Designing and Developing a Mobile Application to Help Improve Time Management and Organisation Skills in Students Diagnosed With ADHD

[DOCUMENT SUBTITLE]

HARRY KIRKWOOD (UG)

To-do / Ideas:

- Diagrams for the structure of the application (Backend – Firebase – Frontend).
-

Abstract

Declaration

Acknowledgements

Table of Contents

1. Introduction

1.1 Motivation

Attention Deficit Hyperactivity Disorder (ADHD) is a neurological condition that affects millions of people worldwide [1]. In the UK, an estimated 3-4% of adults have ADHD, which can have a significant impact on their ability to manage their time and stay organised [2]. As a result, individuals with ADHD often experience feelings of stress and overwhelm when they struggle to stay on top of their responsibilities.

As someone diagnosed with ADHD, I can personally attest to the challenges that come with managing this condition. In particular, time management and organisation have been ongoing struggles throughout my life. These issues have motivated me to develop an application that can help individuals with ADHD improve these skills and feel more in control of their lives.

It's becoming increasingly common for technology to be used as a method of helping individuals with ADHD on their time management and organisation [3]. There are a few existing applications such as Structured – Daily Planner [4] that currently aim to help individuals with ADHD however, many of them are unintuitive to use, or require too many steps to setup. Many of them also have paid features or require a subscription which is an immediate barrier to entry for many users. On top of this, many existing solutions are also mobile only, limiting their availability, which is a problem as access to the tools should be available wherever possible.

1.2 Rationale

For my project, I intend to produce an application, which aims to help improve time management and organisation skills in people with ADHD by including only strictly necessary features to avoid complications and distractions and a user interface (UI) that is intuitive through the use of usability principles. The idea is that the application will provide feedback to the user, using notifications and data visualisation for tasks and jobs the user has inputted. This is important as two of the main symptoms of ADHD according to the NHS are poor organisational skills and continually starting new tasks before finishing old ones [5]. As the application collects more of this data, it will be able to provide daily and weekly reports on the progress made by the user on their organisation and time management. While my initial focus is on creating a solution for students between the ages of 18 and 23, the potential impact of this solution could extend to a much broader range of age groups.

1.3 Aims and Objectives

1.3.1 Aim

The aim of this project is to develop an application aimed at helping students diagnosed with ADHD, improve their organisation and time management skills. I intend on testing and utilising user feedback to extend and improve the application. By collecting feedback, I will be able to effectively evaluate the functionality and performance of the application.

1.3.2 Objectives

1) **Research and identify three current applications and solutions for time management / organisation for users with ADHD.**

   By developing an understanding of existing solutions / methods of improving time management and their features, I will be able to gain an understanding of the features currently implemented by these solutions as well as their effectiveness and performance. I intend on choosing the top three most downloaded solutions on the Apple App Store for this research.

2) **Investigate established HCI usability principles and their practical implementation in order to inform the design and evaluation of a prototype for enhancing the user experience and efficiency.**

   For this objective, I plan to research and analyse established HCI usability principles that can be incorporated into the design of the application, such as user-centred design, simplicity, and consistency by reading research papers. This research will be integrated into the prototype which will enhance the user experience and efficiency and allow for the evaluation of the principles I have incorporated.

3) **Gather feedback from target users by conducting a survey to understand their needs and requirements.**

   This objective will be carried out by collecting data relating to the usability, functionality and design of the application. I intend to do this by carrying out a survey on 5 - 10 students diagnosed with ADHD. This survey will include designs and features from existing solutions and will allow me to refine the requirements for my own solution.

4) **Explore available development frameworks and tools to identify the most suitable and effective options for designing and developing a prototype application.**

   As part of this objective, I aim to explore and evaluate various existing frameworks and tools that can be utilized to create a functional and efficient prototype application, taking into consideration factors such as ease of use, flexibility, compatibility, and scalability.

5) **Design a prototype application based on the features and data gathered in research objectives 1, 2, 3 and 4.**

   In order to be able to produce a prototype, I will need to create some designs using relevant design mock-up software. I intend to experiment with and implement features identified in the previous research objectives, including the HCI usability principles.

6) **Develop a full-stack prototype based on the designs created in objective five.**

   To complete this objective, I will need to implement the designs created in objective five into a prototype application. I will also be using the frameworks and tools researched in objective four. Testing will be carried out throughout development as well as after the prototype is complete.

7) **Summarise and evaluate the solution.**

   Evaluating my solution will allow me to identify any mistakes I made, features overlooked, and whether my solution is adheres to HCI usability principles. Evaluation will be carried out through a user study by allowing individuals to use the app, followed by a survey to collect feedback.

2. Research

This section outlines the strategies and methodologies employed in the research process. It includes the selection and review of relevant literature, the design and implementation of surveys, and the analysis of data collected from these sources. The goal of this section is to provide a comprehensive overview of the methods used to gather and interpret data, ensuring the research is reliable, valid, and applicable to the project's objectives.

2.1 Literature Review

The literature review section provides a critical analysis of existing research relevant to the project. It includes a review of studies on ADHD, time management, and the use of technology in managing ADHD symptoms. This section aims to identify gaps in current knowledge and research, inform the project's direction, and provide a theoretical framework for the study.

Research has shown a significant gap in the field of neurodivergent-led research into technological solutions for individuals with ADHD. This lack of representation can impact the outcome of research and the types of systems being developed within Computing and HCI. It's crucial to involve neurodivergent individuals in the research and development process to ensure that the resulting solutions are tailored to their unique needs and experiences. This insight underscores the importance of user-centred design in the development of the proposed application [3].

Another study evaluated why individuals with ADHD struggle to find effective digital time-management tools. The research identified several areas for improvement in these tools, highlighting the need for more research in this area to better understand the unique challenges faced by this population. By investigating different strategies used by adults with ADHD, the study provides valuable insights that can inform the design and development of more effective digital tools. These insights can also help to identify useful features and strategies used by current solutions, which can be incorporated into the prototype to better serve individuals with ADHD [6].

Research into Activities of Daily Living (ADL) and the difficulties encountered by individuals with ADHD at managing and keeping track of ADLs was also conducted. The study presented five design principles devised by the researchers, aiming to help individuals track and manage their ADLs while being unintrusive. While the study focused on wearable devices, the design principles are highly relevant to this project. They can be implemented and possibly extended or modified as necessary to enhance the usability of the proposed application for individuals diagnosed with ADHD [7].

