NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

**School of Computer Science & Engineering**

AY 2023/24 Semester 2

**SC2006/CZ2007 Software Engineering**

# Software Requirements Specifications for Kanbanize

SCSG / Group 1 / Seow's Team


Prepared by:

Augustine Jesuraj Senchia Gladine (U2222429C)
Samuel Tan (U2223180G)
Seet Tze Shin, Cheyenne (U2222685K)
Seow Ming Han Samuel (U2220546F)
Willy Tang Jing Lin (U2222170F)



Date created: 14 April 2024

# Table of Contents

# Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| All | 22/2 | Drafted Functional & Non-functional Requirements | 1.0 |
| Samuel Seow, Senchia & Willy | 23/2 | Drafted Initial Use Case Descriptions | 1.1 |
| Willy Tang | 23/2 | Drafted Initial Use Case Diagram | 1.2 |
| Cheyenne & Senchia | 23/2 | Drafted UI Mockups | 1.3 |
| Samuel Seow | 29/2 | Refined Functional & Non-Functional Requirements | 2.0 |
| Samuel Seow | 1/3 | Refined Use Case Diagram | 2.1 |
| All | 1/3 | Drafted Sequence Diagrams | 2.2 |
| Senchia & Cheyenne | 1/3 | Drafted Initial Dialog Map | 2.3 |
| Samuel Seow | 1/3 | Drafted Initial Key Boundary & Control Classes Diagram | 2.4 |
| Samuel Seow | 19/3 | Refined Use Case Descriptions and Diagram | 3.0 |
| Samuel Seow | 19/3 | Refined Initial Key Boundary & Control Classes Diagram, Dialog Map | 3.1 |
| Samuel Seow | 19/3 | Drafted System Architecture Diagram | 3.2 |
| Samuel Seow | 27/3 | Refined Use Case Descriptions | 3.3 |
| Samuel Seow | 15/4 | Refined Functional & Non-functional Requirements | 4.0 |
| Samuel Seow, Samuel Tan & Cheyenne | 15/4 | Refined Use Case Descriptions | 4.1 |
| Samuel Seow | 16/4 | Refined Class Diagram and Key Boundary & Control Classes Diagram | 4.2 |
| Samuel Seow | 16/4 | Refined Sequence Diagrams and Dialog Map | 4.3 |
| Samuel Seow & Senchia | 16/4 | Drafted Testing | 4.4 |
| Cheyenne & Samuel Tan | 16/4 | Drafted Demo Script and Slides | 4.5 |
| Willy Tang | 18/4 | Refined Demo Script and Slides | 4.6 |

| | | | |
|---|---|---|---|
| Samuel Seow | 19/4 | Drafted Introduction | 5.0 |
| Willy Tang | 20/4 | Recorded Demo Video | 5.1 |
| Willy & Samuel Tan | 20/4 | Drafted External Interface Requirements | 5.2 |
| Willy & Samuel Seow | 20/4 | Refined Non-functional Requirements | 5.3 |
| Willy Tang | 21/4 | Refined System Architecture Diagram | 5.4 |
| Senchia, Cheyenne & Samuel Seow | 21/4 | Drafted Overall Description | 5.5 |
| Samuel Seow | 21/4 | Completed remaining sections and finalized formatting | 6.0 |

# 1. Introduction

## 1.1. Purpose, Outcome and Benefits

**Kanbanize** is a website that enables its users to perform seamless work management through the use of a Kanban board-style interface. Users can customize their workflows, tasks and permissions within their workspace. Our purpose is to visualize workflow and maximize efficiency for work management.

## 1.2. Document Conventions

Main Header Font: Times New Roman
Main Header Font Size: 20
Subsection Header Font: Times New Roman
Subsection Header Font Size: 16
Subsubsection Header Font: Times New Roman
Subsubsection Header Font Size: 14
Subsubsubsection Header Font: Times New Roman
Subsubsubsection Header Font Size: 11
Content Font: Times New Roman
Content Font: 11

## 1.3. Intended Audience and Reading Suggestions

Our target stakeholders are all types of teams and organizations for school projects and business project management.

This Software Requirement Specification (SRS) document is intended for use by the general public. For the coherence of the project, this document also helps the current developers create a logical development process to produce the website. The following sections of this SRS contains detailed information regarding the user requirements, use cases as well as the user interfaces (UI). It is preferable to follow the sequential flow of this document, however, for those who are interested in the functionalities of this application, kindly refer to Section 4, System Features.

## 1.4. Product Scope

Kanbanize allows users to make use of a Kanban board-style interface to visualize workflow and thus maximize efficiency for work management.

Kanbanize allows users a multiple of task card-based features focused on cross-functional collaboration. These features include drag-and-drop task card feature, user management within a project, and user collaboration via a comment system within each task card.

## 1.5. References


1. Architecture of Mean Stack, retrieved from https://www.javatpoint.com/architecture-of-mean-stack

# 2. Overall Description

## 2.1. Product Perspective

**Kanbanize** is a website designed to assist groups and companies to manage their workflow effectively by using an interface similar to a Kanban board. In the workspace, users are able to prioritize their workflows and customize tasks. The primary goal is to maximize productivity for work management by conceptualizing the workflow.

## 2.2. Product Functions

Our website has the following functions:

- Users must be able to register, and then log in to their personalized accounts using their username and password.
- User must be able to reset their password if forgotten.
- User must be able to create, edit and delete multiple projects, including renaming the project.
- User must be able to add, edit, delete, flag multiple tasks in a particular project, including the task title, description and date it is due.
- User must be able to edit the task status by moving the task card from a one column to the preferred column.
- User must be able to add other users to the project.
- User must be able to view all the current users added to the project.
- User must be able to assign or unassign a task to any user that is in the project.
- User must be able to add comments to a certain task such that all the users in the project are able to view it.
- User must be able to log out from their account.

## 2.3. User Classes and Characteristics

Several user classes can be anticipated based on their interaction frequency, technical expertise, and role within the workflow.

1. Team Members: These are the most frequent users of Kanbanize, utilizing core functionalities like creating tasks, and collaborating with comments and updates. They represent various experience levels (regarding their familiarity with using Kanban boards) and technical expertise.

2. Team Leads: These users will be frequent, utilizing features like adding users, assigning tasks, and monitoring team progress through Kanban boards. They typically represent higher experience levels and technical expertise.

3. Administrators: These infrequent users, likely with strong technical knowledge, will be responsible for managing user accounts, security settings, and overall platform administration.

## 2.4. Operating Environment

Kanbanise runs on any web browser on any device connected to the internet. No additional software is required for the operation of the website.

## 2.5. Design and Implementation Constraints

Several factors will constrain the design and implementation of Kanbanize:

1. Security Considerations: All user data requires robust security measures. Secure communication protocols and user authentication are implemented to comply with data privacy regulations.

2. Scalability: The website should function efficiently for teams of various sizes. Our chosen database (MongoDB) scales to accommodate a growing user base and data volume.

3. Responsiveness and Performance: The website should function smoothly on desktop and across various internet connection speeds.

## 2.6. User Documentation

A simple manual with screenshots is included under Section 3.1, in the form of UI Mockups. In general, the application should be intuitive without a need for a manual, however, you may refer to our demo video (Appendix D) for more information on navigating Kanbanize.

## 2.7. Assumptions and Dependencies

<u>Assumptions:</u>

1. Devices must have stable internet connection to access the website.
2. User must enter a valid email that they have access to during registration.

3. Compatibility with various web browsers and operating systems.

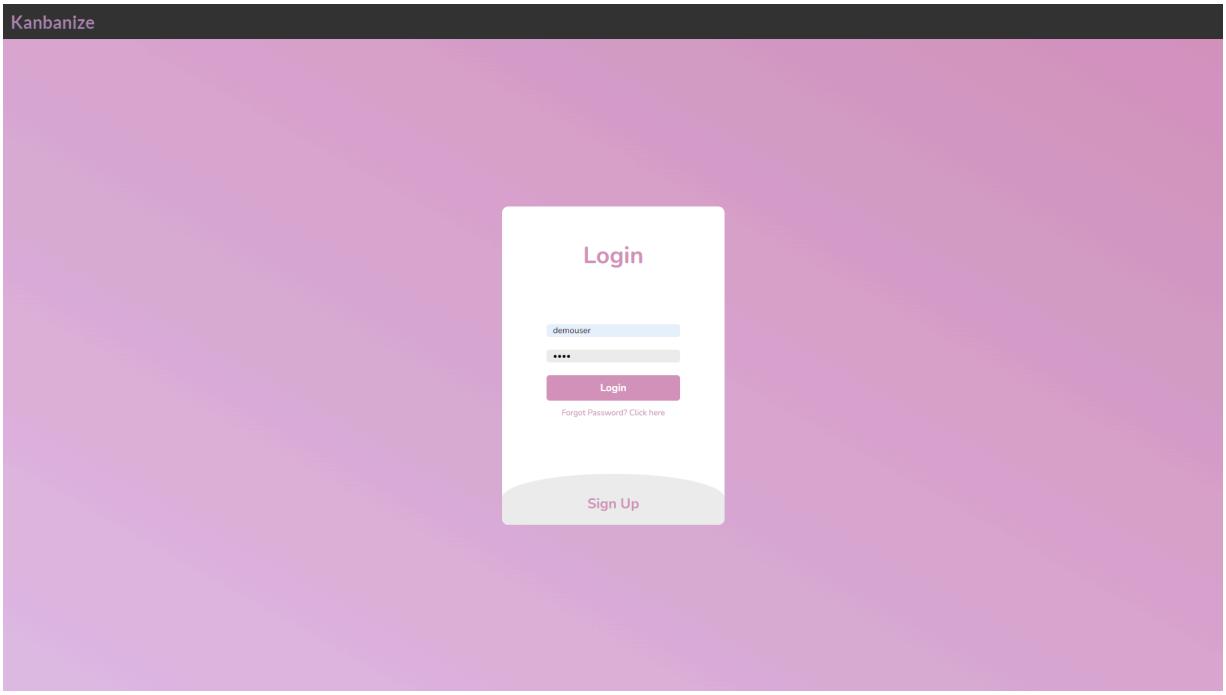
Dependencies:

1. Browser Compatibility: Dependency on cross-browser compatibility for ensuring consistent user experience across different browsers (e.g. Chrome, Firefox, Safari and Microsoft Edge). Testing and addressing certain browser-specific issues are essential dependencies.
2. Hosting Services: Dependency on Angular CLI 17.2.3 & Node 20.11.1, MongoDB 7.0.X and Nodemon are required for hosting services to ensure the website runs properly.

# 3. External Interface Requirements

## 3.1. User Interfaces

**Login Page**



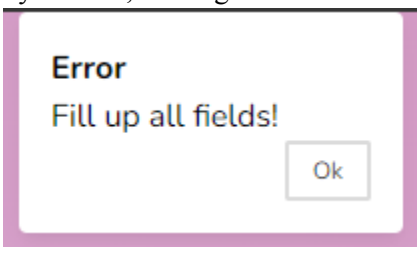

This is the first page the user will encounter after entering our main domain name.

Login Screen has two compulsory fields:

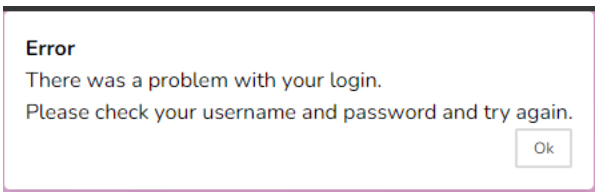
- Username
- Password

In the “Password” field, any characters entered by the user will be hidden. This enforces the security aspect of this application.
Upon successful login, the user will be directed to the Project List Page.

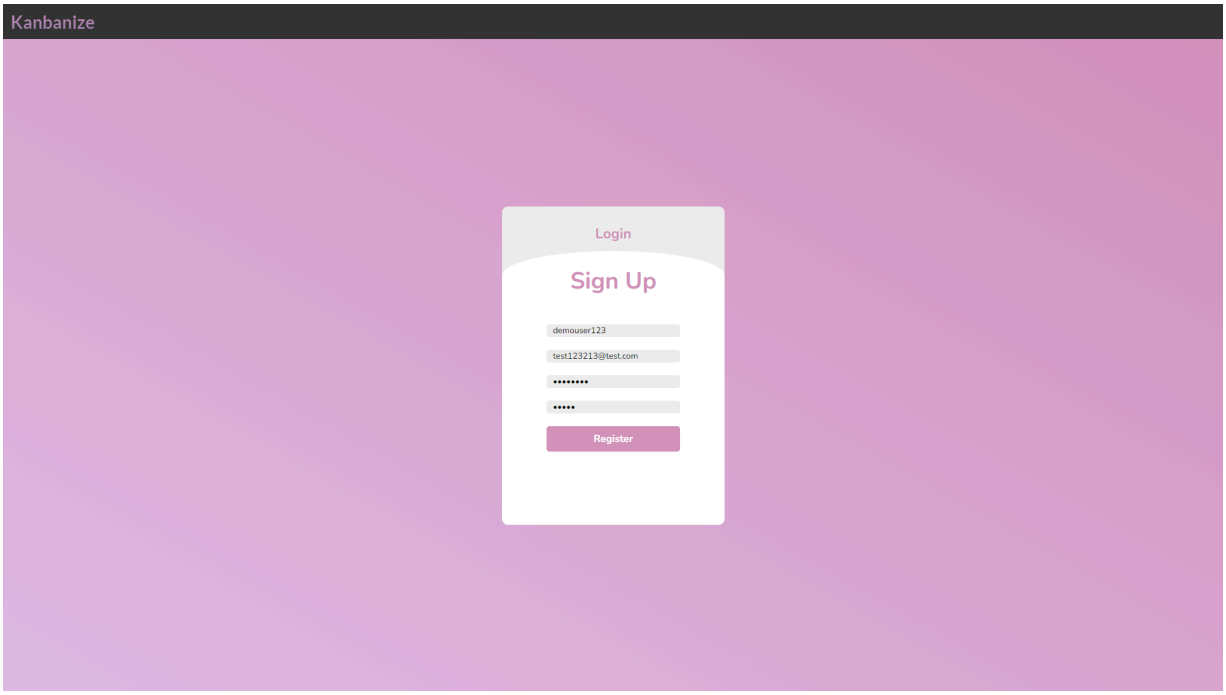
By default, the login will fail if any fields are empty and the following dialog box will be displayed:




If invalid user credentials are entered, the following dialog box will be displayed:




An option to sign up for an account is also available on this screen. When the user clicks on ‘Sign Up’, the Sign Up element will slide up, revealing the registration fields:

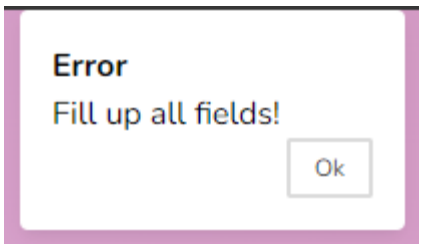



There are four compulsory fields:

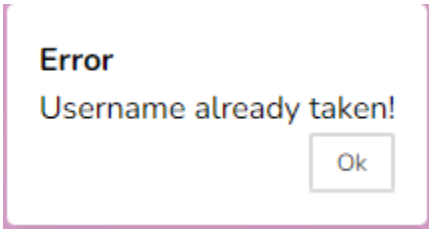
- Username
- Email
- Password
- Confirm Password

In the “Password” and “Confirm Password” field, any characters entered by the user will be hidden. Upon successful registration, the user will be logged in and directed to the Project List Page.

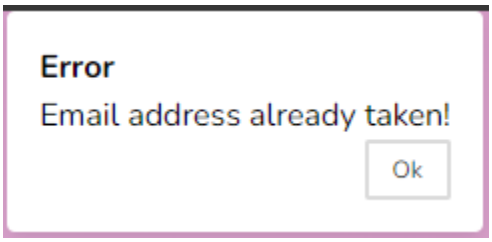
By default, the registration will fail if any fields are empty and the following dialog box will be displayed:



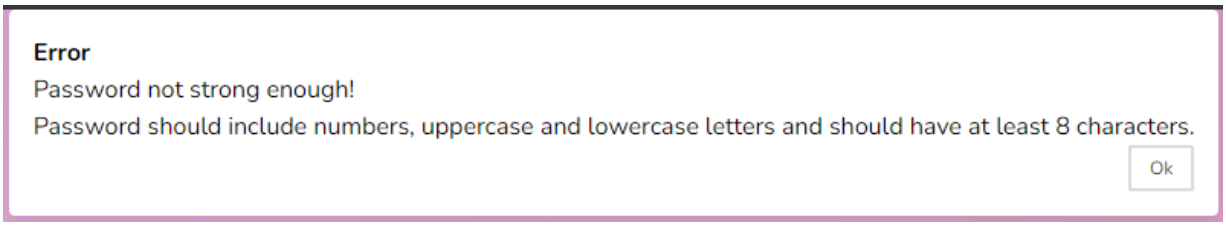

Should the user enter a username that already exists in our database, the following dialog box will appear:



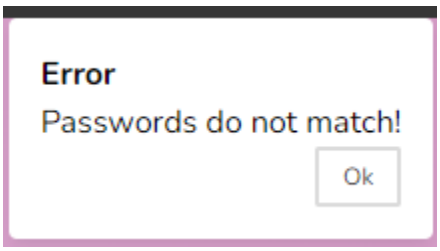

Should the user enter an email address that already exists in our database, the following dialog box will appear:



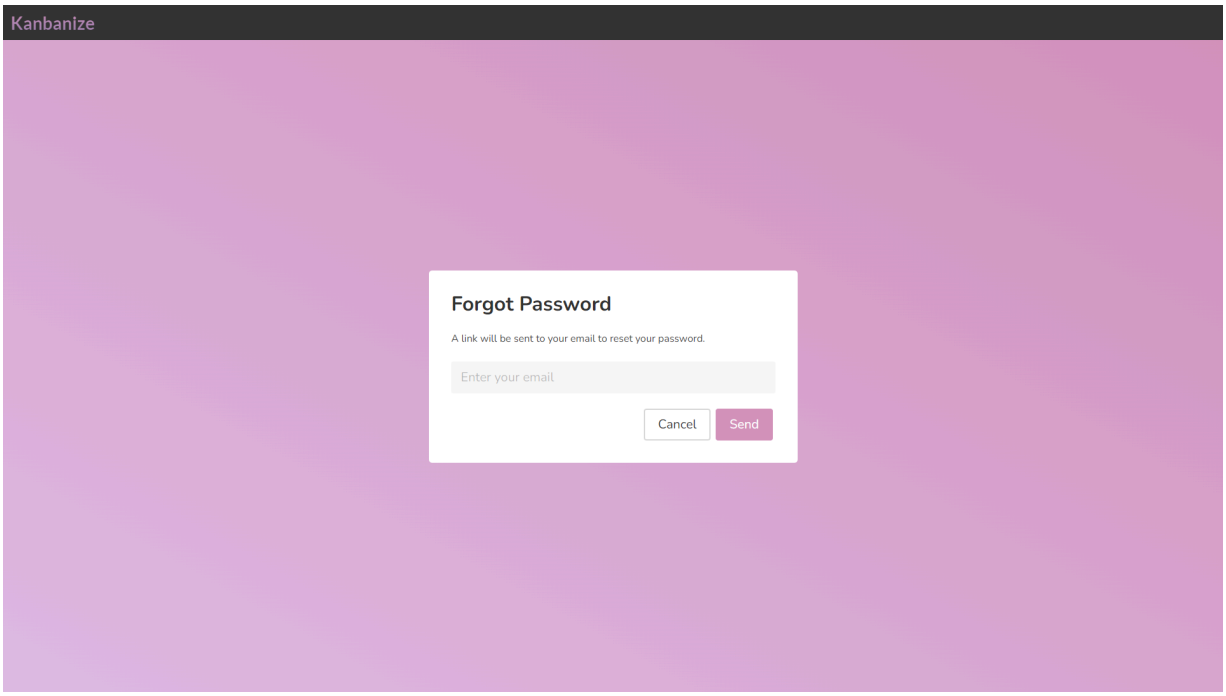

Should the user enter a weak password, the following dialog box will appear:




Should the “Password” and “Confirm Password” field not match, the following dialog box will appear:



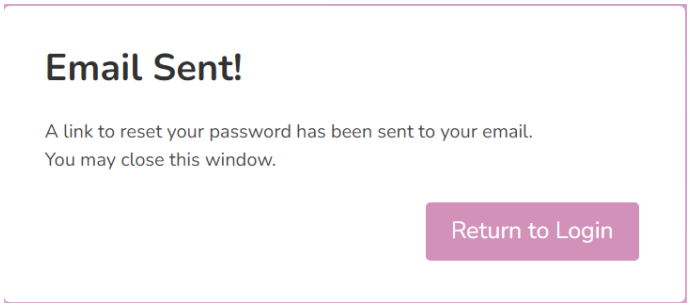
## Forget Password Page




The user can access the Forget Password page by clicking on the “Forgot Password? Click here” button in the Login page.

The Forget Password page contains a compulsory field for the user to enter the email address tied to their account.
The user can cancel the operation and return to the Login page by clicking on the Cancel button.

Upon entering a valid email address and clicking the Send button, an email containing the link to reset password will be sent to the email address. The user will be notified that the email has been sent.



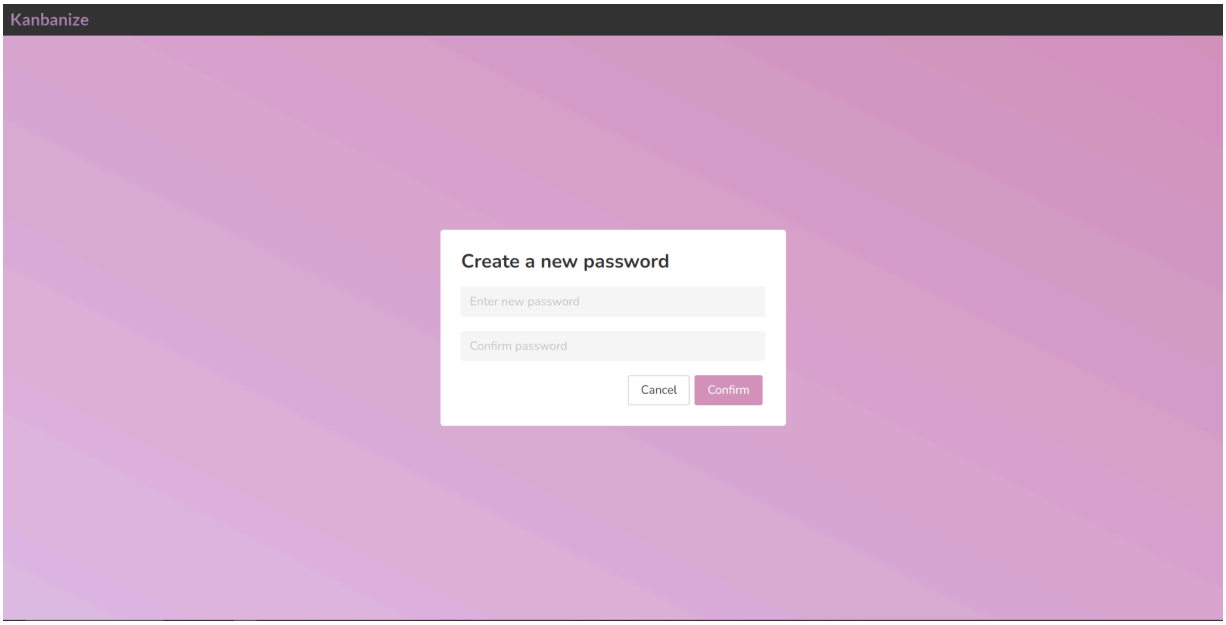

The user can click on the Return to Login button to return to the Login page.

