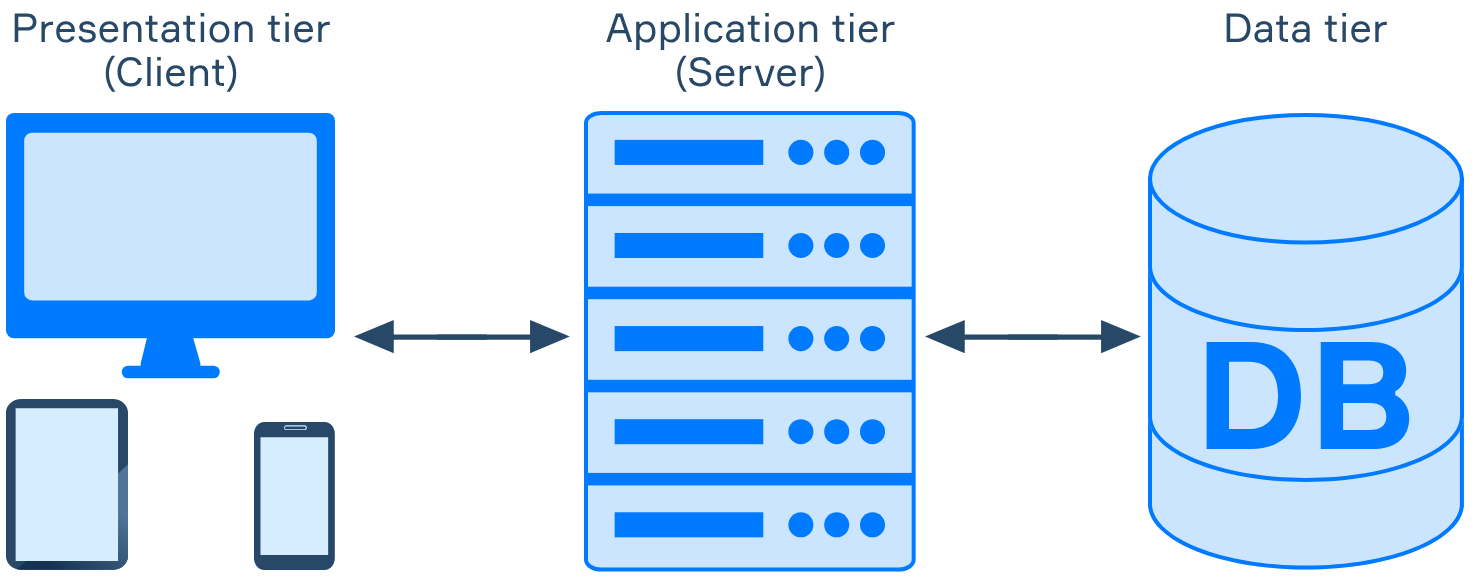


Fig.1 Three tier architecture

### 1.2.1 Three-Tier Architecture

This software application design splits the application into three independent layers or tiers, each accountable for a certain activity.

1. Presentation Tier (User Interface)

User interface. It accepts input and displays data. A critical part in delivering a seamless and intuitive experience for the users.

* User-friendly interfaces for project creation, adding tasks, and managing resources.
* Displaying task lists, timelines, Gantt charts, and allowing users to create, edit, and delete tasks. Also, showing task dependencies and critical paths.
* Presenting resource availability, visualising resource allocation, and enabling drag-and-drop scheduling for tasks and assignments.
* Providing real-time chat, displaying notifications, and showing project-related discussions or comments.
* Summary dashboards, generating printable reports, and visualising project health.
* Login screens, ensuring secure access to project data, and handling user roles and permissions.

1. Application Tier (Business Logic)

* Process new project/tasks creation, editing and deletion and displays in the user interface.
* Calculating project progress based on completed tasks.
* Aggregates data for reporting and analytic purposes.
* Converting user-entered data into a standardised format.
* Integrating with external Services and APIs. (sending notifications to team members via email)

1. Data Tier (Storage)

* Stores the data
* Handles all access requests, ensuring data integrity and consistency
* All the details, tasks, resources and user information resides
* Retrieving data on fetching the tasks list.
* Handles inserts, updates and deletion (CRUD) based on users actions
* Only authorised users can access specific data.

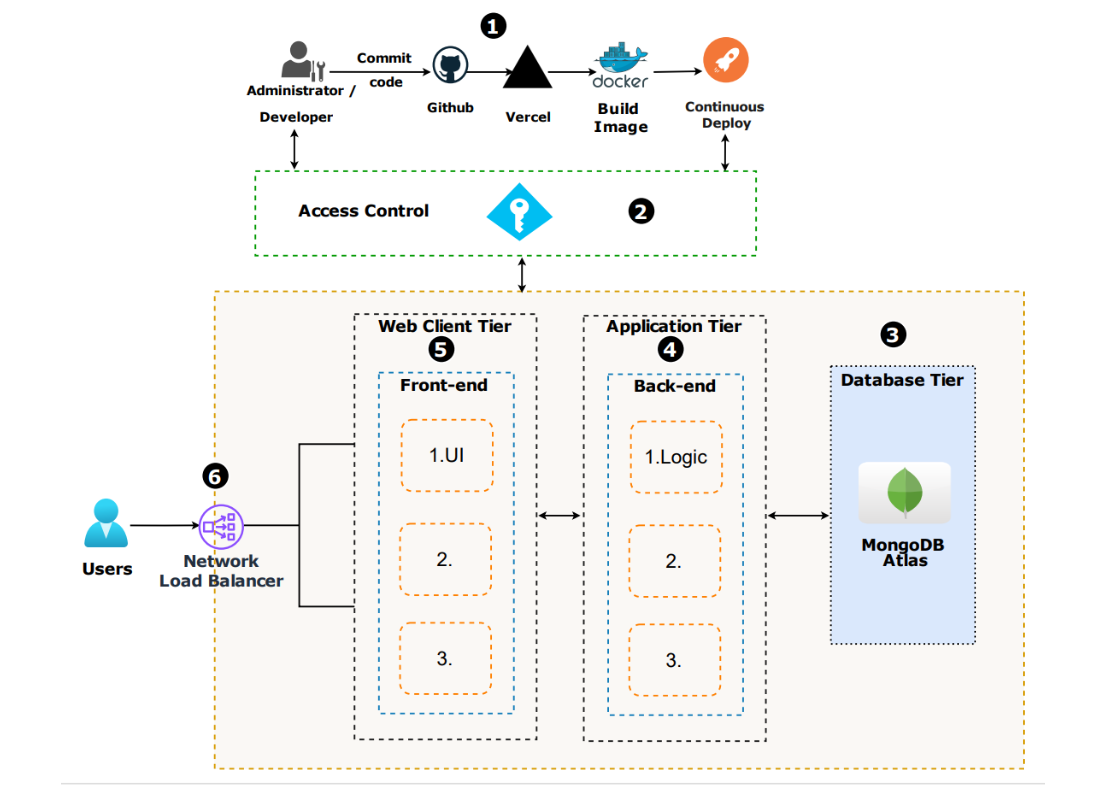
DBMS will be setup on MongoDb, NonSQL document based database. to store all the information related to login credentials, project, tasks, assigned staffs, etc. with built in security, allows simultaneous online access by the developer team and end users securely.

Figure 2: Three-tier architecture of Alpha-PM

### 1.2.2 Software Development Process and Assessment Plan

Our team uses waterfall methodology as our development process model which follows a linear and sequential process as we have already established a clear final product with specific requirements and outcomes. With this method, the individual execution teams aren’t required to be in constant communication and, unless specific integrations are required, are usually self-contained. Individual team members can work independently and are not expected to provide status updates as often as with the agile approach.