Jakob Nielsen's 10 Usability Heuristics, a set of 10 general principles for interactive design, were also reviewed for their relevance to this project. These principles cover different aspects of usability and aim to guide the design of an efficient, usable user experience (UX) and user interface (UI). By following Nielsen's usability heuristics, the application can be designed to ensure that it is easy to use and navigate, with clear and concise instructions, minimal cognitive load, and a visually appealing interface. The effectiveness of these heuristics will be evaluated through a heuristics evaluation and user feedback, allowing for future refinement and improvement of the application [8].

Research still to add:

- ==Good survey methods / questionnaires==
- ==More on usability and HCI.==
- ==Research on existing features==
- ADHD in students?
- Mention primary vs secondary research, qualitative & quantitative
- For each source, add my thoughts on the research and validity / relevance.
- Research on tools & frameworks, alternatives etc:
  - Full-stack web approach.
  - Different stacks I could have used: MEVN, FEVN, etc.
  - Vue.js, React, Angular for front-end.
  - Mongo, RxDB, Firebase for database.
  - Node.js and co for back-end.
- Software development methodology (waterfall).

2.2 Similar Applications Analysis

This section provides an analysis of three existing ADHD time management applications, highlighting their pros and cons.

2.2.1 Structured – Daily Planner
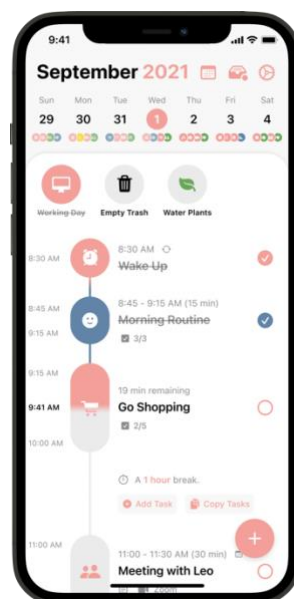
Figure 1: Structured - Daily Planner User Interface

Structured (Figure 1) is an application designed to help individuals with ADHD manage their time effectively. The application's strengths lie in its simplicity and user-friendly interface, making it easy for users to plan their day and track their tasks. However, it lacks advanced features such as data visualisation and progress tracking, which could provide users with more insight into their current time management habits. Additionally, the app is mobile-only, limiting its accessibility for users who prefer to use other devices [4].

2.2.2 Todoist
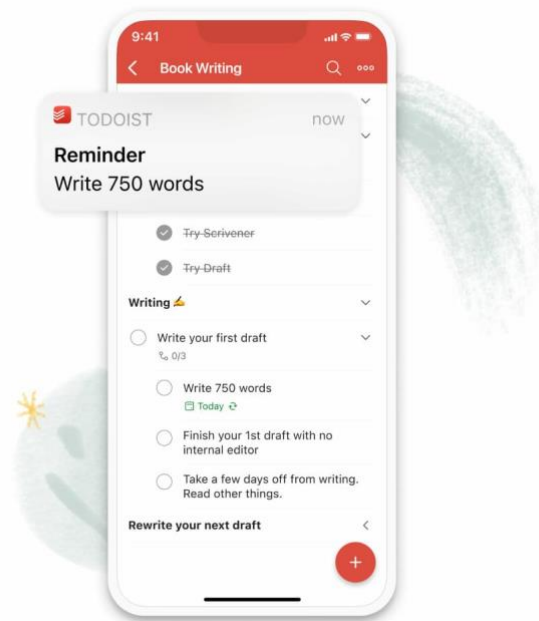
Figure 2: Todoist User Interface

Todoist (Figure 2) is a comprehensive task management app that allows users to create, manage, and organise tasks and projects [9]. It offers features such as due dates, priority levels, and labels, which can be particularly helpful for individuals with ADHD. However, while Todoist is feature-rich, its interface can be overwhelming for some users, especially those new to digital task management. Furthermore, some of its advanced features are only available in the paid version, which may be a barrier for some users.

2.2.3 Tiimo
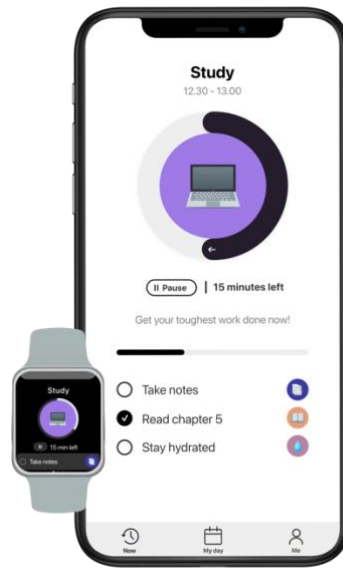
Figure 3: Tiimo User Interface

Tiimo (Figure 3) is a mobile application specifically designed for individuals with ADHD and autism. It offers visual schedules and reminders to help users structure their day, manage tasks, and build routines. The app's visual and auditory cues can be particularly beneficial for individuals with ADHD. However, some users may find the app's design and colour scheme less appealing. Additionally, while Tiimo offers a free version, some features are only available in the paid version [10].

The analysis of existing ADHD time management applications - Structured - Daily Planner, Todoist, and Tiimo - has provided valuable insights into the strengths and weaknesses of current solutions. It is evident that simplicity, user-friendly interfaces, and features tailored to the unique needs of individuals with ADHD, such as visual schedules and reminders, are crucial for the success of these applications. However, the analysis also highlighted areas for improvement, such as the need for more advanced features like data visualisation and progress tracking, and the importance of making all features accessible in free versions of the apps. These insights will inform the design of the questionnaire, guiding the selection of questions to better understand users' needs and preferences. For instance, questions could be included to gauge the importance of different features identified in the app analysis, and to explore any challenges users have experienced with existing apps. This will ensure that the questionnaire captures comprehensive data to inform the design and development of a more effective and user-friendly ADHD time management application.

2.3 Questionnaire

This section outlines the design and implementation of the questionnaire used to gather data from the target population. A comprehensive guide on designing effective questionnaires was reviewed to inform the design of the survey [11]. The questionnaire was designed to be clear, unambiguous, and easy to understand, ensuring that it effectively gathered the necessary data to support the research objectives.

2.3.1 Carrying out the Questionnaire

Considering ethical guidelines and the importance of safeguarding participant privacy, an online questionnaire was deemed the most suitable approach. This method ensures respondents can maintain anonymity while facilitating the automatic organisation of result data. The choice of an online questionnaire platform over in-person interviews or focus groups was motivated primarily by the necessity for anonymity and the available time constraints. Conducting interviews or focus groups would require significant setup time and data recording efforts, potentially yielding a lesser quantity of data. The next stage was to determine the questions to be used in the survey.