## Reset Password Page

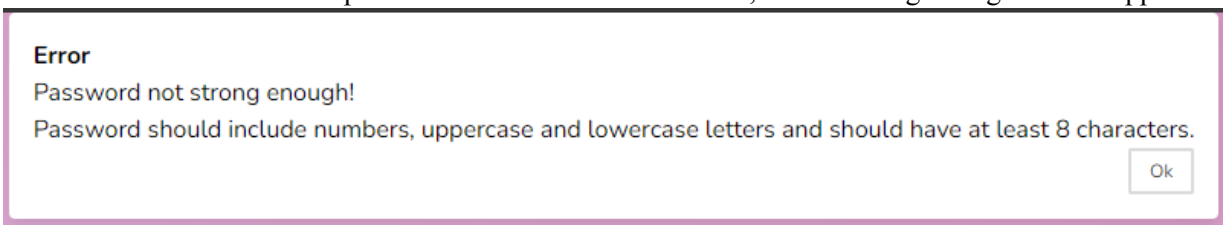

Users can click on the Reset Password button as shown in the email. Users will be redirected to the Reset Password Page.

There are two compulsory fields:

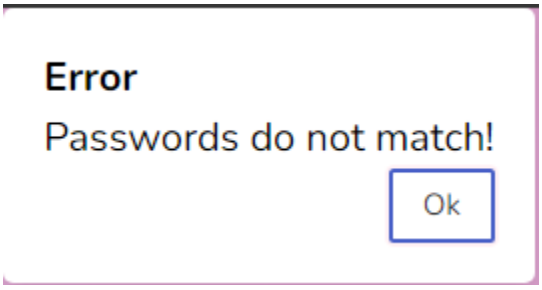
- Password
- Confirm Password



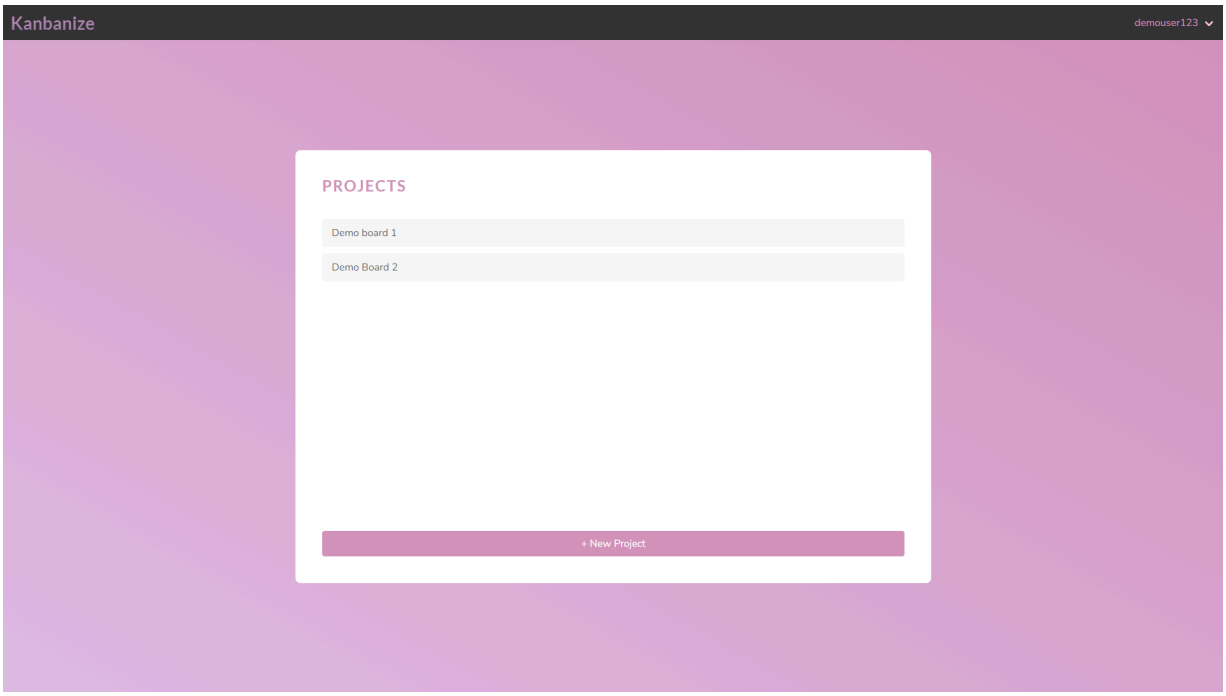

Should the user enter a weak password or leave both field blank, the following dialog box will appear:



Should the “Password” and “Confirm Password” field not match, the following dialog box will appear:



## Project List Page

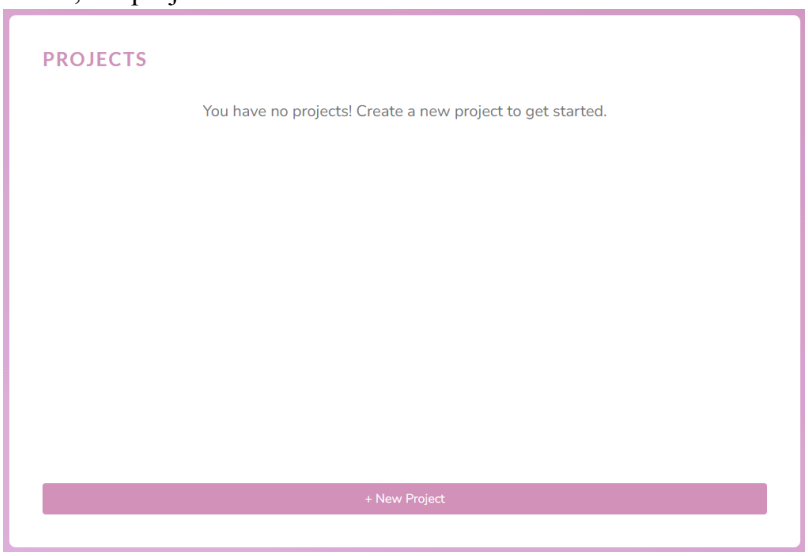



Upon successful login, the user will be directed to the Project List Page.

The Project List Page has the following elements:

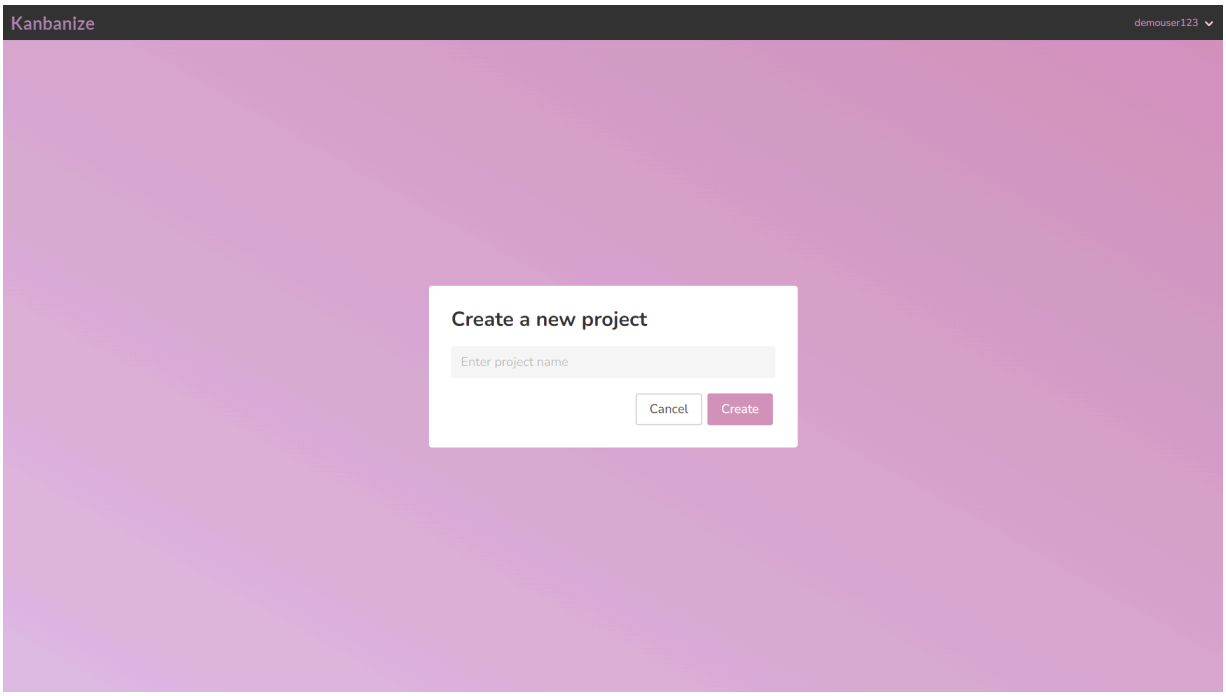
- Project List
- New Project button

The project list displays all the projects that the User has access to. If the User has no boards that he can access, the project list will look as such:




The user can click on the New Project button to create a new project.

## New Project Page



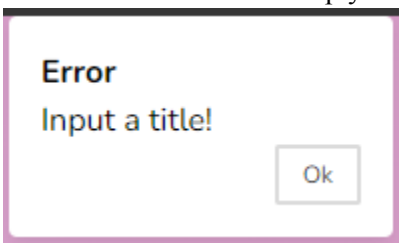

Accessed by clicking on the New Project button on the Project List page.

There is one compulsory field:

- Project Name

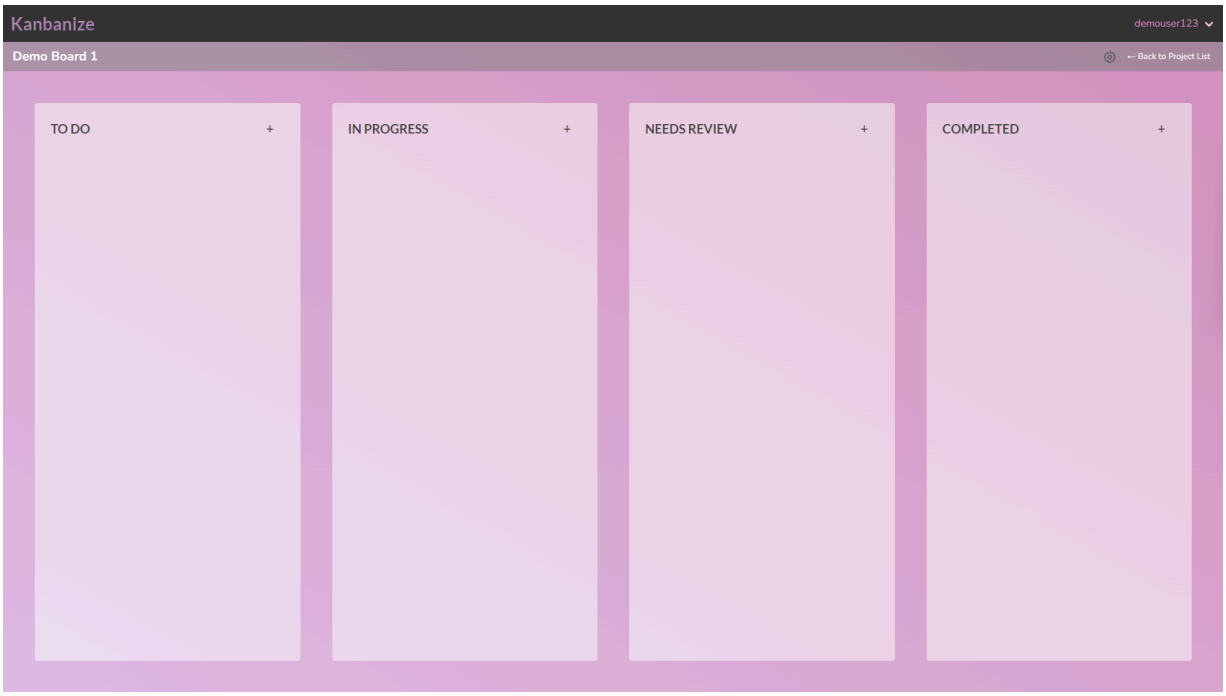
Users are to enter a name for the project to be created and click create. The project will be created and then reflected in the project list.

Should the user enter an empty title, the following dialog box will appear:




Users can access a project board by clicking on a project in the project list.

## Kanban View Page

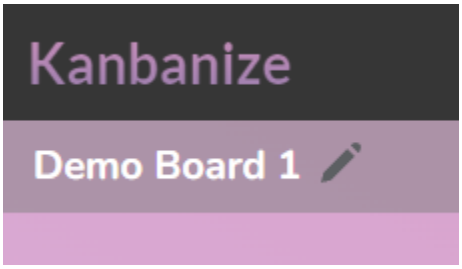



Users can access the Kanban View Page by clicking on a project in the Project List page.
Users can also return to this page from actions carried out on the board.
The board contains the following elements:

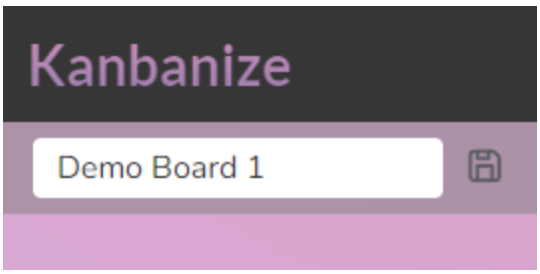
- Board header bar
- Task Columns
- Task Cards

The Board header bar displays the board title and contains a board settings button and a “Back to Project List” button.

To change the board title, the user must bring their cursor close to the title. A edit icon will appear as such:

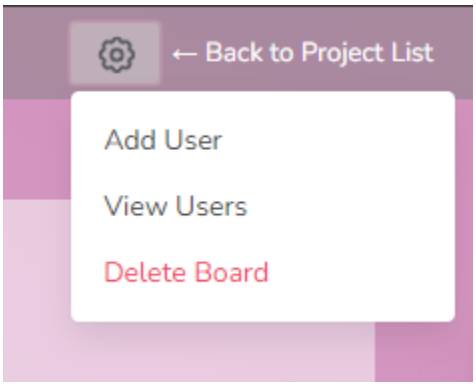



Upon clicking the edit icon, the board title will switch to a text input field, where the user can edit the title.




Users can save the title by clicking on the save icon.

The settings button will trigger a dropdown when clicked. Users can access the Add User, View Users and Delete Board feature here.

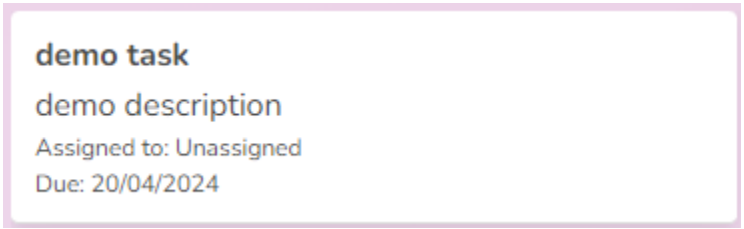



The Back to Project List button will return the user to the Project List page.

The board contains 4 columns, “To Do”, “In Progress”, “Needs Review” and “Completed”.
The header of the bar will display the column title and a “+” icon.


When the user clicks on the “+” icon, they will be directed to the New Task page where they can add a task to the column.

Each column contains a Task Card container. The container contains all the Task Cards that are allocated to the column.

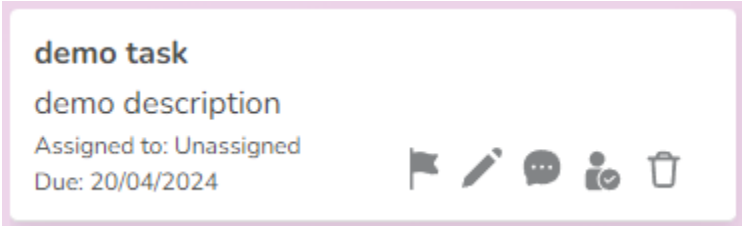



The Task Card contains the following elements:

- Task title
- Task description

- Assigned User
- Due Date

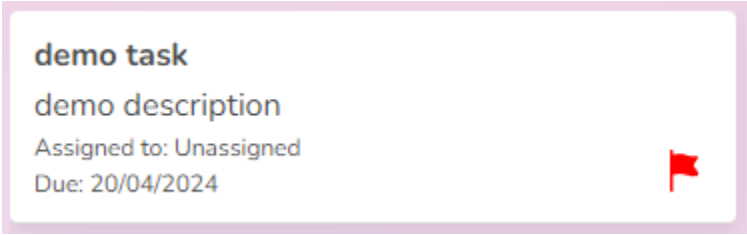
When the user hovers on the Task Card, Task Action buttons will appear as such:




The Task Action buttons are (in order of left to right):

- Task Priority
- Edit Task
- Task Comments
- Assign User
- Delete Task

When the user clicks on the Task Priority button, it should turn red to indicate priority:



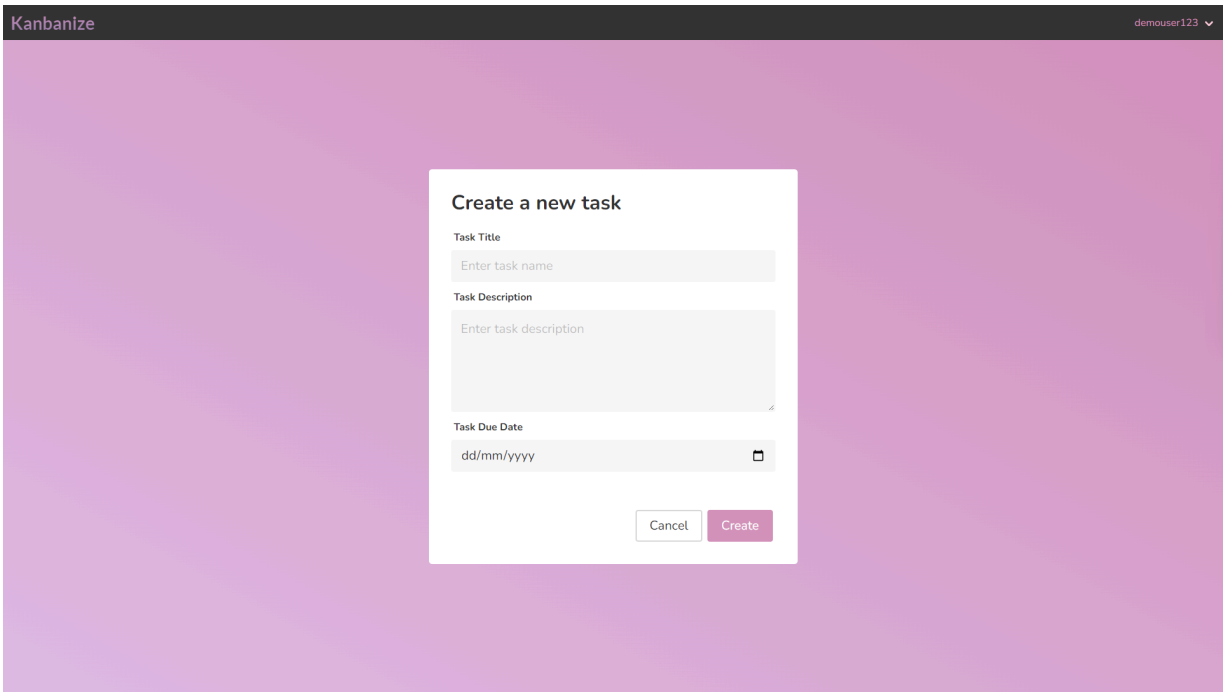

When the user clicks on the Edit Task button, the user will be directed to the Edit Task page.

When the user clicks on the Task Comments button, the user will be directed to the Task Comment page.

When the user clicks on the Assign User button, the user will be directed to the Assign User page.

When the user clicks on the Delete Task button, the user will be directed to the Delete Task page.

## New Task Page




Users can access the New Task page by clicking on the "+" icon in the header of a column in the Kanban View page.

The New Task page contains two compulsory input fields:

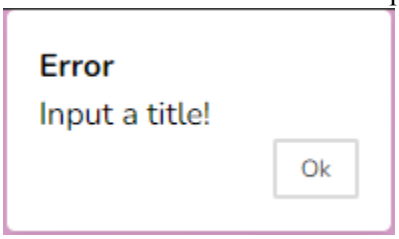
- Task Title (text input)
- Task Due Date (date input)

and an optional input field:

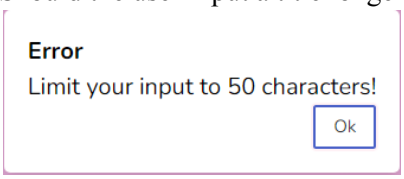
- Text Description (textarea input)

After filling in the fields appropriately and clicking the Create button, the task will be created and the user will be directed back to the Kanban View page.

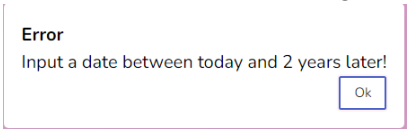
Should the user leave the title empty, the following dialog box will appear:



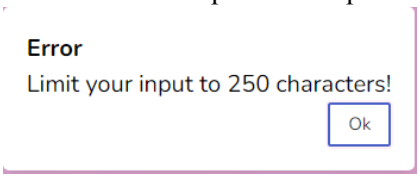

Should the user input a title longer than 50 characters, the following dialog box will appear:



Should the user leave the date empty, input a date before the current date or input a date 2 years after the current date, the following dialog box will appear:

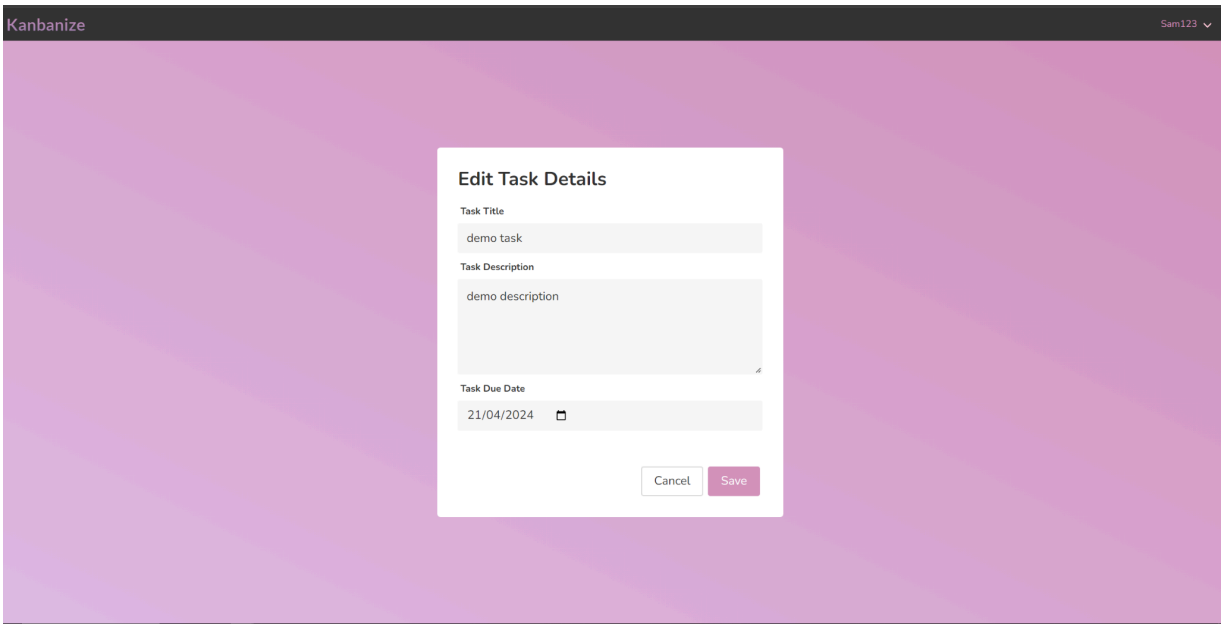



Should the user input a description longer than 250 characters, the following dialog box will appear:




The user can cancel the operation and return to the Kanban View page by clicking on the Cancel button.

## Edit Task Page




Users can edit specific tasks by clicking the Edit Task button on each Task Card in the Kanban View page.

