National College of Ireland

Technical Report:

Synch-Ur-Life

an app that helps users
plan, schedule and manage their life.

BSc (Honours) in Computing

Specializing in Cyber Security

2023/2024

Penuel Maypa

X16382003

X16382003@student.ncirl.ie

Github
Synch-Ur-Life https://github.com/pen-nci-2023/synch-ur-life Webhook Server https://github.com/pen-nci-2023/synch-ur-life-webhook-server

Contents

1.0 Introduction

1.1. Background

I started this project because I noticed that many calendar and task apps don't have all the features needed for a good life management. People often use different apps for tasks, events, and reminders, which can be scattered and overwhelming. As noted in an articles by Groenewegen, C.(2024) and Feyoh, M. (2024) reviewing shared calendar apps for families, managing various aspects of life, such as school activities, sports practices, and medical appointments, can become overwhelming. This shows the difficulty of managing many tasks with different apps. For example, a review mentions how hard it is to keep up with a family's weekly schedule, including activities like ballet recitals, baseball practices, play dates, and family events. This complexity shows that using multiple apps can cause disorganization and more stress.

My goal is to make an all-in-one app that combines these and adds special features like custom tags, categories, and saved filters for a more personalized and efficient way to organize.

The project aims to improve personal productivity and organization with an easy-to-use interface that suits different needs. It will use the latest web and mobile app technologies, mainly ReactJS and React Native, for a smooth experience on all devices, (Bahrynovska, T., 2022.). The focus will be on making the design intuitive and centred around the user, making the app easy to use and flexible for different lifestyles.

1.2. Aims

This project aims to create a straightforward, efficient task management system designed to improve productivity and organizational skills. It focuses on delivering a simple and effective way for users to manage their tasks, emphasizing ease of use, clear interfaces, and reliable functionality. The goal is to offer a solution that helps users streamline their daily routines, enhancing their ability to organize, prioritize, and complete tasks effectively. By providing a tool that is accessible and adaptable, the project seeks to support users in their personal and professional task management, making it easier for them to stay organized and focused.

1.3. Technology

My main programming language of choice is Javascript. Due to time constrain, Javascripts allows for a quicker coding and prototyping.

According to the paper "*An Empirically Based Model of Software Prototyping: A Mapping Study and a Multi-Case Study*" by Bjarnason, E., Lang, F., and Mjöberg, A. (2023) examines how prototyping benefits software development, specifically how technologies like JavaScript can speed up coding and testing.

According to the paper, JavaScript is ideal for quick prototyping in agile development thanks to its comprehensive libraries like React, Angular, and Vue. It allows developers to quickly create and test prototypes, which is essential for getting early feedback. Its simplicity and the wide range of available learning materials also make it popular, especially in startups that prioritize speedy and economical innovation.

The paper highlights JavaScript's ability to work well with different technologies, making it great for prototypes that need to interact with various system parts. JavaScript's flexibility helps create interactive prototypes that simulate real applications, letting developers efficiently showcase and improve their product ideas.

Javascript is versatile language used for both frontend and backend development. It lets you have a consistent development process for your entire application.

For Frontend, I will be using Javascript's ReactJS for the web application. ReactJS is widely used and has strong community support, which offers many resources and libraries that help speed up development and create responsive, user-friendly apps. Its scalability and modularity also make it easier to add new features or services later. I will use React Native to develop the mobile application version of the project.

- Node.js with Express.js is used for back-end, which comes with React JS.
- Expo — Expo is a comprehensive framework designed to simplify the development of mobile applications for Android and iOS using React Native. Expo is used speeds up development by easily setting up environments for Android and iOS, allowing real-time updates with live reloading, and offering APIs for native device features without needing native code — *[https://expo.dev]*
- Firebase – for authentication, real-time database, and hosting.
- Heroku – a cloud hosting solution for backend deployment.
- Git for version control.
- Quire – for tracking progress and managing task.

1.4. Structure

This document is structured to provide a clear and comprehensive understanding of the project, outlining its key components, methodologies, and functionalities. Each section has been meticulously designed to guide the reader through the project's progression, from initial concept to final implementation.

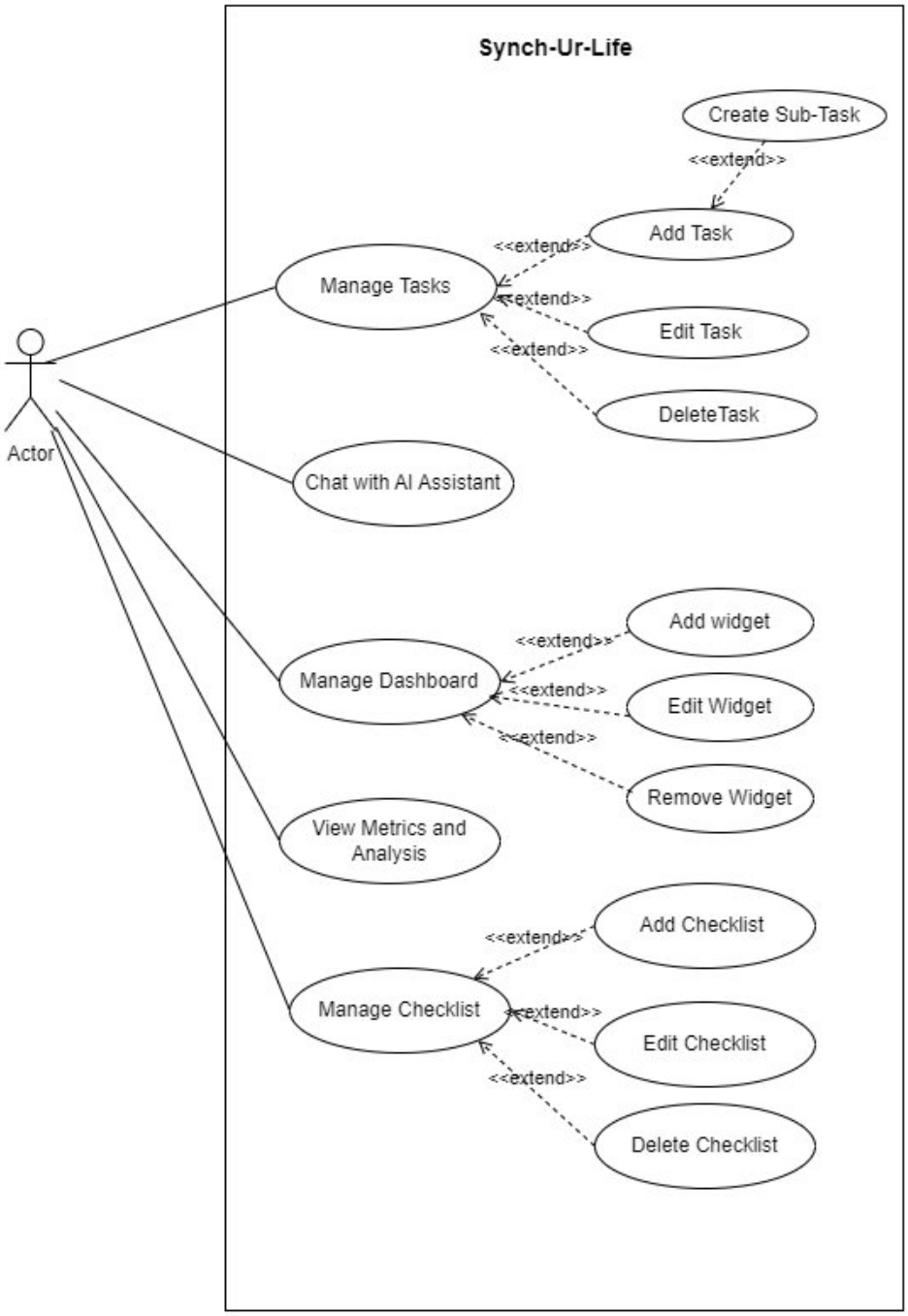
- **Section 1.0 Introduction**: Introduces the project, outlining the background, aims, and the overarching goals, setting the stage for the detailed technical exposition that follows.
- **Section 2.0 System Overview**: Delivers a high-level description of the system, encapsulating the core functionalities, system architecture, and the technology stack employed in the development of the application.
- **Section 2.1 Requirements**: Delineates the specific requirements that the system is designed to fulfill, including both functional and non-functional specifications.
- **Section 2.1.1 Use Case Descriptions**: Explores various use cases such as Manage Task, Add Task, Edit Task, and Delete Task, providing insights into the system's interaction with its users and detailing the user-driven functionalities.

2.0 System

2.1. Requirement

2.1.1. Functional Requirements

2.1.1.1. Use Case Diagram



2.1.1.2. Requirement 1: Manage Task

This use case allows users to manage their tasks within the application. It encompasses creating new tasks, viewing existing tasks, editing tasks, and deleting tasks.

2.1.1.3. Description & Priority

This requirement ensures users can effectively manage their tasks, crucial for the app's core functionality. It allows task creation, viewing, editing, and deletion, vital for user organization and productivity. Its priority is high, reflecting its significance in the system's overall utility and user experience. Ensuring this feature is user-friendly and reliable is essential for the app's success.

2.1.1.4. Use Case 1: Manage Task

Description

This use case describes the process through which users can manage their tasks, including creating new tasks, viewing a list of tasks, editing existing tasks, and deleting tasks.

Flow Description
Precondition

- The user must be logged in to the homepage.
- The system must be able to connect to the firebase services.
- There must be a list of tasks on the task board.

Activation

- This use case starts when the user clicks/selects a task that are list under “Tasks” or when the user selects the task’s add, edit and delete icon.

Main flow: Manage Task

1. The user navigates to the Home page.
2. The system displays a list of existing tasks.
3. The user can select a task on the list and decide whether to edit or delete it.

Alternate flow

- The user must select any of the task and either select “Add”, “Edit” or “Delete” icon.
- Add task – go to Extended Use Case 1.1
- Edit task – go to Extended Use Case 1.2
- Delete task – go to Extended Use Case 1.3

Exceptional flow

- The user accesses the Home page, but due to slow network conditions or server response times, the task list takes an excessive amount of time to load.

- If the system reaches a loading timeout threshold without successfully displaying the task list, it will then display a message "Loading tasks has timed out. Please check your network connection and try again."
- The user can refresh the page.

Termination

- This use case terminates when the user navigates away from the task management page or logs out.

Post condition

- The user has successfully managed tasks, including creating, editing, or deleting tasks.
- The system reflects the changes immediately on the user interface.
- The updated task details are stored persistently in Firebase.

2.1.1.5. Extended Use Case 1.1: Add Task

Description

This use case describes the process use case to enhance their task management by creating new tasks directly from the dashboard, ensuring their task list is always up-to-date and reflective of their current priorities.

Precondition

- The user must be logged into the system.
- The system must be able to connect to the firebase services.
- The user must be on the dashboard page within the "Manage Tasks" use case.

Activation

- The user clicks on the "add" icon next to a task in the task list.

Main flow:

1. The user is viewing their dashboard, which displays today's task list.
2. The user clicks the "add" icon (+) next to an existing task.
3. The system opens a form for task entry.
4. The user fills out the task details, including a description and any other required fields.
5. The user submits the form by selecting the "Add" button.
6. The system validates and saves the new task.
7. The system closes the form and the user sees the newly added task on the list in the dashboard.

Alternate flow

A. Incomplete or invalid form submission:
    1. The system displays an error message detailing what needs to be corrected.

2. The user adjusts the information accordingly and resubmits.

B. User cancels the task creation:
    1. The user selects a "Cancel" option or clicks outside the form.
    2. The system closes the form without saving the new task.

C. User can add a subtask to an existing task which smaller, manageable units of work that are part of a larger task.
    1. During task creation, the user selects the option to "Add Subtask".
    2. The system presents a form for entering subtask details (title, description, deadline, etc.).
    3. The user fills in the subtask details and submits the form.
    4. The system validates the subtask details and, upon success, associates the subtask with the main task.
    5. The system confirms to the user that the subtask has been added.
    6. If the user decides to add another subtask, they repeat steps 1-5 as needed.

Exceptional flow

- If the system encounters an error saving the new task (e.g., database error):
    - The system displays an error message: "Unable to add your task at this time. Please try again later."
    - The form remains open allowing the user to try to save again or cancel.

Termination

- The use case terminates when the new task is successfully added to the task list and the user is notified of the successful addition, or the task creation form is closed.
- If the user cancels the task creation, the use case also terminates once the form is closed without saving any input.
- If an error prevents the task from being saved, and the user chooses not to attempt to save again, the use case terminates when the user exits the error message and closes the form.

Post condition

- A new task is successfully added to the user's task list.
- The task list on the dashboard reflects the addition of the new task.

2.1.1.6. Extended Use Case 1.2: Edit Task

This use case describes the process through which users can create a new task.