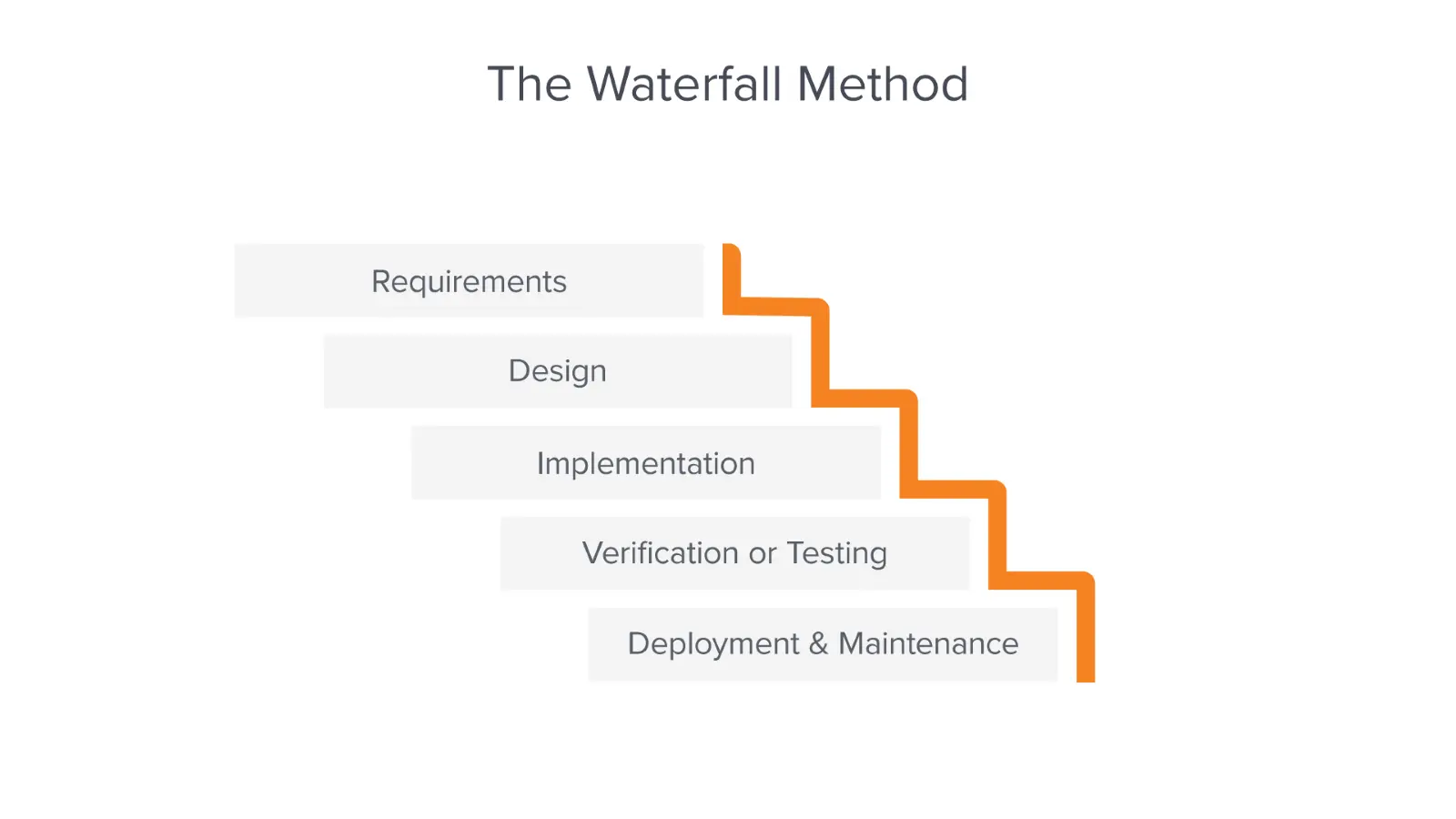


Fig.3 Waterfall Methodologies in software development

Pros and Cons of Waterfall Methodology

Advantages

* Projects are easier to manage. Size of the team does not matter. Once the initial work of planning all the necessary details for successful completion, all that's left is to follow the steps. Throughout the project, everyone is aware of exactly what is being worked on and by whom, as well as any tasks that are dependent.
* Measuring project progress is made simple by it. Project management involves clearly defining all project elements, which simplifies the process of measuring progress. This allows the team to always be aware of the schedule, the number of completed tasks, submitted deliverables, and so on.
* Predictable timeline. Allows for better estimation of project timelines and resource allocation such as having more teammates to focus on the more crucial or time-consuming parts of the project.

Disadvantages

* Requires a lot of planning. Requires project detail upfront. Employ extensive, comprehensive planning efforts. In-depth interviews and multiple brainstorming sessions.
* Requires focus on every phase. Requires every phase to be fully completed before proceeding to the next. Difficult to go back, if anything is missed.
* Limited customer involvement**.** Customers typically see the final product at the end, which can lead to unexpected changes or dissatisfaction.
* Delayed testing. Testing occurs at the end of the development cycle, increasing the risk of discovering critical issues late in the process.
* Potential for scope creep. Changes in requirements can be difficult to accommodate.

### 1.2.3 Resources and Logistics

In this web-based application, we used the following resources:

|  |  |
| --- | --- |
| Javascript / typescript | Front-end and backend language |
| NextJS / Node.js | Backend Framework |
| NextJS / React | Front-end Framework |
| MongoDB | Database |
| MongoDB Atlas | Cloud based database hosting |
| Github | Data repository OSC (one stop centre) |

The project team has 6 members. We conduct regular meetings via Zoom or physical meetings at least once a week to discuss the progress and work together on the roadblocks. Meeting minutes are also recorded and stored in the Github repository. All team members will be involved in user requirements gathering, system design, coding, documentation and testing.

### 1.2.4 Scheduling

The work is divided into small sections such as user requirement data gathering, system design and architecture, entity relationship diagram (ERD), coding & development, testing and debugging.

|  |  |
| --- | --- |
| **Task** | **Target date** |
| Forming of project group and idea pitching | 1-Jul-24 |
| Role & task delegation | 4-Jul-24 |
| System Requirement Specification | 7-Jul-24 |
| Product research and explore suitable tools to use | 14-Jul-24 |
| System design/architecture | 17-Jul-24 |
| User stories creation | 19-Jul-24 |
| Database Schema & Entity Relationship Diagram | 16-Jul-24 |
| Coding & Development | 16-Jul-24 to 8-Aug-24 |
| Testing and Debugging | 8-Aug-24 to 10-Aug-24 |
| Product launch | 12-Aug-24 |