The Edit Task page contains two compulsory input fields:

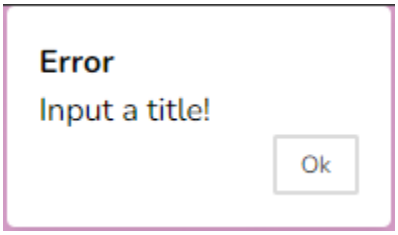
- Task Title (text input)
- Task Due Date (date input)

and an optional input field:

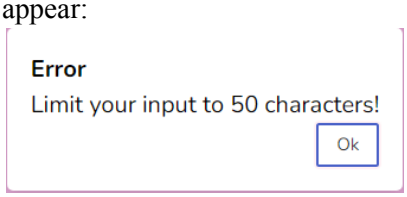
- Text Description (textarea input)

After editing in the fields appropriately and clicking the Save button, the task will be edited and the user will be directed back to the Kanban View page.

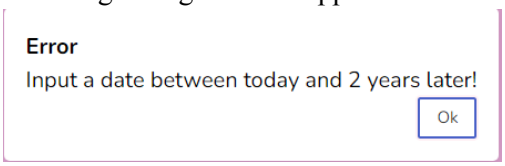
Should the user edit the title such that the title field is empty, the following dialog box will appear:



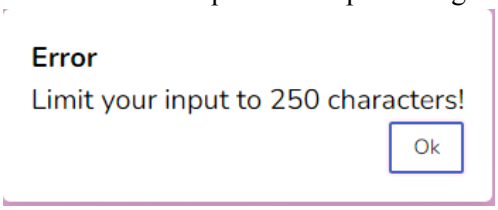

Should the user edit the title such that it is longer than 50 characters, the following dialog box will appear:



Should the user input a date before the current date or input a date 2 years after the current date, the following dialog box will appear:

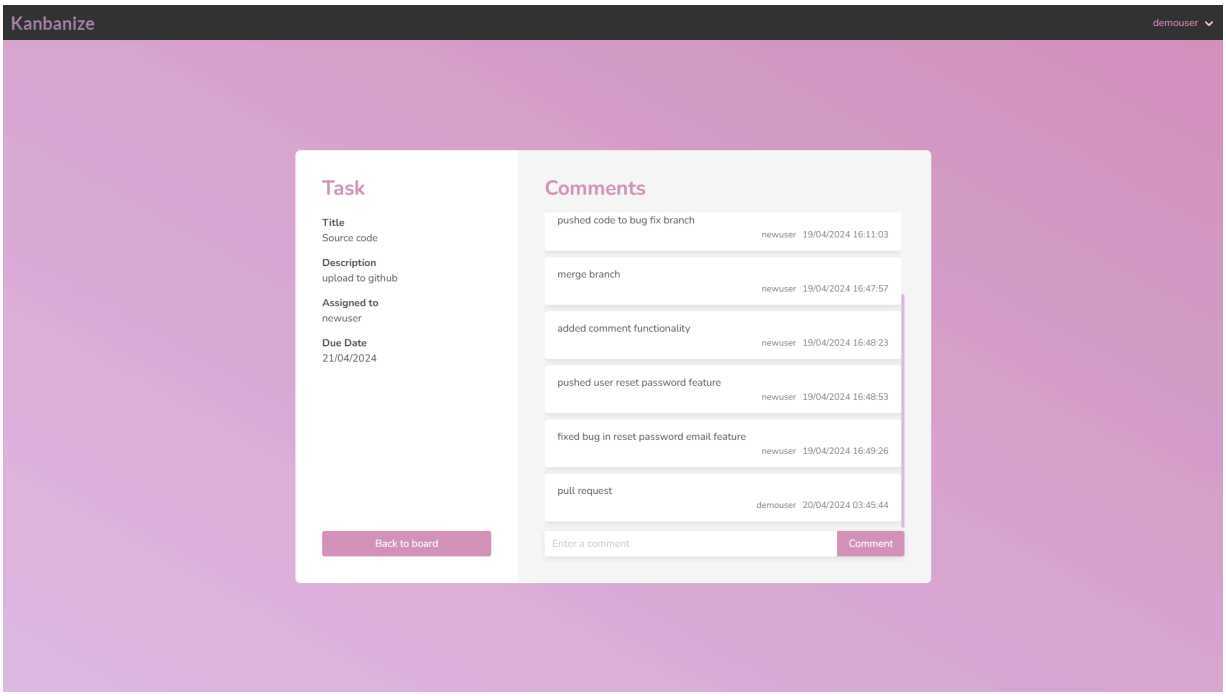



Should the user input a description longer than 250 characters, the following dialog box will appear:




The user can cancel the operation and return to the Kanban View page by clicking on the Cancel button.

## Task Comments Page

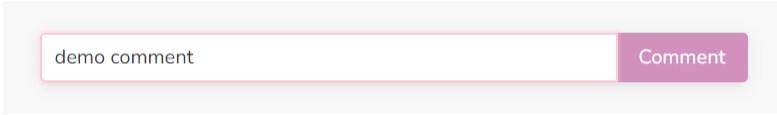



Users can comment on specific tasks by clicking the Task comments button on each Task Card in the Kanban View page.

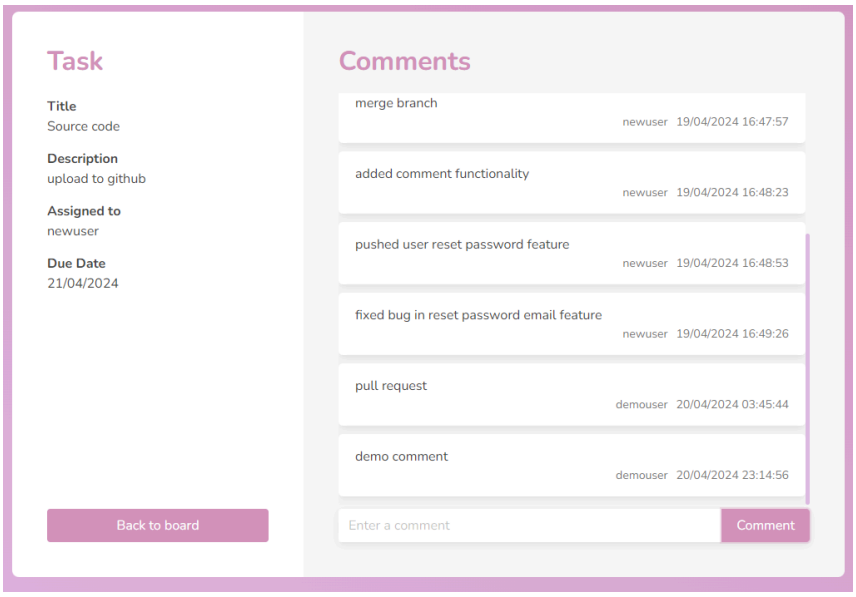
The Task Comments page contains one input fields:

- Enter a comment (text input)

Users can input their comment in the “Enter a comment” input box and click on the Comment button to post the comment to the Task comments page

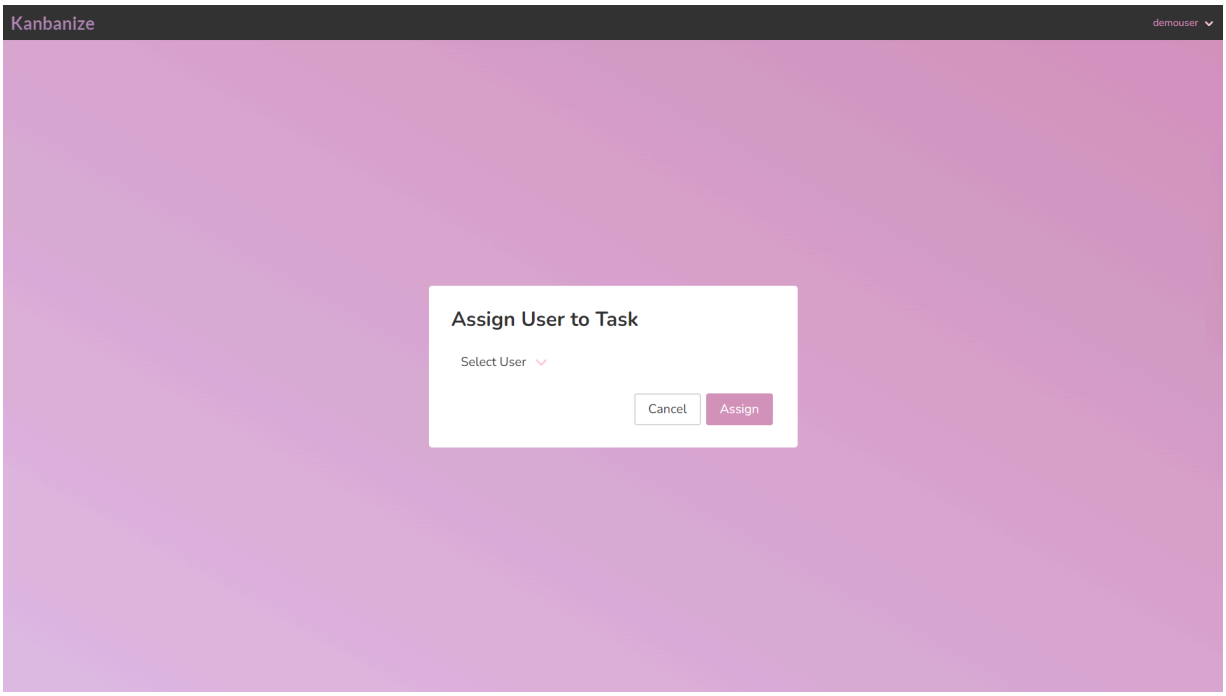



The comment will appear on the Task Comments Page ordered by the date and time they were posted.



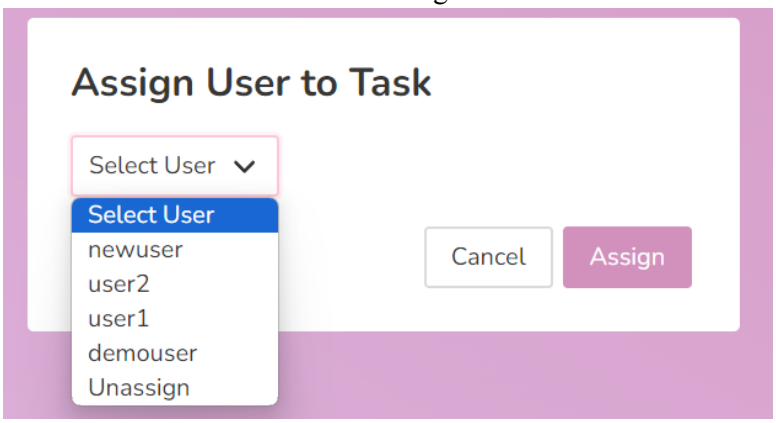

The user can return to the Kanban View page by clicking on the Back to Board button.

## Assign User Page

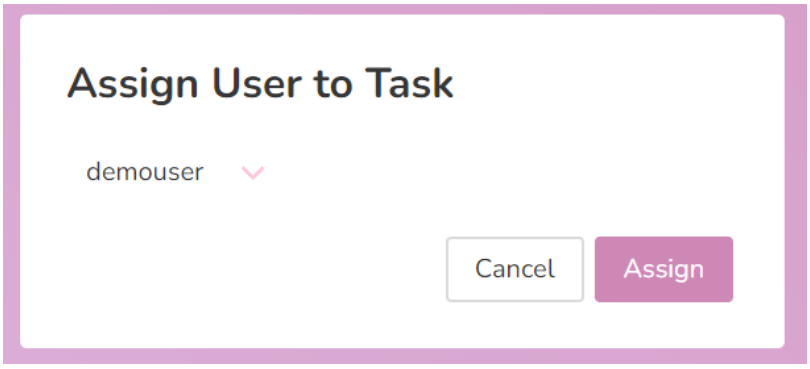



The user can access the Assign User page by clicking on the Assign User button on a Task Card in the Kanban View page.

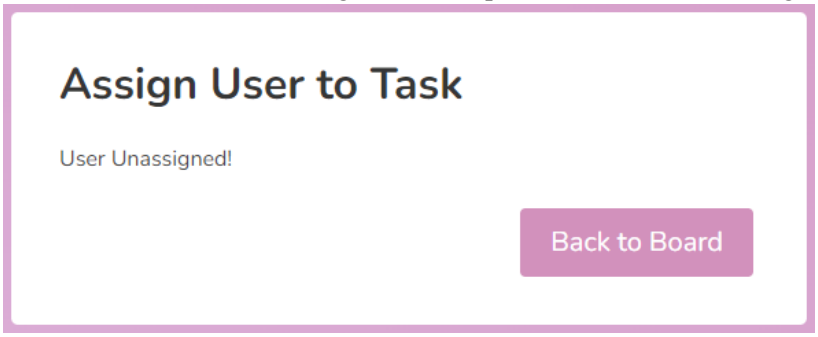
The Assign User page contains a dropdown button that will display the usernames of the users that have access to the board and “Unassign”.



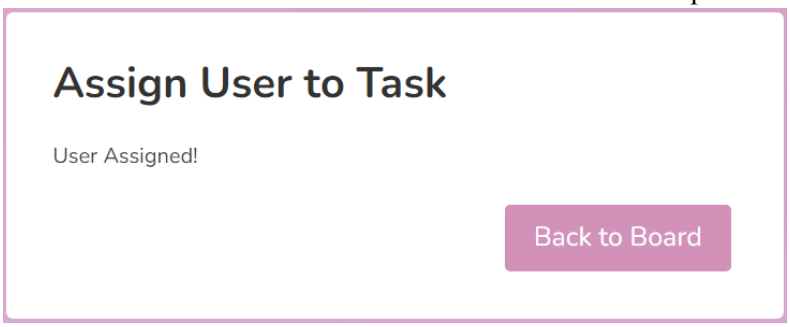

The user can click on a user in the dropdown and click the Assign button to assign the user.




The user can click on Unassign in the dropdown and click the Assign button to Unassign the task.



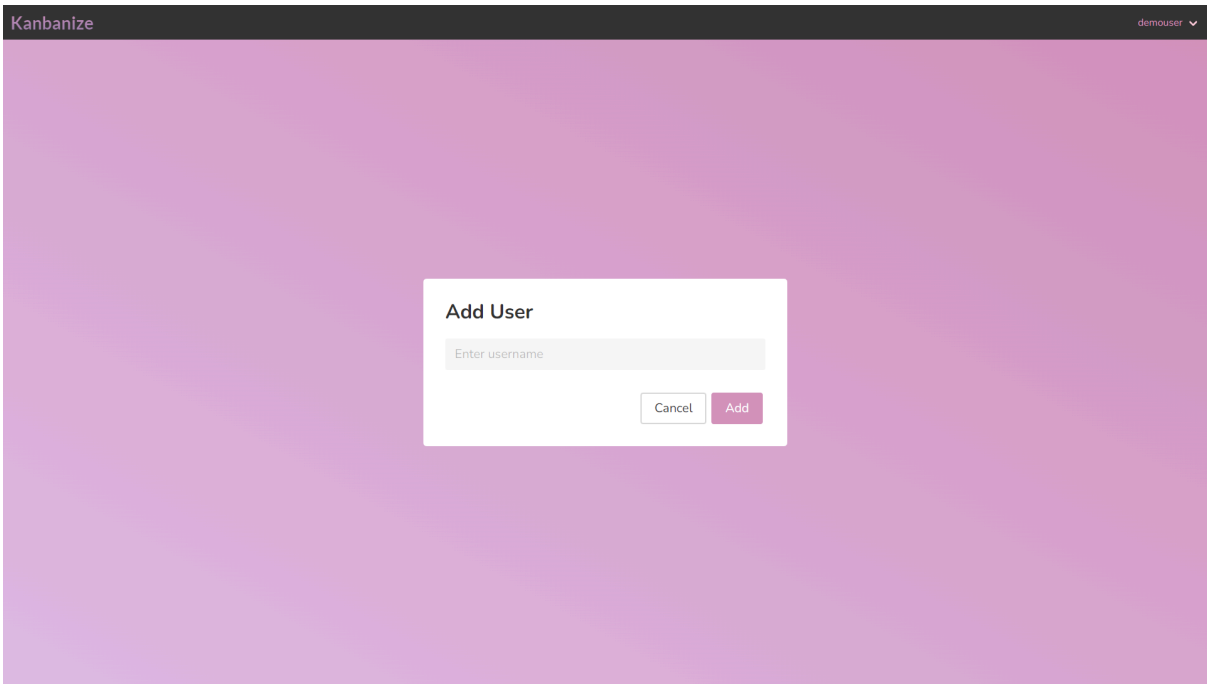

The user will be shown an confirmation of the successful operation as such:




The user can return to the Kanban View page by clicking on the Back to Board button.

Instead of assigning a user, the user can cancel the operation and return to the Kanban View page by clicking on the Cancel button.

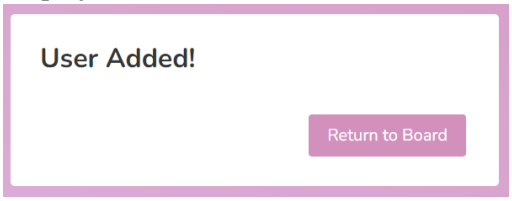
## Add User Page




Users can access the Add User page by clicking on the Add User button in the board settings dropdown in the Kanban View page.
The Add User page contains one compulsory field:

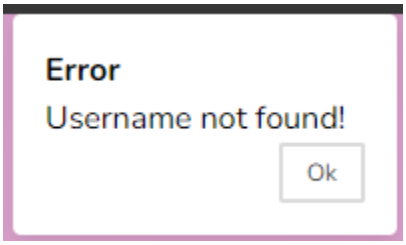
- Username

The user can enter the username of the user they want to add and click the Add button to add them to the project.



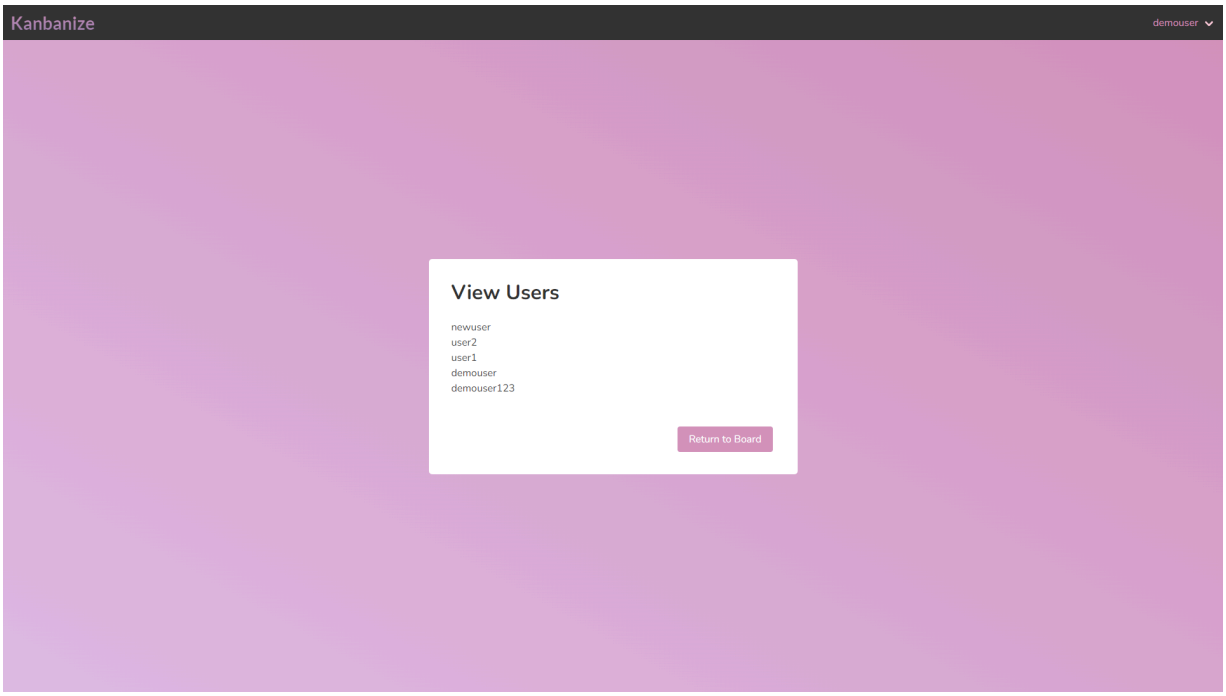

The user will be displayed a confirmation on successful operation. The user can click on the Return to Board button to return to the Kanban View page.

When the user enters a username that does not exist in the database, the following dialog box appears:




The user can cancel the operation and return to the Kanban View Page by clicking on Cancel.

## View User Page



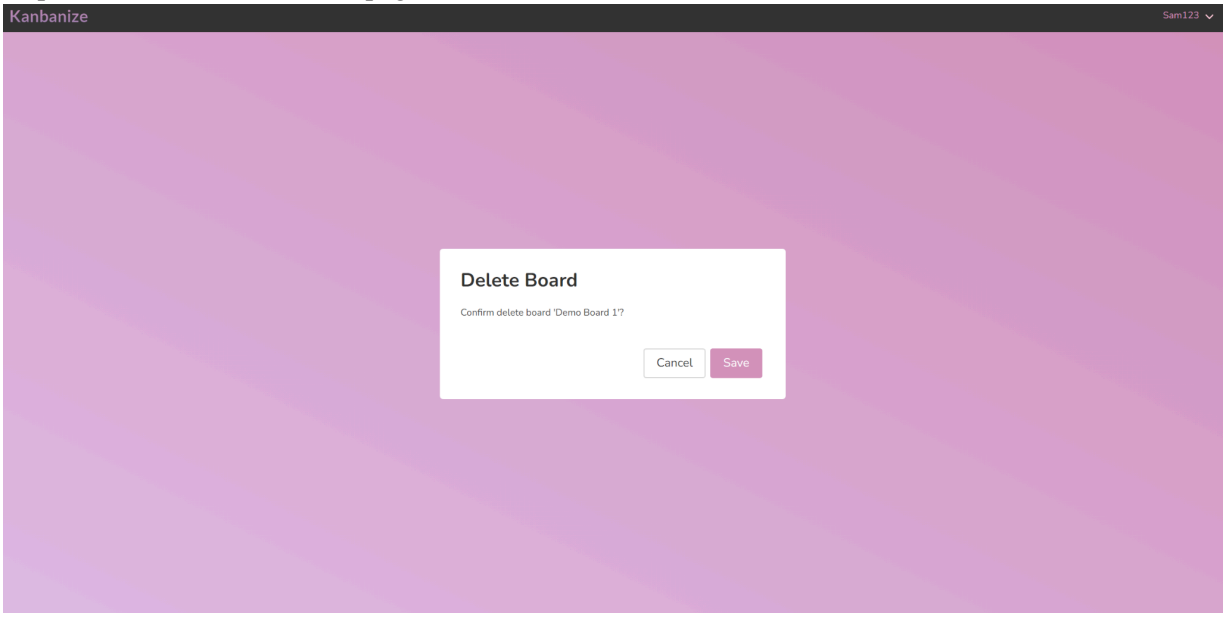

Users can access the View Users page by clicking on the View User button in the board settings dropdown in the Kanban View page.

The View Users page displays the usernames of the users that have access to the project.

The user can return to the Kanban View page by clicking on the Back to Board button.

## Delete Board Page

Users can delete the current board by clicking on the Delete board button in the board settings dropdown in the Kanban View page.




Users can click on the Save button to permanently delete the board from their Project List Page.

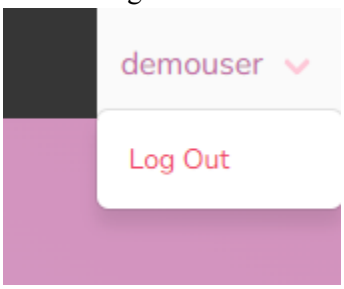
Users will be redirected to the Project List Page once the board is deleted.

**Navigation Bar**


The Navigation bar is present in all our pages. The name of our application is displayed at the left end. Except for in the Login page, Forget Password page and Reset Password page, there is a dropdown button on the right end displaying the username of the user currently logged in.

The dropdown can be triggered on click and will display the following buttons:

- Log Out




Upon clicking the Log Out button, the user will be logged out of the system and returned to the Login page.

## 3.2. Hardware Interfaces

To use Kanbanize, any device with an internet connection capability is a prerequisite for its operation. Users can change the interface displayed by clicking on display elements either through physical touch (e.g. mobile devices) or a mouse (e.g. laptop/desktops). Users are also required to type in their email, password, board names, task title, description and comments with their keyboard.