Description

This use case details the procedure for users to edit an existing task, including altering task details such as title, description, and due date, to ensure the task list remains accurate and reflective of their current objectives.

Precondition

- The user must be logged into the system.
- The system must be able to connect to the firebase services.
- The user must be on the dashboard page within the "Manage Tasks" use case.
- The user must have selected a task to edit from the task list.

Activation

- The user clicks on the "edit" icon next to the task they wish to update.

Main flow:

1. The user is viewing their dashboard, which displays today's task list.
2. The user clicks the "edit" icon (typically a pencil symbol) next to the task they want to modify.
3. The system opens the task in an editable form, pre-filled with the current details of the task.
4. The user modifies the task details as needed, which may include the title, description, due date, priority, etc.
5. The user submits the updated information by selecting the "Save" button.
6. The system validates the updated information.
7. If validation passes, the system saves the updated task.
8. The system confirms the successful update by displaying a success message and updating the task details on the dashboard.

Alternate flow

A. Incomplete or invalid form submission:
    1. The system displays an error message detailing what needs to be corrected.
    2. The user adjusts the information accordingly and resubmits.

B. User cancels the task update:
    1. The user selects a "Cancel" option or clicks outside the form.
    2. The system closes the form without saving changes and reverts to displaying the original task details.

Exceptional flow

- If the system encounters an error saving the updated task (e.g., database error):
    - The system displays an error message: "Unable to save your changes at this time. Please try again later."
    - The form remains open allowing the user to try to save again or cancel.

Termination

- The use case terminates when the user's changes to the task are saved and acknowledged by the system, or the user exits the editing process either by cancelling or closing the form.

Post condition

- The task is updated with new details in the user's task list.
- The task list on the dashboard reflects the changes made to the task.

2.1.1.7. Extended Use Case 1.3 : Delete Task

Description

This use case details the procedure for users to remove a task from their list, ensuring the task list remains current and uncluttered by eliminating completed or obsolete tasks.

Precondition

- The user must be logged into the system.
- The system must be able to connect to the Firebase services.
- The user must be on the dashboard page within the "Manage Tasks" use case.
- The user must have identified a task they wish to delete.

Activation

- The user clicks on the "delete" icon next to the task they intend to remove.

Main flow:

1. The user navigates to the task they want to delete.
2. The user clicks the "delete" icon, prompting a confirmation dialogue to ensure the user wants to proceed with deleting the task.
3. Upon confirmation, the system permanently removes the task from the list.
4. The system updates the task list to reflect the removal.

Alternate flow

1. If the user decides not to delete the task during confirmation, they select the "cancel" option.
2. The system closes the confirmation dialogue and retains the task without changes.

Exceptional flow

- If the system encounters an error during the deletion process (e.g., network error, database issue), it displays an error message: "Unable to delete the task at this time. Please try again later."

Termination

- The user has successfully received the response from the AI Assistant.
- The response from the AI Assistant, including any tasks or information requested, is displayed in the application.
- If the query involved adding or updating tasks, those tasks are now reflected in the Firebase database.
- The interaction session with the AI Assistant ends, and the user can either close the chat interface or continue with other activities within the application.

Post condition

- The user receives a relevant response from the AI Assistant based on their query.
- The application displays the tasks or information requested by the user.

2.1.1.8. Use Case : Chat with AI Assistant

Description

This use case describes how a user interacts with the AI Assistant to add tasks, get reminders, or seek help using natural language queries. The AI Assistant processes the user input through Dialogflow and returns relevant responses that are displayed in the application.

Precondition

- The user is logged into the Synch-Ur-Life application.
- The AI Assistant is integrated and configured properly within the application.
- The webhook server for Dialogflow is running and accessible.

Activation

- The user activates the AI Assistant by navigating to the AI interaction section of the application or clicking on a "Chat with AI" button.

Main flow:

1. The user navigates to the part of the application where they can interact with the AI Assistant, in a dedicated input field.
2. The user types a natural language query into the input field (e.g., "What are my tasks for tomorrow?").
3. The application sends the user's query to the webhook server using a POST request.
4. The webhook server receives the query and forwards it to Dialogflow for natural language processing
5. Dialogflow processes the query, identifies the intent, and prepares a response. If the intent is to retrieve tasks, Dialogflow includes the necessary parameters in the response.
6. The webhook server receives the response from Dialogflow.
7. The webhook server sends the response back to the application.
8. If the response includes an action such as retrieving tasks, the application queries the Firebase database for the relevant tasks.
9. The application displays the response from the AI Assistant to the user, including the list of tasks for tomorrow.

Alternate flow

A. Invalid Query
    1. If the user inputs a query that the AI Assistant cannot understand or process, the application displays an appropriate error message or prompt for clarification.

B. Network/Server Error
    1. If there is a network or server error while processing the query, the application displays an error message indicating that the AI Assistant is currently unavailable

Exceptional flow

- If the system encounters an error saving the updated task (e.g., database error):
    - The system displays an error message: "Unable to save your changes at this time. Please try again later."
    - The form remains open allowing the user to try to save again or cancel.

Termination

- The use case terminates when the user's changes to the task are saved and acknowledged by the system, or the user exits the editing process either by cancelling or closing the form.

Post condition

- The user receives a relevant response from the AI Assistant based on their query.
- The application displays the tasks or information requested by the user.

2.1.1.9. Requirement 2: Data Analysis and Task Metrics

This functionality analyzes user data to enhance productivity, identifying procrastinated tasks and providing insights into time management. It assesses how users allocate time across various activities, offering a detailed view of their task engagement. This features aids users in optimizing their schedules and improving task prioritization, integral to the system's effectiveness.

2.1.1.10. Requirement 2: Data Analysis and Task Metrics

2.1.1.10.1. Description & Priority

This requirement is pivotal, focusing on a simplified yet powerful analysis of user tasks and time management. It highlights delayed tasks and categorizes time spent on activities, crucial for enhancing user productivity and prioritization. Given its direct impact on user engagement and efficiency, this feature is of high priority. It's essential for helping users optimize their schedules and gain meaningful insights into their task management patterns, thereby improving overall time utilization.

2.1.1.10.2. Use Case: View Metrics and Analysis

Description

This use case enables users to view and interpret their task management metrics and analysis, providing insightful data to enhance productivity and task prioritization.

Precondition

- The user must be logged into the system.
- The system has completed the data analysis and generated the relevant metrics.
- The user has access to the dashboard or the section where metrics are displayed.

Activation

- The user selects the option to view their metrics and analysis from the dashboard.

Main flow:

1. The user navigates to the analytics section of the application.
2. The system displays various metrics and analyses, including task completion rates, time allocation, and procrastinated tasks.

Alternate flow

1. The metrics are not updated, and the user refreshes the data.
2. The user customizes the type of data or period they want to view.

Exceptional flow

- If the system cannot retrieve or display the analytics data, it shows an error message and suggests the user try again later.

Termination

- The use case ends when the user exits the analytics view or logs out of the system.

Post condition

- The user gains insights from the displayed metrics, potentially influencing their future task management and scheduling decisions.

2.1.1.10.3. Use Case: Manage Dashboard

Description

This use case involves the configuration and customization of the user's dashboard to manage and organize widgets such as calendars, task lists, checklists, and analysis boards.

Precondition

- User must be logged into the system with appropriate permissions.
- The dashboard is the user's current view.

Activation

- The user initiates the use case by selecting an option to manage widgets from the dashboard.

Main flow:

1. The user selects the manage option for widgets.
2. The system presents available widget options – add, edit, or remove.
3. For adding, the user selects 'Add Widget', chooses the widget type, and positions it on the dashboard.
4. For editing, the user selects an existing widget and modifies its settings.
5. For removal, the user selects 'Remove Widget' and confirms deletion.

Alternate flow

1. The user attempts to add a widget but there is no more space on the dashboard.
2. The user tries to edit a widget but the settings are restricted or unavailable.

Exceptional flow

- If a system error occurs while managing widgets (e.g., server unresponsiveness), an error message is displayed.

Termination

- The use case concludes when the dashboard reflects the user's changes, or the user exits the management mode.

Post condition

- The dashboard is updated to reflect the user's custom arrangement and selection of widgets.

2.1.2. Non-Functional Requirements

2.1.2.1. Data Requirement

The system will handle various data types data primarily stored in Firestore, a non-relational NoSQL database. It ensures data integrity, supports JSON/XML formats, and provides mechanisms for secure data transfer and encryption at rest. There are four types of data entity; Task, Notes, Day Journal and User.

2.1.2.1.1. Types of Data

- Tasks — actionable item that users can create, track, and complete. It serves as a component for task management, enabling users to organize their work efficiently and effectively in the application's ecosystem.
    - Attributes:
        - **ID:** a unigque id, automatically generated by firebase
        - **Date and Time**: Utilizes the date and time data type to record when tasks are set or due.
        - **Location**: Stored as a string, indicating the location associated with the task.
        - T**ask Name**: A string representing the name or title of the task.
        - **Task Description**: A long string providing detailed information about the task.
        - **Category**: A string that categorizes the task for easy sorting and retrieval.
        - **Reminders**: Utilizes date and time data type for setting reminders.
        - **Tags**: Strings used for tagging tasks for easy search and categorization.
            - Add other date: Capability to attach related Notes data.
        - **Priority**: Importance or urgency level.
        - **Status**: Current state of the task (e.g., pending, in progress, completed).
        - **Due** Date: When the task is expected to be completed.

- Notes — serves as a tool within the application, allowing users to create, store, and manage rich-text notes along with relevant attachments and connections to other entities.
    - Attributes:
        - **Title**: A string summarizing the note.
        - **Body**: A long string for the main content of the note.
        - **Attachments**: Ability to include images or videos. Other items such as Tasks and Journal can be attached.
        - **Tags:** Strings for easy categorization and retrieval.
- DayJournal — Represents diary or journal entries, allowing users to document their daily experiences.
    - Attributes:
        - **Title**: A string title for the journal entry.
        - **Body**: Long string for detailed journal content.
        - **Attachments**: Options to attach images, videos, or related tasks.
- User — Data pertaining to user identification and account management.
    - Attributes:
        - **Email**: The user's email address.
        - **Username**: A unique identifier for the user.
        - **Password**: A secure string for user authentication.

2.1.2.1.2. Data Structure and Storage

The project's data storage and management utilize Firebase, a non-relational NoSQL database, to facilitate flexible and scalable data handling. The setup offers immediate data updates and strong performance, free from the limitations typical of standard relational databases. Additionally, users are provided with the functionality to download and save their data locally, enhancing personal data control. For app users on personal devices like Android and iOS, there's an option to save data directly on their devices, where it will be securely encrypted. The system also includes an offline feature for the web app, ensuring continuous accessibility and user engagement, even without an internet connection. This method guarantees a smooth and safe experience for all users, meeting the diverse needs of everyone.

2.1.2.1.3. Security and Compliance

The application’s data management system complies with GDPR and Irish laws, ensuring the protection of user data privacy and security. The application's data management is fortified by Google's security protocols, with Firebase's configuration object securely linking the app to essential Firebase services.

This secure setup ensures safe data handling, keeping data intact during transfer and storage. The system meets legal standards and maintains strict security, allowing trusted data use within the app, customized to user roles and permissions.

2.1.2.2. Performance Requirement

The "Synch-ur-life" app's performance is key for users to manage tasks smoothly, aiming for a straightforward and user-friendly experience. This section defines the key performance requirements to guarantee the app operates swiftly and reliably. According to Dennis et al. (2010), performance requirements focus on issues such as response time, capacity, and reliability, which are essential for maintaining a seamless operation of the system. Meeting these requirements is crucial for ensuring that the app runs smoothly, especially when many users are on it:

- Response Time — The app aims for sub-second response times for all user interactions. Whether it's task manipulation or navigation through the app, users expect immediate feedback.
- Data Handling — Firebase integration must be optimized for quick data fetches and updates. The app requires efficient data handling strategies to prevent delays in task updates and retrievals.
- Offline Performance — The offline capabilities of the web app are designed to offer full functionality without network access. Performance in this mode should mirror the online experience, with changes syncing when reconnecting.

2.1.2.3. Usability Requirement

According "*Systems Analysis and Design: An Object-Oriented Approach with UML*" by Alan Dennis, Barbara Haley Wixom, and David Tegarden, the usability non-functional requirement typically refers to how user-friendly and intuitive the system is for end-users. It encompasses aspects such as the ease of navigating the system, the clarity of the user interface, and how satisfying and straightforward the system is for users to accomplish their tasks, a process which involves participants completing tasks while their interactions are observed to improve the product's design and user experience (Moran, K. 2019). Usability focuses on optimizing the user experience by ensuring the system is accessible, understandable, and efficient, thereby enhancing overall user satisfaction and interaction with the system.