# 

# 2. Project Plan

## 2.1 Work Break-down Structure (WBS)

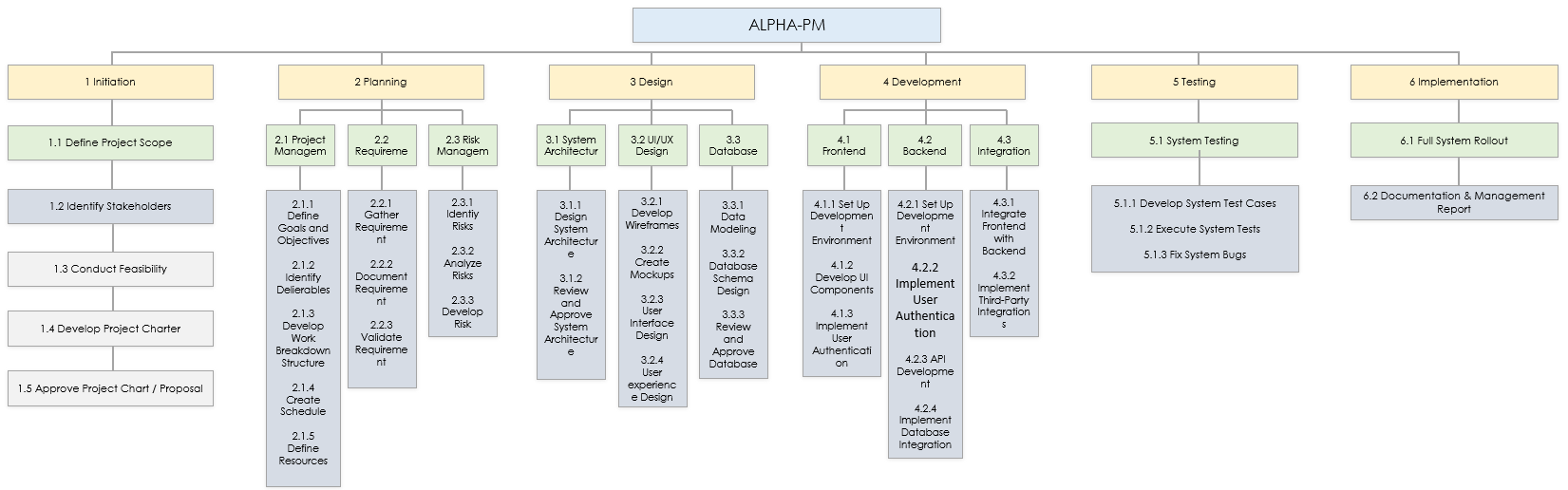
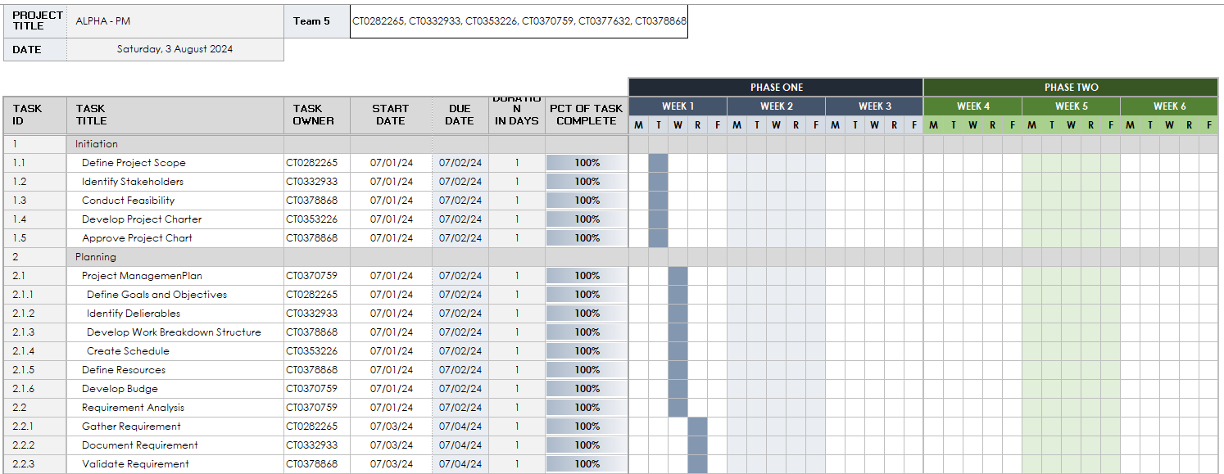
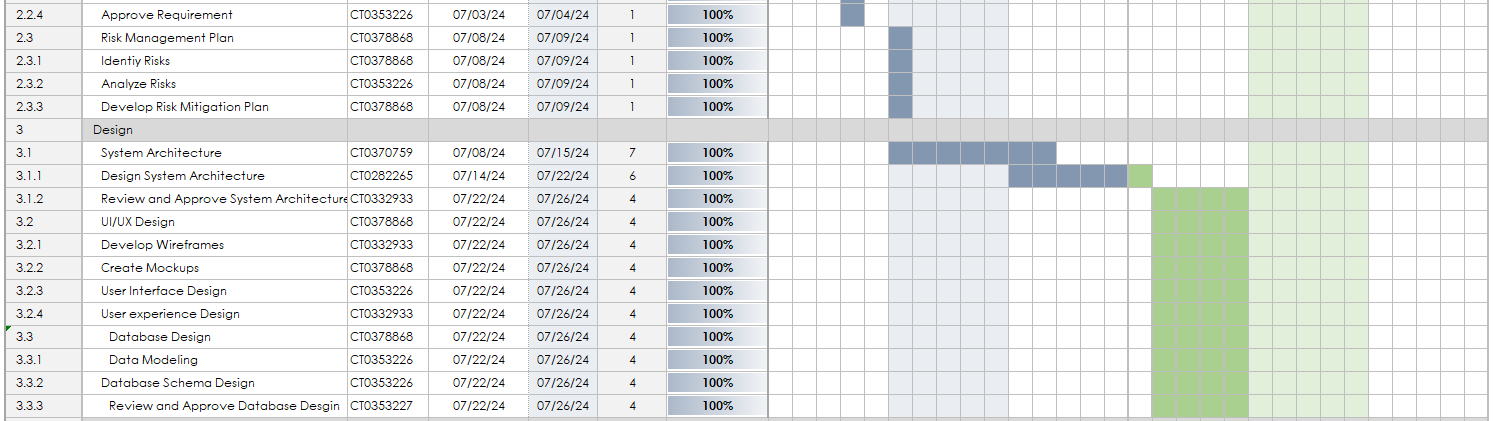


Fig4. WBS of Alpha-PM

## 2.2 Project Gantt chart





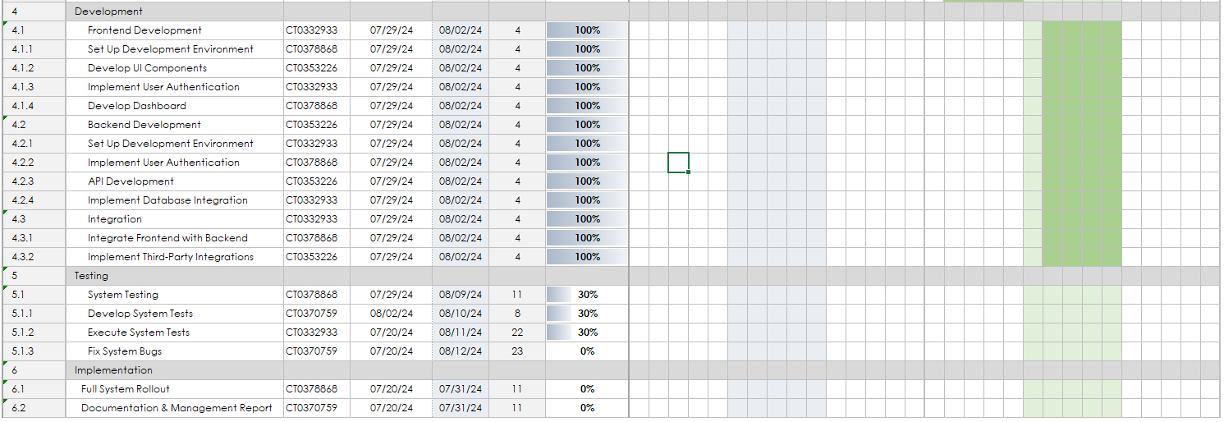


Fig.5 Gantt chart of Alpha-PM

# 3. System Requirement Specification (SRS)

## 3.1 User Needs

### User Needs Identification

* + **Functionality**: Users need to create, edit and delete tasks.
  + **Usability**: The app should have an intuitive interface with clear task labels and deadline
  + **Aesthetics:** Users appreciate a clean, visually appealing design
  + **Accessibility:** The app must be usable by Supervisors and administrators

### Implementation

* + Task creation with due dates
  + Task categorization(eg. work, personal, urgent)
  + Reminders and notifications
  + Task completion tracking
  + Dark mode for usability
  + User-friendly icons and animations

### Prioritisation

* + Must have: Task creation and editing
  + Should have: Task categorization
  + Could have: Dark Mode
  + Won’t have: Complex gamification features

## 

## 

## 3.2 Assumptions and Dependencies

|  |  |
| --- | --- |
| Assumption |  |
| * Users are assumed to have basic knowledge of project management principles * A stable internet connection is needed for real-time updates * Familiarity with project management terms such as milestone, tasks, dependencies * The app cannot handle up to 100 concurrent users without significant degradation in performance * App is compliance with GDPR/PDPA regulations for handling user data | * A short training (orientation) will be conducted for familiarisation of the app * In real-time all handphones are connected to the 5G network, system will update once it gets connected and it will only update as needed * Making sure the min. Requirement of hardware requirements needed to support the app before use. * All users will need to install microsoft authenticator and a 2PA security to protect data |

### 

|  |
| --- |
| Dependencies |
| * Hardware and software requirements * User knowledge * Process knowledge * Approval workflow * Users acceptance test * Timelines * Security |

## 3.3 Functional Requirements

* The task management software must be able to have a login and logout function for users.
* The software will allow for user profile creation and management, including the setting of permissions for different user roles.
* A search function will allow for managers to search for specific team members or tasks.
* The software will allow users to add new tasks, edit existing tasks, and delete tasks when required.
* To track the progress of team members, the software will display the progress of each task.
* For tasks that are completed, users will be able to upload files or images as proof of completion.
* Admins should be able to see a summary of the overall project progress on the landing page

## 3.4 Nonfunctional Requirements

* Performance : Loading speed should be less than 5s on average internet speeds. Necessary steps to ensure smooth 60fps rendering of the application has to be implemented. The system has to be responsive and should work on various device screen sizes. Be able to handle multiple users using at the same time without crashing. The database must be able to handle multiple records without overwriting data.
* Security: Database that stores username and passwords. Application security to prevent attacks such as SQL injection.
* Usability: Easy to navigate webpage. Search/filter functionality. Provide visualisations such as gantt chart and project bar.
* Scalability: Be able to support multiple new users joining each team. Be able to support increasing numbers of data in the database.
* Maintainability: Code is easy to update and maintain, clearly documented for easy testing for maintenance purposes such as writing comments in code.

# 4. System design/Architecture

## 4.1 Context Model

The following context model diagram illustrates how System Admins and Members interact with the GUI to perform various project and task management activities, which are processed by the Server's UI and Logic/APIs components, with data stored and retrieved from the Database.

