3 Design

3.4 Plan

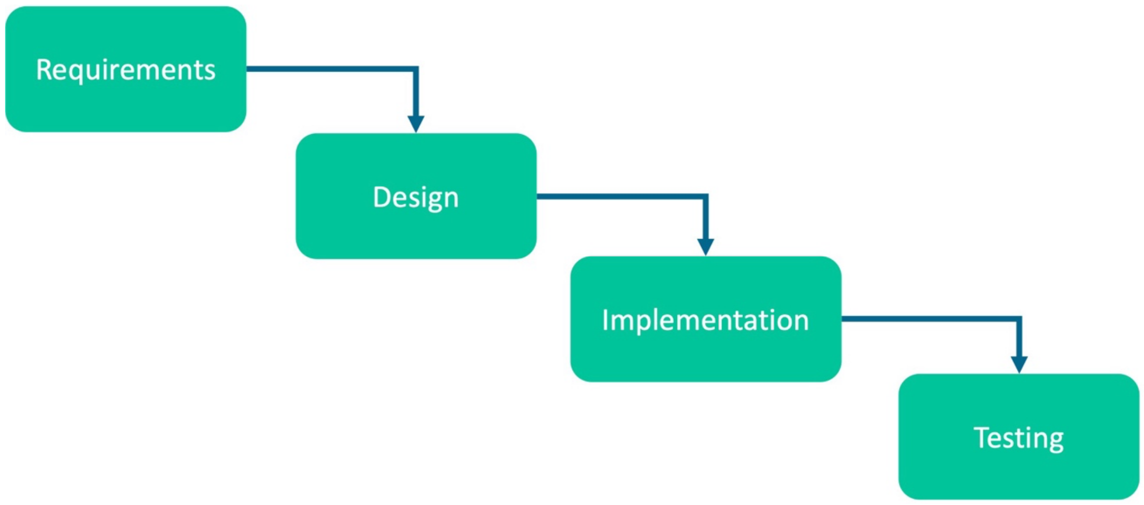

Figure 17: Waterfall SDLC adaptation

Following a comprehensive investigation into a range of software development methodologies, the waterfall development methodology [10] has been adapted and will be used during the development process. This approach segments the project into four principal stages – Requirements, Design, Implementation, and Testing. The rationale behind selecting this particular development methodology lies in the restricted timeframe at hand. Alternative methods, such as Agile, prove to be less suitable due to the excessively brief nature of each iterative phase, potentially leading to an overly convoluted, tight schedule that may adversely affect the project's quality. The waterfall method aligns well with this project since the survey findings were used to determine the requirements as well as to shape the foundational design of the application.