The Synch-Ur-Life application will be designed to offer a simple and intuitive interface, making it easy for users to navigate and complete tasks efficiently, thereby boosting the application's overall productivity.

The application is designed to have a Responsive Design, meaning it will automatically adjust to fit various devices, ensuring a consistent and smooth user experience across different platforms. This design strategy enhances accessibility and user satisfaction by maintaining a

uniform usability and visual appeal. The application will be designed to meet accessibility standards, ensuring it is usable for everyone, including individuals with disabilities. This will broaden the app's usability.

The application is designed to incorporate ReactJS, enhancing the user interface with its dynamic and responsive capabilities, leading to improved user engagement. It also uses Node.js with Express.js for backend efficiency, supporting the front end's need for fast data handling, ensuring the app remains responsive and performs reliably, optimizing the overall user experience.

2.1.2.4. Reliability

Reliability is discussed in *'Systems Analysis and Design with UML, 3rd Edition*' in several contexts, highlighting its importance in system development. In the case of my application, the requirement for reliability is not a top priority. Any minor downtime would not critically affect the user. However, we still need to aim for high reliability. There are several strategies that can be implemented to enhance reliability:

- Using many servers together helps handle requests well. If one server stops working, the others keep the service running smoothly. This method works like AWS or Azure cloud services. The application uses external cloud services —Heroku for deployment and management, it's less likely to experience failures, as Heroku is known for being reliable. Which there is no need for to me manage the physical infrastructure.
- The application uses Google's Firebase, which already provides its self-monitoring tools. Firebase, part of Google's extensive cloud services suite, is renowned for its reliability and comprehensive service offerings. It provides a host of features including real-time databases, user authentication, and hosting solutions, all designed to facilitate smooth app development and deployment.
- One of the most important for reliabiliy is by adopting a strict code standard from the start of development , and implementing , continuous integration (CI) practices, and automated testing to catch potential faults early in the development process - (*IEE |The Importance of Software Testing. ND*.).

2.2. Design & Architecture

2.2.1. System Overview

The Synch-Ur-Life app is a cross-platform tool that offers a consistent user experience on both web and mobile devices. The system uses modern web technologies and a cloud-based backend to ensure it can scale, remain reliable, and synchronize data in real time.

2.2.2. Architectural Design

This section explains the architecture of the Synch-Ur-Life application and how each layers are structured. The architecture also follows the ReactJS way, embracing component-based design, state management, and hooks, which differ from traditional architectural designs.

Presentation Layer:

- **Web Frontend:** Developed using ReactJS, this layer provides a responsive and interactive user interface for the web application. It includes components for task management, calendar views, and user interactions.
    - **Key Files:**
        - **App.js**: The main application file that integrates various components and manages the overall state of the application.
        - **Calendar.j**s: A component that displays the calendar view, allowing users to navigate between months and select dates.
        - **Utilities.js**: Contains utility functions used across the application.
- **Mobile Frontend:** Built with React Native, this layer ensures the application is accessible on both Android and iOS devices. It mirrors the functionality of the web frontend, offering a consistent user experience across platforms

Business Logic Layer:

- This layer handles the core functionality of the application, including task creation, editing, deletion, and interactions with external services. The business logic is embedded within the React components and utility functions, ensuring efficient data processing and management.
    - **Key Functionalities:**
        - **Task Management:** Functions within App.js for adding, updating, and deleting tasks.
        - **Date Management:** Utility functions in Utilities.js for date formatting and manipulation.

Data Access Layer:

- The data access layer is responsible for communicating with the backend services, primarily Firebase Firestore. It manages the data flow between the application and the cloud database, ensuring real-time updates and synchronization.
    - **Key Files:**
        - **firebaseConfig.js:** Configures Firebase and initializes Firestore for real-time database operations.
        - **App.js - CRUD Operations:** Implemented within App.js using Firestore methods for adding, retrieving, updating, and deleting tasks.

Backend Services:

- **Webhook Server:** An Express.js server that handles interactions with Dialogflow for natural language processing. It processes user queries, interacts with Dialogflow, and sends responses back to the application.
    - **Key File:**
        - **index.js**: The main server file that sets up the Express.js server, handles incoming requests, and communicates with Dialogflow.
        - **Firebase:** Provides authentication, real-time database, and cloud functions. Firebase ensures secure data storage, user authentication, and real-time synchronization of data across devices.

2.2.3. Architecture of the ReactJS way

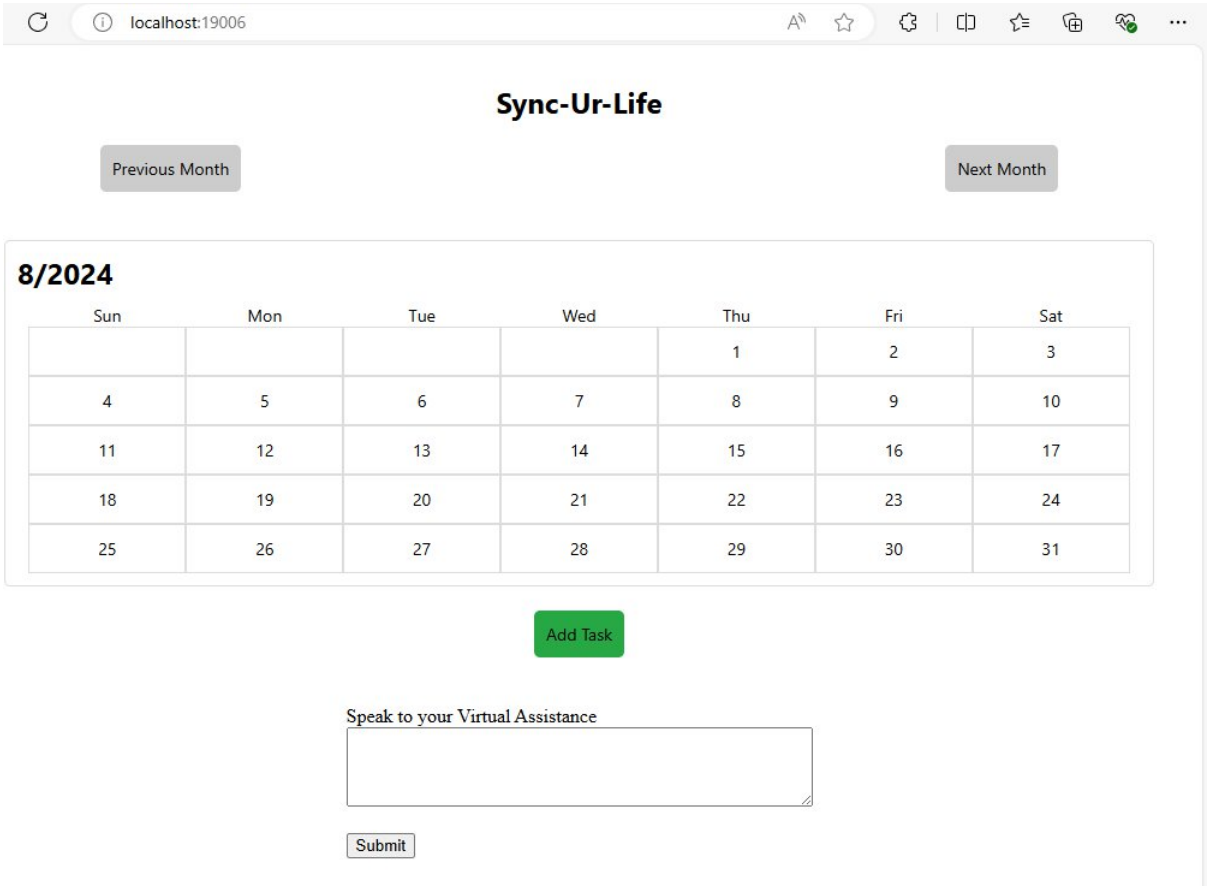
This section explains how the application adopts the ReactJS approach, embracing several key principles that distinguish it from traditional architectural designs, with a focus on the advantages of ReactJS in terms of architecture.

1. Component-Based Architecture:
    - **Reusable Components**: The application is built using reusable components, such as TaskManager and Calendar. This modular approach allows for easier maintenance and scalability.
    - **Separation of Concerns**: Each component encapsulates its own logic and presentation, leading to a clean separation of concerns.
2. State Management:
    - **Hooks**: The application uses React hooks (e.g., useState, useEffect) to manage state and side effects. This provides a functional approach to state management, making the code more readable and maintainable.
    - **Context API**: Where necessary, the Context API can be used to manage global state, although it is not explicitly used in the current codebase.

3. Declarative UI:
    - **JSX Syntax**: The use of JSX syntax allows for declarative UI design. This means that the UI is described in terms of what it should look like, rather than how to achieve it procedurally.
    - **Responsive Updates**: React efficiently updates and renders components when the underlying data changes, ensuring a responsive user interface.
4. Integration with External Services:
    - **Firebase Integration**: The data access layer integrates directly with Firebase Firestore for real-time database operations. This integration is handled within the components using Firestore methods, allowing for real-time updates and synchronization without traditional RESTful API calls.
    - **Dialogflow Integration**: The webhook server interacts with Dialogflow, providing natural language processing capabilities that are seamlessly integrated into the React components.

2.3. Graphical User Interface (GUI)

The image below shows the landing page of the application.



2.4. Testing

Testing is done by Use Case. Each Use Case are tested individually.

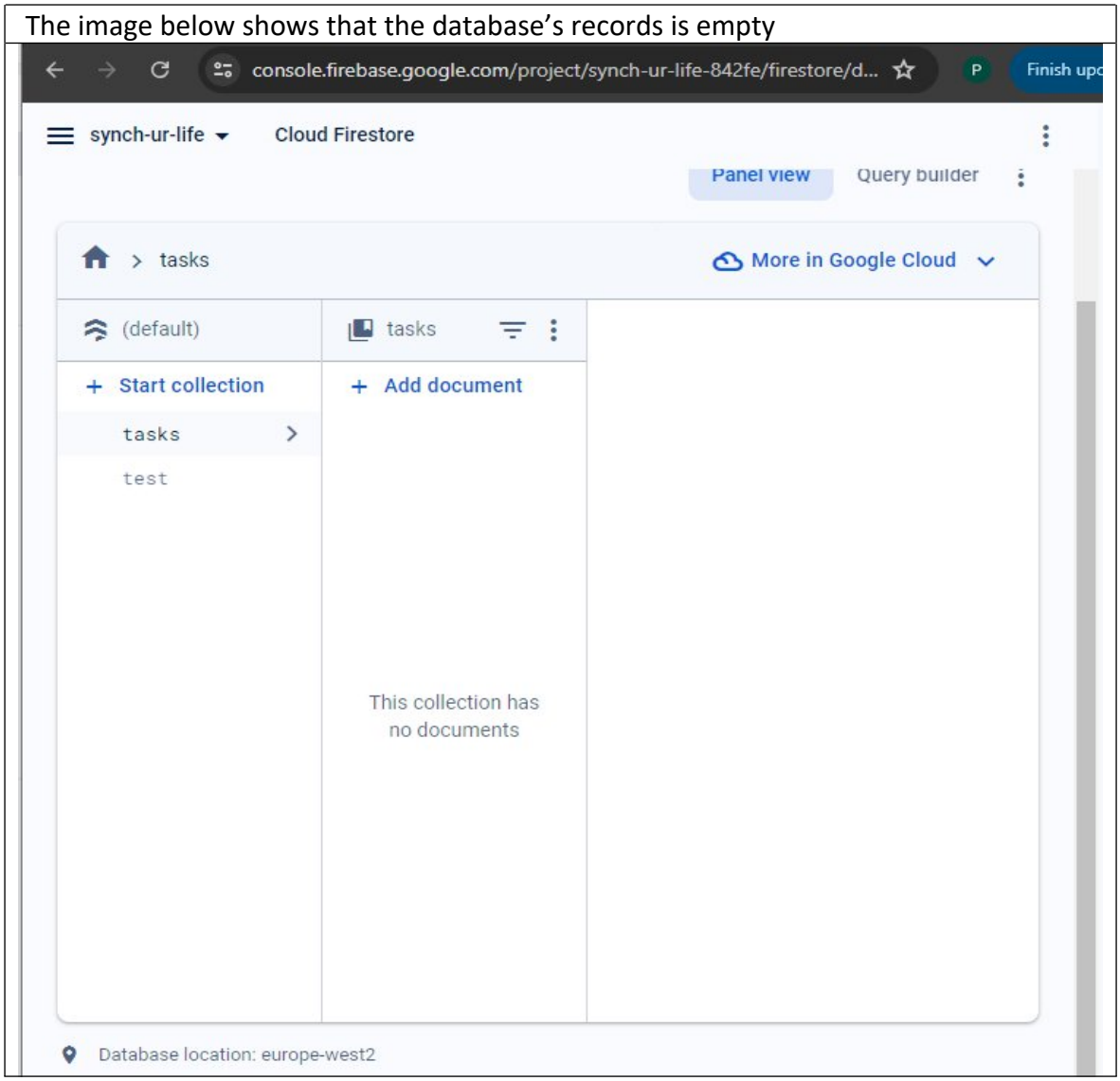
2.4.1. Test Case: Add Task

The testing for “Add Task” item:

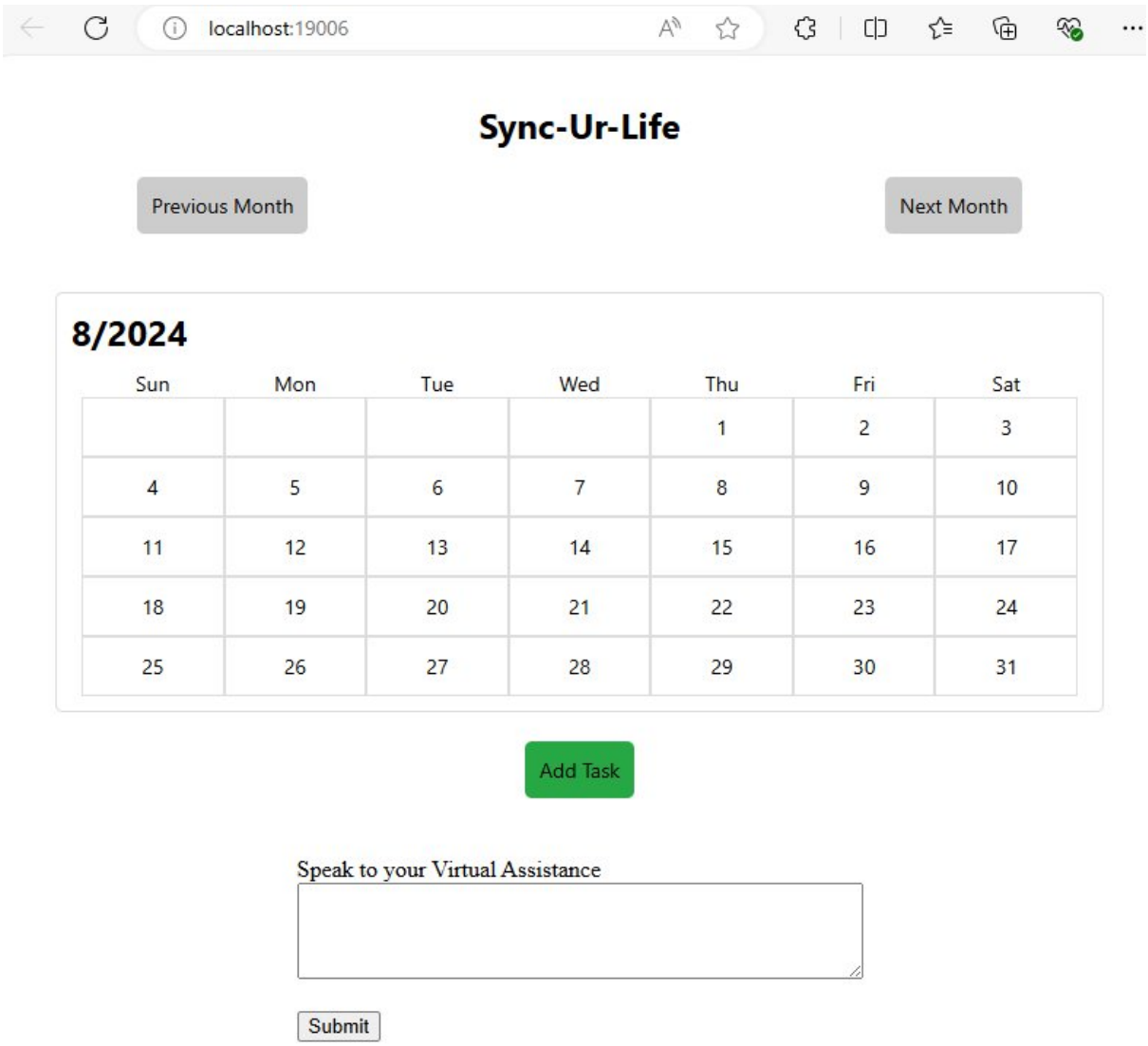
Precondition

- Ensure the database is empty by clearing all records.
- Verify that the database records are empty.

The image below shows that the database’s records is empty



— The image shows a landing page (main page) of the application. There are currently no task being listed.



Steps:

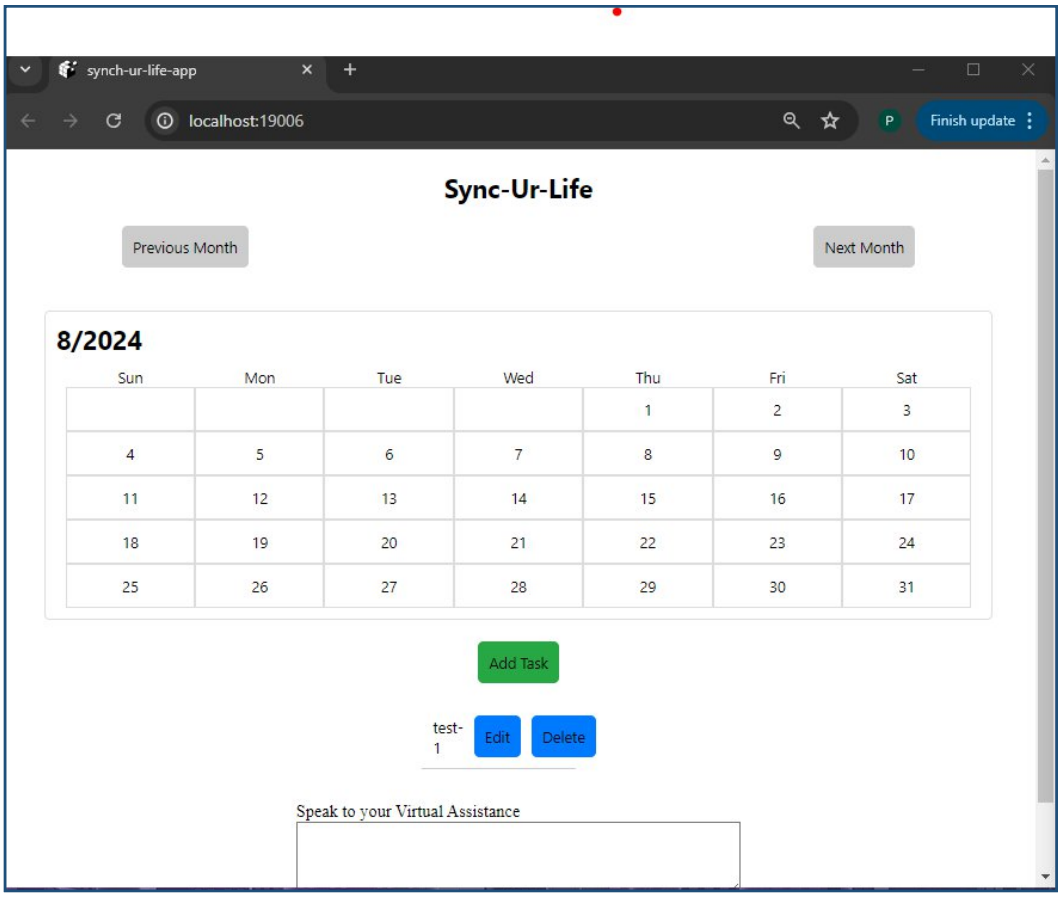
1. Open the application and navigate to the homepage.
2. From the homepage, navigate to the dashboard page .
3. Locate and click on the "add task" button to open the task creation form
4. In the task creation form, fill in the following details:
    - Description: "test-1"
    - Start Date: Select a start date.
    - End Date: Select an end date.
    - Tags: "testing"
5. Click the "submit" button to add the task.

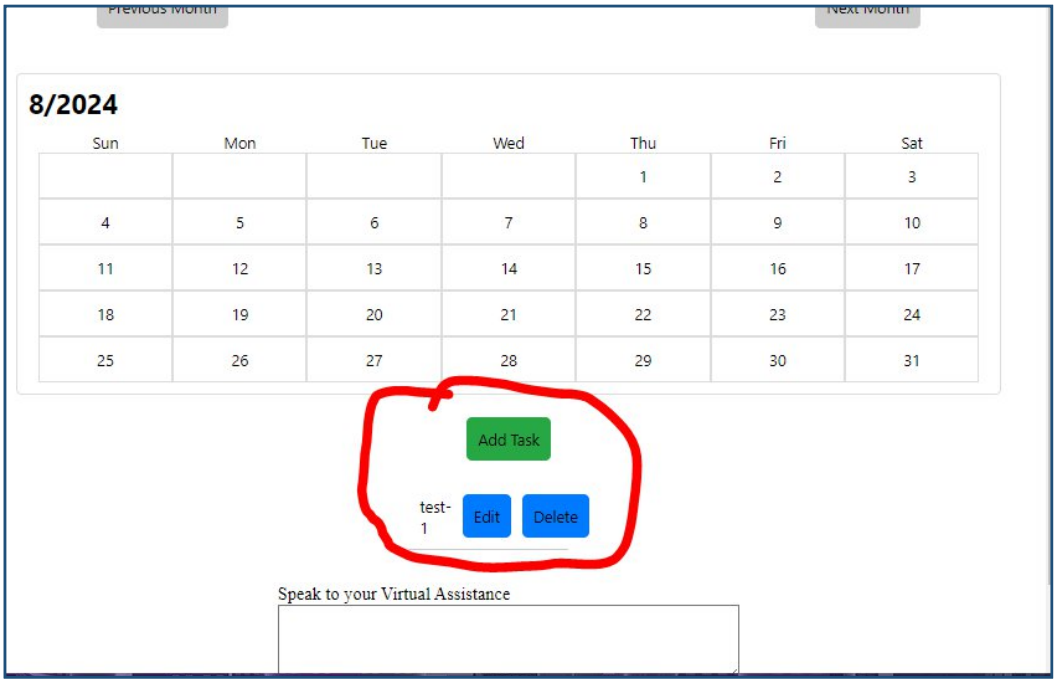
Expected Result:

- The newly added task ("Test Task") should be listed on the dashboard page.
- Verify that the Firebase database has been populated with the new task.
- The database entry should include the task details (description, start date, end date, tags)

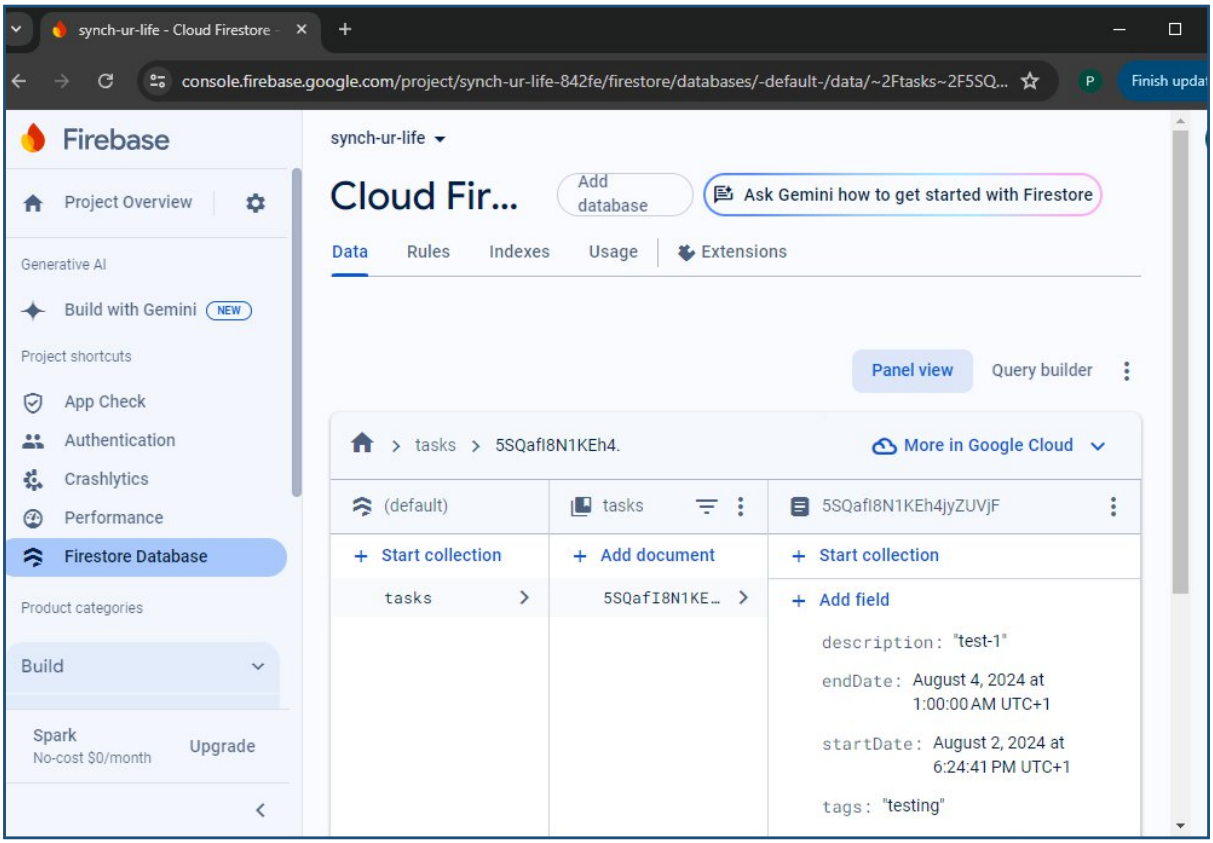
➢ An image displaying the addition of a task through form completion.