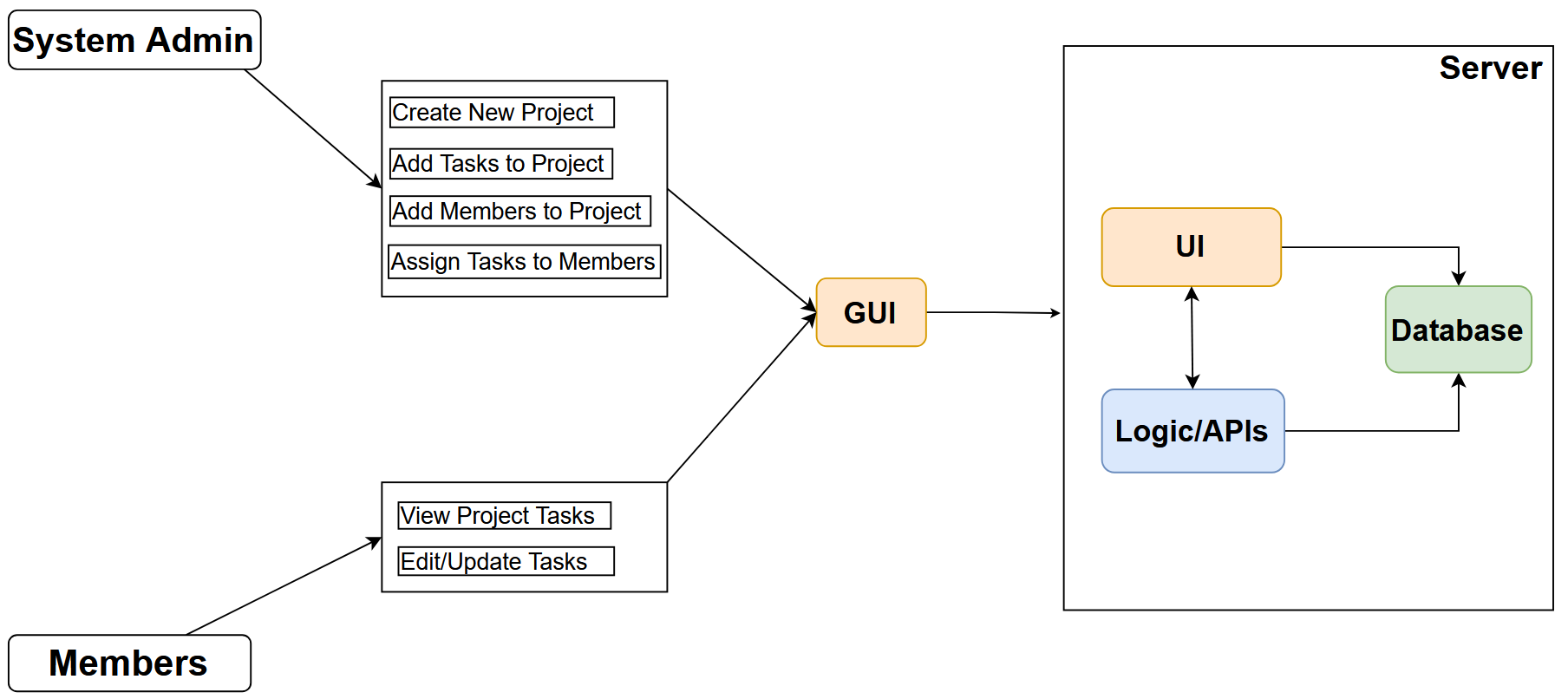
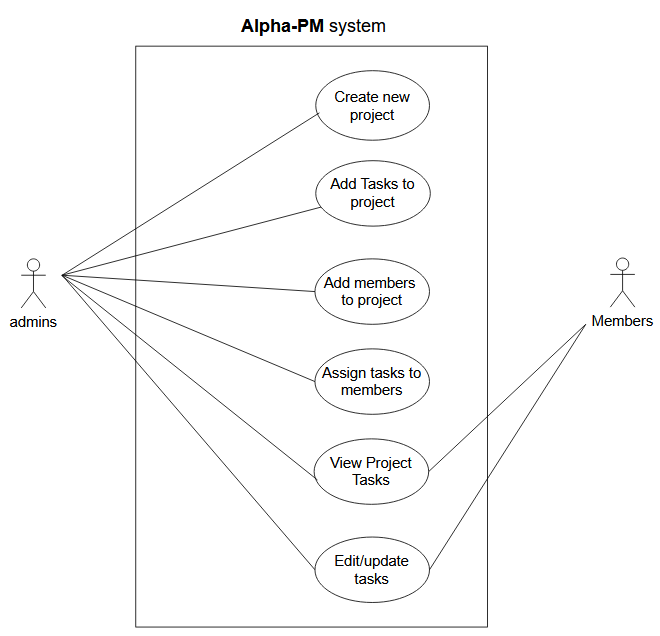


Fig.6 Context model of Alpha-PM

## 4.2 Interaction Model

### 4.2.1 Use case diagram

****

**Fig.7 Use case diagram of Alpha-PM**

### 4.2.2 Use case study

#### Alpha-PM use case study 1

|  |  |
| --- | --- |
| **Use case** | **Create new project** |
| Actors | System Admins |
| Description | A logged in admin user to create a project to add tasks |
| Pre-condition | User is logged in to the app console and on the Alpha-PM page. |
| Post-condition | The project is successfully created and able to add tasks |
| Data | **Input**: project name,description… **Output**: Success/Error message |
| Valid Case | User loads a create project form provided by the web server. The user then inputs the project name and description. The user submits the form, which the web server receives. This form data is then sent to the project manager, who successfully validates the project data according to the validation rules. The manager then creates the project record with the data in the database. A successful project creation message is sent back to the client interface and is displayed to the user. |
| Error Case | Project name is empty, or invalid name length: User loads a create project form provided by the web server. The user inputs the project description but leaves the project name empty or inputs an invalid name. The user submits the form, which the web server receives. This form data is then sent to the project manager, who unsuccessfully validates the project data according to the validation rules. An unsuccessful project creation message is sent back to the client interface and is displayed to the user. |

#### Alpha-PM use case study 2

|  |  |
| --- | --- |
| **Use case** | **Add Tasks to project** |
| Actors | System Admins |
| Description | A logged-in admin user attempts to add tasks to an existing project using an inputted task name, description, and due date. |
| Pre-condition | The user is on the Alpha-PM project page and has selected a project to add tasks to. |
| Post-condition | The task has been successfully added to the project in the database with the task details being displayed to the user by the app. The project now includes the new task |
| Data | **Input:** Task name, description, due date **Output:** Success/Error message |
| Valid Case | User loads the add task form provided by the web server. The user then inputs the task name, description, and due date. The user submits the form, which the web server receives. This form data is then sent to the task manager, who successfully validates the task data according to the validation rules. The manager then creates the task record with the data in the database and associates it with the selected project. A successful task creation message is sent back to the client interface and is displayed to the user. |
| Error Case | Invalid or missing task name, description, or due date: User loads the add task form provided by the web server. The user inputs the task details but leaves one or more fields empty or inputs invalid data. The user submits the form, which the web server receives. This form data is then sent to the task manager, who unsuccessfully validates the task data according to the validation rules. An unsuccessful task creation message is sent back to the client interface and is displayed to the user. |

#### Alpha-PM use case study 3

|  |  |
| --- | --- |
| **Use case** | **Add members to project** |
| Actors | System Admins |
| Description | A logged-in admin user attempts to add members to an existing project using an inputted member username or email. |
| Pre-condition | The user is on the Alpha-PM project page and has selected a project to add members to. |
| Post-condition | The member has been successfully added to the project in the database with the member details being displayed to the user by the app.  The project now includes the new member |
| Data | **Input:** Member username or email **Output:** Success/Error message |
| Valid Case | User loads the add member form provided by the web server. The user then inputs the member's username or email. The user submits the form, which the web server receives. This form data is then sent to the member manager, who successfully validates the member data according to the validation rules. The manager then adds the member record to the project in the database. A successful member addition message is sent back to the client interface and is displayed to the user. |
| Error Case | Invalid or missing member username or email: User loads the add member form provided by the web server. The user inputs the member details but leaves the username or email field empty or inputs invalid data. The user submits the form, which the web server receives. This form data is then sent to the member manager, who unsuccessfully validates the member data according to the validation rules. An unsuccessful member addition message is sent back to the client interface and is displayed to the user. |