3.5 Application Structure

3.5.1 Application Architecture

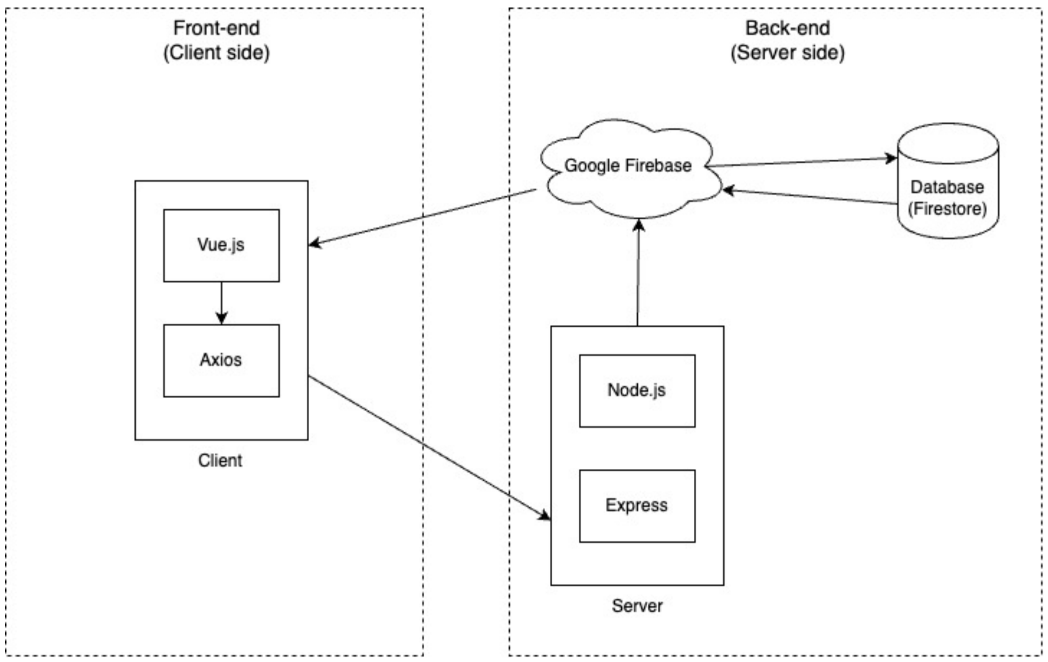

Figure 18: Application Architecture Diagram

As the project will be utilising a full-stack development approach, it will be split into two sections: the front-end and the back-end.

Front-end

The front-end will be built using the Ionic framework along with the Vue.js framework for rendering the user interface, whilst Axios will be used to perform any HTTP requests to the back-end. One advantage of using Vue.js is that it is reactive. This means that Vue.js automatically tracks JavaScript changes and can very quickly update the page to reflect these changes [16]. For example, UI components can be subscribed to a database and reflect any changes made in real-time without any need for refreshing, allowing for an interactive, fluid user experience. This is important as it relates to Nielsen's first usability heuristic: Visibility of system status, which states that any feedback from the system should be provided to the user as quickly as possible [8].

In most typical full-stack applications, the back-end API will provide GET methods which allow for data retrieval from a database. However, this is not necessary when using Google Firebase and Vue.js as it will be able to retrieve any data from the databases directly, circumventing the back-end. One benefit of this approach is speed, as the front-end is able to read the database without having to request the data through the back-end.

Back-end

The back-end will be developed using Node.js and Express in order to handle the API requests from the front-end. Node.js is a JavaScript runtime environment that allows for scalable server-side applications [17] and Express provides a flexible API for building HTTP servers, handling requests and responses, and routing URLs to specific handlers. It is possible to build an application using Vue.js for the front-end and Firebase for the back-end however, this approach does not scale well as it is platform dependant and will only allow for platforms supported by Google Firebase. The back-end developed, will host the API methods used to interact with the database, as well as any assets that may require it.