2.3.2 Questionnaire Flow

In order to obtain more useful data from the questionnaire, the branching feature will be added. This allows condition-based flow through a questionnaire depending on the responses given. The flow for the questionnaire is as follows:
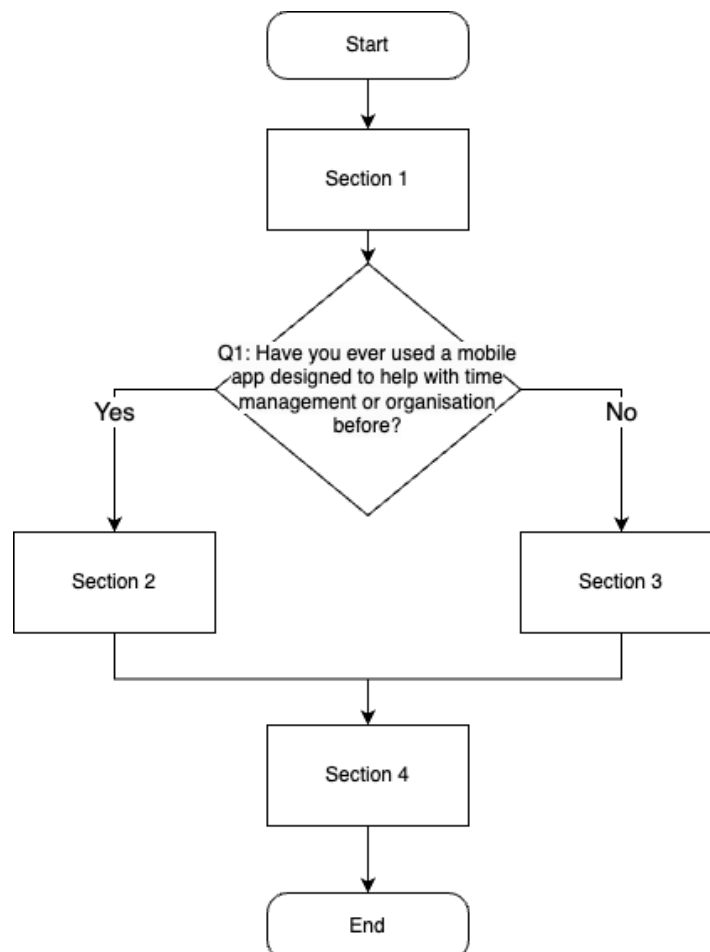

Figure 4: Questionnaire Flow Diagram

2.3.3 Section 1.

Question 1.

Have you ever used a mobile app designed to help with time management or organisation before?

<u>Rationale</u>

This question is designed to establish whether the respondent has used a time management or organisational app before. It's a critical first step in understanding the respondent's previous experience with such applications, providing context for their subsequent answers. This question can provide insights into how prevalent these types of apps are among the target demographic. According to the principle of learnability in HCI, users should be able to operate a system after learning it once [12]. Therefore, understanding the user's prior experiences can provide insights into how quickly they may adapt to a new app.

2.3.4 Section 2.

The second section is for participants who have used time management or organisation apps before.

Question 2.

Which features of these apps have you found most beneficial?

<u>Rationale</u>

This question aims to identify the features that users find most useful in current time management apps. By aligning the features of the new app with those that users find beneficial, the developers can enhance the principle of satisfaction [12], which refers to the user's subjective satisfaction when using the system.

Question 3.

What challenges, if any, did you face when using these time management / organisation apps?

<u>Rationale</u>

Question 3 seeks to understand the pain points and difficulties that users experience with existing apps, providing valuable information to avoid these issues when designing the new application, enhancing its usability [12].

2.3.5 Section 3.

Section 3 is for participants who have not previously used a time management or organisation application.

Question 4.

Would you use a time management / organisation mobile application?

<u>Rationale</u>

The only question in this section asks if they would use such an application. This question aims to gauge the interest and potential demand for the proposed app.

2.3.6 Section 4.

The fourth section is aimed at all participants.

Question 5.

Which three features are most important to you in a time management / organisation application?

<u>Rationale</u>

The fifth question asks participants to choose the three most important features in a time management or organisation app. This question is based on the concept of prioritising user needs and ensuring that the most essential features are included in the application [13].

Question 6.

Would you find a priority ranking system for tasks beneficial?

<u>Rationale</u>

This question aims to understand if users would find a specific feature useful, following the principle of utility in HCI, which refers to whether the system provides the features you need [12].

Question 7.

What type of visual design and layout would be most appealing to you in a time management app?

<u>Rationale</u>

The next question addresses visual design and layout preferences. Aesthetics and visual design play a significant role in the overall user experience and can impact the usability of the application [12] [14]. Aesthetic design can affect a user's perception of usability, while minimalist design reduces cognitive load.

Question 8.

What would make an application easy for you to navigate?

<u>Rationale</u>

Question eight focuses on navigation preferences, as navigation is a key aspect of usability [13].

Question 9.

How important is the ease of use when you select an application?

<u>Rationale</u>

The penultimate question assesses the importance of ease of use when selecting an application. This question helps to understand how significant usability is for the target user group [12].

Question 10.

Are there specific features or improvements not mentioned earlier that you think would be beneficial for a time management and organisation application targeted for individuals with ADHD?

<u>Rationale</u>

Finally, the last question inquires about any specific features or improvements not mentioned earlier that the participants think would be beneficial for a time management and organisation application targeted for individuals with ADHD. This open-ended question allows participants to provide valuable insights and feedback that may not have been covered in the earlier questions [15].

2.3.7 Questionnaire Results

In total, 15 responses were recorded. Ideally a higher number would be better however, given the participant requirement of having an ADHD diagnosis, 15 should be more than enough. The questionnaire was distributed through Instagram stories.

Question 1.

1. **Have you ever used a mobile app designed to help with time management or organisation before?**

More Details   Insights

- Yes   11
- No   4

Figure 5: Questionnaire Question 1 Results

A high number of participants responses (73%) suggested that they already use or have tried to use a time management / organisation application.

Which features of these apps have you found most beneficial?

Figure 6: Questionnaire Question 2 Results

This question was qualitative in order to allow respondents more flexibility with their responses. A predominant theme that emerged across the responses was the value placed on reminders and notifications for tasks and deadlines. This functionality appears to be integral to helping the students maintain their focus, avoid forgetting important tasks, and adhere to set timelines.

Secondly, the ability to create, manage, and prioritise to-do lists was highly appreciated. It was evident that this feature assists in the organisation of tasks and helps to structure students' work. It also allows for effective prioritisation based on importance or urgency, thereby enabling efficient allocation of time and effort.