#### Alpha-PM use case study 4

|  |  |
| --- | --- |
| **Use case** | **Assign tasks to members** |
| Actors | System Admins |
| Description | A logged-in admin user attempts to assign tasks to members of an existing project using inputted task and member details. |
| Pre-condition | The user is on the Alpha-PM project page and has selected a project with existing tasks and members. |
| Post-condition | The task has been successfully assigned to the member in the database with the assignment details being displayed to the user by the app.  The member is now responsible for the assigned task. |
| Data | **Input:** Task ID, Member ID **Output:** Success/Error message |
| Valid Case | User loads the assigned task form provided by the web server. The user then selects the task and the member from dropdown lists or inputs their IDs. The user submits the form, which the web server receives. This form data is then sent to the task manager, who successfully validates the assignment data according to the validation rules. The manager then updates the task record with the member ID in the database. A successful task assignment message is sent back to the client interface and is displayed to the user. |
| Error Case | Invalid or missing task ID or member ID: User loads the assign task form provided by the web server. The user inputs the task and member details but leaves one or more fields empty or inputs invalid data. The user submits the form, which the web server receives. This form data is then sent to the task manager, who unsuccessfully validates the assignment data according to the validation rules. An unsuccessful task assignment message is sent back to the client interface and is displayed to the user. |

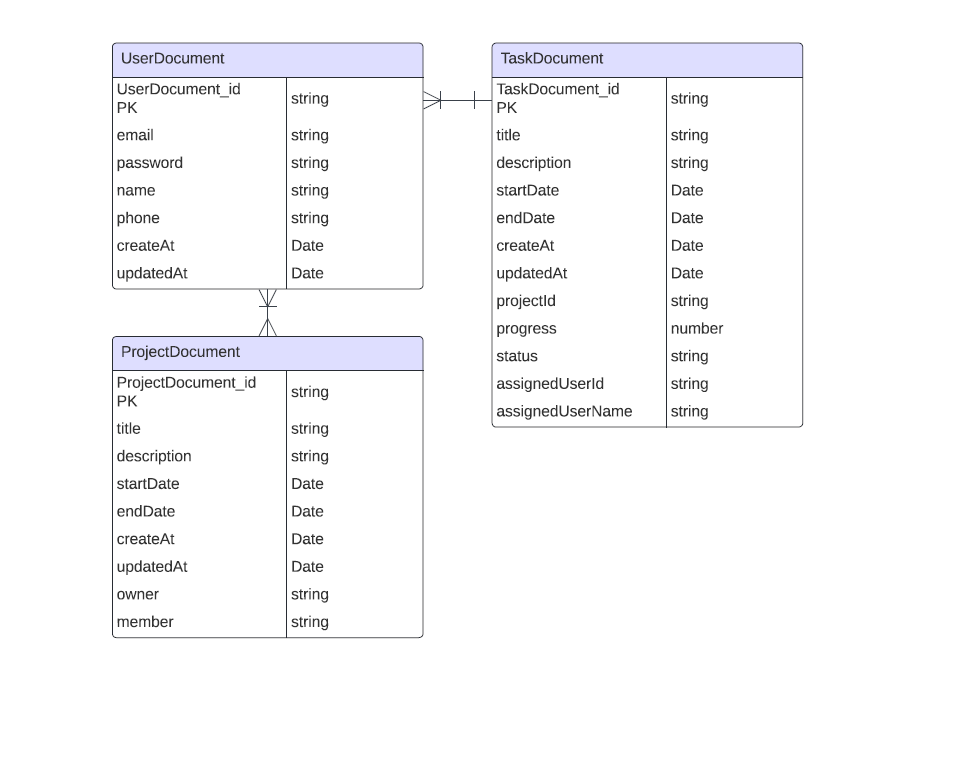
#### Alpha-PM use case study 5

|  |  |
| --- | --- |
| **Use case** | **View Project Tasks** |
| Actors | Members |
| Description | A logged-in member user attempts to view tasks associated with a specific project they are part of. |
| Pre-condition | The user is on the Alpha-PM project page and has selected a project they are a member of. |
| Post-condition | The tasks associated with the selected project are successfully displayed to the member. |
| Data | **Input:** Project ID  **Output:** List of tasks associated with the project, Success/Error message |
| Valid Case | User navigates to the project page and selects a project they are a member of. The system retrieves the tasks associated with the selected project from the database. The task list is successfully displayed on the project page, showing task details such as task name, description, due date, and assigned members. |
| Error Case | Invalid or missing project ID: User navigates to the project page but the system encounters an issue retrieving the project tasks due to an invalid or missing project ID. An error message is displayed to the user indicating that the tasks could not be retrieved. |

#### Alpha-PM use case study 6

|  |  |
| --- | --- |
| **Use case** | **Edit/update tasks** |
| Actors | Members |
| Description | A logged-in member user attempts to edit or update the progress of a task they are responsible for in an existing project. |
| Pre-condition | The user is on the Alpha-PM project page and has selected a task they are responsible for. |
| Post-condition | The task details have been successfully updated in the database with the updated details being displayed to the user by the app |
| Data | **Input:** Task ID, updated task name, updated description, updated due date  **Output:** Success/Error message |
| Valid Case | User navigates to the task they are responsible for and selects the edit option. The user updates the task details such as the task name, description, and due date. The user submits the form, which the web server receives. This form data is then sent to the task manager, who successfully validates the updated task data according to the validation rules. The manager then updates the task record with the new details in the database. A successful task update message is sent back to the client interface and is displayed to the user. |
| Error Case | Invalid or missing updated task details: User navigates to the task they are responsible for and selects the edit option. The user inputs invalid or missing task details such as an empty task name or due date. The user submits the form, which the web server receives. This form data is then sent to the task manager, who unsuccessfully validates the updated task data according to the validation rules. An unsuccessful task update message is sent back to the client interface and is displayed to the user. |

### 4.2.3 ERD Diagram



**Fig.8 ERD diagram of Alpha-PM**

**Description:**

**UserDocument**

1. **Unique Identification**: UserDocument\_id is a unique identifier for each user.
2. **Authentication**: email and password fields are included for user authentication purposes.
3. **Contact Information**: name and phone fields are used to store user contact information.
4. **Timestamps**: createAt and updatedAt fields are used to track when a user record is created and last updated.

**ProjectDocument**

1. **Unique Identification**: ProjectDocument\_id is a unique identifier for each project.
2. **Project Details**: title and description fields capture the basic details of the project.
3. **Project Timeline**: startDate and endDate fields are used to define the project's timeline.
4. **Ownership and Membership**: owner and member fields identify the user who owns the project and the members involved in it.
5. **Timestamps**: createAt and updatedAt fields are used to track when a project record is created and last updated.

**Task Document**

1. **Unique Identification**: TaskDocument\_id is a unique identifier for each task.
2. **Task Details**: title and description fields capture the basic details of the task.
3. **Task Timeline**: startDate and endDate fields define the task's timeline.
4. **Project Association**: projectId field is used to associate a task with a specific project.
5. **Progress Tracking**: progress field indicates the task's completion percentage.
6. **Task Status**: status field is used to track the current status of the task (e.g., pending, in progress, completed).
7. **Assignment**: assignedUserId and assignedUserName fields identify the user to whom the task is assigned.
8. **Timestamps**: createAt and updatedAt fields are used to track when a task record is created and last updated.

**Relationship Assumptions**

1. **User-Project Relationship**: Each project has an owner and potentially multiple members, implying a one-to-many or many-to-many relationship between users and projects.
2. **Project-Task Relationship**: Each task is associated with a specific project, indicating a one-to-many relationship between projects and tasks.**User-Task Relationship**: Each task is assigned to a user, suggesting a one-to-many relationship between users and tasks.