It is not recommended for users to use Kanbanize with a mobile device as it is not developed with mobile users in mind.

## 3.3. Software Interfaces

Kanbanize is compatible with any device which has access to an internet connection. Kanbanize can be accessed through the user's browser such as Google Chrome, Safari or Firefox.

## 3.4. Communications Interfaces

Kanbanize will be available via accessing the website. Users would also be able to access all of its features within the website, with the exception of password reset which requires users to authenticate via the email address used to create their account.

User's data will be maintained in the server's database with HTTP requests facilitating the interaction between the website and the database.

# 4. System Features

## 4.1. Login Page

4.1.1. The system must have a Login interface within the Login Page.

4.1.1.1. The system must display one text field for Username.

4.1.1.2. The system must display one text field for Password.

4.1.2. The system must display a “Login” button.

4.1.2.1. When the user clicks “Login”, the system must validate that all required fields are not empty.

4.1.2.2. When the user clicks “Login”, the system must validate whether the username is valid.

4.1.2.2.1. If it is invalid, the system must prompt the user to “Check your username and try again.”

4.1.2.3. When the user clicks “Login”, the system must validate whether the corresponding password is valid.

4.1.2.3.1. If it is invalid, the system must prompt the user to “Check your password and try again.”

4.1.2.4. When the user enters a username in the database and the correct password, the system must redirect the user to the Project List Page.

4.1.3. The system must have a Sign Up interface within the Login Page.

4.1.3.1. The system must display one text field for Username.

4.1.3.2. The system must display one text field for Email Address.

4.1.3.3. The system must display one text field for Password.

4.1.3.4. The system must display one text field for Confirm Password.

4.1.4. The system must display a “Register” button.

4.1.4.1. When the user clicks “Register”, the system must validate that all required fields are not empty.

4.1.4.1.1. If any field is empty, the system must prompt the user to “Fill up all fields.”

4.1.4.2. When the user clicks “Register”, the system must validate that the username has not been used before.

4.1.4.2.1. If the username has been used before, the system must prompt the user “Username already taken.”

4.1.4.3. When the user clicks “Register”, the system must validate that the email address entered is valid.

4.1.4.3.1. If the email address is invalid, the system must prompt the user to “Enter a valid email address.”

4.1.4.3.2. An input which does not contain the characters “@” and “.” in the email address field will prompt a failed registration attempt.

4.1.4.4. When the user clicks “Register”, the system must validate that the password entered must meet the following requirements:

4.1.4.4.1. A minimum of 8 characters in length.

4.1.4.4.2. Contains at least one character from three of the following categories: Uppercase letter (A-Z) Lowercase letter (a-z) Digit (0-9).

4.1.4.4.3. If the password does not satisfy the requirements, the system must prompt the user “Password not strong enough.” and include the default password validation practices.

4.1.5. The system must allow users to reset their account passwords through a “Forgot Password” button.

## 4.2. Forget Password Page

4.2.1. The system must have a Forget Password interface within the Forget Password Page.

4.2.1.1. The system must display one text field for the user to enter their registered email address.

4.2.1.1.1. When the user enters an email that is not in the database, the system must prompt the user “No users with email found.”

4.2.1.1.2. When the user enters an email that is in the database, the system must display “Email sent!”

4.2.1.2. The system must send an email to the inputted email address with instructions on how to change their password.

4.2.1.2.1. When the user clicks “Reset Password” in the email, they must be redirected to the Reset Password Page.

4.2.1.2.2. The system must display a text field for the user to enter their new password.

4.2.1.2.3. The system must require the user to type in the new password twice to ensure the accuracy of the password.

4.2.1.2.4. The system must follow the same password validation practices as the Registration Page.

## 4.3. Navigation Bar

4.3.1. The system must have a navigation bar where the user can log out of their account.

4.3.1.1. The system must display the Username of a currently logged in user.

4.3.1.1.1. When the user clicks on their username, the system must display “Log out”.

## 4.4. Project List Page

4.4.1. The system must have a Project List Page.

4.4.1.1. The system must display a button to create new projects.

4.4.1.1.1. The system must display one text field for Project Name.

4.4.1.2. The system must display the current projects the user is a part of, including projects that the user has been added to.

4.4.1.3. The system must redirect the user to the Kanban View Page of the project that the user selected.

## 4.5. Kanban View Page

4.5.1. The system must have a Kanban View Page, which must contain the main functionalities of the Kanban Board that the user can use.

4.5.1.1. The system must allow the user to manage team members of the project.

4.5.1.1.1. The system must allow the user to add new users to the project.

4.5.1.1.2. The system must allow the user to view users of the project.

4.5.1.2. The system must allow the user to delete the current board.

4.5.1.3. The system must display 4 distinct columns to represent different stages of the workflow. (TO DO, IN PROGRESS, NEEDS REVIEW, COMPLETED)

4.5.1.4. The system must be able to display task cards.

4.5.1.4.1. Users must be able to create, edit or delete existing task cards.

4.5.1.4.2. Users must be able to add/edit the title and description of a task.

4.5.1.4.3. Users must be able to set/edit a deadline for a task.

4.5.1.4.4. Users must be able to flag a task as high priority.

4.5.1.4.5. Users must be able to add comments for a task card.

4.5.1.4.6. Users must be able to assign members of the project to a task card.

4.5.1.4.7. Users must be able to drag and drop tasks between different columns to update workflow progress.

# 5. Other Non-functional Requirements

## 5.1. Performance Requirements

5.1.1. When the user uses task-based functionalities, changes must be reflected within at least 1 minute.

5.1.2. The database must be updated upon every modification made by users.

5.1.3. The system must be free of any deadlocks.

## 5.2. Reliability Requirements

5.2.1. The system should not crash more than thrice daily.

## 5.3. Security Requirements

5.3.1. The system should prevent unauthorized personnel from accessing user data.

5.3.2. The system must not share the user's data with any external organization.

5.3.3. The system should only allow registered users to view their account information.

5.3.4. The system must protect user data, including but not limited to the following:

5.3.4.1. Email Address

5.3.4.2. Password

5.3.4.2.1. The system must use bcrypt's salt and hash method to encrypt all passwords in our database.

## 5.4. Scalability Requirements

5.4.1. The website must be able to support 10 active users while meeting all real-time requirements.

5.4.2. The website must be able to support at least 20 tasks on a board while meeting all real-time requirements.

## 5.5. Usability Requirements

5.5.1. Features on the website must be responsive within 3 seconds of being clicked.

## 5.6. Software Quality Attributes

5.6.1. The system must retrieve and display information from the database correctly.

5.6.2. The system must be able to continue to function with minimal maintenance.

5.6.3. The system must ensure uninterrupted operation and not experience crashes.

5.6.4. Users must be able to use the website without guidance from a manual.

5.6.5. The user interface must be user-friendly with clear button placements and succinct prompts such that users will not get lost within the application.

## 5.7. Business Rules

5.7.1. Users should be able to use all of Kanbanize's features once they are logged in.

## 5.8. Design Patterns

### 5.8.1. Observer Pattern

Observer pattern is a software design pattern in which an object, named the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods.

Problem:

In the design phase, we noticed that many of our methods were asynchronous as they were requests to our web server. Let's say: the size of our app grows and many features need to be loaded into a page. To retrieve this data we would need to call many asynchronous operations, and this would result in a long loading time if we await for each operation.

Solution:

Therefore, we use observables to handle such operations. By subscribing to the asynchronous operations, we can queue multiple asynchronous operations and when they finally complete, the subscriptions will be notified and update the viewmodel. As such, the page will appear to be more responsive.
An example of this: Drag-and-drop feature of task cards.

### 5.8.2. Service Layer Pattern

Service layer is an architectural pattern, applied within the service-orientation design paradigm, which aims to organize the services, within a service inventory, into a set of logical layers. Services that are categorized into a particular layer share functionality.

Problem:

Our frontend and backend systems are separated and communicate with each other through API calls. All our functionalities utilize this, especially the Logout feature which is present in every page after the user logs in.

Solution:

Therefore we made use of the Service Layer Pattern. We have a web request service to handle our RESTful API requests to our backend server. We also have 2 additional services to handle the requests related to tasks and authentication. These 2 services utilize the web request service. These services handle CRUD (create, read, update, delete) operations and manage data synchronization between the frontend and backend for our tasks, boards and users.

### 5.8.3. Dependency Injection

Dependency injection is a programming technique that makes a class independent of its dependencies. It achieves that by decoupling the usage of an object from its creation.

Reasoning:

We used dependency injection to achieve loose coupling between our service layer and components. Services are injected into components using Angular's dependency injection feature.

Benefits:

Coupled with the service layer pattern, this means that any operations related to API calls are relegated to the service layer, which enforces strong SOLID design principles. This ensures our program is modular, extensible and we would have a codebase that is easier to maintain.

## 5.9. System Architecture

### 5.9.1. MEAN Stack

MEAN is an acronym for:

- MongoDB
- Express.js
- Angular
- Node.js

Reasoning:

We used the MEAN stack which is a Typescript and Javascript based framework for developing scalable web apps. MongoDB is used for our database, Express is used for creating our API routes, Angular powers the frontend of our application and Node is used to build our web server.

With this, our system takes a layered approach, with each tech responsible for a layer.
The client uses the frontend of the app and if needed Angular will make a HTTP request to the server which is then handled by Node. Based on what the request is, the request is then routed and Express will then make the relevant query to the database. The result is then sent back to the frontend through a server response. This response will then be parsed by the frontend and then displayed to the user.

### 5.9.2. Model-View-ViewModel

The MVVM architectural style is an extension of the Model-View-Controller (MVC) style. MVC separates an app into three components: model, view, and controller, while MVVM adds a ViewModel layer between the view and the model.
In MVC, the controller handles user input and updates the view accordingly, while in MVVM, the ViewModel acts as a mediator between the view and the model, managing data binding and providing a clean separation of concerns.

Reasoning:

Angular follows a Model-View-ViewModel rather than a conventional Model-View-Controller. The model is our Typescript model classes and MongoDB database, the view is the HTML template and CSS styling, and the viewmodel is the Typescript component classes.

### 5.9.3. User Authentication

User Authentication is performed via:

1) Bcrypt's salt and hash method
2) JSON web tokens (JWT)

Method:

Firstly, passwords are hashed in our database using bcrypt's salt and hash method. We use bcrypt's compare method to check login credentials. Upon successful login, we create a session. We will generate a random hex string for each successful login and store it in our database. Using this hex string, we create an access token by signing a JWT with a secret string and return this alongside the UserId and refresh token. These will then be stored in the browser's local storage and appended to the header of every request via an interceptor. Every protected API route will check for a valid access token. If the access token is expired, a new access token will be generated and the request will be rerun. This gives us the ability to manage user login sessions in the future, similar to the likes of Whatsapp, Telegram and many other apps.

# 6. Testing

## 6.1. Black Box Testing

### 6.1.1. Registration

| **Location** | Login Interface → Register Interface |
|---|---|

**General cases**

| **ID** | **Input** | **Oracle** | **Log** |
|---|---|---|---|
| #1 | Register with valid username, email address and password | System redirects to Project List Page. | System redirects to Project List Page. |
| #2 | Register without populating username, email address, password and/or confirm password fields | System displays error message, “Fill up all the fields!” prompting user. | System displays error message, “Fill up all the fields!” prompting user. |
| #3 | Register with invalid email address format | System displays error message, “Enter a valid email address!” prompting user to re-enter email address. | System displays error message, “Enter a valid email address!” prompting user to re-enter email address. |
| #4 | Register with an existing email address | System displays error message “email address already taken!” prompting user to change their username. | System displays error message “email address already taken!” prompting user to change their username. |

| | | | |
|---|---|---|---|
| #5 | Registers with an existing username | System displays error message “username already taken!” prompting user to change their username. | System displays error message “username already taken!” prompting user to change their username. |
| #6 | Registers without meeting password criteria | System displays error message “ Password not strong enough!<br>Password should include numbers, uppercase and lowercase letters and should have at least 8 characters.” prompting user to choose another password that follows the criteria. | System displays error message “Password not strong enough!<br>Password should include numbers, uppercase and lowercase letters and should have at least 8 characters.” prompting user to choose another password that follows the criteria. |
| #7 | Registers with different inputs in “Password” and “Confirm Password” fields | System displays error message, “Passwords do not match!” prompting user to recheck. | System displays error message, “Passwords do not match!” prompting user to recheck. |

**Specific cases**

| ID | Username | Email | Password | Confirm Password | Expected Output | Test Result |
|---|---|---|---|---|---|---|
| #1 | {Empty} | {Empty} | {Empty} | {Empty} | Error! Fill up all the fields! | Error! Fill up all the fields! |
| #2 | {Empty} | username@gmail.com | P@ssw0rd | P@ssw0rd | Error! Fill up all the fields! | Error! Fill up all the fields! |

| | | | | | | |
|---|---|---|---|---|---|---|
| #3 | username | {Empty} | P@ssw0rd | P@ssw0rd | Error! Fill up all the fields! | Error! Fill up all the fields! |
| #4 | username | username@gmail.com | {Empty} | P@ssw0rd | Error! Fill up all the fields! | Error! Fill up all the fields! |
| #5 | username | username@gmail.com | P@ssw0rd | {Empty} | Error! Fill up all the fields! | Error! Fill up all the fields! |
| #6 | username | user@gmail | P@ssw0rd | P@ssw0rd | Error! Enter a valid email address! | Error! Enter a valid email address! |
| #7 | registeredUsername | username@gmail.com | P@ssw0rd | P@ssw0rd | Error username already taken! | Error username already taken! |
| #8 | username | registeredEmail@gmail.com | P@ssw0rd | P@ssw0rd | Error! Email address already taken! | Error! Email address already taken! |
| #9 | username | username@gmail.com | password | password | Error! Password not strong enough!<br><br>Password should include numbers, uppercase and lowercase letters and should have at least 8 characters. | Error! Password not strong enough!<br><br>Password should include numbers, uppercase and lowercase letters and should have at least 8 characters. |

| | | | | | | |
|---|---|---|---|---|---|---|
| #10 | username | username@gmail.com | P@ssw0rd | P@ssw0rd | System redirects to Project List Page. | System redirects to Project List Page. |
| #11 | username | username@gmail.com | P@ssw0rd | password | Error! Passwords do not match! | Error! Passwords do not match! |

6.1.2. Login

| **Location** | Login Interface |
|---|---|

**General cases**

| **ID** | **Input** | **Oracle** | **Log** |
|---|---|---|---|
| #1 | Login with registered username and valid password | System redirects to Project List Page. | System redirects to Project List Page. |
| #2 | Login with unregistered username | System displays error message “There was a problem with your login. Please check your username and password and try again.” prompting user to recheck their login credentials. | System displays error message “There was a problem with your login. Please check your username and password and try again.” prompting user to recheck their login credentials. |
| #3 | Login with registered username but invalid password | System displays error message “There was a problem with your login. Please check your username and password and try again.” prompting user to recheck their login credentials. | System displays error message “There was a problem with your login. Please check your username and password and try again.” prompting user to recheck their login credentials. |
| #4 | Login with invalid username and password | System displays error message “There was a problem with your login. Please check your username and password and try again.” prompting user to recheck their login credentials. | System displays error message “There was a problem with your login. Please check your username and password and try again.” prompting user to recheck their login credentials. |

| | | | |
|---|---|---|---|
| #5 | Login with username or/and password field(s) unpopulated | System displays error message, “Fill up all the fields!” prompting user. | System displays error message, “Fill up all the fields!” prompting user. |

**Specific cases**

| ID | Username | Password | Expected Output | Test Result |
|---|---|---|---|---|
| #1 | registered username | P@ssw0rd | System redirects to Project List Page. | System redirects to Project List Page. |
| #2 | unregistered | P@ssw0rd | Error! There was a problem with your login. Please check your username and password and try again | Error! There was a problem with your login. Please check your username and password and try again |
| #3 | registered username | password | Error! There was a problem with your login. Please check your username and password and try again | Error! There was a problem with your login. Please check your username and password and try again |
| #4 | invalid username | password | Error! There was a problem with your login. Please check your username and password and try again | Error! There was a problem with your login. Please check your username and password and try again |
| #5 | {Empty} | P@ssw0rd | Error! Fill up all the fields! | Error! Fill up all the fields! |
| #6 | registered username | {Empty} | Error! Fill up all the fields! | Error! Fill up all the fields! |

| | | | | |
|---|---|---|---|---|
| #7 | {Empty} | {Empty} | Error! Fill up all the fields! | Error! Fill up all the fields! |

### 6.1.3. Forget Password

| **Location** | Register Interface → Forgot Password Interface → Login Interface |
|---|---|

**General cases**

| **ID** | **Input** | **Oracle** | **Log** |
|---|---|---|---|
| #1 | Change password with valid password | System displays password reset message and prompts user to return to login page. | System displays password reset message and prompts user to return to login page. |
| #2 | Change password without meeting password criteria | System displays error message “Password not strong enough!<br>Password should include numbers, uppercase and lowercase letters and should have at least 8 characters.” prompting user to choose another password that follows the criteria. | System displays error message “Password not strong enough!<br>Password should include numbers, uppercase and lowercase letters and should have at least 8 characters.” prompting user to choose another password that follows the criteria. |

**Specific cases**

| **ID** | **Password** | **Confirm Password** | **Expected Output** | **Test Result** |
|---|---|---|---|---|
| #1 | P@ssw0rd | P@ssw0rd | Password Reset!<br>Your password has been successfully reset.<br>Please log in with your updated credentials. | Password Reset!<br>Your password has been successfully reset.<br>Please log in with your updated credentials. |

| | | | | |
|---|---|---|---|---|
| #2 | password | password | Error!<br>Password not strong enough!<br><br>Password should include numbers, uppercase and lowercase letters and should have at least 8 characters. | Error!<br>Password not strong enough!<br><br>Password should include numbers, uppercase and lowercase letters and should have at least 8 characters. |
| #3 | P@ssw0rd | password | Error!<br>Passwords do not match! | Error!<br>Passwords do not match! |
| #4 | {Empty} | {Empty} | Error!<br>“Password not strong enough!<br><br>Password should include numbers, uppercase and lowercase letters and should have at least 8 characters.” | Error!<br>“Password not strong enough!<br><br>Password should include numbers, uppercase and lowercase letters and should have at least 8 characters.” |

6.1.4. Create New Project

| **Location** | Project List Page |
|---|---|

**General cases**

| **ID** | **Input** | **Oracle** | **Log** |
|---|---|---|---|
| #1 | Create a project with name | System creates a new Project and adds it to the list of projects. | Creates a new Project and adds it to the list of projects. |
| #2 | Create a project with {Empty} name field | System displays error message, “Fill up all the fields!” prompting user. | System displays error message, “Fill up all the fields!” prompting user. |

**Specific cases**

| **ID** | **Project Name** | **Expected Output** | **Test Result** |
|---|---|---|---|
| #1 | Project1 | Project “Project1” gets added to the project list. | Project “Project1” gets added to the project list. |
| #2 | {Empty} | Error! Input a title! | Error! Input a title! |

6.1.5. Create New Task Card

| **Location** | Kanban View Page |
|---|---|

**General cases**

| **ID** | **Input** | **Oracle** | **Log** |
|---|---|---|---|
| #1 | Create a task in the preferred column | System creates a new task and adds the task to the preferred column. | System creates a new task and adds the task to the preferred column. |
| #2 | Create a task with {Empty} title, description and date | System displays error message, “Input a title!” and prompts user. | System displays error message, “Input a title!” and prompts user. |
| #3 | Create a task with title, description and {Empty} date | System displays error message, “Input a date between today and 2 years later!” and prompts user. | System displays error message, “Input a date between today and 2 years later!” and prompts user. |

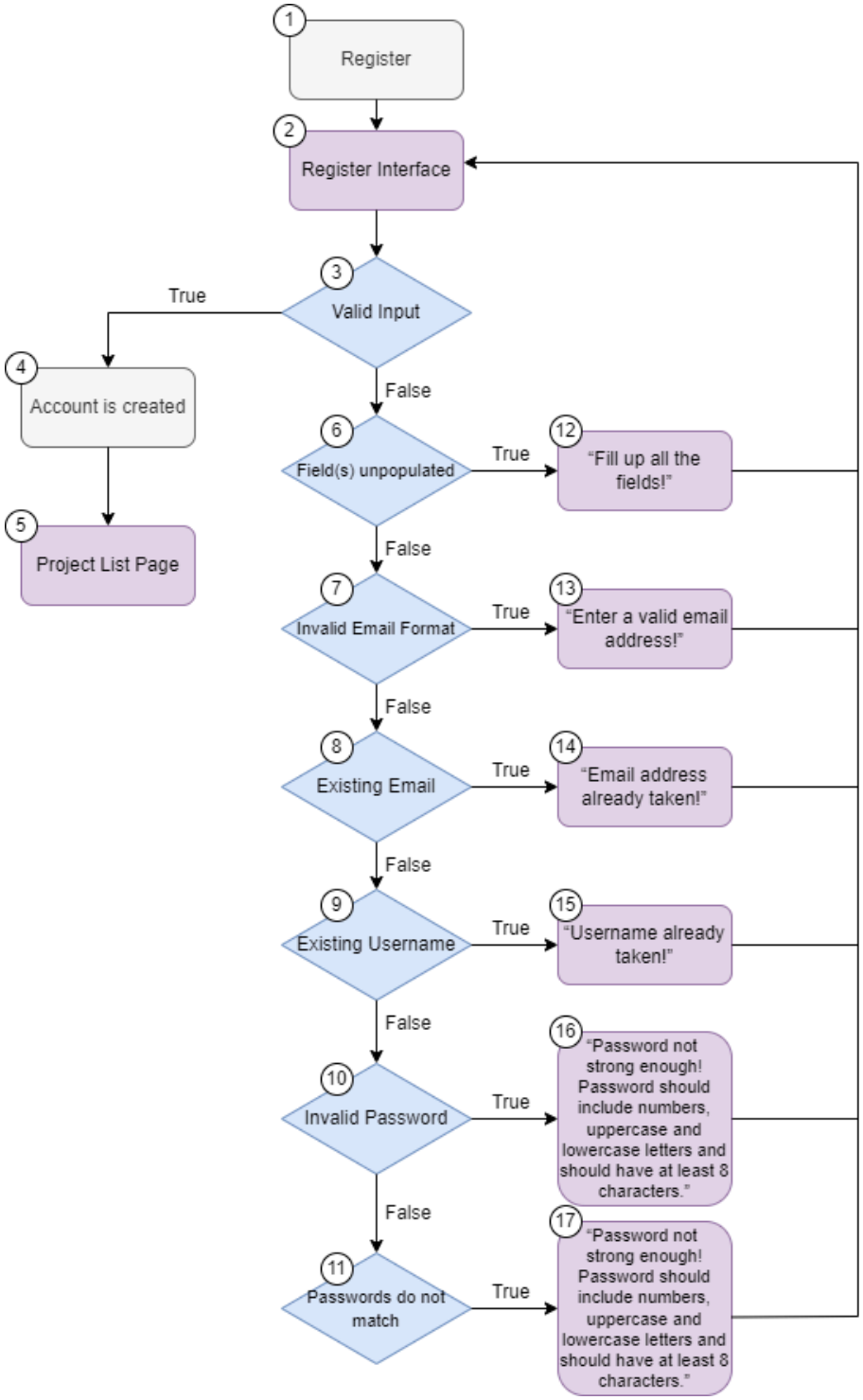
**Specific cases**

| **ID** | **Task Title** | **Task Description** | **Task Due Date** | **Expected Output** | **Test Result** |
|---|---|---|---|---|---|
| #1 | Task 1 | {Empty} | 16/4/2024 (Current date) | Task gets added to preferred column. | Task gets added to preferred column. |
| #2 | {Empty} | abc | 16/4/2024 | Error! Input a title! | Error! Input a title! |
| #3 | Task 1 | abc | 16/4/2024 | Task gets added to preferred column. | Task gets added to preferred column. |
| #4 | Task 1 | abc | 10/4/2024 | Error! Input a date | Error! Input a date |

| | | | | | |
|---|---|---|---|---|---|
| | | | (date before current date) | between today and 2 years later! | between today and 2 years later! |
| #5 | Task 1 | abc | 17/4/2026 (date beyond 2 years from current date) | Error! Input a date between today and 2 years later! | Error! Input a date between today and 2 years later! |