Database

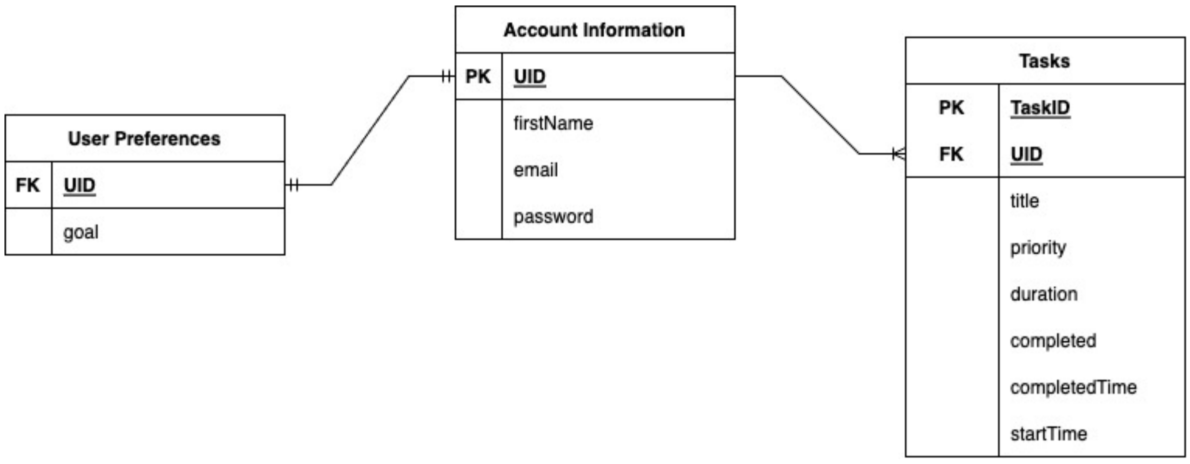

Figure 19: Entity Relationship (ER) Database Diagram

The database will use Firestore, a real-time database solution offered as part of the Firebase suite. It will consist of three tables: Account Information, Tasks and User Preferences. Figure 19 represents an abstraction of an ER diagram for the database, as there is no relationship feature, however relationships and retrieval of data can be achieved using unique IDs.

<u>Account Information</u>

Used to store the first name of the user, along with their email and password, however this is an abstraction as the Firebase authentication tools will be used to handle authentication which will store the hashed passwords separately and securely. Each user is assigned a UID upon creation which is used as a foreign key by the back-end for identifying user data throughout other tables.

<u>Tasks</u>

Each user will be assigned their own Tasks table, allowing for data separation. Each task entry will contain a TaskID, used as the primary key along with the user UID as a foreign key for identification as well as the title, priority, duration, completed Boolean, completedTime and startTime.

<u>User Preferences</u>

The user preferences table will be used to store any user preferences saved by the user such as their goal. In the future this can be expanded as more customisation options and features are added to the application, for example dark mode. The reason behind storing this information in a database rather than locally is because it allows for a seamless user experience across platforms and reduces the risk of a user losing any progress or data should something happen to their device.

3.5.2 User Flow Diagram

With the features, non-functional and functional requirements determined, a user flow diagram of the application was produced (Figure 20). In the development of a prototype application, a user flow diagram serves as an invaluable tool for ensuring a seamless and intuitive user experience. By mapping out the various pathways users may take to complete tasks within the application, a user flow diagram will identify potential bottlenecks, redundancies, or points of confusion.