A visual calendar or planner view was another highly valued feature. The participants indicated that these tools provide an overview of their schedule and help them to plan and structure their day. The integration of the to-do lists with these calendar apps for scheduling and planning was seen as an enhancement to this functionality.

The incorporation of focus or time management tools, such as a focus timer, was also acknowledged as beneficial. These features aid in concentration and time management, both of which can be challenging for students with ADHD.

What challenges, if any, did you face when using these time management / organisation apps?
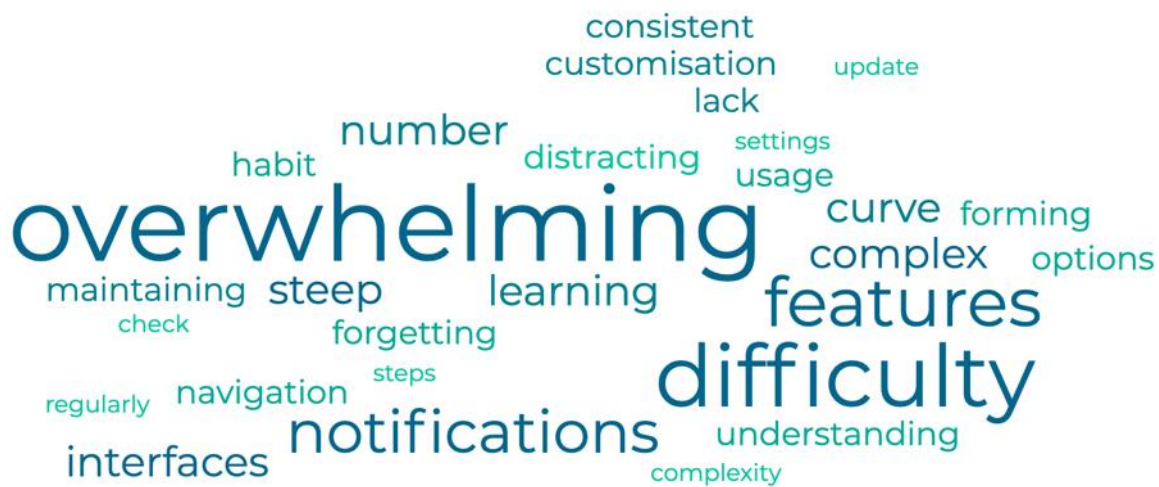

Figure 7: Questionnaire Question 3 Results

In analysing the survey responses to question 3, a number of challenges experienced by students with ADHD when using time management and organisation apps were identified.

One of the most prominent issues reported was the complexity of these apps, often due to a multitude of features and options. This complexity not only made the apps difficult to navigate, but also overwhelming for the users. This response suggests that overly complex interfaces can inhibit the effectiveness of the app and may deter consistent use.

The aforementioned complexity contributed to another significant challenge, which was the learning curve associated with these apps. Many students reported that it took considerable time and effort to understand and navigate the features of the apps effectively. This initial difficulty could cause frustration and potentially affect the regular usage of the app.

Interestingly, a feature often praised in the previous question – notifications and reminders – was also identified as a challenge. Some respondents found the volume of notifications overwhelming, to the point of being counterproductive, and in some instances, a source of distraction. This suggests that while reminders are beneficial, there needs to be a balance or the ability to customise to prevent them from becoming a nuisance.

Question 4.

4. **Would you use a time management / organisation mobile application?**

More Details

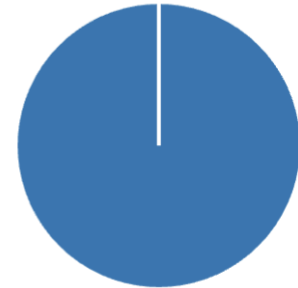|		
🔵 Yes	4	
🟠 No	0	
🟢 Maybe	0	

Figure 8: Questionnaire Question 4 Results

Out of those identified in question 1 that have not used a time management / organisation application before, 100% responded that they would use one if a suitable application became available.

Question 5.

5. **Which three features are most important to you in a time management / organisation application?**

More Details

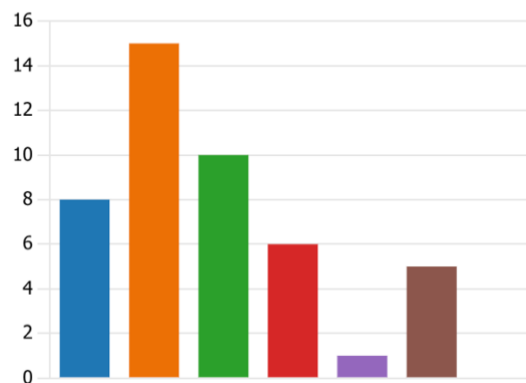|	
🔵 Focus Timer	8
🟠 Todo List	15
🟢 Timetable / Planner	10
🔴 Progress analysis / Reports (a br...	6
🟣 Gamification	1
🟤 Reminders	5
🟣 Other	0

Figure 9: Questionnaire Question 5 Results

The most favoured features in a time management/organisation application, as indicated by the survey respondents, were the "Todo List" (15 selections), "Timetable/Planner" (10 selections), and "Focus Timer" (8 selections). These tools are evidently valued for their ability to aid in task organisation, time management, and maintaining focus. "Progress analysis/Reports" and "Reminders" were also selected by a subset of participants (6 and 5 respectively), indicating their usefulness for tracking productivity and remembering tasks. "Gamification" was chosen by only one participant, suggesting it is not a critical feature for this user group. No participants selected the "Other" option.

Question 6.

6. **Would you find a priority ranking system for tasks beneficial?**

More Details   Insights

● Yes   15
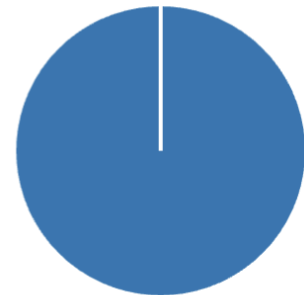● No   0

Figure 10: Questionnaire Question 6 Results

In response to question 6, 100% of participants indicated that they would find a priority ranking system for tasks beneficial. This aligns with responses from question 2 which suggested respondents appreciated having the ability to prioritise tasks based on importance.

Question 7.