## 6.2. White Box Testing

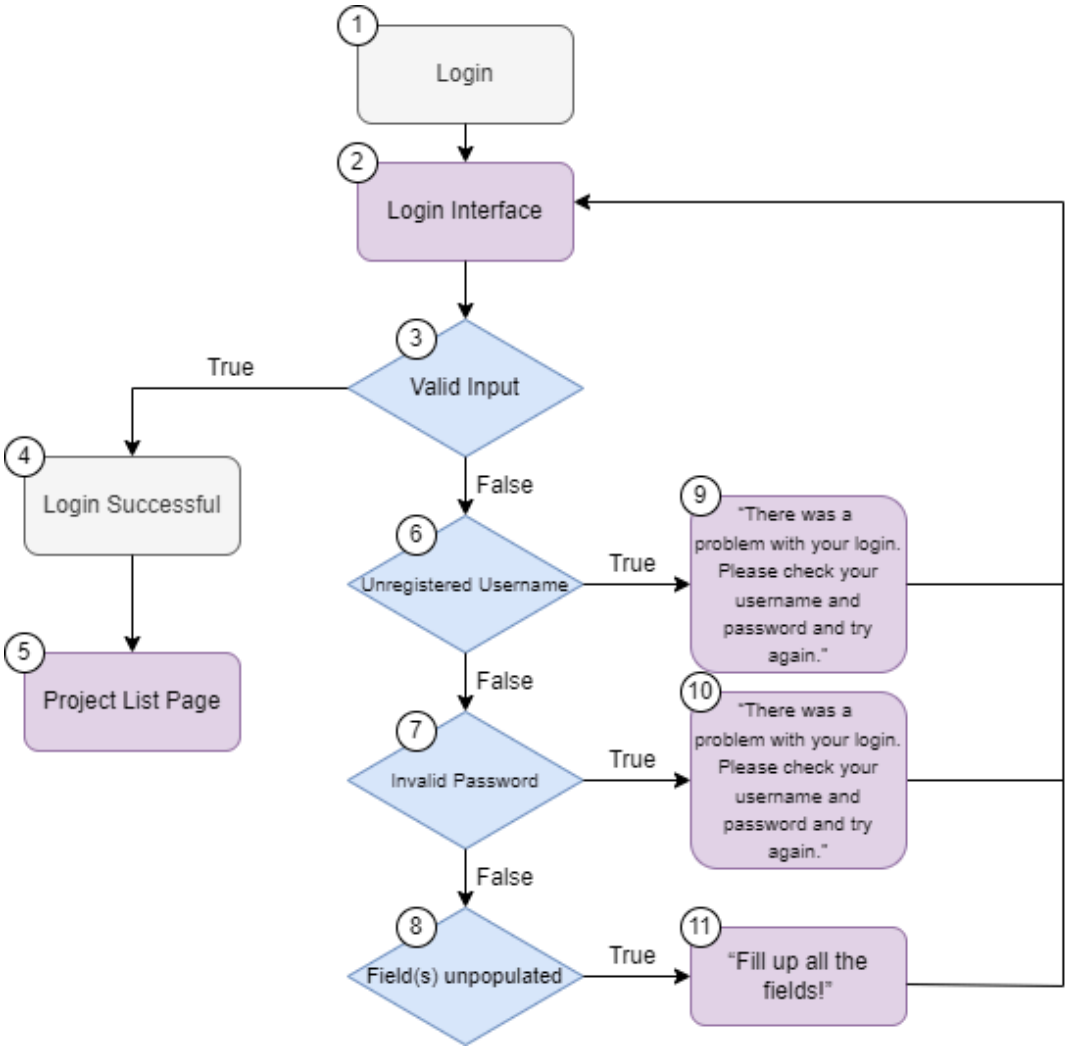
### 6.2.1. Registration



Cyclomatic complexity = 22 - 17 + 2 = 7

| ID | Basis paths | Real execution paths |
|---|---|---|
| #1 | 1, 2, 3, 4, 5 | 1, 2, 3, 4, 5 |
| #2 | 1, 2, 3, 6, 12, 2, 3, 4, 5 | 1, 2, 3, 6, 12, 2, 3, 4, 5 |
| #3 | 1, 2, 3, 6, 7, 13, 2, 3, 4, 5 | 1, 2, 3, 6, 7, 13, 2, 3, 4, 5 |
| #4 | 1, 2, 3, 6, 7, 8, 14, 2, 3, 4, 5 | 1, 2, 3, 6, 7, 8, 14, 2, 3, 4, 5 |
| #5 | 1, 2, 3, 6, 7, 8, 9, 15, 2, 3, 4, 5 | 1, 2, 3, 6, 7, 8, 9, 15, 2, 3, 4, 5 |
| #6 | 1, 2, 3, 6, 7, 8, 9, 10, 16, 2, 3, 4, 5 | 1, 2, 3, 6, 7, 8, 9, 10, 16, 2, 3, 4, 5 |
| #7 | 1, 2, 3, 6, 7, 8, 9, 10, 11, 17, 2, 3, 4, 5 | 1, 2, 3, 6, 7, 8, 9, 10, 11, 17, 2, 3, 4, 5 |

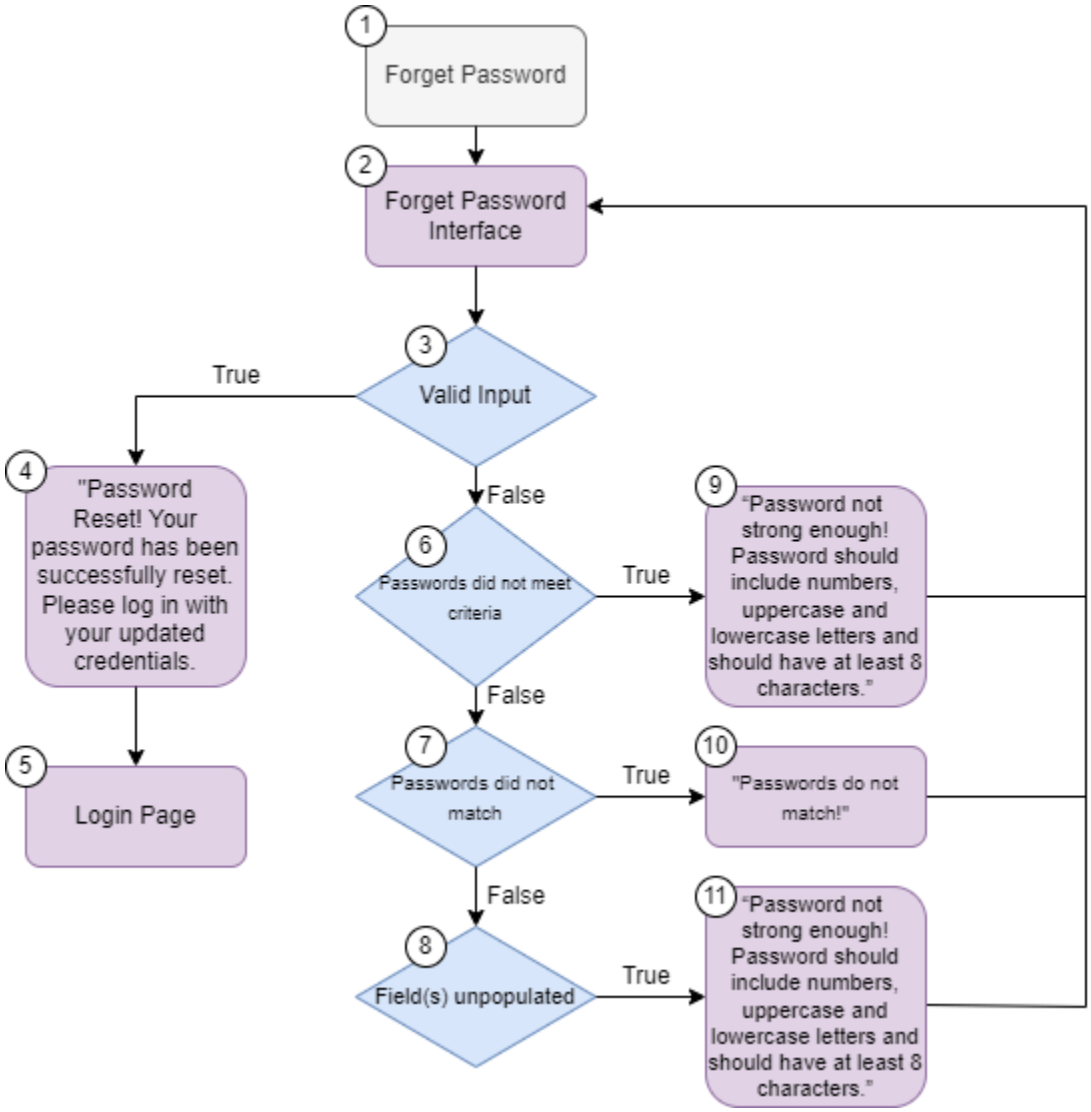
6.2.2. Login




Cyclomatic complexity = 13 - 11 + 2 = 4

| ID | Basis paths | Real execution paths |
|---|---|---|
| #1 | 1, 2, 3, 4, 5 | 1, 2, 3, 4, 5 |
| #2 | 1, 2, 3, 6, 9, 2, 3, 4, 5 | 1, 2, 3, 6, 9, 2, 3, 4, 5 |
| #3 | 1, 2, 3, 6, 7, 10, 2, 3, 4, 5 | 1, 2, 3, 6, 7, 10, 2, 3, 4, 5 |
| #4 | 1, 2, 3, 6, 7, 8, 11, 2, 3, 4, 5 | 1, 2, 3, 6, 7, 8, 11, 2, 3, 4, 5 |

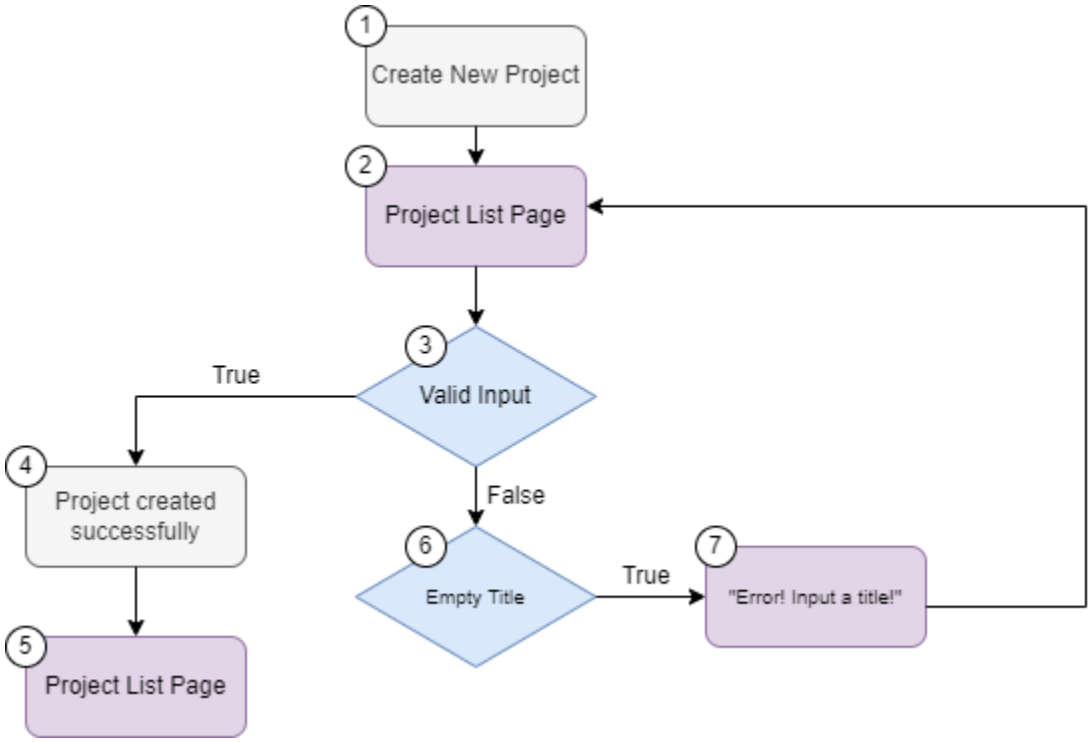
6.2.3. Forget Password




Cyclomatic complexity = 13 - 11 + 2 = 4

| ID | Basis paths | Real execution paths |
|---|---|---|
| #1 | 1, 2, 3, 4, 5 | 1, 2, 3, 4, 5 |
| #2 | 1, 2, 3, 6, 9, 2, 3, 4, 5 | 1, 2, 3, 6, 9, 2, 3, 4, 5 |
| #3 | 1, 2, 3, 6, 7, 10, 2, 3, 4, 5 | 1, 2, 3, 6, 7, 10, 2, 3, 4, 5 |
| #4 | 1, 2, 3, 6, 7, 8, 11, 2, 3, 4, 5 | 1, 2, 3, 6, 7, 8, 11, 2, 3, 4, 5 |

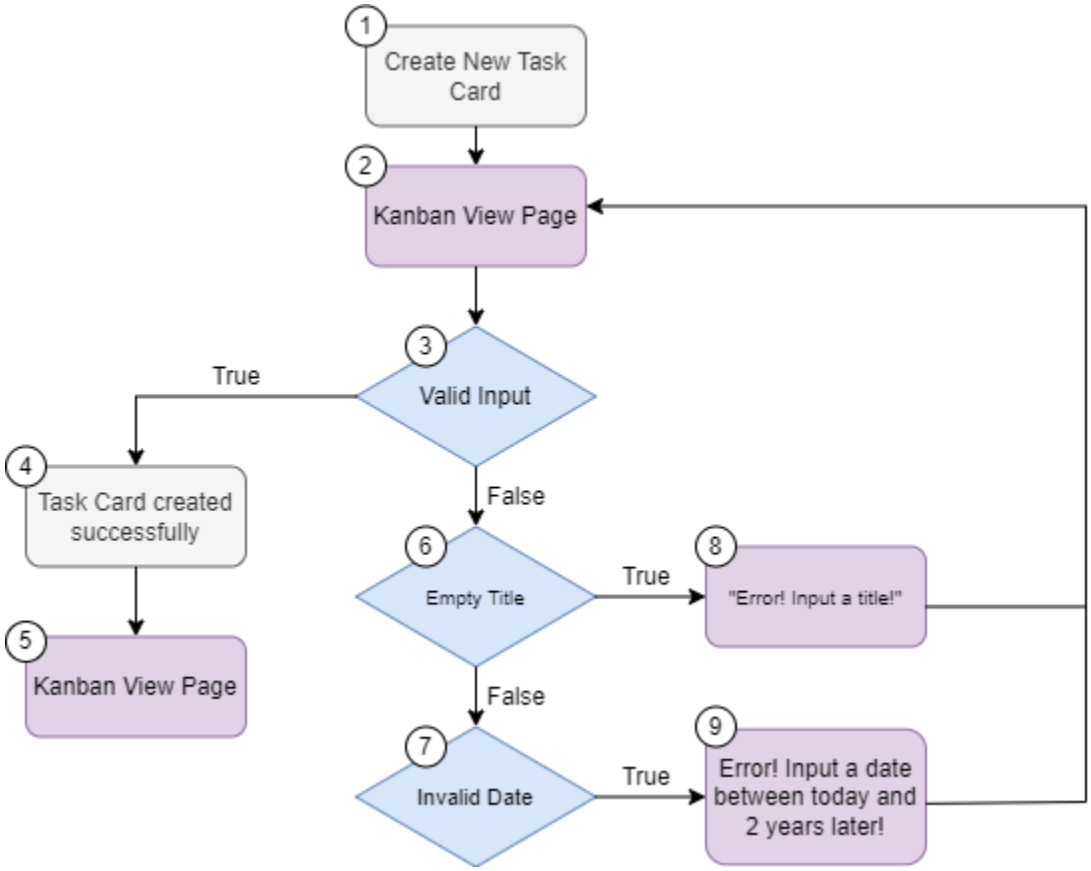
6.2.4. Create New Project




Cyclomatic complexity = 7 - 7 + 2 = 2

| ID | Basis paths | Real execution paths |
|---|---|---|
| #1 | 1, 2, 3, 4, 5 | 1, 2, 3, 4, 5 |
| #2 | 1, 2, 3, 6, 7, 2, 3, 4, 5 | 1, 2, 3, 6, 7, 2, 3, 4, 5 |

6.2.5. Create New Task Card




Cyclomatic complexity = 10 - 9 + 2 = 3

| ID | Basis paths | Real execution paths |
|---|---|---|
| #1 | 1, 2, 3, 4, 5 | 1, 2, 3, 4, 5 |
| #2 | 1, 2, 3, 6, 8, 2, 3, 4, 5 | 1, 2, 3, 6, 8, 2, 3, 4, 5 |
| #3 | 1, 2, 3, 6, 7, 9, 2, 3, 4, 5 | 1, 2, 3, 6, 7, 9, 2, 3, 4, 5 |

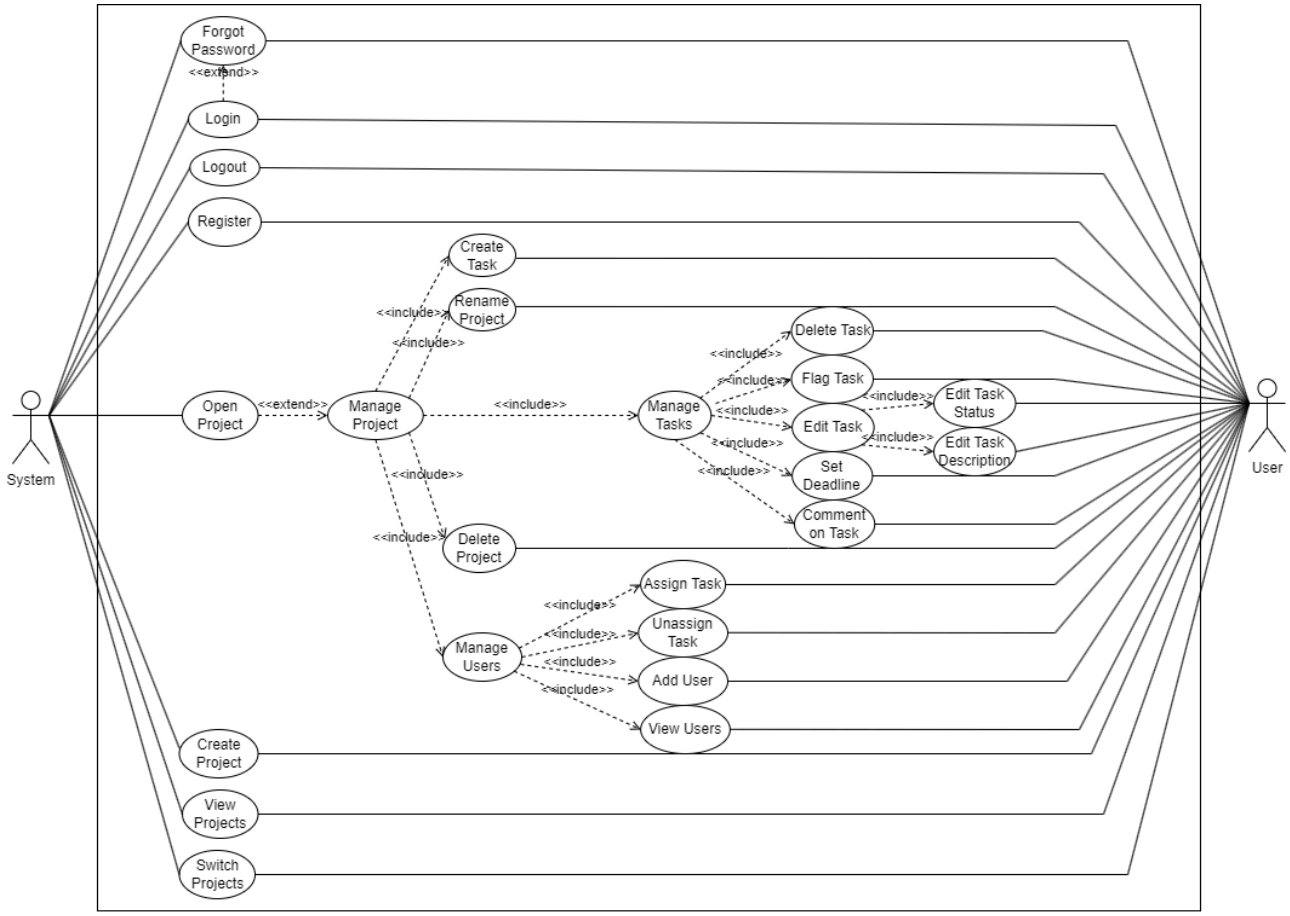
# Appendix A: Glossary

| Term | Definition |
|---|---|
| Kanban board | A project management tool which is the main feature of our website. Kanban boards visually depict work at various stages of a process using task cards to represent work items and columns to represent each stage of the process. |
| Column | Vertical division of a Kanban board that allows users to visualize different stages of workflow. |
| Task Card | Digital cards to represent a task in a project. |
| Login Page | The page where users can either register, login or access the Forget Password page. Found on this page are: Login Interface and Registration Interface. |
| Forget Password Page | The page where users can enter their email should they forget their password. Found in this page is: Forget Password Interface. |
| Reset Password Page | The page where users can enter their new password. Only retrieved when the user requests a password change and an email is sent to the user's inbox. |
| Project List Page | The page where users are directed to after a successful login. Users can view their projects or create a new project here. |
| Kanban View Page | The main page of a project. The main functionalities of the Kanban board will be performed here. |
| Navigation bar | A black bar loaded at the top of the page with the website's name. Users that have logged in are able to see their username and log out. |

# Appendix B: Analysis Models

## 1. Use Case Model

### 1.1. Use Case Diagram



## 1.2. Use Case Descriptions

### 1.2.1. Registration

| | | | |
|---|---|---|---|
| **Use Case ID:** | 1 | | |
| **Use Case Name:** | Registration | | |
| **Created By:** | Samuel Seow | Last Updated By: | Samuel Tan |
| **Date Created:** | 08/02/2024 | Date Last Updated: | 16/04/2024 |

| | |
|---|---|
| **Actor:** | User, System |
| **Description:** | Registration of a new user account |
| **Preconditions:** | ● User has an email account.<br>● User does not have an account registered with this email.<br>● User has a unique username.<br>● User has a password with a minimum of 8 characters that contains 1 of each: uppercase, lowercase and number.<br>● Connection to mobile data/WiFi. |
| **Postconditions:** | ● Users are able to login to access Kanbanize services. |
| **Priority:** | High |
| **Frequency of Use:** | Once |
| **Flow of Events:** | 1. User is able to view the Login Page.<br>2. User clicks on “Sign Up”.<br>3. Sign Up interface is displayed to the user.<br>4. User will be prompted to enter a “Username”, “Email”, “Password” and “Confirm Password”.<br>5. User clicks on the “Register” button.<br>6. User’s account is created and logged into his account, and brought to the Project List Page. |
| **Alternative Flows:** | NIL |
| **Exceptions:** | 1. User’s Username is not unique:<br>a. Pop-up displays: “Error. Username already taken!”<br>2. User’s Email Address already exists in the database:<br>a. Pop-up displays: “Error. Email address already taken!”<br>3. User’s Password does not fulfill the requirements:<br>a. Pop-up displays: “Error. Password not strong enough! Password should include numbers, uppercase and lowercase letters and should have at least 8 characters.”<br>4. User’s Password and Confirm Password are not the same:<br>a. Pop-up displays: “Error. Passwords do not match!” |
| **Includes:** | NIL |
| **Special Requirements:** | ● User has a password that contains 1 of each: uppercase, lowercase, number and minimum length of 8 characters. |
| **Assumptions:** | ● User is connected to the internet. |
| **Notes and Issues:** | NIL |