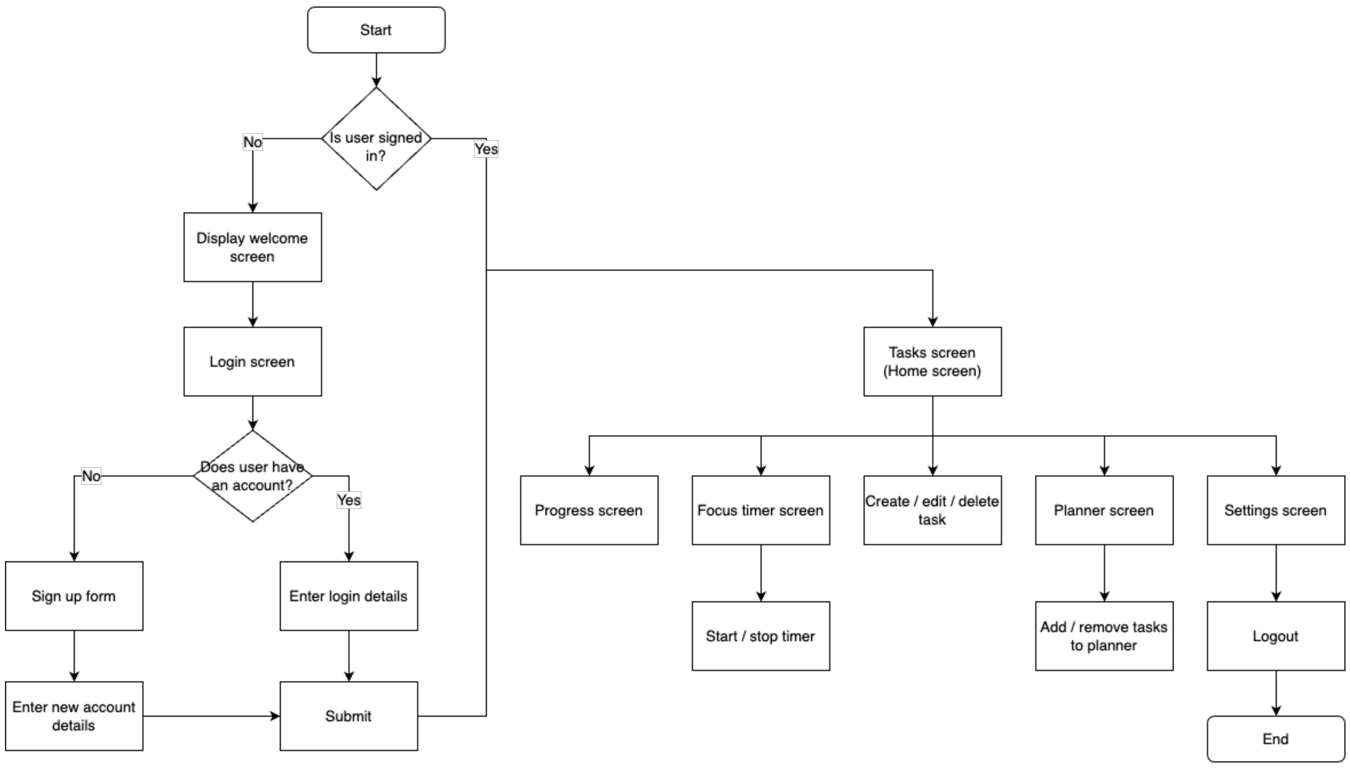

Figure 20: User Flow Diagram

3.5.3 UML Use Case Diagram

In addition to a user flow diagram, a UML use case diagram (Figure 21) has also been produced in order to aid development of the application.