7. **What type of visual design and layout would be most appealing to you in a time management app?**

More Details   Insights

● Minimalist design with simple c...   9
● Modern and trendy design with ...   0
● Classic design with traditional c... 0
● Fun and playful design with cart... 0
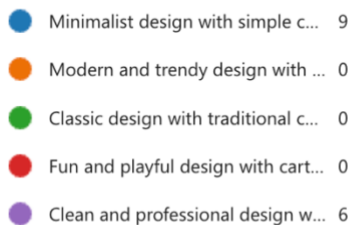● Clean and professional design w...   6

Figure 11: Questionnaire Question 7 Results

The majority of respondents, nine in total, expressed a preference for a minimalist design with simple colour schemes. This suggests that the students value simplicity and clarity in the user interface of the app. A minimalist design can help reduce cognitive load, which might be particularly beneficial for students with ADHD, as it can make the app more intuitive and easy to navigate [8].

Six participants expressed a preference for a clean and professional design, emphasising functionality over aesthetics. This suggests that these students value an application that is straightforward and practical, and does not distract from its primary purpose of helping them manage their time and organise their tasks.

Interestingly, no respondents selected a modern and trendy design with bright colours and bold typography. Similarly, no participants favoured a classic design with traditional colours and fonts, or a fun and playful design with cartoonish characters or illustrations. These preferences could be interpreted as a desire for a design that is not overly stimulating or distracting, which could be especially important for students with ADHD who might find such designs overwhelming or distracting.

8. What would make an application easy for you to navigate?

More Details   💡 Insights

🔵	Swipe-based pages	8
🟠	Navigation bar	7
🟢	Menu	0
🔴	Other	0

Figure 12: Questionnaire Question 8 Results

Based on the results, respondents appear divided between "Swipe-based pages" (53%) and "Navigation Bar" (47%). As a result, the final navigation implementation will attempt to incorporate both aspects in order to make it as easy to navigate as possible for all users.

9. How important is the ease of use when you select an application?

More Details

🔵	Extremely important	15
🟠	Somewhat important	0
🟢	Neutral	0
🔴	Somewhat unimportant	0
🟣	Extremely unimportant	0

Figure 13: Questionnaire Question 9 Results

Question 9 also further reiterates the findings from question 2 that complexity is unpopular in users diagnosed with ADHD. 100% of participants answered that ease of use is extremely important when choosing a time management / organisation application.

Are there specific features or improvements not mentioned earlier that you think would be beneficial for a time management and organisation application targeted for individuals with ADHD?

Figure 14: Questionnaire Question 10 Results

An analysis of the responses to question 10 reveals a variety of suggested features and improvements that the participants believe would be beneficial for a time management and organisation application specifically targeted at individuals with ADHD.

A recurring theme among the responses was the suggestion for a feature that allows users to break down larger tasks into smaller, more manageable subtasks. This feature would potentially help users to manage overwhelming tasks more effectively by providing a step-by-step approach, making it easier to focus and progress towards completion.

Another commonly suggested feature was the incorporation of a progress tracking system, with an emphasis on visual representations of accomplishments. This could provide motivation and a sense of achievement, enhancing engagement with the app. Some respondents further suggested that this progress tracking could be paired with a rewards or incentives system, thereby introducing a gamified element to encourage task completion.

Notifications and reminders were also highlighted as potentially beneficial, not only for tasks but also for other important aspects such as medication intake. This indicates a desire for customisable notifications that can support a variety of needs in individuals with ADHD.

Several participants proposed features related to time management techniques, such as time-blocking or chunking tasks into manageable portions. These techniques can help maintain focus and avoid overwhelm, which can be particularly beneficial for individuals with ADHD.

3 Design

3.1 Plan
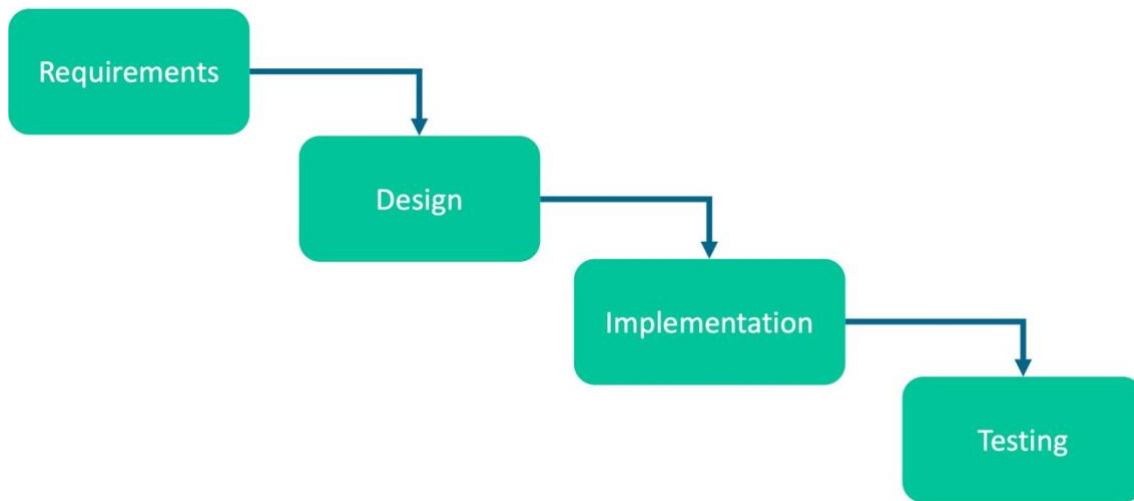
Figure 15: Waterfall SDLC adaptation

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 will be used to determine the requirements as well as to shape the foundational design of the application.

3.2 Tools and Technologies

Microsoft Forms

Microsoft Forms has been chosen for the purpose of conducting an online survey. The primary reasons for choosing Microsoft Forms include its cost-free nature and its ability to generate a cover page – a feature not available in Google Forms. This cover page can serve as a platform to disclose the survey's brief and ethics, and to obtain consent from respondents.

Adobe XD

Adobe XD is a very powerful wireframing and prototyping tool which is perfectly suited to this project. Unlike alternatives such as Balsamiq, it is very easy to quickly create components from scratch and reuse them throughout your designs. Prior experience with Adobe XD will also speed up the design process.

Draw.io

Draw.io is a tool that will be employed for the creation of high-quality diagrams such as an ER diagram, UML class diagram, and a user flow diagram.