1.2.2. Login

| | | | |
|---|---|---|---|
| **Use Case ID:** | 2 | | |
| **Use Case Name:** | Login | | |
| **Created By:** | Senchia | Last Updated By: | Samuel Tan |
| **Date Created:** | 08/02/2024 | Date Last Updated: | 16/04/2024 |

| | |
|---|---|
| **Actor:** | User, System |
| **Description:** | User logins to an existing account. |
| **Preconditions:** | ● User has already registered for an account.<br>● User information is stored in the database.<br>● Connection to mobile data/WiFi. |
| **Postconditions:** | ● User is redirected to the Project List Page. |
| **Priority:** | High |
| **Frequency of Use:** | Once until logged out |
| **Flow of Events:** | 1. User is on the Login Page.<br>2. User clicks on “Login”.<br>3. Login interface is displayed to User.<br>4. User inputs their unique username and password in text boxes and clicks on the Login button.<br>5. Kanbanize system verifies users login credentials.<br>6. User is redirected to the Project List Page. |
| **Alternative Flows:** | NIL |
| **Exceptions:** | 1. Incorrect username or password:<br>a. Pop-up displays: “ Error. There was a problem with your login. Please check your username and password and try again.”<br>b. System returns to step 1. |
| **Includes:** | NIL |
| **Special Requirements:** | NIL |
| **Assumptions:** | ● User has already successfully registered for an account.<br>● User is connected to the internet. |
| **Notes and Issues:** | NIL |

### 1.2.3. Logout

| | | | |
|---|---|---|---|
| **Use Case ID:** | 3 | | |
| **Use Case Name:** | Logout | | |
| **Created By:** | Samuel Seow | Last Updated By: | Samuel Seow |
| **Date Created:** | 19/03/2024 | Date Last Updated: | 15/04/2024 |

| | |
|---|---|
| **Actor:** | User, System |
| **Description:** | User logs out of the system. |
| **Preconditions:** | ● User is logged in. |
| **Postconditions:** | ● User is redirected to the Login Page. |
| **Priority:** | Low |
| **Frequency of Use:** | Once until logged in |
| **Flow of Events:** | 1. User clicks on their username in the Navigation bar.<br>2. A drop-down menu is displayed with “Log out”<br>3. User clicks on “Log out”.<br>4. User is redirected to the Login Page. |
| **Alternative Flows:** | ● User clears browser cookies for Kanbanize and is logged out. |
| **Exceptions:** | NIL |
| **Includes:** | NIL |
| **Special Requirements:** | NIL |
| **Assumptions:** | ● User has an account and is logged in.<br>● User is connected to the internet. |
| **Notes and Issues:** | NIL |

### 1.2.4. Forget Password

| | | | |
|---|---|---|---|
| **Use Case ID:** | 4 | | |
| **Use Case Name:** | Forget Password | | |
| **Created By:** | Samuel Seow | Last Updated By: | Samuel Tan |
| **Date Created:** | 19/03/2024 | Date Last Updated: | 15/04/2024 |

| | |
|---|---|
| **Actor:** | User, Database, System |
| **Description:** | User resets their password with “Forget Password” |
| **Preconditions:** | ● User is on the Login Page and is not logged in.<br>● User has an existing account.<br>● User remembers the Email Address used to register for the account. |
| **Postconditions:** | ● User is prompted to return to the Login Page. |
| **Priority:** | High |
| **Frequency of Use:** | Low, only when user forgets password |
| **Flow of Events:** | 1. User clicks on “Login”.<br>2. Login interface is displayed to User.<br>3. User clicks on “Forgot Password? Click here”.<br>4. User enters his email address in the textbox.<br>5. System sends an email to the user’s email address.<br>6. The email includes a hyperlink to the Reset Password Page.<br>7. User clicks on “Reset Password” and is redirected to the Reset Password Page.<br>8. User will be prompted to enter a new Password, and once more to confirm the new password.<br>9. User clicks “Confirm”.<br>10. User is redirected back to the Login Page. |
| **Alternative Flows:** | NIL |
| **Exceptions:** | 1. User’s Password does not fulfill the requirements:<br>a. Pop-up displays: “Error. Password not strong enough! Password should include numbers, uppercase and lowercase letters and should have at least 8 characters.”<br>2. User’s Password and Confirm Password are not the same:<br>a. Pop-up displays: “Error. Passwords do not match!”<br>3. User enters the incorrect email address:<br>a. Pop-up displays: “Error. No users with email found!” |
| **Includes:** | NIL |
| **Special Requirements:** | ● User has a password that contains 1 of each: uppercase, lowercase, number and minimum length of 8 characters. |
| **Assumptions:** | ● User is connected to the internet.<br>● User has already successfully registered for an account. |
| **Notes and Issues:** | NIL |

### 1.2.5. Create Project

| **Use Case ID:** | 5 | | |
|---|---|---|---|
| **Use Case Name:** | Create Project | | |
| **Created By:** | Samuel Seow | Last Updated By: | Samuel Seow |
| **Date Created:** | 08/02/2024 | Date Last Updated: | 15/04/2024 |

| | |
|---|---|
| **Actor:** | User, Database, System |
| **Description:** | User creates a project. |
| **Preconditions:** | ● User must have a Kanbanize account.<br>● User must be logged in.<br>● User is at the Project List Page. |
| **Postconditions:** | ● A new project is created. |
| **Priority:** | High |
| **Frequency of Use:** | Medium |
| **Flow of Events:** | 1. User is at the Project List Page.<br>2. User clicks on “+ New Project”.<br>3. User is prompted to enter project name.<br>4. A new project is created and the user returns to the Project List Page. |
| **Alternative Flows:** | NIL |
| **Exceptions:** | 1. User did not input a Project Name.<br>a. Pop-up displays: “Error. Input a title!” |
| **Includes:** | NIL |
| **Special Requirements:** | NIL |
| **Assumptions:** | ● User has an account and is logged in.<br>● User is connected to the internet. |
| **Notes and Issues:** | NIL |

### 1.2.6. Switch Project

| | | | |
|---|---|---|---|
| **Use Case ID:** | 6 | | |
| **Use Case Name:** | Switch Project | | |
| **Created By:** | Samuel Seow | Last Updated By: | Samuel Seow |
| **Date Created:** | 19/03/2024 | Date Last Updated: | 15/04/2024 |

| | |
|---|---|
| **Actor:** | User, System |
| **Description:** | User wants to switch between projects. |
| **Preconditions:** | ● User must have a Kanbanize account.<br>● User must be logged in. |
| **Postconditions:** | ● User is redirected to the desired project's Kanban View Page. |
| **Priority:** | High |
| **Frequency of Use:** | Medium |
| **Flow of Events:** | 1. User is at the Project List Page and sees multiple projects which are either under their name or they have been added to.<br>2. User clicks on the desired project to switch to.<br>3. User is redirected to that project's Kanban View Page. |
| **Alternative Flows:** | 1. User is at the Kanban View Page in another project.<br>2. User clicks on "← Back to Project List"<br>3. User is redirected to the Project List Page.<br>4. User is at the Project List Page and sees multiple projects which are under their name.<br>5. User clicks on the desired project to switch to.<br>6. User is redirected to that project's Kanban View Page. |
| **Exceptions:** | NIL |
| **Includes:** | NIL |
| **Special Requirements:** | NIL |
| **Assumptions:** | ● User has an account and is logged in.<br>● User is connected to the internet.<br>● User has more than 1 project. |
| **Notes and Issues:** | NIL |

### 1.2.7. Delete Project

| | | | |
|---|---|---|---|
| **Use Case ID:** | 7 | | |
| **Use Case Name:** | Delete Project | | |
| **Created By:** | Samuel Seow | Last Updated By: | Samuel Seow |
| **Date Created:** | 15/04/2024 | Date Last Updated: | 15/04/2024 |

| | |
|---|---|
| **Actor:** | User, System |
| **Description:** | User deletes the current project board. |
| **Preconditions:** | ● User must be logged in.<br>● User must have a registered account.<br>● User is at the Kanban View Page of the project to be deleted. |
| **Postconditions:** | ● User is redirected to the Project List Page. |
| **Priority:** | High |
| **Frequency of Use:** | Medium |
| **Flow of Events:** | 1. User clicks on the “Settings” icon at the top right of the page.<br>2. A dropdown menu is shown with “Delete Board” at the bottom.<br>3. User clicks on “Delete Board”.<br>4. User is redirected to a page to confirm deletion of the board.<br>5. User clicks “Delete” and deletes the project.<br>6. User is then redirected back to the Project List Page. |
| **Alternative Flows:** | NIL |
| **Exceptions:** | NIL |
| **Includes:** | NIL |
| **Special Requirements:** | NIL |
| **Assumptions:** | ● User has an account and is logged in.<br>● User is connected to the internet. |
| **Notes and Issues:** | NIL |

1.2.8. Add Task

| | | | |
|---|---|---|---|
| **Use Case ID:** | 8 | | |
| **Use Case Name:** | Add Task | | |
| **Created By:** | Senchia Gladine | Last Updated By: | Samuel Tan |
| **Date Created:** | 08/02/2024 | Date Last Updated: | 16/04/2024 |

| | |
|---|---|
| **Actor:** | User, System |
| **Description:** | User adds a task in a selected project. |
| **Preconditions:** | ● User must be logged in.<br>● User must have a registered account.<br>● User is at the Kanban View Page. |
| **Postconditions:** | ● New task card shows up at the chosen column. |
| **Priority:** | High |
| **Frequency of Use:** | High |
| **Flow of Events:** | 1. User click on “+” on the top right corner of the particular column.<br>2. User is redirected to new task creation page.<br>3. User fills in “Title”, “Description” and “Due Date”.<br>4. Task is created once User clicks “Create”. |
| **Alternative Flows:** | NIL |
| **Exceptions:** | 1. User did not input a Task Title:<br>a. Pop-up displays: “Input a title!”<br>2. User inputs a title more than 50 characters long:<br>a. Pop-up displays: “Error. Limit your input to 50 characters!”<br>3. User inputs a description more than 250 characters long:<br>a. Pop-up displays: “Error. Limit your input to 250 characters!”<br>4. User did not input a valid date:<br>a. Pop-up displays: “Input a date between today and 2 years later!” |
| **Includes:** | ● Title<br>● Description<br>● Due Date |
| **Special Requirements:** | NIL |
| **Assumptions:** | ● User has an account and is logged in.<br>● User is connected to the internet. |
| **Notes and Issues:** | NIL |

### 1.2.9. Flag Task

| | | | |
|---|---|---|---|
| **Use Case ID:** | 9 | | |
| **Use Case Name:** | Flag Task | | |
| **Created By:** | Samuel Seow | Last Updated By: | Samuel Seow |
| **Date Created:** | 15/04/2024 | Date Last Updated: | 15/04/2024 |

| | |
|---|---|
| **Actor:** | User, System |
| **Description:** | User flags a task in a selected project. |
| **Preconditions:** | ● User must be logged in.<br>● User must have a registered account.<br>● User is at the Kanban View Page. |
| **Postconditions:** | ● Task is now flagged with a red flag icon. |
| **Priority:** | Low |
| **Frequency of Use:** | High |
| **Flow of Events:** | 7. User clicks on the “Flag” icon on a task card.<br>8. “Flag” icon is now red. |
| **Alternative Flows:** | NIL |
| **Exceptions:** | NIL |
| **Includes:** | NIL |
| **Special Requirements:** | NIL |
| **Assumptions:** | ● User has an account and is logged in.<br>● User is connected to the internet. |
| **Notes and Issues:** | NIL |

### 1.2.10. Edit Task Details

| | | | |
|---|---|---|---|
| **Use Case ID:** | 10 | | |
| **Use Case Name:** | Edit Task Details | | |
| **Created By:** | Samuel Seow | Last Updated By: | Samuel Seow |
| **Date Created:** | 19/03/2024 | Date Last Updated: | 15/04/2024 |

| | |
|---|---|
| **Actor:** | User, System |
| **Description:** | User updates the details of the task in a project. |
| **Preconditions:** | ● User must be logged in.<br>● User must have a registered account.<br>● User must be a member of the project.<br>● User is at the Kanban View Page. |
| **Postconditions:** | ● Task card is updated with new details. |
| **Priority:** | High |
| **Frequency of Use:** | High |
| **Flow of Events:** | 1. User clicks on the “Pen” icon on a task card.<br>2. User is redirected to edit task details page.<br>3. User is able to edit the Task Title, Task Description and Task Due Date.<br>4. User saves changes by clicking “Save”. |
| **Alternative Flows:** | NIL |
| **Exceptions:** | 1. User removed the previous Task Title:<br>a. Pop-up displays: “Input a title!”<br>2. User inputs a title more than 50 characters long:<br>a. Pop-up displays: “Error. Limit your input to 50 characters!”<br>3. User inputs a description more than 250 characters long:<br>a. Pop-up displays: “Error. Limit your input to 250 characters!”<br>4. User did not input a valid date:<br>a. Pop-up displays: “Input a date between today and 2 years later!” |
| **Includes:** | ● Title<br>● Description<br>● Due Date |
| **Special Requirements:** | NIL |
| **Assumptions:** | ● User has an account and is logged in.<br>● User is connected to the internet.<br>● User has already created a task. |
| **Notes and Issues:** | NIL |

### 1.2.11. Edit Task Status

| | | | |
|---|---|---|---|
| **Use Case ID:** | 11 | | |
| **Use Case Name:** | Edit Task Status | | |
| **Created By:** | Willy Tang | Last Updated By: | Samuel Seow |
| **Date Created:** | 08/02/2024 | Date Last Updated: | 01/04/2024 |

| | |
|---|---|
| **Actor:** | User, System |
| **Description:** | User updates the status of the task in a project. |
| **Preconditions:** | ● User must be logged in.<br>● User must have a registered account.<br>● User must be a member of the project.<br>● User is at the Kanban View Page. |
| **Postconditions:** | ● The status of the task is updated. |
| **Priority:** | High |
| **Frequency of Use:** | High |
| **Flow of Events:** | 1. User clicks and holds on a Task Card.<br>2. User drags the Card to another Column.<br>3. User releases the click. |
| **Alternative Flows:** | NIL |
| **Exceptions:** | NIL |
| **Includes:** | NIL |
| **Special Requirements:** | NIL |
| **Assumptions:** | ● User has an account and is logged in.<br>● User is connected to the internet.<br>● User has already created a task prior. |
| **Notes and Issues:** | ● User must drop the task into 1 of the 4 columns: TO DO, IN PROGRESS, NEEDS REVIEW, COMPLETED. |

### 1.2.12. Assign Task to User

| | | | |
|---|---|---|---|
| **Use Case ID:** | 12 | | |
| **Use Case Name:** | Assign Task to User | | |
| **Created By:** | Samuel Seow | Last Updated By: | Samuel Seow |
| **Date Created:** | 15/04/2024 | Date Last Updated: | 15/04/2024 |

| | |
|---|---|
| **Actor:** | User, System |
| **Description:** | User assigns a task in a selected project to a member. |
| **Preconditions:** | ● User must be logged in.<br>● User must have a registered account.<br>● User is at the Kanban View Page.<br>● User to be assigned must be part of the project. |
| **Postconditions:** | ● Task is now assigned to a user. |
| **Priority:** | Low |
| **Frequency of Use:** | High |
| **Flow of Events:** | 1. User clicks on the “person with a tick” icon on a task card.<br>2. User is redirected to Assign User to Task Page where the user can choose which user to assign the task to.<br>3. System displays a drop down menu called “Select User”.<br>4. User clicks the menu and selects the user to assign to the task.<br>5. User clicks “Assign” and is shown a popup that says “User Assigned!” |
| **Alternative Flows:** | NIL |
| **Exceptions:** | NIL |
| **Includes:** | NIL |
| **Special Requirements:** | NIL |
| **Assumptions:** | ● User has an account and is logged in.<br>● User is connected to the internet. |
| **Notes and Issues:** | NIL |

### 1.2.13. Unassign Task from User

| | | | |
|---|---|---|---|
| **Use Case ID:** | 13 | | |
| **Use Case Name:** | Unassign Task from User | | |
| **Created By:** | Cheyenne Seet | Last Updated By: | Cheyenne Seet |
| **Date Created:** | 16/04/2024 | Date Last Updated: | 16/04/2024 |

| | |
|---|---|
| **Actor:** | User, System |
| **Description:** | User unassigns a task in a selected project from a member. |
| **Preconditions:** | ● User must be logged in.<br>● User must have a registered account.<br>● User is at the Kanban View Page.<br>● User to be unassigned must already be assigned to the Task. |
| **Postconditions:** | ● Task is now unassigned from a member. |
| **Priority:** | Low |
| **Frequency of Use:** | High |
| **Flow of Events:** | 1. User clicks on the “person with a tick” icon on a task card.<br>2. User is redirected to Assign User to Task Page where the user can choose to unassign a user from the task.<br>3. System displays a drop down menu called “Select User”.<br>4. User clicks the menu and selects “Unassign” at the bottom of the menu.<br>5. User clicks “Unassign” and is shown a popup that says “User Unassigned!” |
| **Alternative Flows:** | NIL |
| **Exceptions:** | NIL |
| **Includes:** | NIL |
| **Special Requirements:** | NIL |
| **Assumptions:** | ● User has an account and is logged in.<br>● User is connected to the internet. |
| **Notes and Issues:** | NIL |

### 1.2.14. Delete Task

| | | | |
|---|---|---|---|
| **Use Case ID:** | 14 | | |
| **Use Case Name:** | Delete Task | | |
| **Created By:** | Samuel Seow | Last Updated By: | Samuel Seow |
| **Date Created:** | 19/03/2024 | Date Last Updated: | 01/04/2024 |

| | |
|---|---|
| **Actor:** | User, System |
| **Description:** | User deletes a task from the project. |
| **Preconditions:** | ● User must be logged in.<br>● User must have a registered account.<br>● User must be a member of the project.<br>● User is at the Kanban View Page. |
| **Postconditions:** | ● The task is removed from the system. |
| **Priority:** | Medium |
| **Frequency of Use:** | Medium |
| **Flow of Events:** | 1. User clicks on a task card.<br>2. User clicks on the “Trash can” icon on the bottom right-hand of the Task Card window.<br>3. User is redirected to a page to confirm deletion.<br>4. User clicks on “Delete” and deletes the task. |
| **Alternative Flows:** | NIL |
| **Exceptions:** | NIL |
| **Includes:** | NIL |
| **Special Requirements:** | NIL |
| **Assumptions:** | ● User has an account and is logged in.<br>● User is connected to the internet.<br>● User has already created a task prior that can be deleted. |
| **Notes and Issues:** | NIL |

### 1.2.15. Add User to Project

| | | | |
|---|---|---|---|
| **Use Case ID:** | 15 | | |
| **Use Case Name:** | Add User to Project | | |
| **Created By:** | Samuel Seow | Last Updated By: | Samuel Seow |
| **Date Created:** | 15/04/2024 | Date Last Updated: | 15/04/2024 |

| | |
|---|---|
| **Actor:** | User, System |
| **Description:** | User wants to add a user to a selected project. |
| **Preconditions:** | ● User must be logged in.<br>● User must have a registered account.<br>● User is at the Kanban View Page. |
| **Postconditions:** | ● |
| **Priority:** | Low |
| **Frequency of Use:** | Medium |
| **Flow of Events:** | 1. User clicks on the “Settings” icon at the top right of the page.<br>2. A dropdown menu is shown with “Add User” at the top.<br>3. User clicks on “Add User”.<br>4. User is redirected to Add User page to input a user to add.<br>5. User is prompted to enter the username of the user to be added.<br>6. User clicks “Add” and adds a new user to the project. |
| **Alternative Flows:** | NIL |
| **Exceptions:** | 1. User enters a username not found in the database<br>a. Pop-up displays: “Username not found!” |
| **Includes:** | NIL |
| **Special Requirements:** | NIL |
| **Assumptions:** | ● User has an account and is logged in.<br>● User is connected to the internet. |
| **Notes and Issues:** | NIL |

### 1.2.16. View Users in Project

| | | | |
|---|---|---|---|
| **Use Case ID:** | 16 | | |
| **Use Case Name:** | View Users in Project | | |
| **Created By:** | Samuel Seow | Last Updated By: | Samuel Seow |
| **Date Created:** | 15/04/2024 | Date Last Updated: | 15/04/2024 |

| | |
|---|---|
| **Actor:** | User, System |
| **Description:** | User wants to view all users in a project. |
| **Preconditions:** | ● User must be logged in.<br>● User must have a registered account.<br>● User is at the Kanban View Page. |
| **Postconditions:** | ● |
| **Priority:** | Low |
| **Frequency of Use:** | Medium |
| **Flow of Events:** | 1. User clicks on the “Settings” icon at the top right of the page.<br>2. A dropdown menu is shown with “View Users” in the middle.<br>3. User clicks on “View Users”.<br>4. User is redirected to View Users page to view users in the current project.<br>5. User can see the usernames of all the members of the project. |
| **Alternative Flows:** | NIL |
| **Exceptions:** | NIL |
| **Includes:** | NIL |
| **Special Requirements:** | NIL |
| **Assumptions:** | ● User has an account and is logged in.<br>● User is connected to the internet. |
| **Notes and Issues:** | NIL |

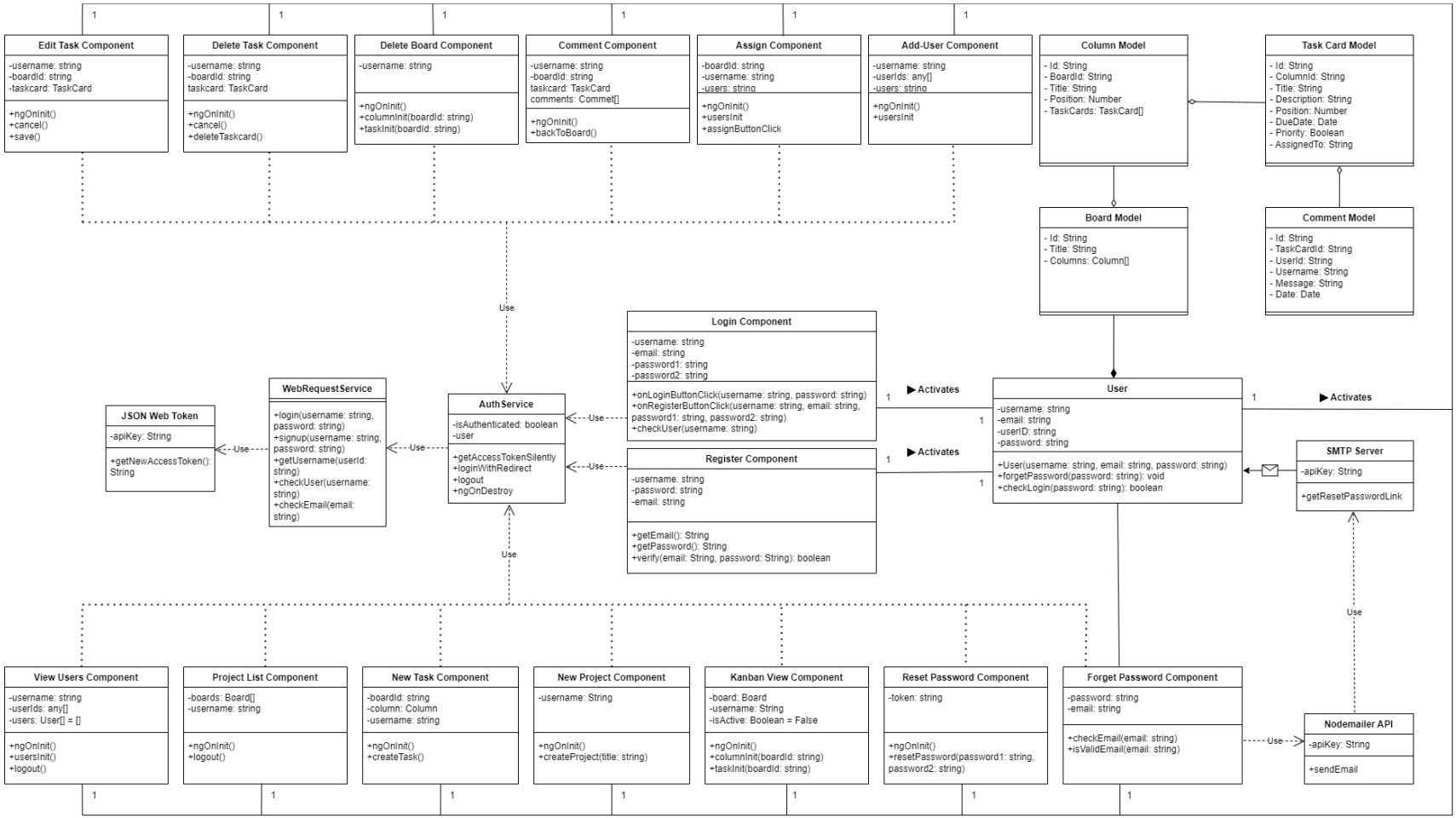
### 1.2.17. Rename Project

| | | | |
|---|---|---|---|
| **Use Case ID:** | 17 | | |
| **Use Case Name:** | Rename Project | | |
| **Created By:** | Samuel Seow | Last Updated By: | Samuel Seow |
| **Date Created:** | 17/04/2024 | Date Last Updated: | 17/04/2024 |