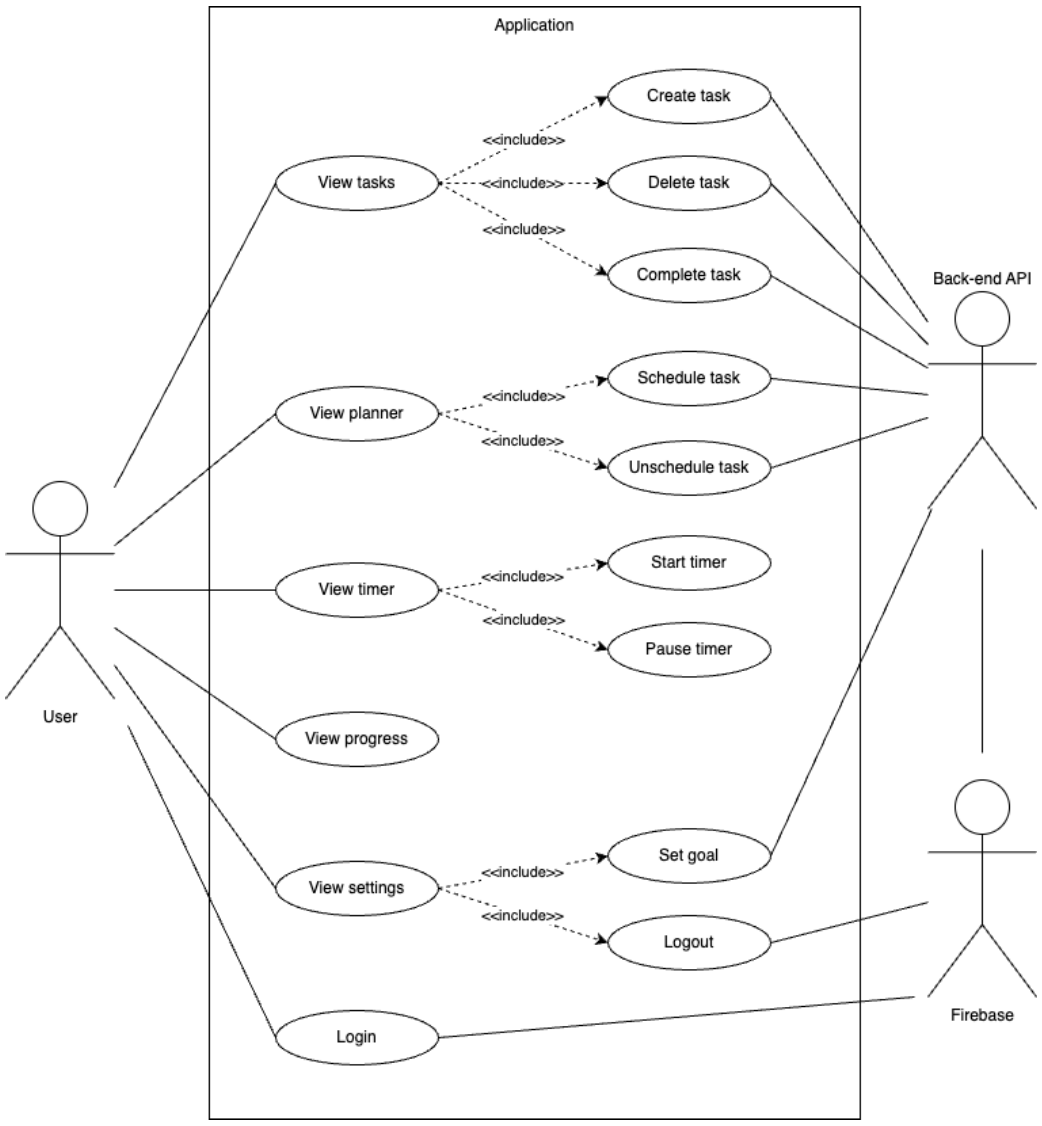

Figure 21: Use Case Diagram

3.6 Colour Palette

When creating the UI prototype, it was important to consider what colours to use for the final prototype. Ideally, this colour scheme needs to be consistent throughout, but also appropriate.

Adobe XD provides an integrated colour scheme tailored to each specific artboard. This feature not only promotes uniformity in design, adhering to one of Shneiderman's principles [18], but also facilitates the creation of visually appealing layouts. Moreover, the scheme enables swift alterations of individual colours for the purpose of experimenting with various themes. The ultimate colour scheme can be seen below in Figure 22.