3.3 Functional Requirements

No.	Requirement	Priority (H, M, L)
FR01	The application should allow users to create accounts with secure login and authentication features, ensuring data privacy.	H
FR02	Users should be able to set reminders for upcoming tasks, classes, or events, with the option to customise notification types (e.g., in-app or push notifications) and frequency.	M
FR03	The application should provide configurable timers that allow users to focus for a period of time. Visual and/or auditory cues should signal the end of a work or break session, with an option to pause or reset the timer if needed.	H
FR04	The application should support creating, and deleting tasks, as well as setting priorities. Users should be able to organize tasks by categories and filter tasks based on priority or due date.	H
FR05	The app should provide a user-friendly interface to create, edit, and view their schedules.	H
FR06	The application should track users' progress on tasks and display visual representations of their productivity over time. Users should be able to view this progress in the form of charts.	H
FR07	Users should be able to customise the appearance, and settings to ensure a personalized experience.	M
FR08	Users should earn awards for completing tasks or multiple tasks to encourage increased engagement with the application.	L
FR09	Include educational resources and practical tips on time management and organization strategies for students with ADHD.	L

Figure 16: Functional Requirements Table

3.4 Non-functional Requirements

No.	Description	Priority (H, M, L)
NFR01	The application should consider and implement Jakob Nielsen's Usability Heuristics where suitable to improve the usability experience [8].	H
NFR02	The application should adhere to industry-standard security practices to protect user data and maintain privacy. This includes encryption of data in transit and at rest and secure authentication mechanisms.	H
NFR03	The application should work seamlessly across different devices, operating systems, and web browsers, ensuring that all users have access to its features regardless of their preferred platform.	M
NFR04	The application should be designed to work well on different screen sizes and resolutions, providing a consistent and enjoyable user experience across various devices.	M

Figure 17: Non-functional Requirements Table

3.5 Application Structure
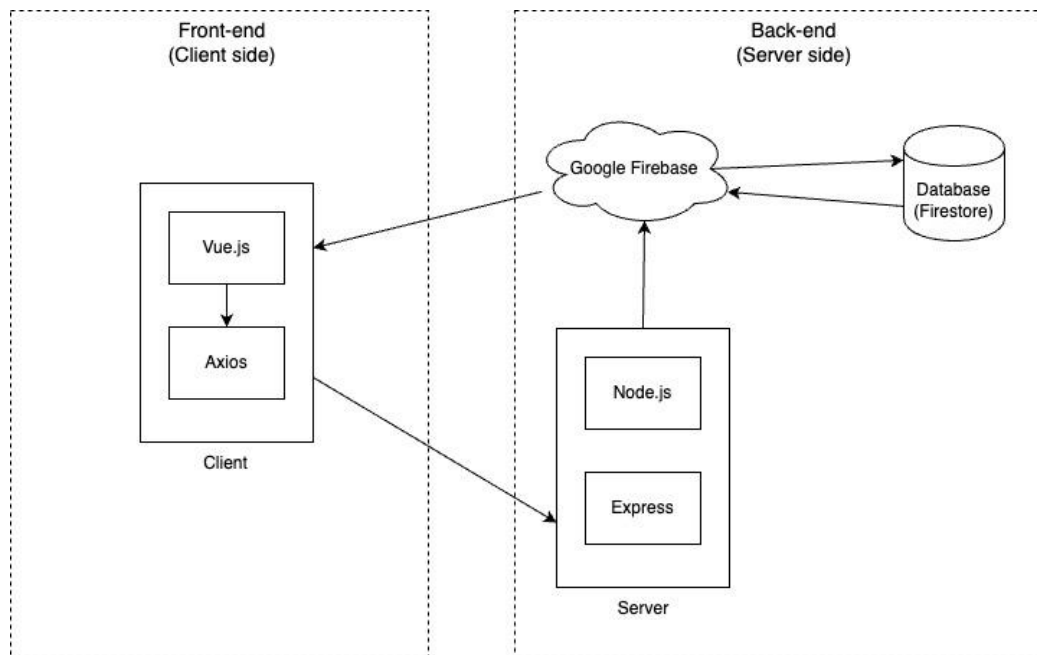
3.5.1 Application Architecture

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.
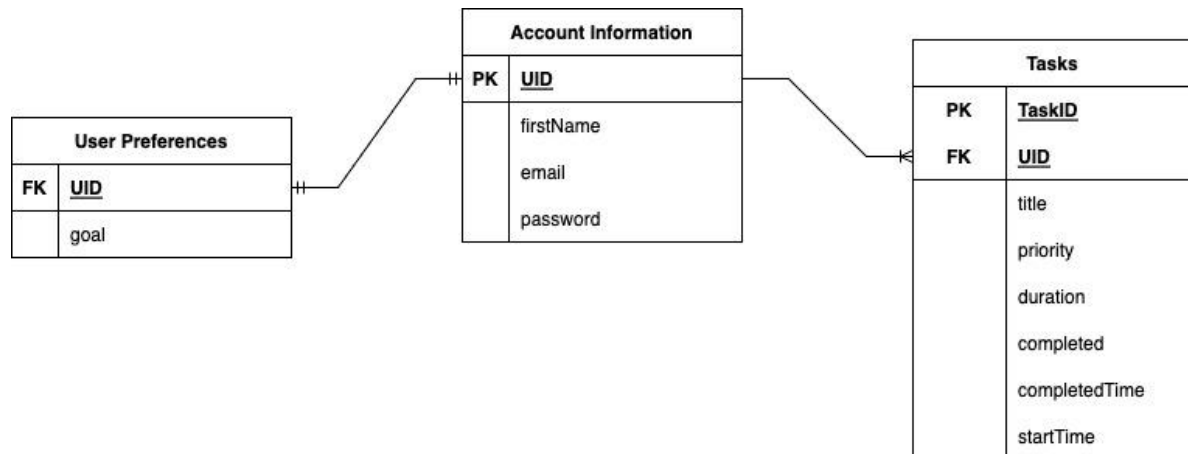
Figure 19: Entity Relationship (ER) Database Diagram

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.

Account Information

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.

Tasks

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.