# 5. Implementation and Testing

## 5.1 Development testing

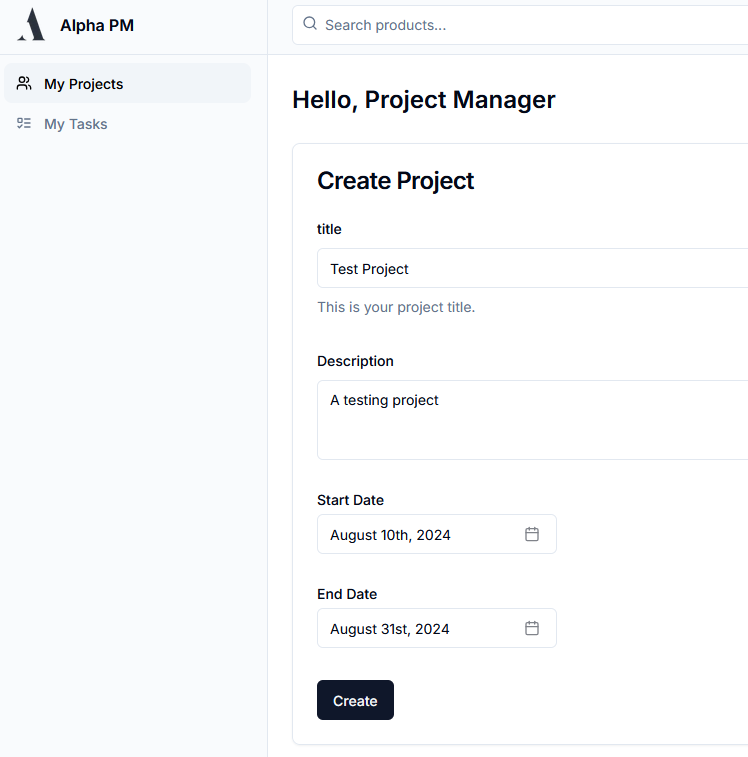
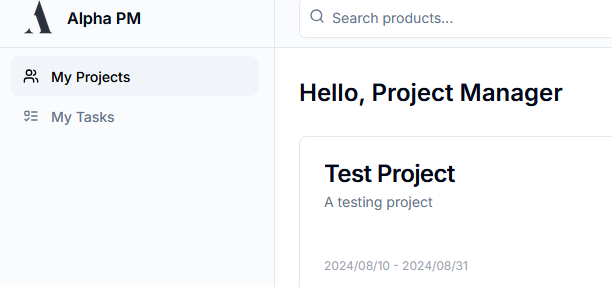
|  |  |  |  |
| --- | --- | --- | --- |
| No | Test Case | Status | Comments |
| 1. | Project/tasks can not be created before the date of the current day. If users forgot to create the project/tasks prior to the current day, they would not be able to set an older date. | Fixed | Users can now add tasks/projects prior to the current date. |
| 2. | When creating tasks/project/adding team members, you would not be able to click out of the form. If users change their mind, they would not be able to leave the form page. | Fixed | Users can now click out of the form page. |
| 3. | Users with different screen resolutions are not able to see some of the options.  Smaller resolution update task:  Smaller resolution add members:  Larger resolution update task:  Larger resolution add members: | Fixed | Added scroll wheel.  All users (regardless of screen resolution) can now see all options. |
| 4 | New user accounts cannot be seen immediately when trying to add users in the Add New Member page. | Fixed | Default caching option by the Cloud provider was disabled. Add new members page can now see new user accounts. |

## 5.2 Manual test cases

#### Create new project

|  |  |
| --- | --- |
| Test Case ID | TC001 |
| Test Case Title | Verify Create New Project functionality |
| Test Description | This test case verifies that a logged-in PM user can create a new project. |
| Preconditions | The user is logged in to the app console and on the Alpha-PM page. |
| Test Data | Project name: Test Project,  Description: A testing project |
| Steps to Execute | 1. Navigate to the 'Create Project' 2. Enter a project name and description, Start Date and End Date 3. Create |
| Expected Result | Project should be created successfully, and a new project should be displayed. |
| Actual Result | Project was created successfully, and the Test Project was displayed. |
| Status | Pass |
| Test Environment | OS: Windows 11, Browser: Chrome |
| Attachments | screenshot\_project\_creation.png |
| Date of Execution | 2024-08-10 |
| Comments | N/A |

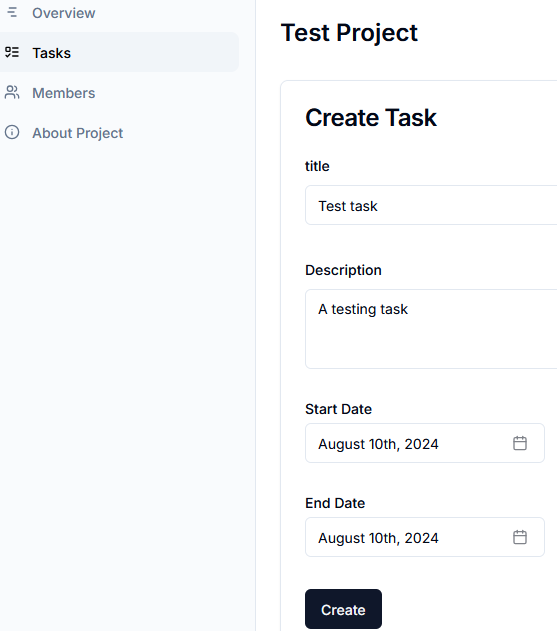
**Screenshot\_project\_creation**

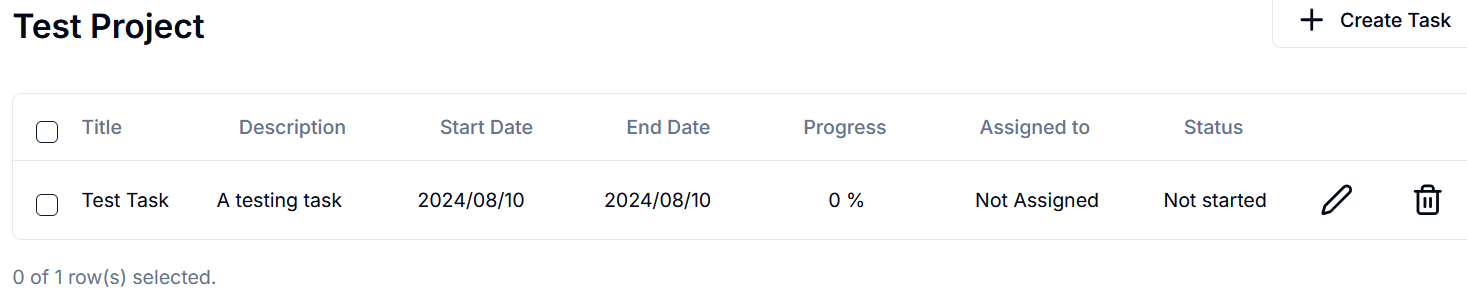
 ****

#### Add tasks in the project

|  |  |
| --- | --- |
| Test Case ID | TC002 |
| Test Case Title | Verify Add Tasks to Project functionality |
| Test Description | This test case verifies that a logged-in admin user can add tasks to an existing project. |
| Preconditions | User is on the Alpha-PM project page and has selected a project to add tasks to. |
| Test Data | Task name: Test task, Description: A testing task  Start Date: 2024/08/10  End Date: 2024/08/31 |
| Steps to Execute | 1. Navigate to the “Create Task”' form 2. Enter a task name, description, and Start Date and End Date  3. Create |
| Expected Result | Task should be added to the project successfully, and a new task should be displayed. |
| Actual Result | Task was added to the project successfully, and the Test task was displayed. |
| Status | Pass |
| Test Environment | OS: Windows 11, Browser: Chrome 103 |
| Attachments | screenshot\_add\_task.png |
| Date of Execution | 2024-08-10 |
| Comments | N/A |