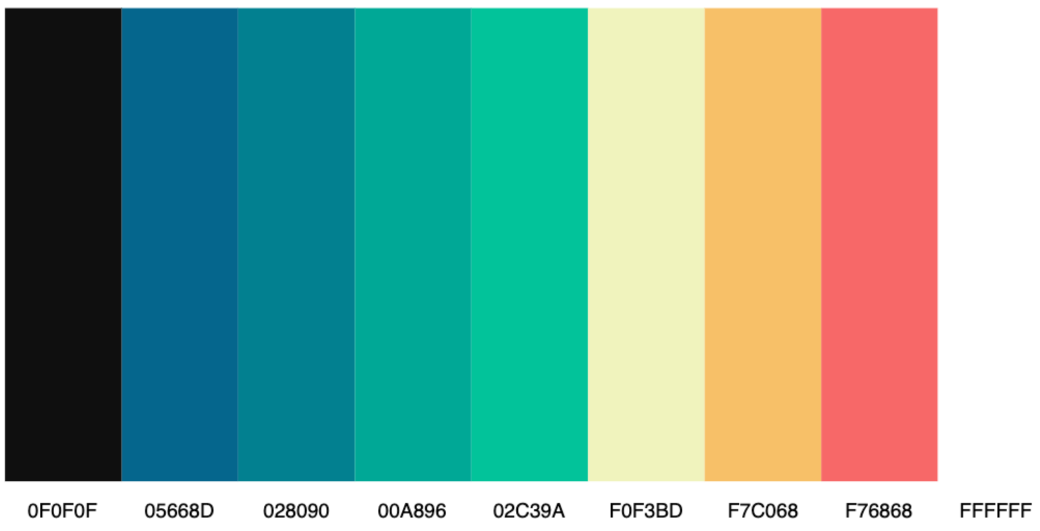

Figure 22: Prototype UI Colour Palette

As Figure 22 demonstrates, the colour palette uses a wide range of colours however, this palette includes status colours, background and font colours. The UI will predominantly make use of the colours 05668D to F0F3BD as shown above. By utilising a combination of blue and green throughout the user interface, users may associate the colours with calmness, growth, health and healing [19].

3.7 Mock-ups

In order to create the wireframes for the application, Adobe XD was used due to previous experience with the software, as well as ease of use. After determining the functional and non-functional requirements for the application based on the results from the questionnaire, the mock-ups for the application could be started.

3.7.1 Login Page

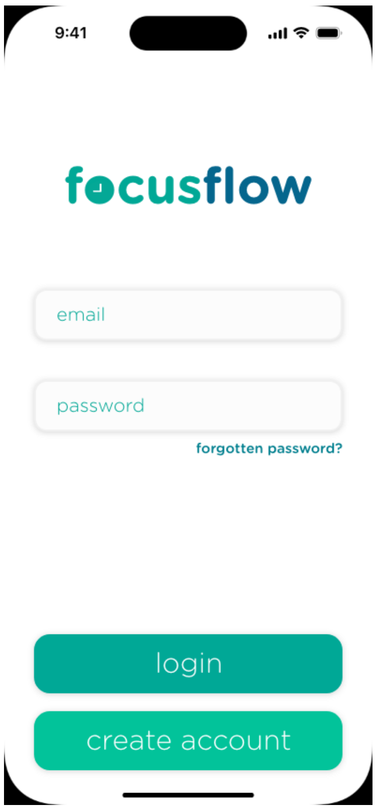

Figure 23: Login Page Mock-up

Figure 23 shows the mock-up login page for the app, displayed when a user is not logged in. When designing the login page, the decision was made to keep it simple and familiar to users in order to implement Nielsen's consistency and standards heuristic [8]. The logo has been centred near the top of the page which was created using Adobe Photoshop. Large, clear buttons have been used to complete actions whilst using flat colours from the colour palette making them easy to see. A mock-up for the create account page has not been produced, as it will be largely based on this login page design.