User Preferences

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.
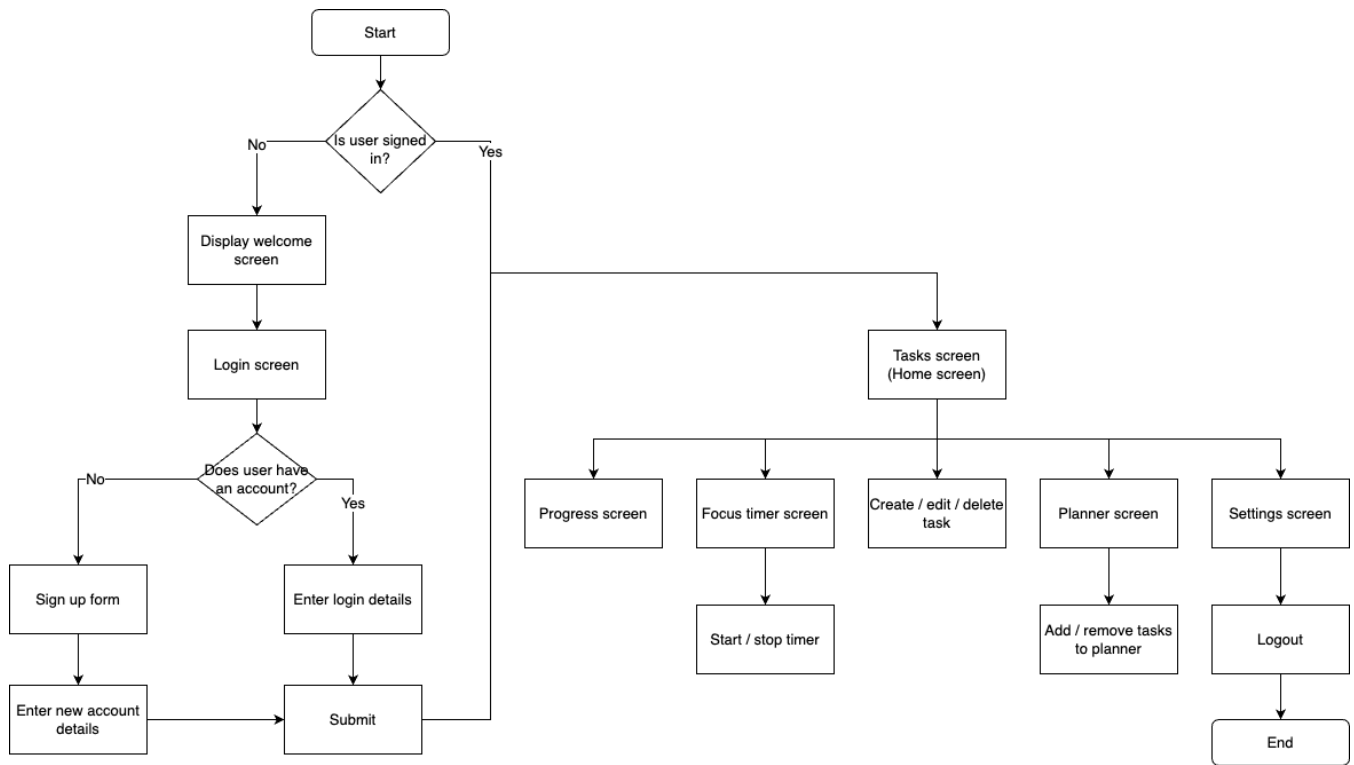

Figure 20: User Flow Diagram

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.
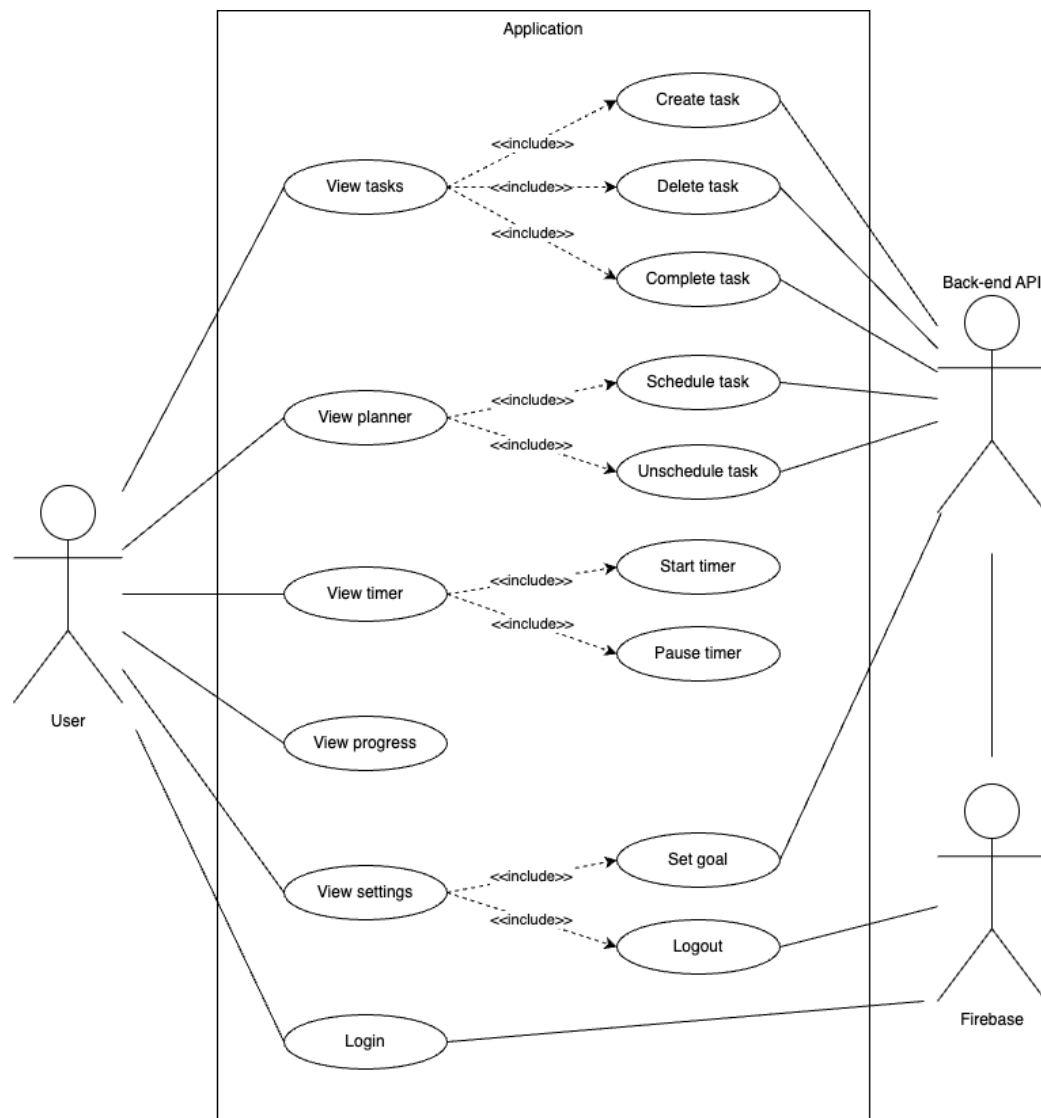

Figure 21: Use Case Diagram

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.