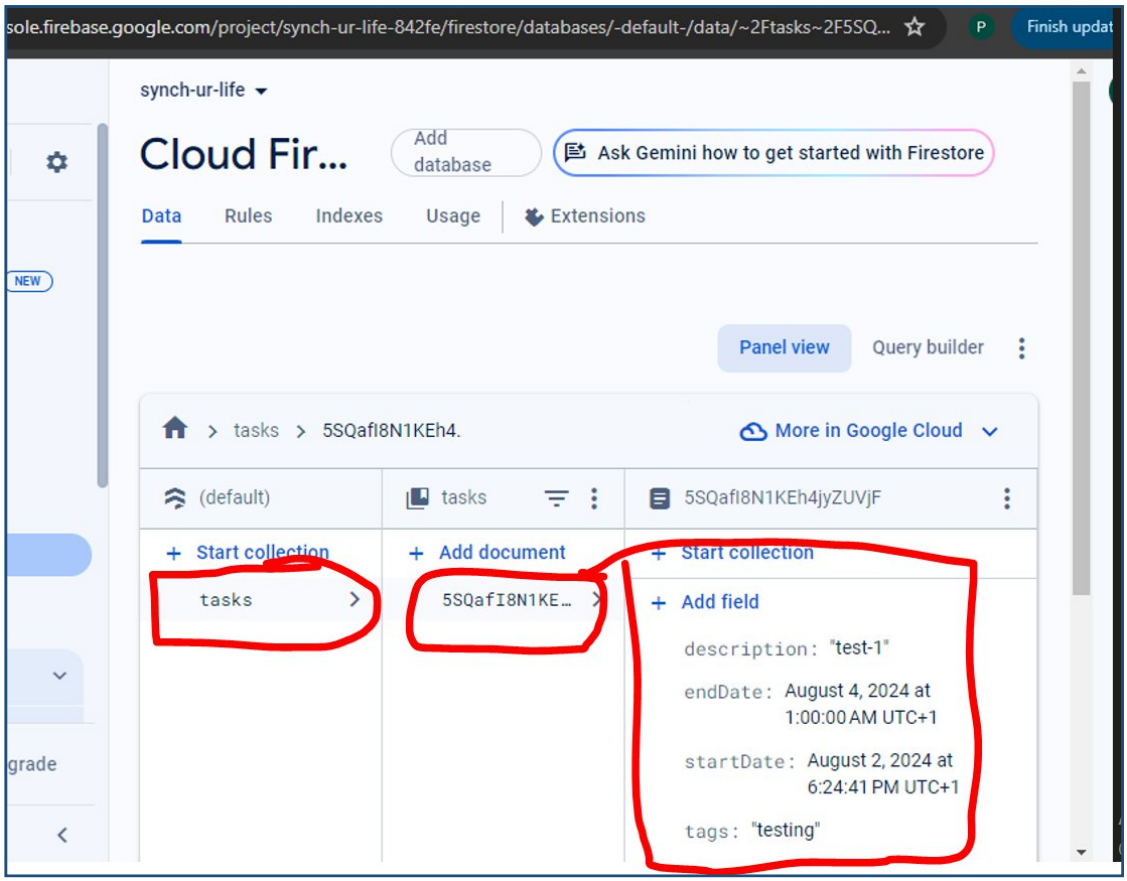
- An image displaying the item listed after selecting 'Confirm'.





- A task was added, and a corresponding record was also created in Firebase's Firestore Database, as illustrated in the images below:





- It seems that expected result was met. I will consider this test as successful.

2.4.2. Test Case: Chat with AI Assistance

This test case verifies that the AI Assistant correctly responds to a user's query about their tasks for the next day.

Precondition:

- Ensure there are tasks scheduled for the next day in the Firebase database.
- Ensure the AI Assistant is integrated and the webhook server is operational.

Steps:

1. Navigate to the part of the application where the AI Assistant can be accessed
2. Type the query "What are my tasks for tomorrow?" into the input field and submit.
3. Ensure that the webhook server sends the query to Dialogflow for natural language processing.
4. Ensure that Dialogflow processes the query and returns a response with the tasks for the next day.
5. Ensure that the application shows the AI Assistant's response, including the list of tasks for the next day.

Expected Result:

1. The AI Assistant accurately interprets the user's query and lists the tasks scheduled for the next day in its response.
2. The application interface displays these tasks, including accurate details such as description, time, and other relevant information.

2.5. Evaluation

Functionality

The primary goal of the project was to create an all-in-one task management and scheduling application. Each of the core functionalities—task creation, editing, deletion, and viewing—was implemented and tested. The integration of the AI Assistant using Dialogflow is only partially working; there were issues with Dialogflow parsing the queries correctly. Although some intents were successfully handled, further refinement and debugging are necessary to achieve full functionality.

Usability

Usability testing was conducted by myself, and while most functionalities are working, broader user testing is required to gain comprehensive feedback. The interface is designed to be clean and intuitive, with clear logical flow.

Testing Strategies

The testing strategy employed was primarily functional testing. The main funtionalities were tested to ensure it met the specified requirements and worked as expected. While this ensured that the core functionalities were operational, additional testing strategies, such as unit testing, integration testing, and user acceptance testing, would be beneficial for more comprehensive validation.

3.0 Conclusion

The Synch-Ur-Life project aimed to create a comprehensive task management and scheduling application that integrates various aspects of daily life into one seamless platform. The project has demonstrated several advantages, strengths, as well as limitations and areas for improvement.

Advantages

- The application successfully combines calendar functions, task management, reminders, and basic data analytics into a single, user-friendly interface. This integrated approach simplifies life management for users.
- By using ReactJS for the web frontend and React Native for the mobile application, the project ensures a consistent and seamless user experience across different devices, including both Android and iOS.
- The integration of the AI Assistant, though partially functional, showcases the potential for advanced user interaction through natural language processing. This feature aims to make task management more intuitive and hands-free.

Disadvantage

- The AI Assistant integration using Dialogflow is only partially working, with issues in parsing user queries accurately. This limitation affects the overall functionality and user experience of the AI feature.
- Usability testing was primarily conducted by the developer (myself), limiting the scope of feedback and potential usability issues that could be identified by a broader user base. More extensive user testing is needed to refine the interface and features.
- The project relied mainly on functional testing. While this ensured core functionalities worked as intended, additional testing strategies such as unit testing, integration testing, and user acceptance testing would provide a more robust validation of the application.

In summary, the Synch-Ur-Life project has successfully laid the groundwork for an integrated task management and scheduling application. The strengths of the project lie in its user-friendly design, cross-platform accessibility, and innovative features like AI assistance. However, there are notable limitations, including partial AI integration, limited usability testing, and basic testing strategies that need to be addressed in future iterations. By focusing on these areas for improvement, the project has the potential to evolve into a highly effective and reliable life management tool, meeting the diverse needs of its users

REFERENCES

Any.do, n.d. 'Any.do', Any.do, accessed 17 December 2023. Available at: https://www.any.do/en.

Bahrynovska, T., 2022. 'React Application Development for Web: Benefits, Challenges, Examples', Forbytes, 28 November. Available at: https://forbytes.com/blog/react-application-development-for-web/ [Accessed 17 December 2023].

Bjarnason, E., Lang, F. & Mjöberg, A. (2023) 'An empirically based model of software prototyping: a mapping study and a multi-case study', *Empirical Software Engineering*, 28(115). Available at: https://link.springer.com/article/10.1007/s10664-023-10331-w [Accessed: 8 May 2024].

BusyCal, nd. 'BusyCal for macOS', BusyCal, 17 December. Available at: https://www.busymac.com/busycal/ [Accessed 17 December 2023].

Cuthbert, E. (2024). Why Choosing JavaScript For Front-End Development? 10 Best Tools and Techniques. Available at: https://medium.com/quick-code/why-choosing-javascript-for-front-end-development-10-best-tools-and-techniques [Accessed 8 May 2024].

Dennis, A., Wixom, B.H. & Tegarden, D. (2010). Systems Analysis and Design with UML. 3rd ed. John Wiley & Sons, p.482.

Feyoh, M. (2024) '15 Best Shared Family Calendar Apps & Organizers [2024 Update]', Develop Good Habits. Available at: https://www.developgoodhabits.com/family-calendar-apps/ [Accessed 25 July 2024].

Firebase. (2024) Understand Firebase Projects. Available at: https://firebase.google.com/docs/projects/learn-more [Accessed 8 May 2024].

geeksforgeeks.org, 2023. 'Integration Testing – Software Engineering', geeksforgeeks.org, 16 November. Available at: https://www.geeksforgeeks.org/software-engineering-integration-testing/ [Accessed 18 December 2023].

geeksforgeeks.org, 2023. 'Unit Testing – Software Testing', geeksforgeeks.org, 16 November. Available at: https://www.geeksforgeeks.org/unit-testing-software-testing/ [Accessed 18 December 2023].

Groenewegen, C. (2024) 'I Tested The Best Shared Calendar Apps For Families. Here Are My Favorites', Motion. Available at: https://www.usemotion.com/blog/best-shared-calendar-app-for-families [Accessed 25 July 2024].

IEE |The Importance of Software Testing. No date. [Online]. Available at: https://www.computer.org/resources/importance-of-software-testing [Accessed: 28-Feb-2024].

Moran, K. (2019) 'Usability Testing 101', Nielsen Norman Group. Available at: https://www.nngroup.com/articles/usability-testing-101/ (Accessed: 27-Feb-2024).

Mowat, A. (2014). 12 Features Every Task Management Tool Should Have. Prialto. Available at: https://www.prialto.com/blog/12-features-team-task-management-tool [Accessed 9 May 2024].

React Dev Team. (2024). React. Available at: https://react.dev [Accessed 8 May 2024].

teamhood.com, n.d. '14 Scrum Advantages and Disadvantages in 2023', teamhood.com, accessed 18 December 2023. Available at: https://teamhood.com/agile/scrum-advantages-disadvantages/#:~:text=,difficult%20to%20identify%20clear%20requirements.

APPENDIX A – PROJECT PROPOSAL

Objectives

The application I will be developing is called Synch-Ur-Life, an application that helps user plan, schedule and manage their life. A calendar, task manager, scheduler and planner in one app. The application will be a cross platform application, which allows users to easily manage their calendar, tasks, reminders, and daily activities all in one place. The app's customizable options, like tailored dashboard designs and colour-coded categories, adapt to personal needs and styles, making it a flexible life management tool.

One of the main feature of the application is the ability to add and create costume category and tagging on the items — task, event, to-do list and meetings. The application feature saved filter, which enable users to quickly access and view items based on predefined criteria, such as category, tag, date, or priority level.

Background

I started this project because I noticed that many calendar and task apps don't have all the features needed for good life management. People often use different apps for tasks, events, and reminders, which can be scattered and overwhelming. My goal is to make an all-in-one app that combines these and adds special features like custom tags, categories, and saved filters for a more personalized and efficient way to organize.

The project aims to improve personal productivity and organization with an easy-to-use interface that suits different needs. It will use the latest web and mobile app technologies, mainly ReactJS and React Native, for a smooth experience on all devices, (Bahrynovska, T., 2022. [1]). The focus will be on making the design intuitive and centred around the user, making the app easy to use and flexible for different lifestyles.

State of the Art

Current apps like BusyCal3 and Any.do have some similar features. BusyCal3 combines tasks and calendars, showing due dates and carrying tasks until done, and includes travel time, (BusyCal, nd.[2]). Any.do lets users customize its look and feel, gives smart task suggestions, integrates a calendar, and merges social and business events with daily tasks, along with location-based reminders, (Any.do, n.d. [3]).