3.7.2 Tasks Page

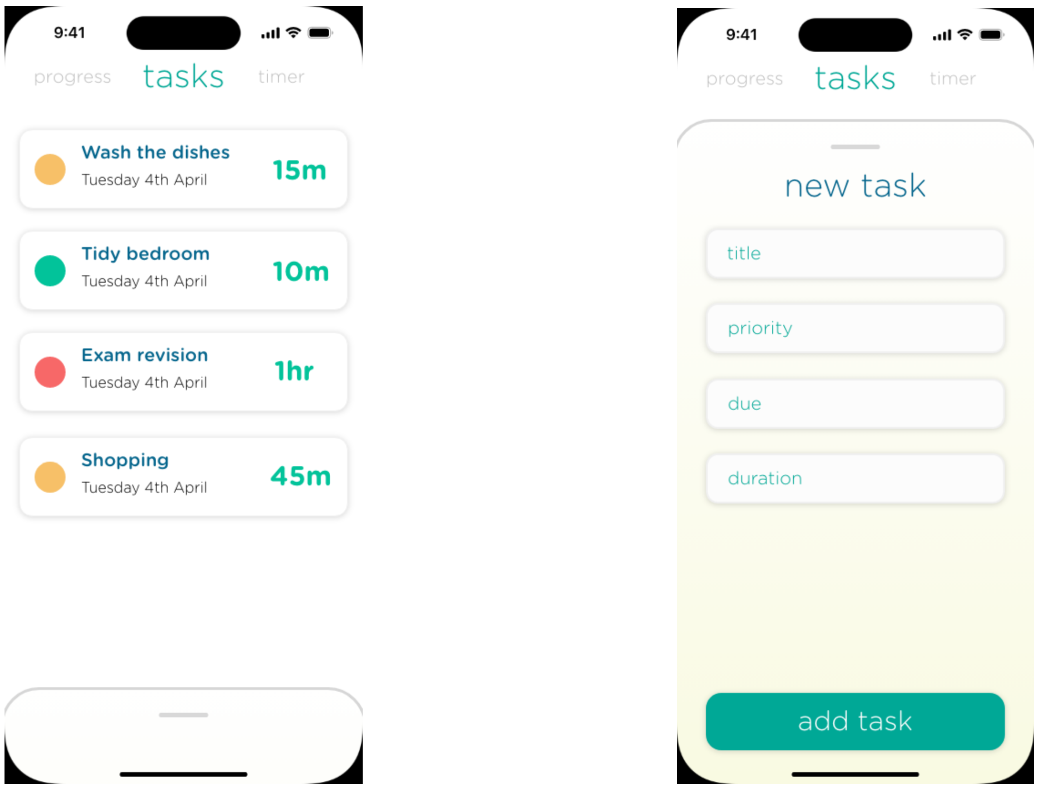

Figure 24: Tasks Page Mock-ups

The most important screen of the app is the tasks page. The tasks page will serve as the home page for the application when a user is signed in and will allow a user to quickly view any tasks they have to complete, along with the duration and priority, displayed as a coloured circle. When a user swipes up from the bottom of the tasks page, a ‘drawer’ will open allowing a task to be easily created and added to the tasks list. When designing this feature, simplicity and ease of use were the main design considerations taken into account to ensure its intuitive for the end user. When a user clicks on a task, it will direct them to the timer view for that task and allow the user to start a focus timer for the predetermined duration assigned by the user.

3.7.3 Timer Page

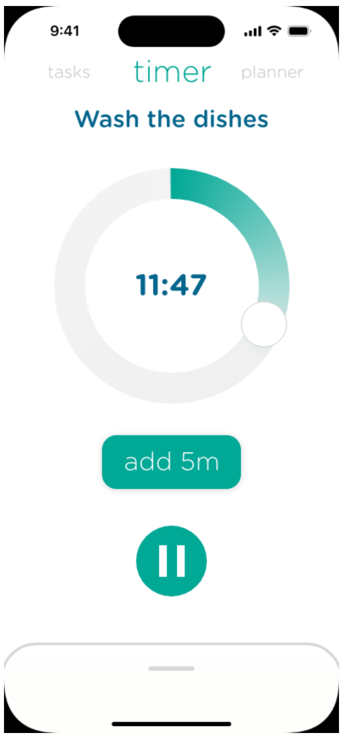

Figure 25: Timer Page Mock-up

For the timer page, the same design principles of simplicity and ease of use were used. As this is meant to be a focus timer, the page should be free of any distractions or clutter. The timer element is designed with a circular progress bar with the remaining time in the centre, which is an appropriate method of visualising time as it is similar to a dial-based kitchen timer, something users will be familiar with. Two HCI principles met by this design are recognition rather than recall and visibility of system status. Additionally, there is also a button which allows users to add 5 minutes to the timer if they have not allocated enough time should they need it, and a button to pause or resume the timer if necessary.