**Screenshot\_add\_task**



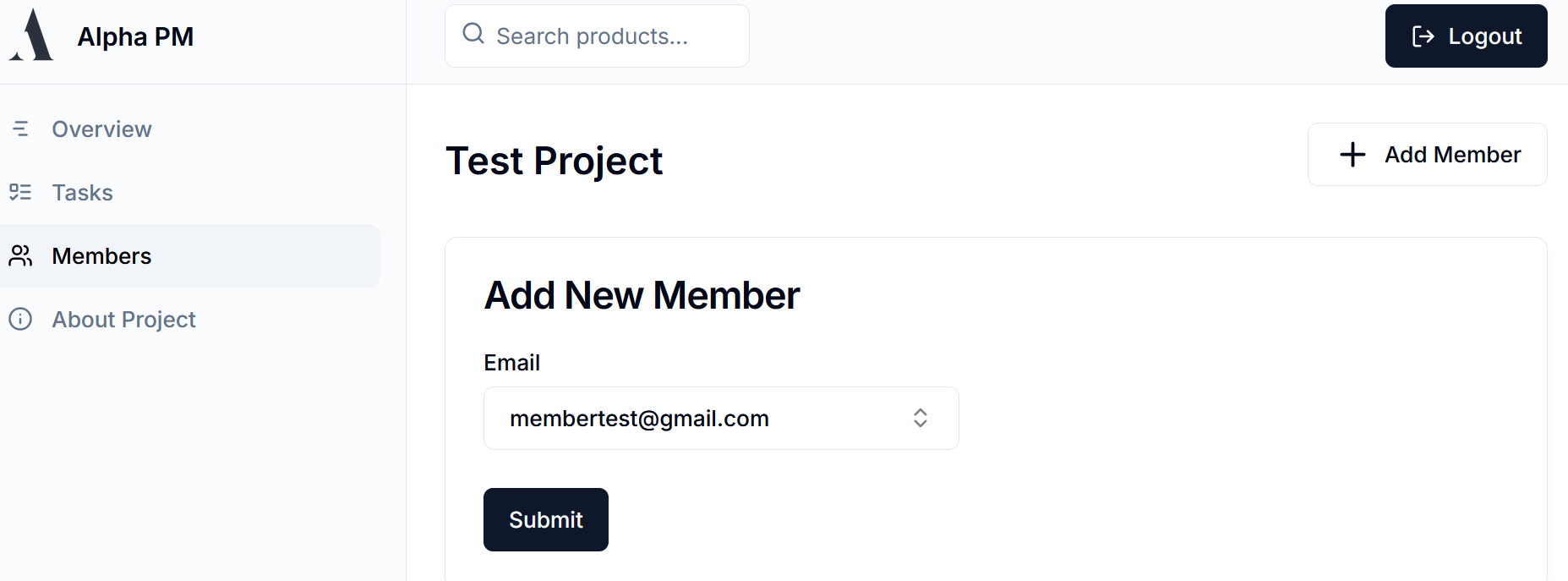


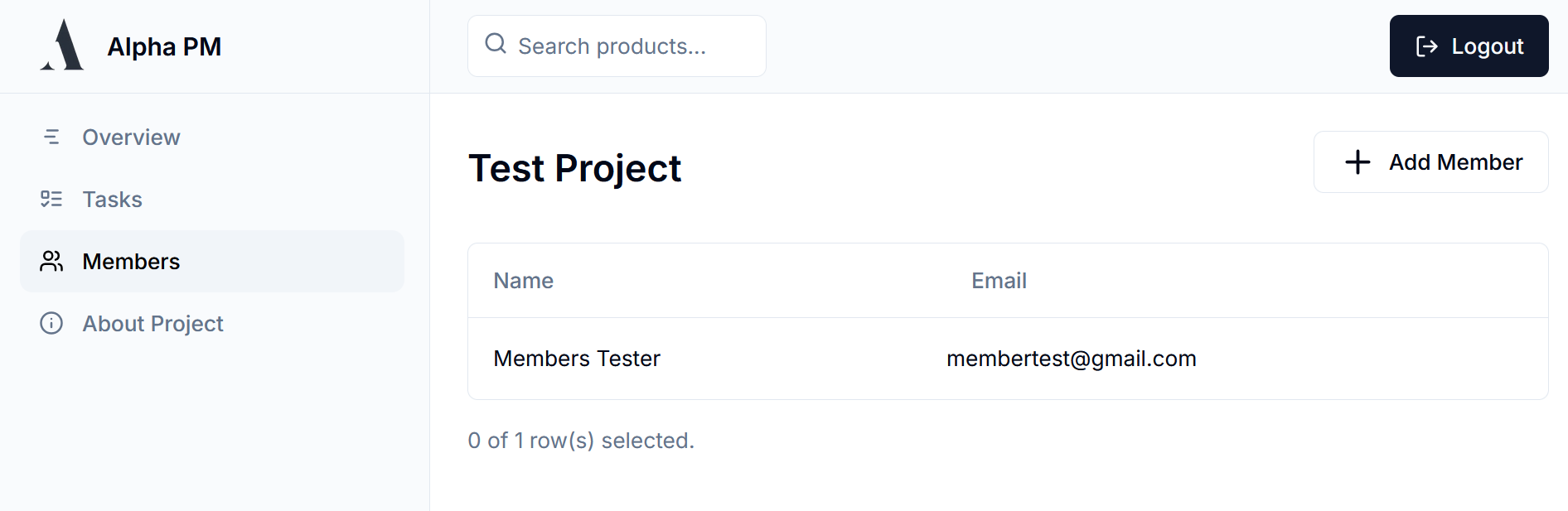
#### Add members to the project

|  |  |
| --- | --- |
| Test Case ID | TC003 |
| Test Case Title | Verify Add Members to Project functionality |
| Test Description | This test case verifies that a logged-in admin user can add members to an existing project. |
| Preconditions | The user is on the Alpha-PM project page and has selected a project to add members to. |
| Test Data | Member Username: Members tester,  Email: membertest@gmail.com |
| Steps to Execute | 1. Navigate to the 'Add Member' form  2. Select a member username or email  3. Submit the form |
| Expected Result | Member should be added to the project successfully, and a member username should be displayed. |
| Actual Result | The member was added to the project successfully, and the “Members tester” was displayed. |
| Status | Pass |
| Test Environment | OS: Windows 11, Browser: Chrome 103 |
| Attachments | screenshot\_add\_member.png |
| Date of Execution | 2024-08-10 |
| Comments | N/A |

# 

**Screenshot\_add\_member**





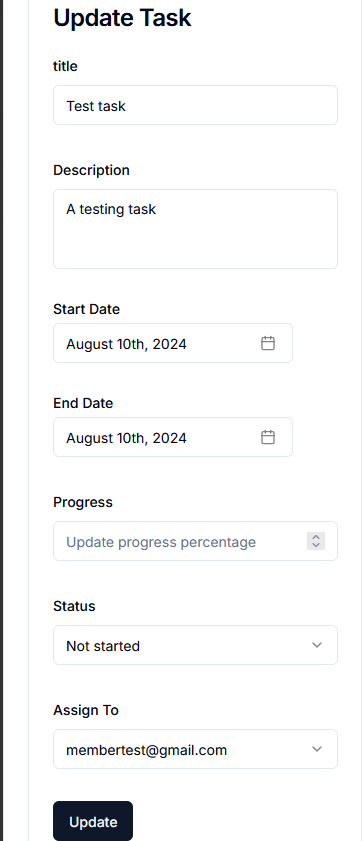
# 

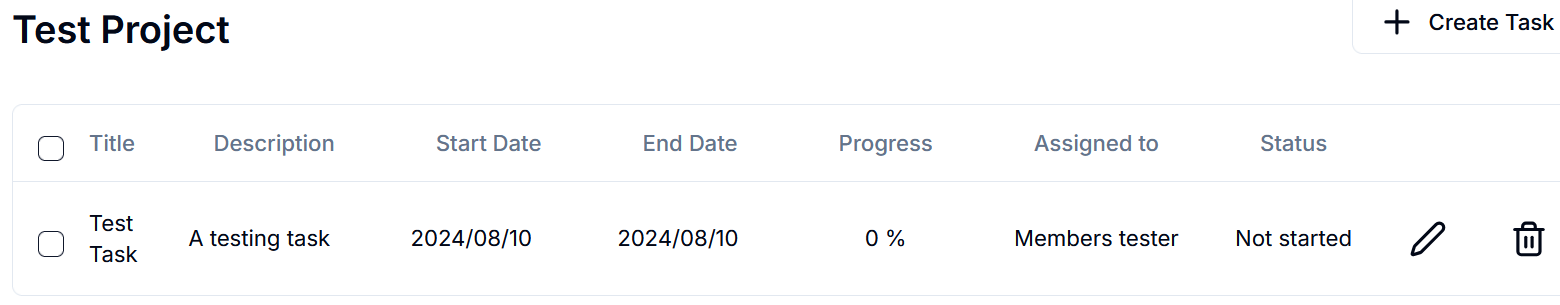
# 

#### Assign tasks to members

|  |  |
| --- | --- |
| Test Case ID | TC004 |
| Test Case Title | Verify Assign Tasks to Members functionality |
| Test Description | This test case verifies that a logged-in admin user can assign tasks to members of an existing project. |
| Preconditions | The user is on the Alpha-PM project page and has selected existing tasks and members. |
| Steps to Execute | 1. Navigate to the 'Update Task' form  2. Select a member to assign  3. Update the form |
| Expected Result | The task should be assigned to the member successfully, and the member assigned should be displayed. |
| Actual Result | The task was assigned to the member successfully, and “Members tester” was assigned and displayed. |
| Status | Pass |
| Test Environment | OS: Windows 11, Browser: Chrome 103 |
| Attachments | screenshot\_assign\_task.png |
| Date of Execution | 2024-08-10 |
| Comments | N/A |