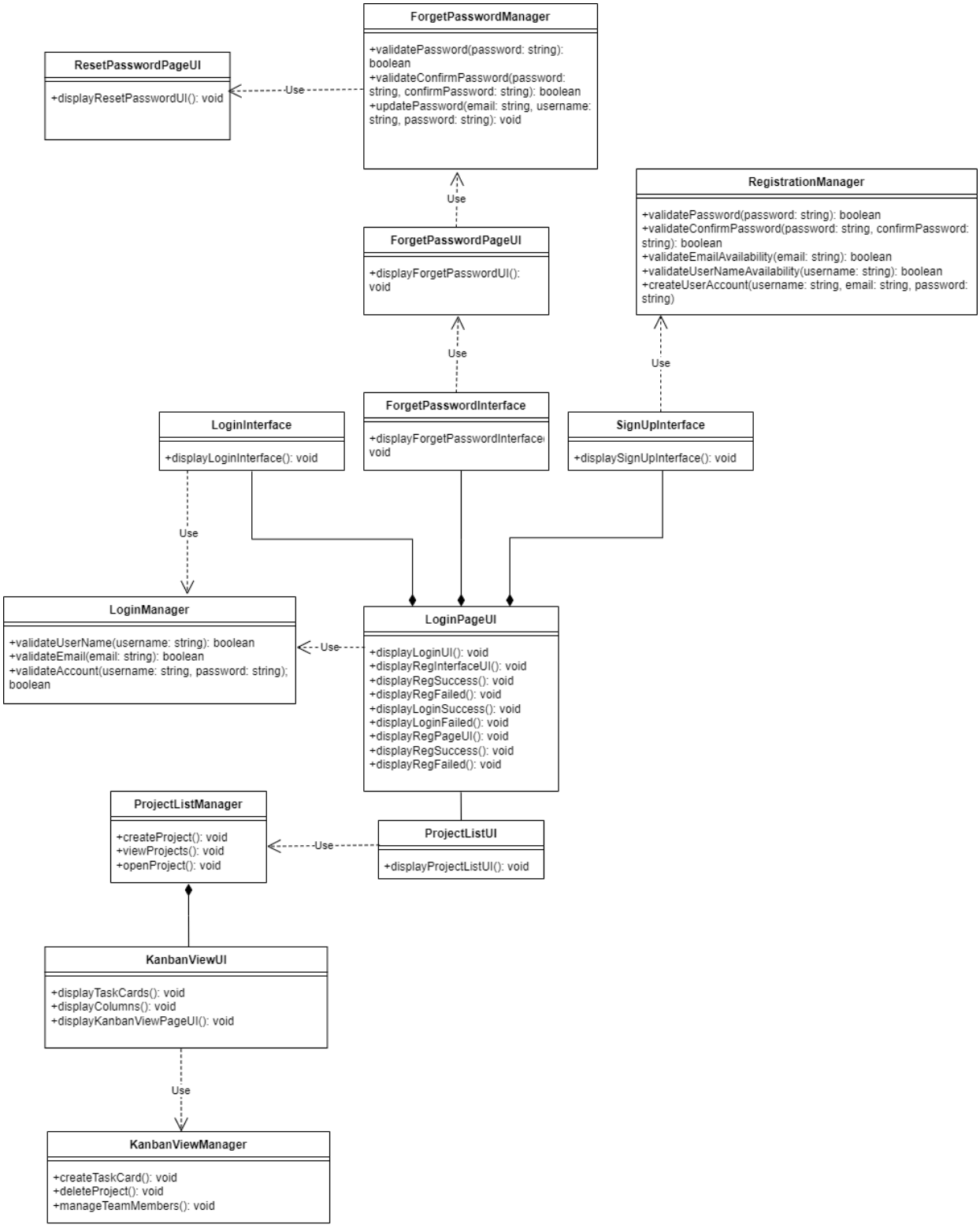
| | |
|---|---|
| **Actor:** | User, System |
| **Description:** | User renames the current project board. |
| **Preconditions:** | ● User must be logged in.<br>● User must have a registered account.<br>● User is at the Kanban View Page of the project to be renamed. |
| **Postconditions:** | NIL |
| **Priority:** | Low |
| **Frequency of Use:** | Low |
| **Flow of Events:** | 9. User hovers over board name at the top left hand corner of the Kanban View Page.<br>10. A “pen” icon is displayed.<br>11. User clicks on “pen” icon to edit the board title.<br>12. User renames the project.<br>13. User saves changes by clicking the “save” icon.<br>14. New project title is reflected. |
| **Alternative Flows:** | NIL |
| **Exceptions:** | NIL |
| **Includes:** | NIL |
| **Special Requirements:** | NIL |
| **Assumptions:** | ● User has an account and is logged in.<br>● User is connected to the internet. |
| **Notes and Issues:** | NIL |

# 2. Conceptual Model

## 2.1. Class Diagram



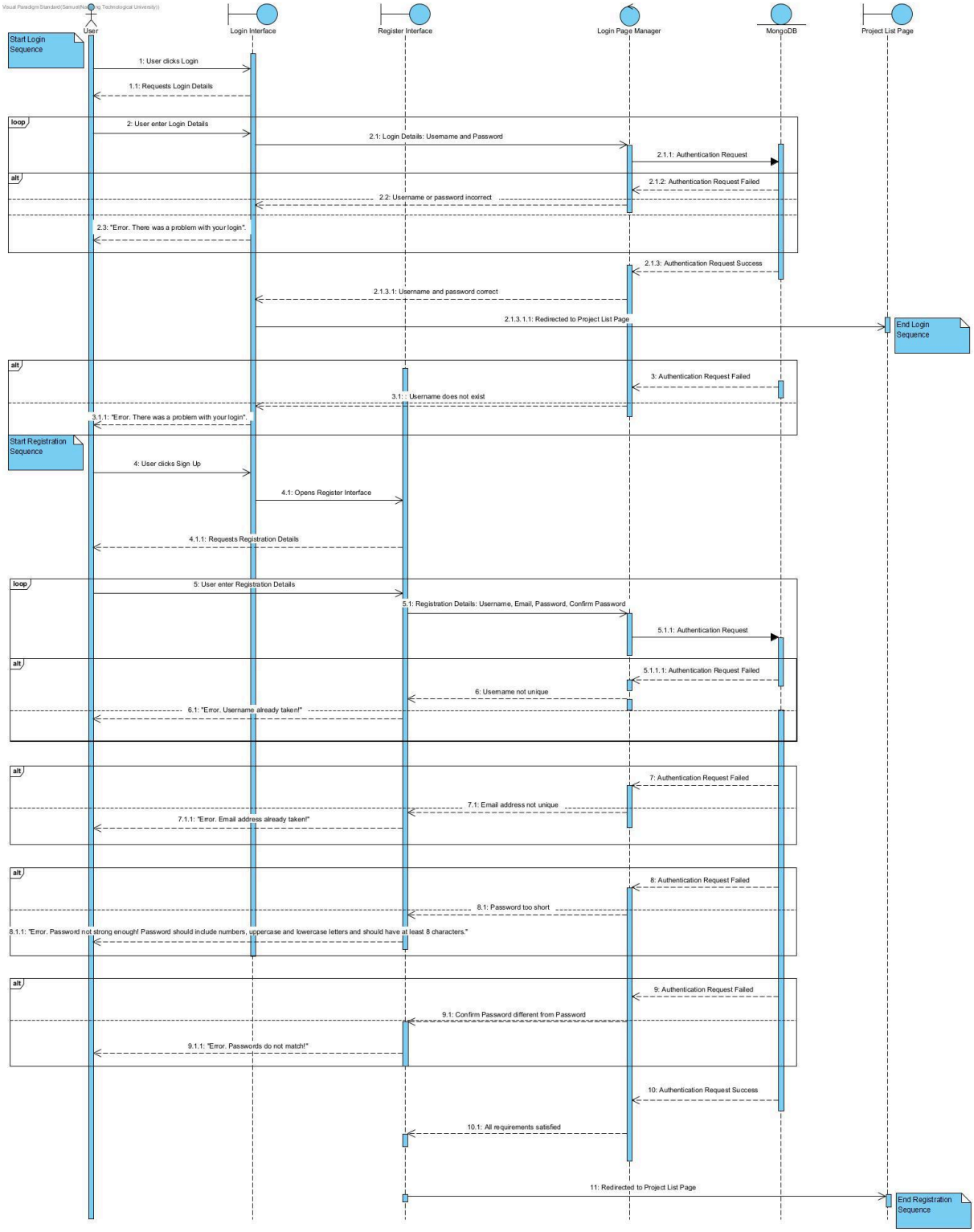
## 2.2. Key Boundary and Control Classes



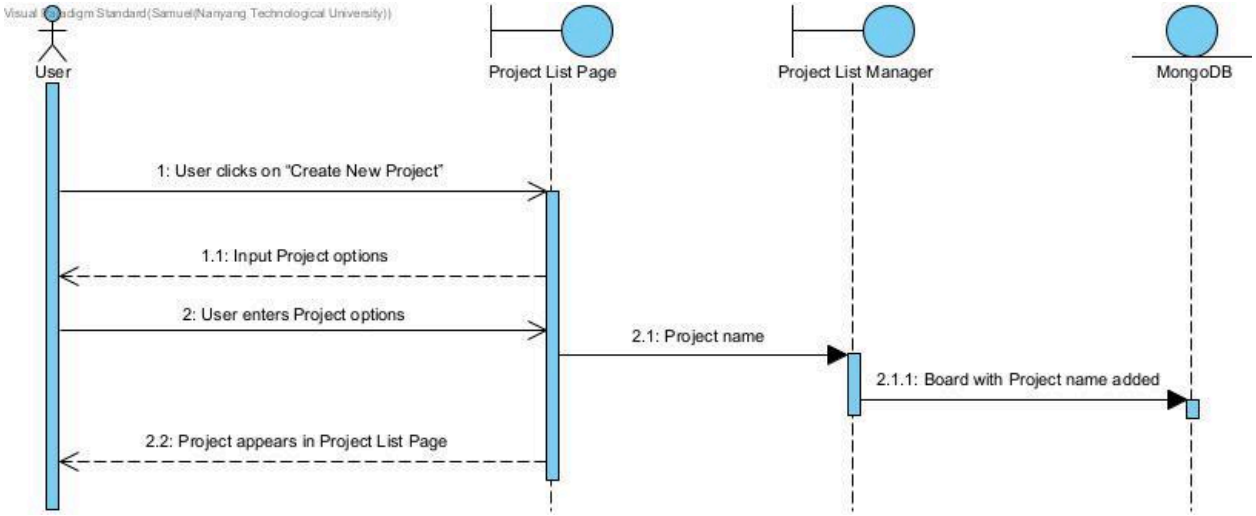
# 3. Dynamic Model

## 3.1. Sequence Diagrams

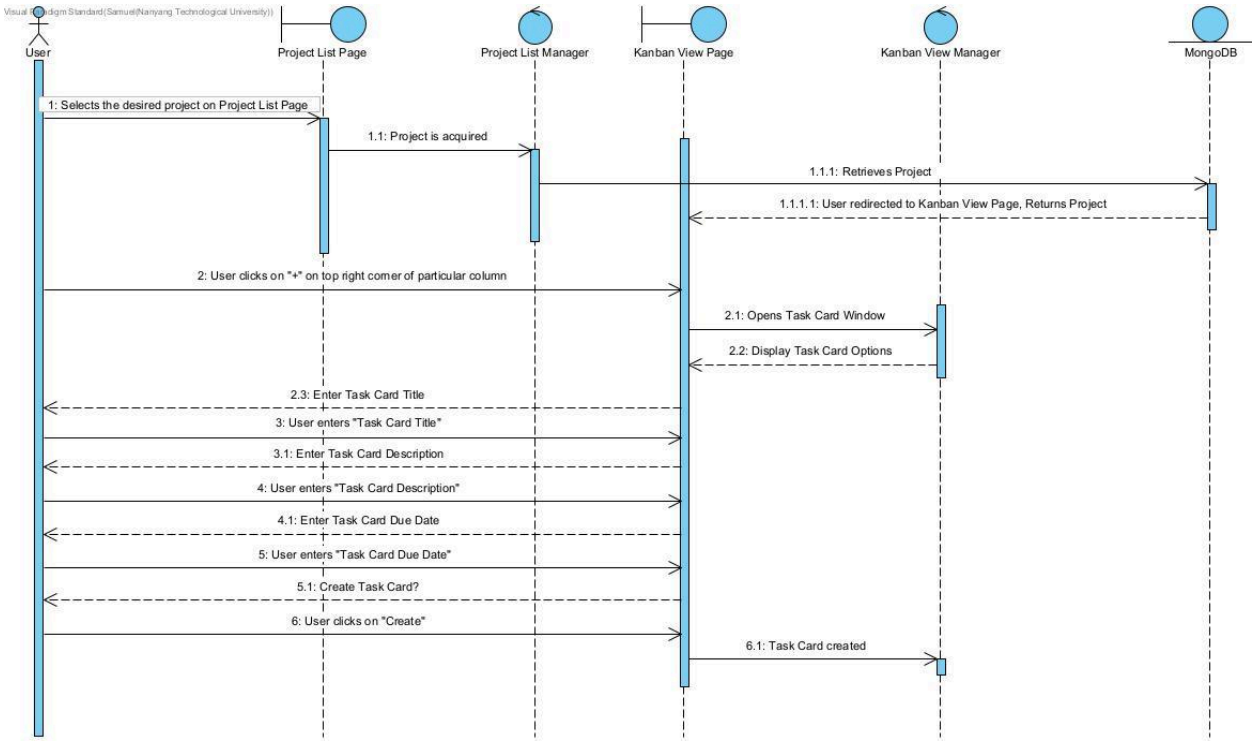
### 3.1.1. Use Case 1&2: Registration and Login



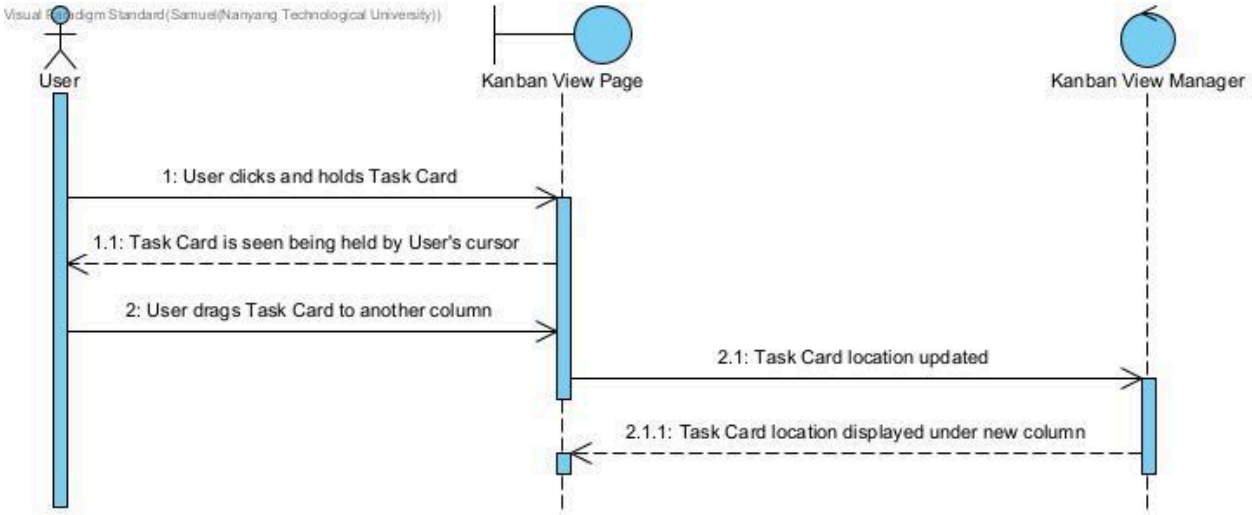
### 3.1.2. Use Case 5: Create Project



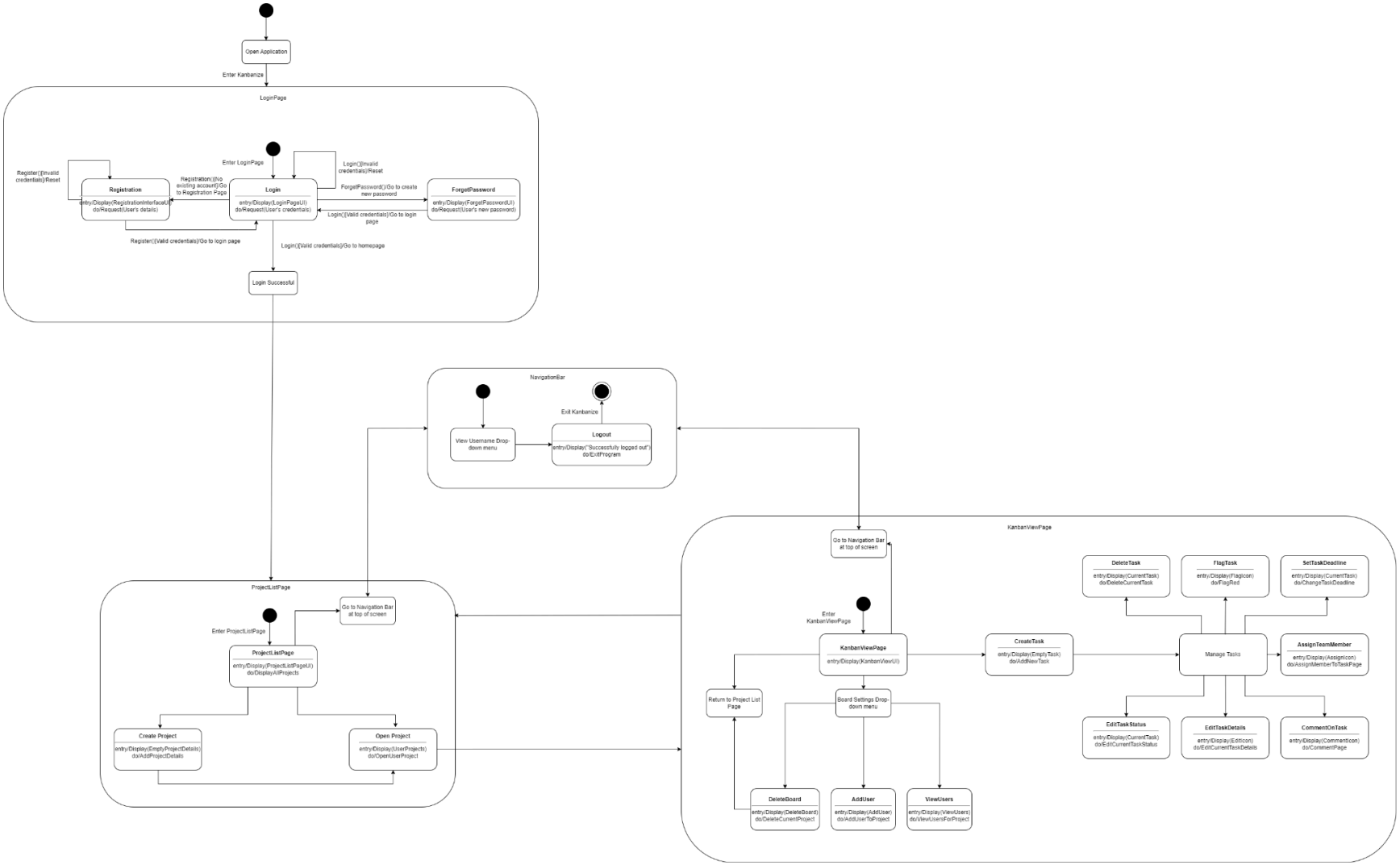

### 3.1.3. Use Case 8: Add Task



### 3.1.4. Use Case 11: Edit Task Status

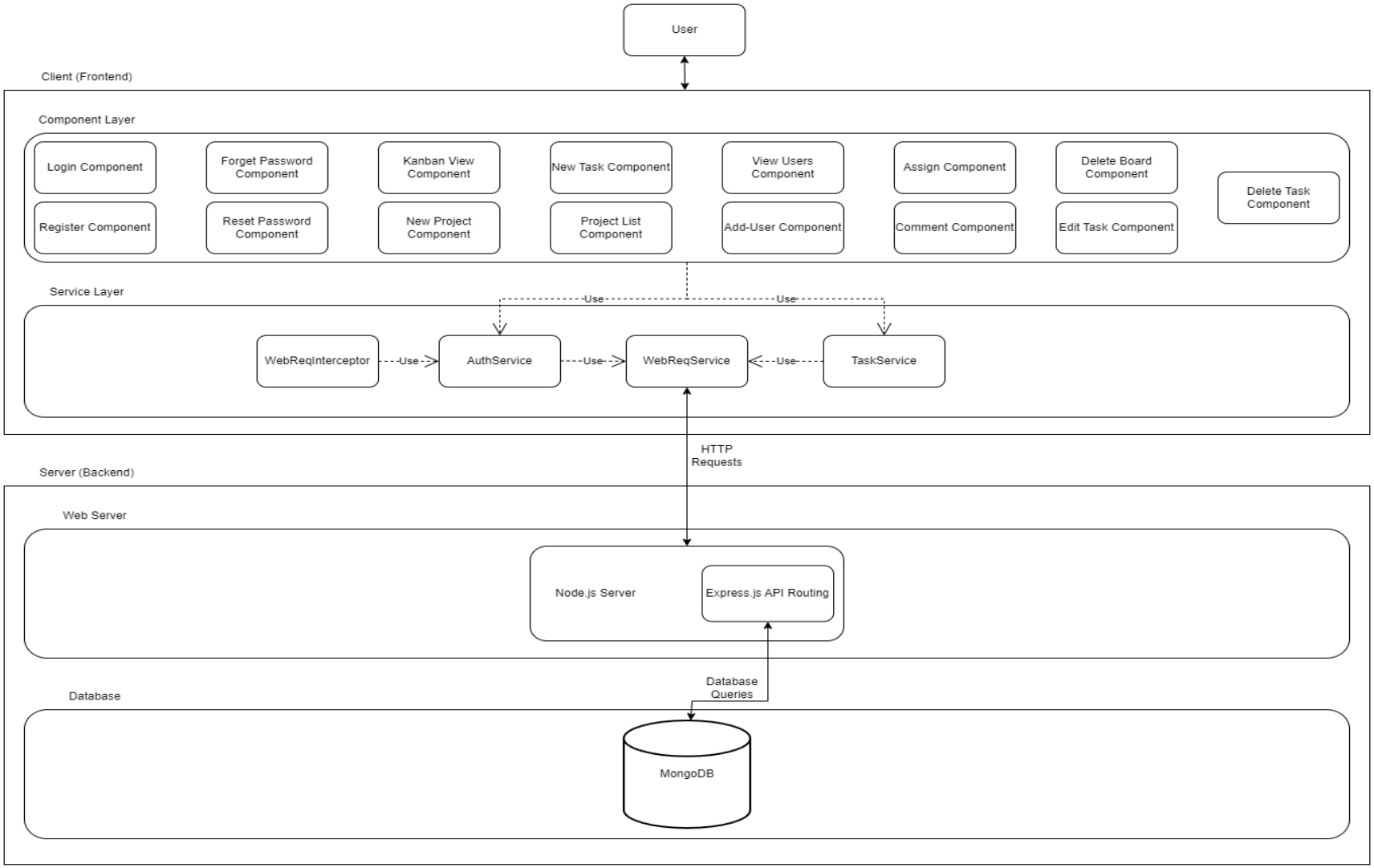


## 3.2. Dialog Map



# 4. Architecture Design

## 4.1. System Architecture Diagram



# Appendix C: Link to Source Code

The code is included in *2006-SCSG-Seow-s-Team*'s GitHub submission repository.