3.7.4 Planner Page

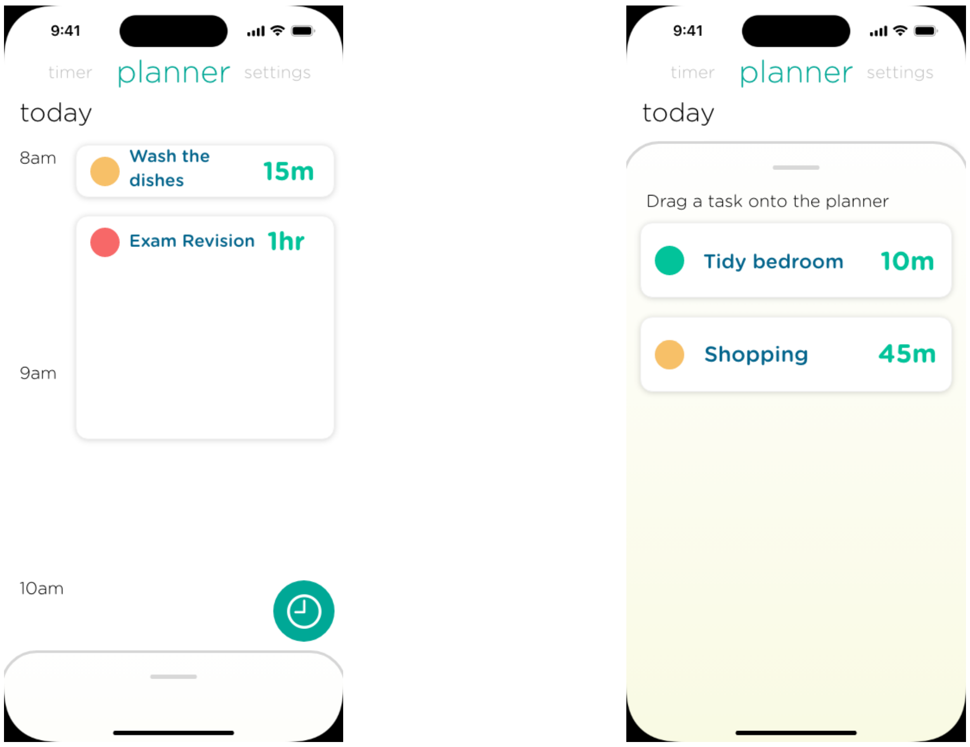

Figure 26: Planner Page Mock-ups

The planner page will provide users with a visual plan / timetable of their day and allow users to plan tasks throughout the day based on their duration. By swiping up or tapping the bottom of the display, the task drawer will be revealed displaying any user created tasks. Users will then be able to drag and drop a task onto the planner, allocating a time slot for the task. The height of the task will automatically be scaled depending on the duration allocated for the task. Users can also create “Breaks” by tapping the floating clock button which will allow them to easily and quickly add a break to the timetable.

3.7.5 Progress Page

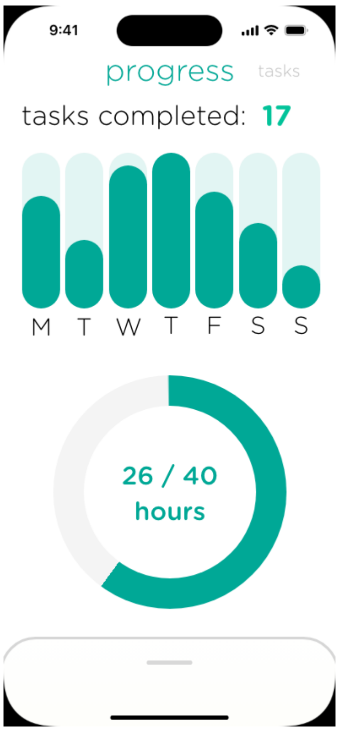

Figure 27: Progress Page Mock-up

Finally, the progress page will allow users to view their progress within the application, giving them a day-by-day breakdown of the number of tasks completed, as well as the number of hours spent on tasks against a goal specified by the user. This allows users to instantly assess their productivity levels and patterns throughout the week, offering a significant advantage over textual or numerical data representations. This design decision is rooted in the HCI principle of recognition rather than recall [8], aiding the users to understand the information more easily and swiftly. It may be possible to include the ability for weekly comparisons, as well as viewing monthly progress in the future, furthering the usefulness of this feature. The mock-up produced fulfils FR06 – progress tracking / reports.