My project plans to go beyond these apps by providing a more complete life management solution. It will mix different features and add new ones such as saved filters for faster access, making it more user-friendly. The app will stand out by letting users highly personalize their experience. Using the latest technology and focusing on ease of use and intuitive design, this project aims to create a new standard in life management apps, improving on what's currently available.

Technical Approach

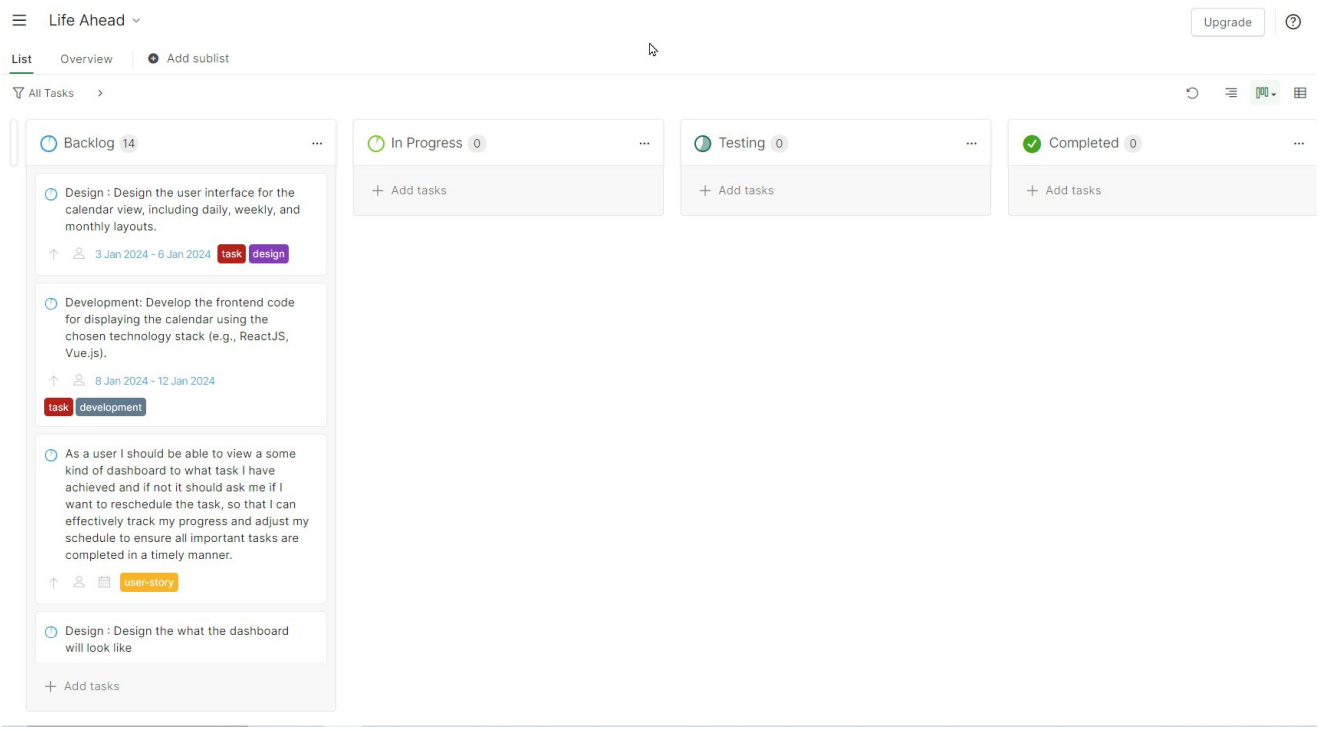
I will be using the Scrum method for developing the application. Using Scrum SDLC for the app project gives advantages, particularly due to its flexibility and adaptability in the ever-changing world of software development (teamhood.com, n.d. [4]).

The iterative process is divided in short development cycles called sprints, which allows for regular assessment and adaptation to changing requirements and priorities. This incremental approach not only facilitates early and frequent delivery of functional app components but also ensures continuous improvement based on regular feedback. Such approach is well suited for this project as I will be working under short time deliverables and will have to present some working functionalities and features.

Scrum also improves overall project management and product quality through its structured yet flexible practices.

I will be identifying requirements based on the User Story. The User Stories will be piled on Product Backlog. The Product Backlog will contain User Stories, Task and Bugs. Each User Story will consist of a user, a goal and a reason. The backlog will have tasks, which often relates to a user story, and is more technical and specific in details.

Each User Stories mark as a milestone.



Technical Details

STACK, tools and Technologies

My main programming language of choice is Javascript. Due to time constrain, Javascripts allows for a quicker coding and prototyping. Javascript is versatile language used for both frontend and backend development. It lets you have a consistent development process for your entire application.

- Frontend
    - For Frontend, I will be using Javascript's ReactJS for the web application. ReactJS is widely used and has strong community support, which offers many resources and libraries that help speed up development and create responsive, user-friendly apps. Its scalability and modularity also make it easier to add new features or services later.
    - I will use React Native to develop the mobile application version of the project.
- Backend
    - Node.js with Express.js
- Git for version control.
- Visual Studio Code as an IDE.
- Quire – for tracking progress and managing task.

Special Resources Required

- Firebase – for authentication, real-time database, and hosting.
- Heroku – a cloud hosting solution for backend deployment.

Project Plan

Our project will use the Scrum method, organizing work into 4 to 5 week periods called sprints. In each sprint, we'll focus on one or two specific user needs, giving us time to develop them carefully. The first sprint starts on January 2nd and will last 4 to 5 weeks. The main goal for this first sprint is to create a working prototype that includes at least two user needs. This method helps us focus on important features first, building a strong base for the rest of the project and steadily working towards an application that really meets user needs.

Product Backlog

- Main GUI / UI
    - Design the Main GUI, what the GUI will look like when the user open the app

- Database Design : Plan and design the database whether to implement a relation or non relational

- USER STORIES
    - As a user, I should be able to use and view a calendar, so that I can easily track and manage my personal and professional events, ensuring that I am organized and can effectively plan my daily activities and appointments.
        - Design : Design the user interface for the calendar view, including daily, weekly, and monthly layouts.
        - Development: Develop the frontend code for displaying the calendar using the chosen technology stack (e.g., ReactJS, Vue.js).
        - CRUD : Develop backend APIs to handle operations like retrieving, adding, updating, and deleting calendar events.
        - Development : Develop the UI components and logic to allow users to add, edit, and delete events in the calendar.
    - As a user, I should be able to add tagging and category on task, meetings, and events to my calendar, so that I can easily organize, search, and prioritize my activities based on their type and relevance.
        - Design : Create user interface designs for adding tags and categories to tasks, meetings, and events within the calendar. This includes designing the layout for the tagging interface, category selection, and how they will be displayed on the calendar.
        - Develop the frontend functionality that allows users to add tags and categories to their calendar entries. This involves coding the interface elements based on the UI/UX designs.
        - Update the database schema to include structures for storing tags and categories, and their association with calendar entries.

- Develop the functionality that allows users to filter their calendar view based on selected tags or categories.

Testing

Unit Testing

CRUD – My main focus in unit testing will be on checking the basic functions of managing calendar events: Create, Read, Update, and Delete (CRUD). I'll thoroughly test how the system adds, shows, changes, and removes events. This way, I can make sure the key features of the app, which users rely on to interact with their calendars, work well and without errors. (geeksforgeeks.org, 2023)

Integration Testing

Ensuring Compatibility with New Features: In our integration testing phase, special attention will be paid each time a new function is added to the application. I will systematically reuse and re-purpose the existing unit tests, particularly those developed for CRUD operations, to verify that these core functionalities continue to work seamlessly with any new additions. This process is critical to confirm that the integration of new features does not disrupt or negatively impact the existing, well-established functionalities of the app. (geeksforgeeks.org, 2023 [6])

REFERENCES of Appendix A

Any.do, n.d. [3] 'Any.do', Any.do, accessed 17 December 2023. Available at: https://www.any.do/en.

Bahrynovska, T., 2022. [1] 'React Application Development for Web: Benefits, Challenges, Examples', Forbytes, 28 November. Available at: https://forbytes.com/blog/react-application-development-for-web/ [Accessed 17 December 2023].

BusyCal, nd.[2] 'BusyCal for macOS', BusyCal, 17 December. Available at: https://www.busymac.com/busycal/ [Accessed 17 December 2023].

geeksforgeeks.org, 2023 [5]. 'Unit Testing – Software Testing', geeksforgeeks.org, 16 November. Available at: https://www.geeksforgeeks.org/unit-testing-software-testing/ [Accessed 18 December 2023].

geeksforgeeks.org, 2023 [6]. 'Integration Testing – Software Engineering', geeksforgeeks.org, 16 November. Available at: https://www.geeksforgeeks.org/software-engineering-integration-testing/ [Accessed 18 December 2023].

teamhood.com, n.d. [4] '14 Scrum Advantages and Disadvantages in 2023', teamhood.com, accessed 18 December 2023. Available at: https://teamhood.com/agile/scrum-advantages-

APPENDIX B – Reflective Journals

REFLECTIVE JOURNAL

2023/2024

Student Name	Penuel Maypa
Student Number	X16382003
Course	BSc (Honours) in Computing
Supervisor	William Clifford
Month:	November 2023 – April 2024

November 2023

This month, my project took a new direction after a tough meeting with my supervisor. My original idea to automate financial trading was evaluated and found to be less suitable for academic purposes. I felt disappointed , having to let go of my initial concept despite the time and hope I had invested in it.

In reaction to this setback, I dived into research and reading to come up with a new idea that would meet academic rubric and requirement and be practically useful. After reflecting on my daily needs, I came up with a new app concept that combines a calendar, planner, diary, and schedule assistant into one tool aimed at improving life management. This shift from a financial tool to a personal organization app was not only enlightening but also freeing, allowing me to explore how technology can enhance everyday life. I am excited about this new application but also a bit worried about whether I can successfully develop it and meet the upcoming deliverables in the next few weeks.

Learning to align my project with academic standards was an important lesson. Discussing this with my peers, I found many of them were also adjusting their projects.

Looking ahead, I plan to organize the development process into smaller, weekly tasks to keep the project moving forward. I'll also consult with my supervisor to fine-tune the app's design and functions to better meet user needs. This guidance will be essential for managing the complexities of developing the app.

I am also outlining the app's key features, such as customizable reminders, integration with other tools, and easy-to-use interfaces. These features are designed to meet a broad range of needs. The opportunity to create a tool that genuinely helps people organize their lives motivates me to face the upcoming challenges.

Student Signature	Penuel Maypa

DECEMBER 2023

December marks the end of the first semester and brings a rush of assignments and deadlines for all my courses. This has forced me to shift most of my effort away from my main project, slowing down the project's progress and increasing my stress. Balancing all these tasks is challenging, as each course demands a lot of attention. Additionally, balancing a full-time job and part-time college is stressful and often leads to sleep loss as I strive to fulfill all my responsibilities.

During this busy time, I've had to improve my time management and set strict priorities to keep up with everything. I've allocated specific, albeit brief, time slots for my project, helping me maintain some progress. Although it's tough to manage so many responsibilities, having a structured plan is crucial for staying on track. But the lack of sleep makes it harder for me to focus and stay energized, further complicating my busy schedule.

Looking ahead, I plan to incorporate what I've learned in other courses into my main project to make the most of my time. This not only helps me handle my workload but also enhances my project by making it more comprehensive and relevant to my studies. As the semester ends, I'm struggling to stay motivated with my project due to the lack of sleep and the overwhelming stress. The thought of completing these tasks and focusing more on my project keeps me going despite the stress.

However, having a few days off from work during Christmas provided a much-needed break and was helpful in regaining some focus.

Student Signature	Penuel Maypa

JANUARY 2024

Now that I've submitted all my assignments for other modules, I can finally focus on my main project. This shift has been a big relief, allowing me to work on the technical parts of development I've been eager to start. I've begun coding, writing the basic lines of code that will make my app function.

I have begun developing the application, starting with setting up the main folders to structure the project effectively. I created the calendar interface and enhanced it with CSS to improve both aesthetics and functionality. However, developing the GUI presented ongoing challenges that required constant refinement and problem-solving to achieve a working version.

As I progressed, I encountered significant bugs in one of the main codebase, which I initially managed to resolve, allowing the application to function. However, persistent issues led me to start over. Despite the redevelopment efforts, the calendar still didn't display correctly, prompting me to revert to stable versions and engage in extensive debugging. The debugging phase was particularly demanding, with the application able to build and run but failing to display the calendar correctly. This highlighted difficulties in effectively integrating JavaScript and CSS. I eventually narrowed down the issue to the CSS module not being called, which prevented styles from applying when the page loaded, pinpointing the root of the problem.

Throughout this journey, each bug encountered underscored not only the progress made but also the need for continual reassessment and adaptation. This experience has been about more than just overcoming technical challenges; it has been a testament to the perseverance and evolving experience that have driven the development of my application.