**Screenshot\_assign\_task**

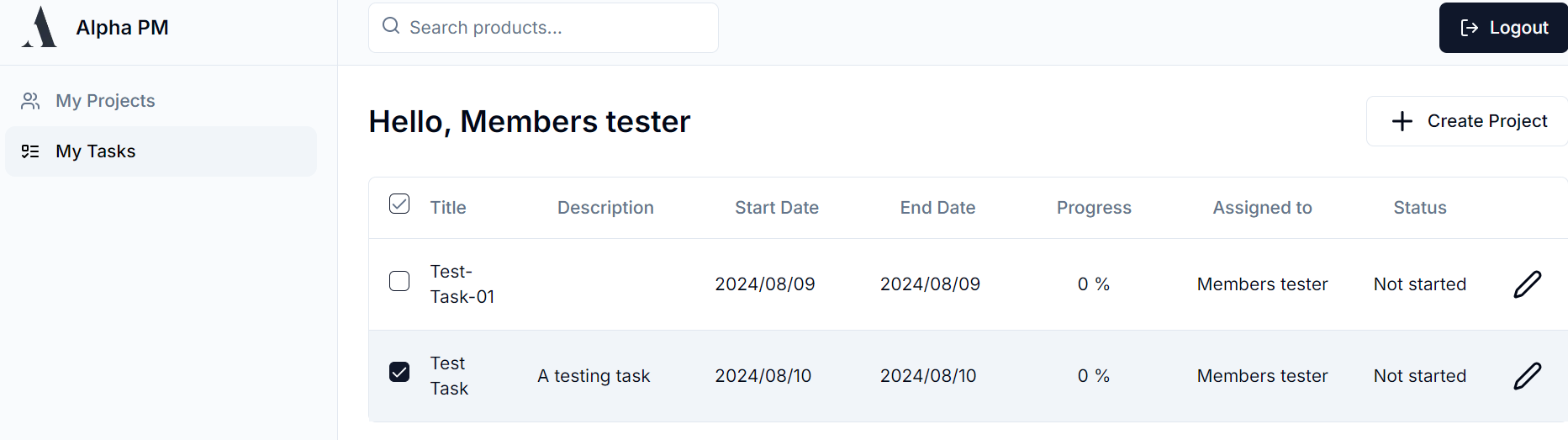




#### View Project Tasks

|  |  |
| --- | --- |
| Test Case ID | TC005 |
| Test Case Title | Verify View Project Tasks functionality |
| Test Description | This test case verifies that a logged-in member user can view tasks associated with a specific project. |
| Preconditions | User is on the Alpha-PM My Tasks page |
| Test Data | Task Title: Test Task |
| Steps to Execute | 1. Navigate to the 'My Tasks' page 3. Tasks should be displayed |
| Expected Result | Tasks associated with the selected project should be displayed successfully. |
| Actual Result | The task “Test task” was displayed successfully. |
| Status | Pass |
| Test Environment | OS: Windows 11, Browser: Chrome 103 |
| Attachments | screenshot\_view\_tasks.png |
| Date of Execution | 2024-08-10 |
| Comments | N/A |

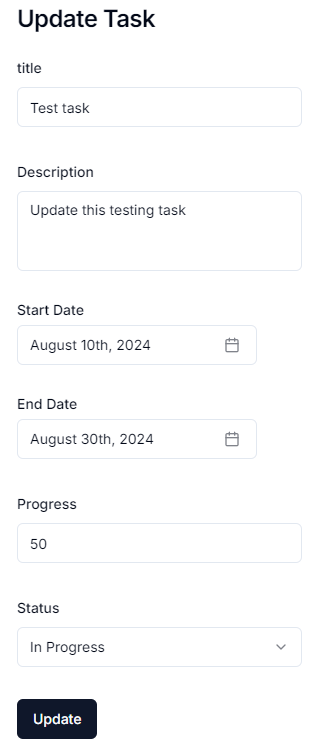
**Screenshot\_view\_tasks**

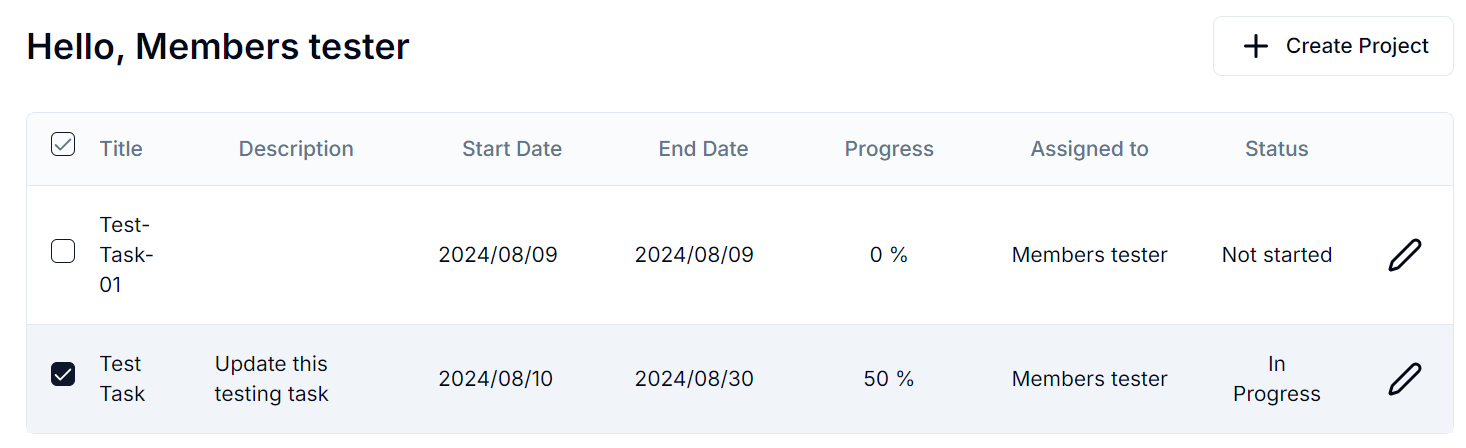


#### Edit/update tasks

|  |  |
| --- | --- |
| Test Case ID | TC006 |
| Test Case Title | Verify Edit/Update Tasks functionality |
| Test Description | This test case verifies that a logged-in member user can edit or update the progress of a task they are responsible for. |
| Preconditions | The user is on the Alpha-PM My Tasks page and has selected a task they are responsible for. |
| Test Data | Task Title: Test Task,  Updated Description: Update this testing task Updated Start Date: 2024-08-10  Updated End Date: 2024-08-30  Updated Progress: 50%  Updated Status: In Progress |
| Steps to Execute | 1. Navigate to the “My Tasks”  2. Select the update task button  3. Update task details  4. Update |
| Expected Result | Task details should be updated successfully, and the updated task details should be displayed. |
| Actual Result | Task details were updated successfully, Updated Description: Update this testing task Updated Start Date: 2024-08-10  Updated End Date: 2024-08-30  Updated Progress: 50%  Updated Status: In Progress |
| Status | Pass |
| Test Environment | OS: Windows 11, Browser: Chrome 103 |
| Attachments | screenshot\_edit\_task.png |
| Date of Execution | 2024-08-10 |
| Comments | N/A |

**Screenshot\_edit\_task**





# 

# 6. References

Source code repo: <https://github.com/SETP-Team-5/Alpha-PM-App.git>

<https://hyperskill.org/learn/step/25083>

<https://business.adobe.com/blog/basics/waterfall>

<https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/>

<https://nextjs.org/docs>

<https://www.mongodb.com/docs/>

<https://mongoosejs.com/docs/documents.html>

<https://zod.dev/>

<https://react.dev/reference/react>

Kan, S. (2002). *Metrics and Models in Software Quality Engineering, Second Edition*.

Bass, L., Clements, P., & Kazman, R. (2013). *Software Architecture in Practice* (3rd ed.). Addison-Wesley.

*Software architecture.* (2016). Willford Pr.

Sommerville Ian. 2010. Software Engineering, 9th Edition, Pearson

Michael Joseph Ingeno. (2018). *Software architect’s handbook : become a successful software architect by implementing effective architecture concepts*. Packt Publishing.

M J D Powell, & Scholtes, S. (2000). *System modelling and optimization : methods, theory, and applications : 19th IFIP TC7 Conference on System Modelling and Optimization, July 12-16, 1999, Cambridge, UK*. Kluwer Academic Publishers.

‌Jose Nathan Kutz. (2013). *Data-Driven Modeling & Scientific Computation : Methods for Complex Systems & Big Data*. Oxford University Press.

‌Thayer, R. H., & Christensen, M. J. (2005). *Software engineering*. Ieee Computer Society Press.

‌Otero, C. (2016). *Software Engineering Design*. CRC Press.

‌Hilburn, T. B., & Massood Towhidnejad. (2020). *Software Engineering Practice*. CRC Press.

Rohit Khurana. (2010). *Software engineering*. Vikas Publishing House.