Refer to https://github.com/softwarelab3/2006-SCSG-Seow-s-Team for source code.

Instructions to run the program are included in the readme file.

# Appendix D: Demo Video Documentation

## 1. Link to Demo Video

Refer to https://www.youtube.com/watch?v=G8lqwwTZkqo for demo video.

## 2. Demo Video Script

Hello, we are Seow's Team and now I will be demonstrating our web application, Kanbanize. Kanbanize is a website that is designed to help teams and organizations perform seamless workflow management through the use of a Kanban board-style interface.

Moving on to our features, the first thing the user will encounter is our Login and register page. Firstly, we will create an account. Our system will check for existing usernames and email, so as to ensure that there are no conflicts. We will also check for password strength and coherency. After a successful registration, the user will be logged in and will be directed to the project list page.
I will now demonstrate the logout feature. The logout button is situated in the dropdown button on the right side of the navigation bar. Upon pressing the logout button, the user will be logged out and redirected to the login page.
Now, I want to log back in, but i forgot my password. Luckily, we have a forgot password feature. Just click the forgot password button and the user will be redirected to the forget password page. User will be prompted to enter their email that they registered the account with. To protect the integrity of our api routes, we implemented checks for invalid email syntax and emails that are not in our database. After entering a valid email address, an email with the link to reset password will be sent to the email address. As can be seen here, a new email has been sent. When we click on the link, the user will be brought to the reset password page where they will be prompted to enter a new password.

Now I will demonstrate our kanban board features. On successful login, the user will be redirected to the project list page. Here, users can view their boards or create a new one. Since this is a fresh account, there are no projects. So we will create one. User will click on the new project button and will be directed to the new project page. User will enter a project name and click create. The user will then be sent back to the project list page and we can see that the project is now in the project lists. We will check for cases where the user does not enter a name, as we take it to be invalid.

To view the kanban board, the user will click on the project and will be redirected to the kanban view page. Here, the user can edit the board name. This can be done by bringing the mouse close to the board name and clicking the edit icon. A text box will appear and the user can then enter a new name and save. If the user wants to access their other boards, they can click the back to project list button and they will be brought back to the project list page. If the user wishes to delete the board, they can do so by clicking the settings icon and then the delete board button in the dropdown. The user will be prompted to confirm the delete the board and when clicked, it will be deleted from the list.

Let's create another board to demonstrate the other features. Now, the board is empty so let's create some tasks. Users are allowed to create tasks in any column so I will pick the todo column first. I will click the + icon and will be brought to the new task page. Here I will enter The title of my task and the due date. The description is optional and can be updated at a later date. Upon clicking create, the taskcard will be created and the details are reflected on the card. The system will check for invalid title and date.

Our key feature is the drag and drop feature. Users can change the state of completion by dragging the tasks to another column. You may have noticed that when we hover over the cards icons appear. These icons represent the actions that the user can perform on the card. Users can click on the flag icon and the priority status will be toggled. These changes are reflected in the database.
Another feature is to edit the taskcard details. So to demonstrate this feature, I will add a description to the taskcard. Click on the pencil icon and we will be brought to the edit task page. Here I will enter the description and click save. The updated description is now reflected on the card.
Lets say, the task is not needed anymore. Users can delete the task by clicking on the trashcan icon. Similarly to the delete board function, Users will be prompted to confirm the deletion and upon confirmation, the task card will be removed from the board. Before I show the other 2 features, I will demonstrate the collaborative aspects of Kanbanize.

Kanbanize aims to be used in a team setting. To achieve this, we first need to be able to add users to the board. We can do so by clicking on the settings icon and then the add users button in the dropdown. Users will then be prompted to enter the username of the team member to add. The system will check if the username is valid. To check what users have access to the board, we can click on the settings icon and then the view users button in the drop down. Here, the usernames of the team members will be listed.

Before we proceed, let's move on to a board that is more fleshed out. Here, this is how it would look when used in a real setting. This is how the drag and drop feature will look when there are more cards. Users are

not only able to move the cards between columns but are also able to change the order of the cards as such.
In this board, we have 4 members as shown.
I will now demonstrate the other 2 features that we skipped over. The first one is the assign user feature. Users are able to assign responsibility of the task to a user. This is done by clicking on the user icon and selecting the user to assign to. The task can be reassigned to another user or unassigned if desired.
Lastly, we have the comment feature. By clicking on the comment button, the user will be redirected to the comment page. On the left, the details of the cards are displayed and on the right the comments are displayed. The comments are sorted by time sent. At the bottom right, users can add a comment by entering their message and clicking comment.

I have come to the end of my demonstration. We hope that this app can be used to enable organizations to better their workflow and team management.

# Appendix E: Live Demo Documentation

## 1. Link to Live Demo Slides

Refer to https://docs.google.com/presentation/d/1GenS74lX2g1gk6nqpNsx5UChWi4YJ7Zb7Hvw3V--Z-c/ for demo slides.

# Kanbanize

SC2006 Project, SCSG Team 1

Senchia Gladine, Samuel Tan, Cheyenne Seet, Samuel Seow, Willy Tang

## Table of contents



## Overview

**Kanban board**-style task and **workflow manager website**

# 01
# Use Case Diagram

## Use Case Diagram

# 02
# Innovative Features

## Innovative Features - Collaborative Capabilities

**Add User**
Multiple registered users from the database can be added into a singular project

**Task Assign**
Tasks can be assigned from anyone to anyone

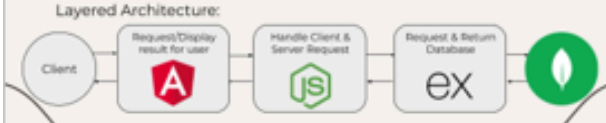
**Comment**
Users can comment on individual tasks

# 03
# System Architecture

## System Architecture

MEAN Stack:
- MongoDB - Database
- Express.js - API routing
- Angular - Frontend
- Node.js - Web server

Layered Architecture:




## Architecture Style

Model-View-ViewModel(MVVM) Architecture:
- Model: Typescript Model Classes & MongoDB
- View: HTML & CSS
- View-Model: Typescript Component Classes

Angular Framework:
- Components
- 2-Way Data Binding
- Dependency Injections

## User Authentication

Bcrypt's Salt & Hash function

JSON Web Tokens(JWT):
- Refresh & Access Tokens
- HTTP header
- Web Request Interceptor


# 04
# Design Patterns

## Design Patterns

**Observer Pattern**

Observers are used to handle asynchronous events to facilitate a responsive system.

**Service Layer Pattern**

Services are separated into a service layer to simplify service maintenance.

**Dependency Injection**

Achieve loose coupling between our ViewModel components and service components.

# 05
# Traceability

## Use Case Diagram

Registration

## Use Case Diagram

Login

## Class Diagram

## Sequence Diagram

Registration and Login

## Black Box Testing

Registration

## Black Box Testing

Registration

## Black Box Testing

Login

## White Box Testing

Registration

## White Box Testing

Login

## 06

## Software Engineering Practices

### Good Software Engineering Practices

**Version Control**

Manage Changes to the Codebase

**Documentation and Communication**

Maintains shared understanding of the software development process

## 07

## Future Upgrades

### Future Upgrades

**Advanced Filtering**

Filter and find specific tasks

**Mobile Accessibility**

Implement Progressive web app features

**Customizable UI**

Customize task cards and background

## 08

## Live Demonstration

# 2. Live Demo Script

## Introduction

**Slide 1 Tan:** Good afternoon everyone, we are Team Kanbanize and our team consists of Samuel Seow, Willy, Samuel Tan, Cheyenne and Senchia. Today, we will be presenting our SC2006 project, Kanbanize.

**Slide 2 Tan:** These are what we will be running through today with regards to our project.

**Slide 3 Tan:** Before we begin, here is a brief overview of what Kanbanize is about. Kanbanize is a website that is designed to help teams and organizations perform seamless workflow management through the use of a Kanban board-style interface. Users can customize their tasks and prioritize workflows within their workspace. Our main purpose is to visualize workflow and maximize efficiency for work management.

## Use Case Diagram and Main Functionalities

**Slide 4 Tan:** Firstly, let's refer to our Use Case Diagram

**Slide 5 Tan:** Our functionality consists of login, logout, registering of a new account and displaying of current ongoing projects and the milestones of each deliverable task on each project.

**Board Creation**

Upon creation of a valid account, users can begin to create Project Boards.

**Tasks**

Once inside a Project Board, users can create tasks, represented by sticky notes and add them to the one of the workflow columns - (To do, In progress, needs review and completed) .

When creating a task, users can set the task title, description and due date. Users can also flag/unflag and comment on tasks. Once a task is added, its details can be edited and it can be moved between the workflow columns as work progresses. Tasks can also be deleted.

**Manage Users**

Users can add other users to, or view current users of a Project Board. They can be assigned or unassigned specific tasks.

**Board Management**

Users can create additional Project Boards, view or rename their existing boards and also delete boards

**Forgot Password**

Users can reset their password using their email if they ever forget it.

## Innovative Features

**Slide 6 Cheyenne:** So what is it that really sets Kanbanize apart from all other similar applications? It is the following features that adds value to our application:

**Slide 7 Cheyenne:** The most innovative feature of Kanbanize its focus on cross-functional collaboration, which presents itself in many of its features.

**Click**

The project owner has the option to add any registered user in our database into the project as a collaborator, meaning they can add, edit, delete tasks; same as the project owner.

**Click**

Another special feature that promotes user collaboration includes the Task Assign feature, which allows any user to assign a task to any other user, encouraging user interaction within the website. The user is also allowed to unassign a task from someone to switch task allocations.

**Click**

Finally, each task card carries a comment feature which allows all users to input text separate from the task description, providing a means for users to give feedback on each assigned task.

## System Architecture

**Slide 8/9** Willy: Before going into our architecture, I will first introduce our tech stack. We used the MEAN stack which is a typescript and javascript based framework for developing scalable web apps. MEAN is an acronym for MongoDB, Express, Angular, and Node which are the four key technologies that make up the layers of the technology stack. MongoDB is used for our database, express is used for creating our api routes, Angular powers the frontend of our application and node is used to build our web server.

With this, our system takes a layered approach, with each tech responsible for a layer.

The client uses the frontend of the app and if needed angular will make a HTTP request to the server which is then handled by node. Based on what the request is, the request is then routed and express will then make the relevant query to the database. The result is then sent back to the frontend through a server response. This response will then be parsed by the frontend and then displayed to the user.

**Slide 10:** Willy

Our system follows a Model-View-ViewModel rather than a conventional Model-View-Controller. The MVVM architectural style is an extension of the MVC style. MVC separates an app into three components: model, view, and controller, while MVVM adds a ViewModel layer between the view and the model. In MVC, the controller handles user input and updates the view accordingly, while in MVVM, the ViewModel acts as a mediator between the view and the model, managing data binding and providing a clean separation of concerns.

The model is our Typescript Model Classes and mongodb database, the view is the html template and css styling, and the viewmodel is the Typescript component classes.

So why did we choose to use Angular to power our frontend? It is because of 3 main reasons. Firstly it is because of angular's component based approach. The component contains the view and viewmodel part of the system. This gives our system great modularity, as each page is a component by itself. Angular also utilizes 2-way data-binding. This means that the view can bind to a value in the viewmodel and events can be bound to trigger a specific method. Another key feature of Angular is its dependency injection capabilities. More will be explained in the design patterns.

**Slide 11:** Willy

I will now talk about how we designed our user authentication. Firstly, passwords are hashed in our database using bcrypts salt and hash method. We use bcrypt's compare method to check login credentials. Upon successful login, we create a session. We will generate a random hex string for each successful login and store it in our database. Using this hex string, we create access token by signing a jwt with a secret string and return this alongside the userid and and refresh token. these will then be stored in the browsers local storage and appended to the header of every request via a interceptor. every protected api route will check for a valid access token. if the access token is expired, a new access token will be generated and the request will be rerun. This gives us the ability to manage user login sessions in the future, similar to the likes of whatsapp, telegram and many other apps.

## Design Patterns

Model-View-ViewModel (MVVM) Pattern

**Slide 12** Willy**:** Moving on to our design patterns.

**Slide 13** Willy:

In the design phase, we noticed that many of our methods were asynchronous as they were requests to our web server. Let's say the size of our app grows and many features need to be loaded into a page. To retrieve the data we would need to call many async operations, this will result in a long loading time if we await for each operation. Therefore, we use observables to handle such operations. By subscribing to the async operations, we can queue multiple async operations and when they finally complete, the subscriptions will be notified and update the viewmodel. As such, the page will appear to be more responsive. This is also enabled by the data binding between the view and viewmodel.

click

We also used the Service Layer Pattern. We have a web request service to handle our restful api requests to our backend server. We also have 2 additional services to handle the requests related to tasks and authentication. These 2 services utilise the web request service. These services handle CRUD(create,read,update,delete) operations and manage data synchronization between the frontend and backend for our tasks, boards and users.

click

Lastly, we used dependency injection to achieve loose coupling between our service layer and components. Services are injected into components using Angular's dependency injection feature. Coupled with the service layer pattern, this means that any operations related to api calls are relegated to the service layer, which enforces strong SOLID design principles. This ensures our program is modular, extensible and we would have a codebase that is easier to maintain.

## Traceability

**Slide 12 Seow :** To demonstrate the traceability of our project, we will be going through the registration and login features.

**Slide 13 Seow:** This is the use case for the registration of an account. The flow of the events are as stated here: The user is brought to the login page first, and clicks sign up if they do not have an account. The sign up interface is displayed to the user, where they will fill up their details. The user clicks on Register, which creates their account and brings them to the project list page.

Exceptions include username or email already existing in the database, password not fulfilling the special requirement, and “password” and “confirm password” not matching.

**Slide 14 Seow :** Next, the use case for login. The flow of events are; the user is brought to the login page first, where they click login. Login interface is displayed, where users input their unique username and password in the respective fields and clicks on Login. After the Kanbanize database verifies the user’s login credentials, the user is then brought to the project list page. There is no alternative way to login.

Exception occurs when the username or password does not match the one in the system. The user will be returned to the login page to re-enter all details again.

**Slide 15 Seow:** Next will be the class diagram for this use case.

So in our diagram these are all our model view components, including registration and login. Login button calls auth service and waits for a reply, which then calls webreqservice, which then makes a HTTP request to the API. So this request will be received by the backend server to authenticate the user. Successful login will create a session for the user.

**Slide 16 Seow:** Moving onto the sequence diagram, this is the sequence of messages exchanged between the User, the interface, and Database. Our sequence diagram follows the use case flow of events, including loops where user inputs incorrect details and is prompted to redo until a successful registration or login.

**Slide 17-19 Senchia:** The testing for this function is relatively simple, using equivalence class testing techniques to generate these test cases since the inputs are discrete values. As you can see, the general cases are testing the aforementioned input requirements, namely whether the password meets requirements, matches, all fields are filled with the correct format, and email is already linked to an account. The specific cases test the validity of each user input field, first email and password, and then a combination of both.

**Slide 20 Seow:** Here is the control flow graph for user registration, with all the decision nodes numbered. We have 22 edges and 17 nodes, therefore a cyclomatic complexity value of 7. Which means 7 basis paths to test.

**Slide 21** Seow: Here is the control flow graph for user login, with all the decision nodes numbered. We have 13 edges and 11 nodes, therefore a cyclomatic complexity value of 4. Which means 4 basis paths to test.

## Good Software Engineering Practices

**Slide 22 Cheyenne:** Having good software engineering practice is vital when developing any software.

**Slide 23 Cheyenne:** When developing this system, our team utilized version control systems such as Github to manage changes to the codebase. This allows for collaboration, tracking changes, and reverting to previous versions if necessary.

**click**

Another credible SE practice is good documentation and communication throughout the course of the project. This ensures the traceability of our progress and that information is well-documented and effectively communicated among team members and stakeholders. Clear and comprehensive documentation helps in maintaining a shared understanding of the software development process and facilitates collaboration.

## Future Upgrades

**Slide 24 Senchia:** Finally, for the future development plans of our website, we aim to improve the functionality and design of our website.

**Slide 25 Senchia:** Firstly, we want to Implement advanced filtering and search capabilities that allow users to quickly find and focus on specific tasks. The tasks can be filtered based on due date, priority status and name.

**click**

Secondly, we want to ensure accessibility on mobile devices by implementing progressive web app features for a seamless mobile experience.

**click**

Lastly, users can create and customize task card fields such as attaching files, checkboxes, subtasks, labels, tags, location, customizable priority levels and activity logs for each task. Users can also change and personalize the colors of their Kanban boards.

### Slide 26: Live Demo starts here

Now we will be going into the live demonstration of our project.

Registration Do:Willy, Narrate: Seow

Hi I'm John, I am doing a software engineering project and I want to use a project management tool to help visualize workflow. I came across this website called Kanbanize that I can use.

1. First, the user will register for an account.
    a. To demonstrate error checking, we will go through a few exception cases first.

i. User will enter a username that already exists in the database
    1. Error message pops up
ii. User enters an email that already exists in the database
    1. Error message pops up
iii. User will input a password that doesn't fulfill the password requirements (8 characters minimum; one upper and lower case and one number)
    1. User will be prompted to re-enter a password.
iv. User will input password such that password and confirm password are different
    1. User will be prompted to re-enter a password

b. Now, User will enter all details correctly
c. User is brought to project list page

Logout Do:Willy, Narrate: Seow

2. Now we will demonstrate our logout feature.
    a. User clicks on their username in the Navigation bar.
    b. A drop-down menu is displayed with "Log out"
    c. User clicks on "Log out".
    d. User is redirected to the Login Page.

Login and Forget Password Do: Willy, Narrate: Seow

3. Now we will properly log into the account we created just now, using the updated password.
    a. Lets say John forgets his password and enters incorrect password
        i. Error message pops up
4. Since John forgot his password, he will reset it
    a. User clicks on "Forgot Password? Click here".
    b. User enters his email address in the textbox.
    c. System sends an email to the user's email address.
    d. The email includes a hyperlink to the Reset Password Page.
    e. User clicks on "Reset Password" and is redirected to the Reset Password Page.
    f. User will be prompted to enter a new Password, and once more to confirm the new password.
    g. User clicks "Confirm".
    h. User is redirected back to the Login Page.
    i. John will login to his account using his new password

i. User inputs their unique username and password in text boxes and clicks on the Login button.
ii. Kanbanize system verifies users login credentials.
iii. User is redirected to the Project List Page.

Create Project Do:Seow, Narrate: Senchia

5. Now we will create a new project.
    a. User is at the Project List Page.
    b. User clicks on "+ New Project".
    c. User is prompted to enter project name.
    d. A new project is created and the user returns to the Project List Page.

Switch Project Do: Seow, Narrate: Senchia

6. Now we will demonstrate how to switch between projects
    a. First create a second project to switch between projects
        i. User will end up at project list page after second project is completed
    b. User is at the Project List Page and sees the 2 projects created.
    c. User clicks on the desired project to switch to.
    d. User is redirected to that project's Kanban View Page.
    **e. Alternatively,** if the user is already in the project's Kanban View Page
        i. User clicks "← Back to Project List"
        ii. User is redirected to the Project List Page.
        iii. User switches to other project

Delete Project Do:Seow, Narrate: Senchia

7. Now we will delete the project we are currently on.
    a. User clicks on the "Settings" icon at the top right of the page.
    b. A dropdown menu is shown with "Delete Board" at the bottom.
    c. User clicks on "Delete Board".
    d. User is redirected to a page to confirm deletion of the board.
    e. User clicks "Save" and deletes the project.
    f. User is then redirected back to the Project List Page.

Add task Do: Senchia Narrate: Cheyenne

8. Now we will add tasks to the remaining project
    a. Select the desired project.
    b. Click on "+" on the  top right corner of the particular column.
    c. We will go through all exception cases first.
        i. User does not put in a Task Title:
            1. Pop-up displays: "Input a title!"
        ii. User did not input a valid date:
            1. Pop-up displays: "Input a date between today and 2 years later!"
    d. Now, we properly Fill in "Title", "Description", "Due Date" and "Progress" status.
    e. Task is created once User clicks "Create".
    f. User is brought back to Kanban View page.

Flag task Do: Senchia Narrate: Cheyenne

9. Now we will demonstrate our flag task feature
    a. User clicks on the "Flag" icon on a task card.
    b. "Flag" icon is now in red.

Edit task details Do: Senchia Narrate: Cheyenne

10. Now we will edit task details
    a. User clicks on the "Pen" icon on a task card.
    b. User is redirected to a page to edit Task Details.
    c. We will go through all exception cases first.
        i. User removed the previous Task Title:
        ii. Pop-up displays: "Input a title!"
        iii. User did not input a valid date:
        iv. Pop-up displays: "Input a date between today and 2 years later!"
    d. Now we properly edit the Task Title, Task Description and Task Due Date.
    e. User saves changes by clicking "Save".
    f. User is brought back to Kanban view page

Edit task status Do: Senchia Narrate: Cheyenne

11. Now we will edit the task status, meaning to change from to-do to completed
    a. User clicks and holds on a Task Card.
    b. User drags the Card to another Column.
    c. User releases the click.
    d. Refresh to show that its saved

Add user to project Do: Cheyenne Narrate:Tan

12. Now we will add other users in the database to our project
    a.
    b. User clicks on the “Settings” icon at the top right of the page.
    c. A dropdown menu is shown with “Add User” at the top.
    d. User clicks on “Add User”.
    e. User is redirected to a page to input a user to add.
    f. User is prompted to enter the username of the user to be added.
    g. We will go through all exception cases first.
        i. User enters a username not in database
            1. Pop-up displays: “Username not found!
    h. Now, User correctly enters the username of the user to be added
        i. User clicks “Add” and adds a new user to the project.
    i. User returns to board.

View users Do: Cheyenne Narrate: Tan

13. Now we will view all users in the current project
    a. User clicks on the “Settings” icon at the top right of the page.
    b. A dropdown menu is shown with “View Users” in the middle.
    c. User clicks on “View Users”.
    d. User is redirected to a page to view users in the current project.
    e. User can see the usernames of all the members of the project.
    f. User returns to board.

Assign task to user Do: Cheyenne Narrate:Tan

14. Now we will assign a task to the user.
    a. User clicks on the “person with a tick” icon on a task card.
    b. User is redirected to Assign User Page where the user can choose which user to assign the task to.
    c. System displays a drop down menu called “Select User”.
    d. User clicks the menu and selects the user to assign to the task.
    e. User clicks “Assign” and is shown a popup that says “User Assigned!”
    f. User returns to board

Unassign task from user Do: Cheyenne Narrate:Tan

15. Now we will unassign a task from a user
    a. User clicks on the “person with a tick” icon on a task card.
    b. User is redirected to Assign User Page where the user can choose to unassign a user from the task.
    c. System displays a drop down menu called “Select User”.
    d. User clicks the menu and selects “Unassign” at the bottom of the menu.
    e. User clicks “Assign” and is shown a popup that says “User Unassigned!”
    f. User returns to board

Delete task Do: Cheyenne Narrate:Tan

16. Now we will delete a task
    a. User clicks on a task card.
    b. User clicks on the “Trash can” icon on the bottom right-hand of the Task Card window.
    c. User is redirected to a page to confirm deletion.
    d. User clicks on “Save” and the task is deleted.
    e. User is automatically redirected to the board.

With that, we have covered the main features of Kanbanize and this concludes the end of our demonstration, thank you.

**– END –**