Student Signature	Penuel Maypa

FEBRUARY 2024

As February starts the second semester, my schedule is getting tighter due to the demands of attending classes for other modules. This change means I have a lot less time for my main project, which needs careful balancing. Despite these scheduling challenges, I made some strides in the project this month. I focused on enhancing the functionality of the application by integrating CRUD operations with the Firebase database.

I started the backend infrastructure using Google Firebase, specifically aiming to implement Firebase CRUD functionalities. After successfully linking the app to the Firestore database, I integrated CRUD operations that enable users to create, read, update, and delete tasks directly within the application. This setup allows for real-time interaction with the database, enhancing user engagement by allowing the user to manage their tasks dynamically and efficiently.

This development phase was not without its hurdles, as I encountered some bugs and errors. After resolving a series of errors related to Firebase and React integration, I managed to stabilize the core functionalities. Even with a hectic semester, I managed to fix some bugs and add key database features, which really helped me make solid technical progress and learn a lot along the way. Each small victory is an advance in my project amid a demanding academic schedule.

The tight schedule is key not only for keeping up with progress but also for making sure the quality of my work doesn't drop. The challenge is tough but it's teaching me a lot about managing my time and prioritizing effectively in real-life scenarios. Every day is a chance to get better at juggling these responsibilities and staying on track with both my college and project goals.

Student Signature	Penuel Maypa

MARCH 2024

February has been a tough month with a heavy workload from ongoing coursework in different subjects. With tons of assignments piling up from different classes. Because of this, I've had little time to work on my main project. Jumping from one deadline to the next has not only slowed down my project but also really ramped up my stress.

The pressure of maintaining a standard in all areas of my studies is mounting. Each day has become a challenge to balance, where I strive to meet the expectations of my courses while the progress of my main project remains stagnant. This imbalance is beginning to wear on me, not just intellectually but emotionally as well. The frustration of not being able to work on my project has been disheartening.

Despite these challenges, this month has also been a profound lesson in resilience. I am learning to navigate the pressures of a demanding academic schedule, finding small ways to manage stress and keep my goals in sight. While the progress on my main project has paused, my commitment to it remains consistent. I am determined to find a way to reintegrate it into my daily routine as soon as the rush of assignments subsides. In the meantime, I hope this tough time will help me improve my time management and prioritization skills, which will be good for my academic and career future.

Student Signature	Penuel Maypa

APRIL 2024

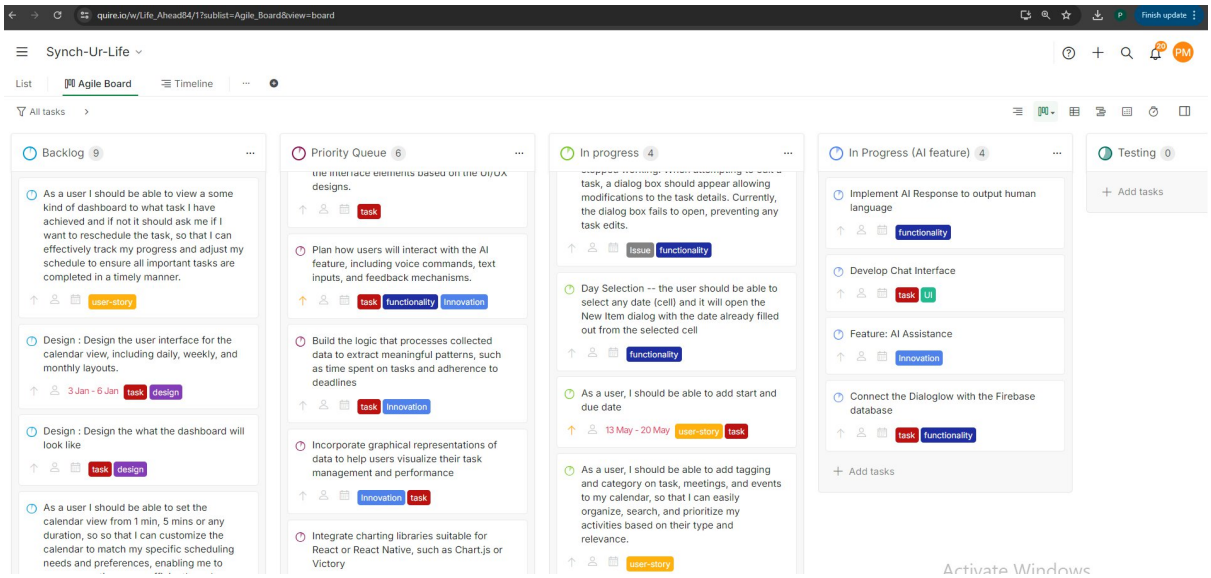
This month, April, the most challenging month of the semester, filled with report submissions and TABA deadlines across all my courses, leading to significant stress. The constant demands resulted in many sleepless nights and forced me to skip my usual exercise routine, increasing my physical and mental fatigue. Due to the intense workload, I also had to cancel all my weekly meetings with my project supervisor for the entire month.

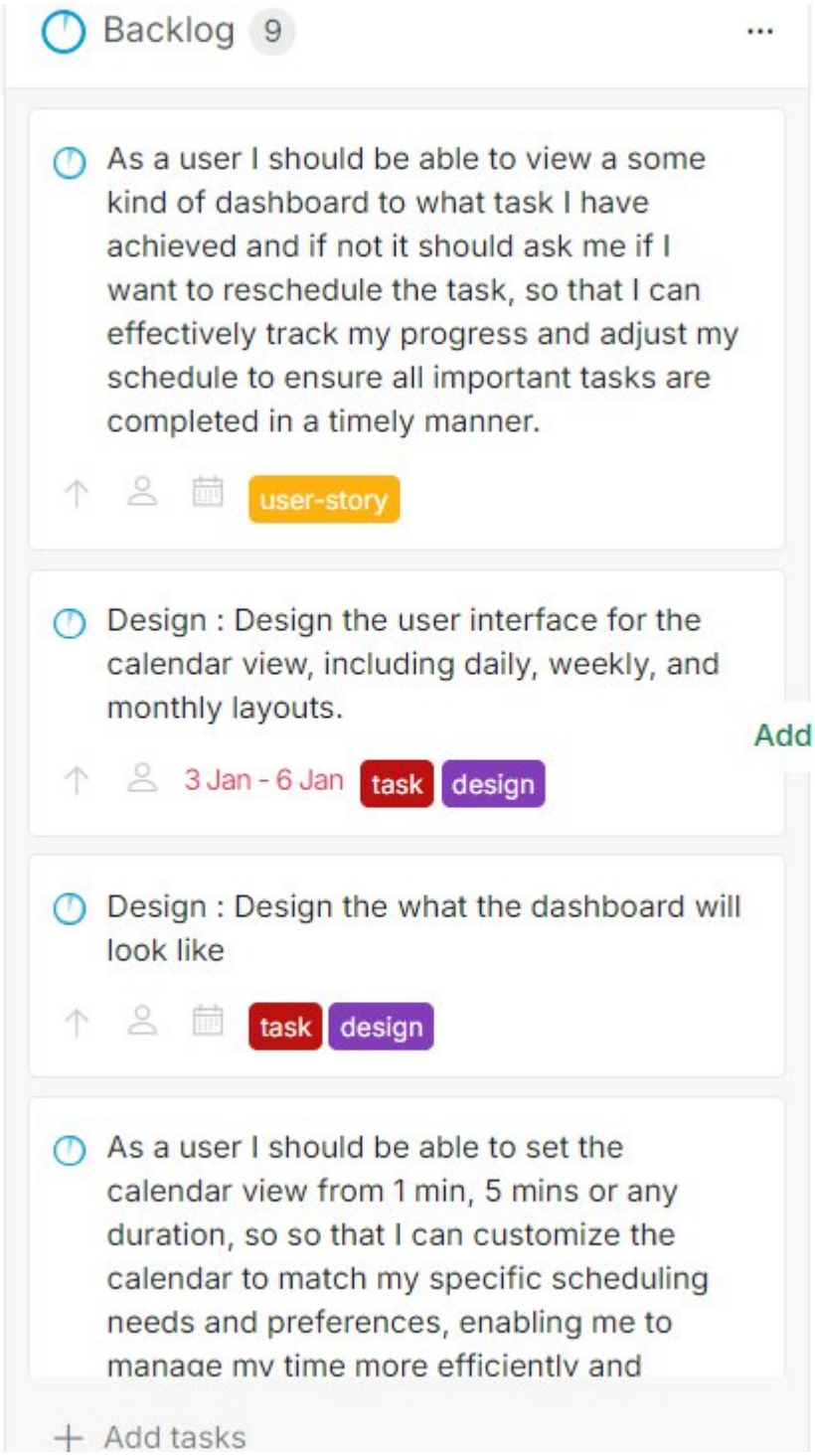
The combination of stress, pressure, and insufficient sleep significantly impacted my health, leading to me becoming ill that further complicated my situation. Even while I was on sick leave from work, I was overwhelmed with assignments for other modules, leaving me no chance to properly recover. This relentless workload prevented any progress on my main project, which was both frustrating and disheartening.

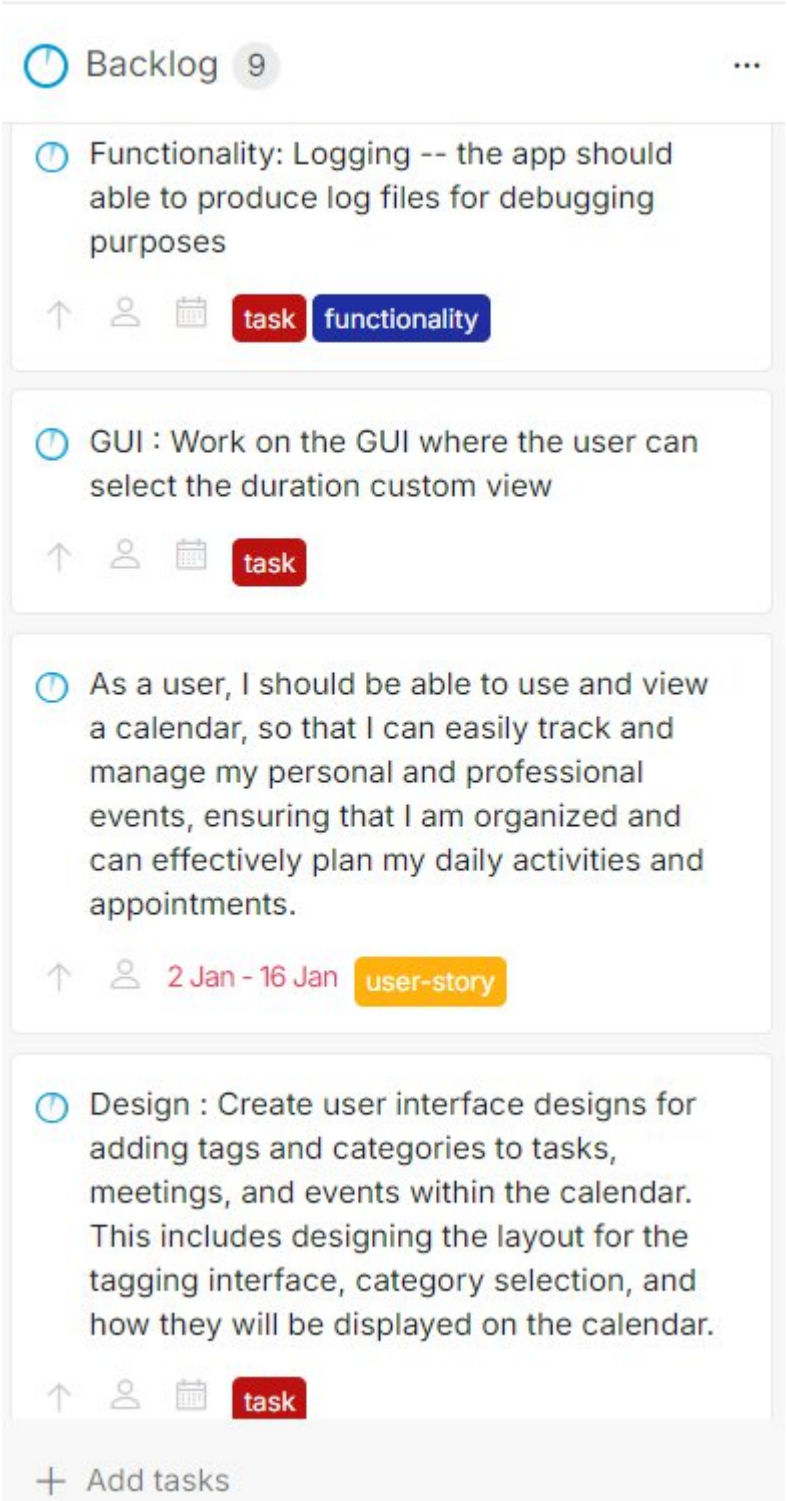
During this time of non-stop deadlines, I felt completely overwhelmed, which made me question my decisions and whether my academic goals were achievable. The constant pressure this month made me seriously consider my capabilities and whether my current academic and career path is sustainable in the long run. It's been a very difficult period, but it's also an important time to rethink how I deal with stress and organize my tasks moving forward.

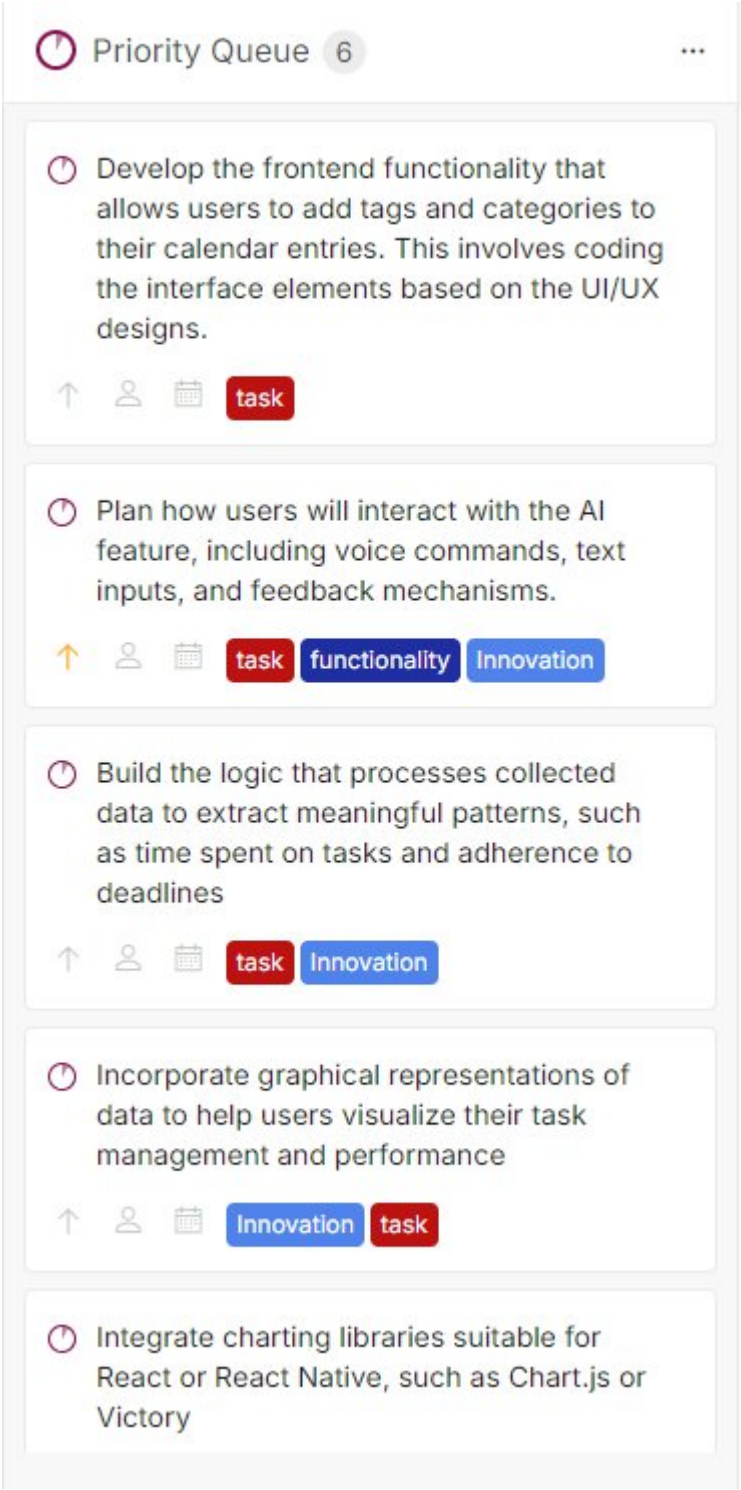
Student Signature	Penuel Maypa

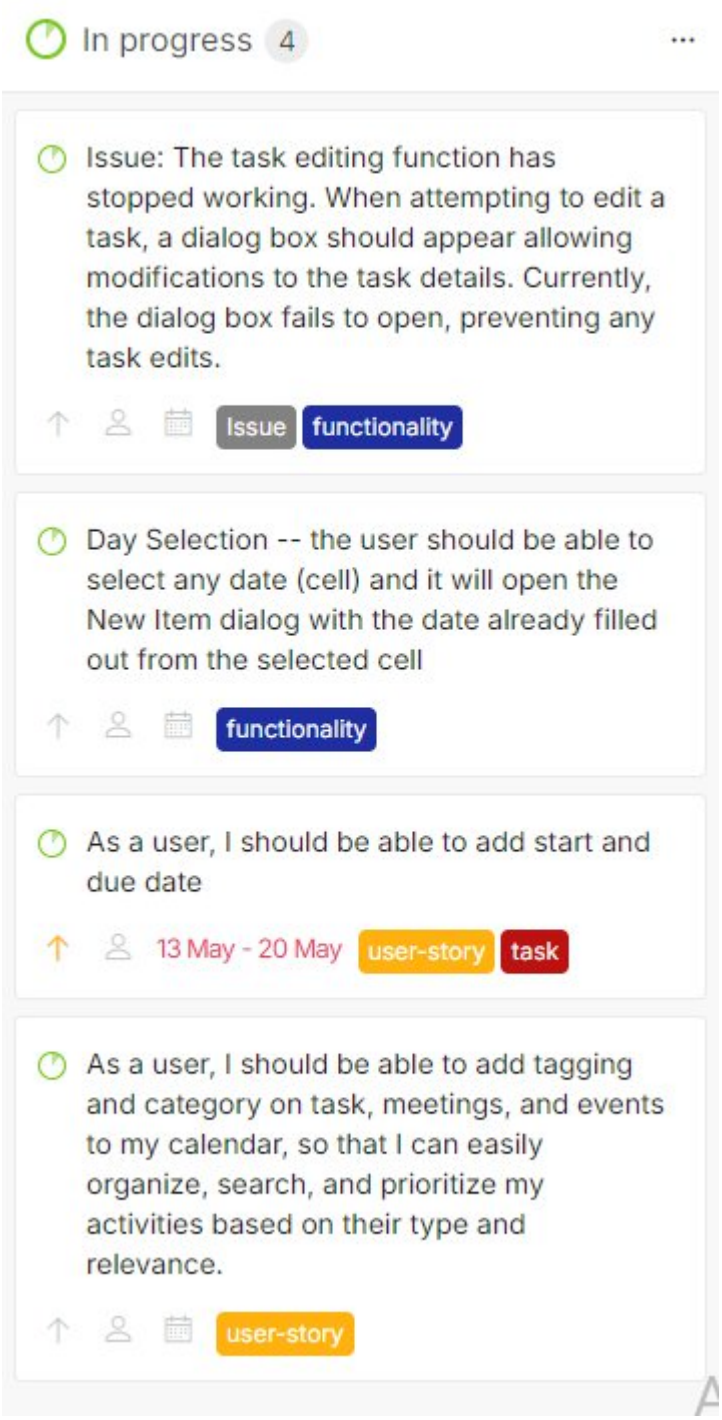
APPENDIX C – Agile Board

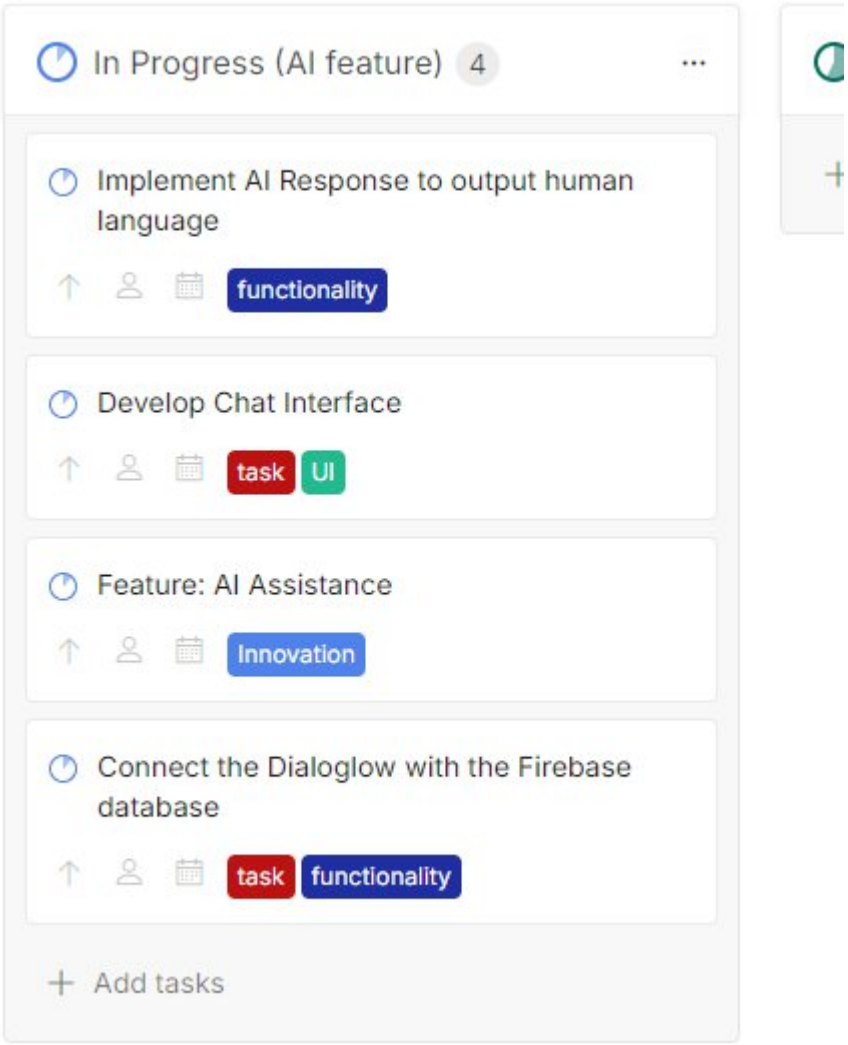



Backlog 9
As a user I should be able to view a some kind of dashboard to what task I have achieved and if not it should ask me if I want to reschedule the task, so that I can effectively track my progress and adjust my schedule to ensure all important tasks are completed in a timely manner.
user-story
Design : Design the user interface for the calendar view, including daily, weekly, and monthly layouts.
3 Jan - 6 Jan
task
design
Add
Design : Design the what the dashboard will look like
task
design
As a user I should be able to set the calendar view from 1 min, 5 mins or any duration, so so that I can customize the calendar to match my specific scheduling needs and preferences, enabling me to manage my time more efficiently and
Add tasks


Backlog 9
Functionality: Logging -- the app should able to produce log files for debugging purposes
task
functionality
GUI : Work on the GUI where the user can select the duration custom view
task
As a user, I should be able to use and view a calendar, so that I can easily track and manage my personal and professional events, ensuring that I am organized and can effectively plan my daily activities and appointments.
2 Jan - 16 Jan
user-story
Design : Create user interface designs for adding tags and categories to tasks, meetings, and events within the calendar. This includes designing the layout for the tagging interface, category selection, and how they will be displayed on the calendar.
task
Add tasks


Priority Queue 6
Develop the frontend functionality that allows users to add tags and categories to their calendar entries. This involves coding the interface elements based on the UI/UX designs.
task
Plan how users will interact with the AI feature, including voice commands, text inputs, and feedback mechanisms.
task functionality Innovation
Build the logic that processes collected data to extract meaningful patterns, such as time spent on tasks and adherence to deadlines
task Innovation
Incorporate graphical representations of data to help users visualize their task management and performance
Innovation task
Integrate charting libraries suitable for React or React Native, such as Chart.js or Victory


In progress 4
Issue: The task editing function has stopped working. When attempting to edit a task, a dialog box should appear allowing modifications to the task details. Currently, the dialog box fails to open, preventing any task edits.
Issue
functionality
Day Selection -- the user should be able to select any date (cell) and it will open the New Item dialog with the date already filled out from the selected cell
functionality
As a user, I should be able to add start and due date
13 May - 20 May
user-story
task
As a user, I should be able to add tagging and category on task, meetings, and events to my calendar, so that I can easily organize, search, and prioritize my activities based on their type and relevance.
user-story


In Progress (AI feature) 4
Implement AI Response to output human language
functionality
Develop Chat Interface
task
UI
Feature: AI Assistance
Innovation
Connect the Dialoglow with the Firebase database
task
functionality
Add tasks

END OF THE PAGE