| 0F0F0F | 05668D | 028090 | 00A896 | 02C39A | F0F3BD | F7C068 | F76868 | FFFFFF |
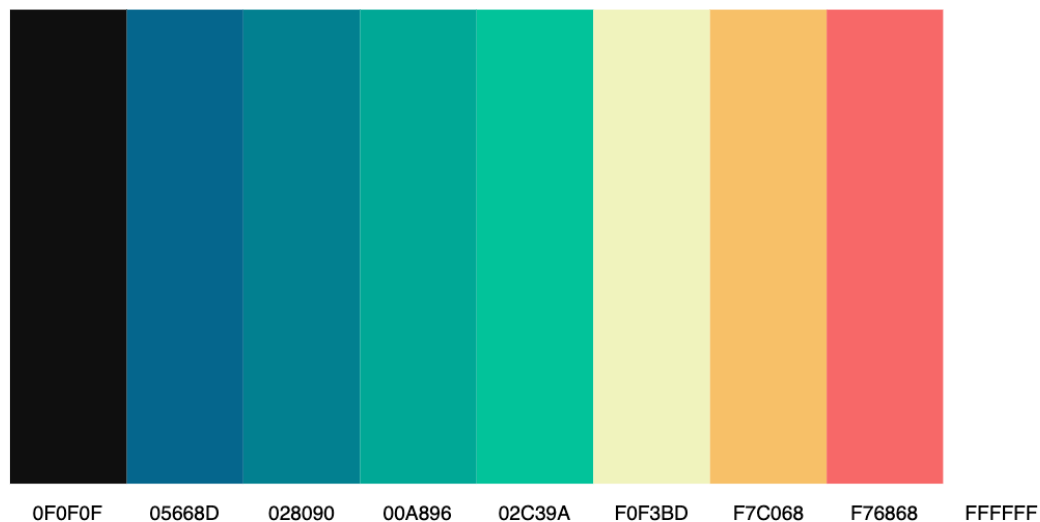
Figure 22: Prototype UI Colour Palette

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
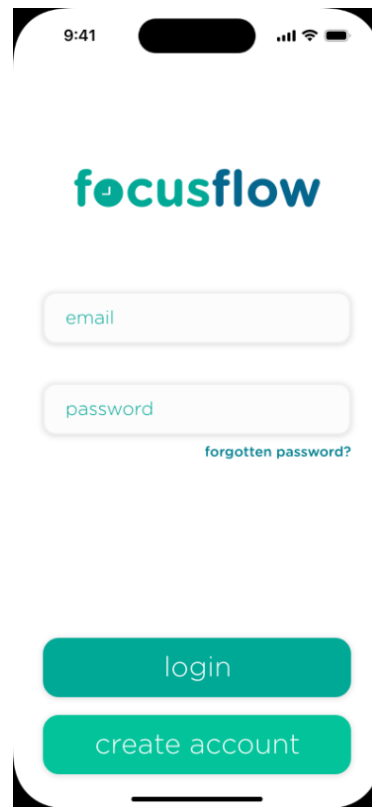
Figure 23: Login Page Mock-up

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
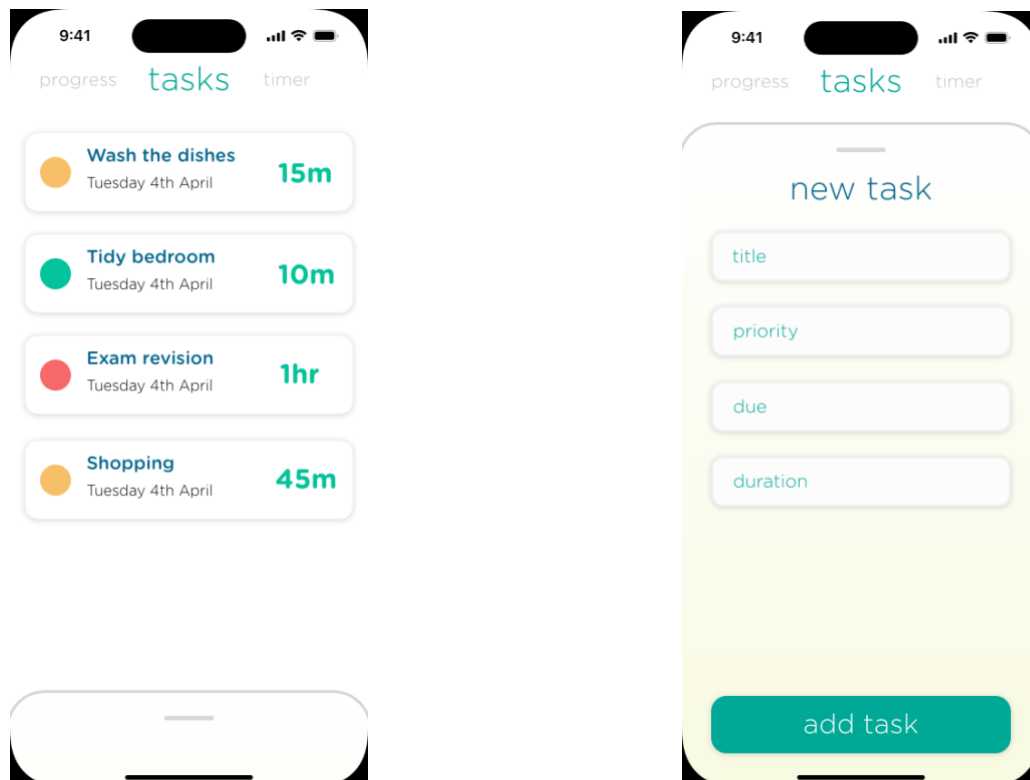
Figure 24: Tasks Page Mock-ups

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
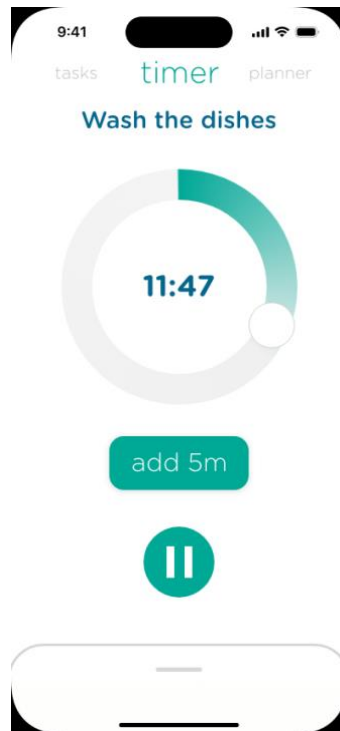
Figure 25: Timer Page Mock-up

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
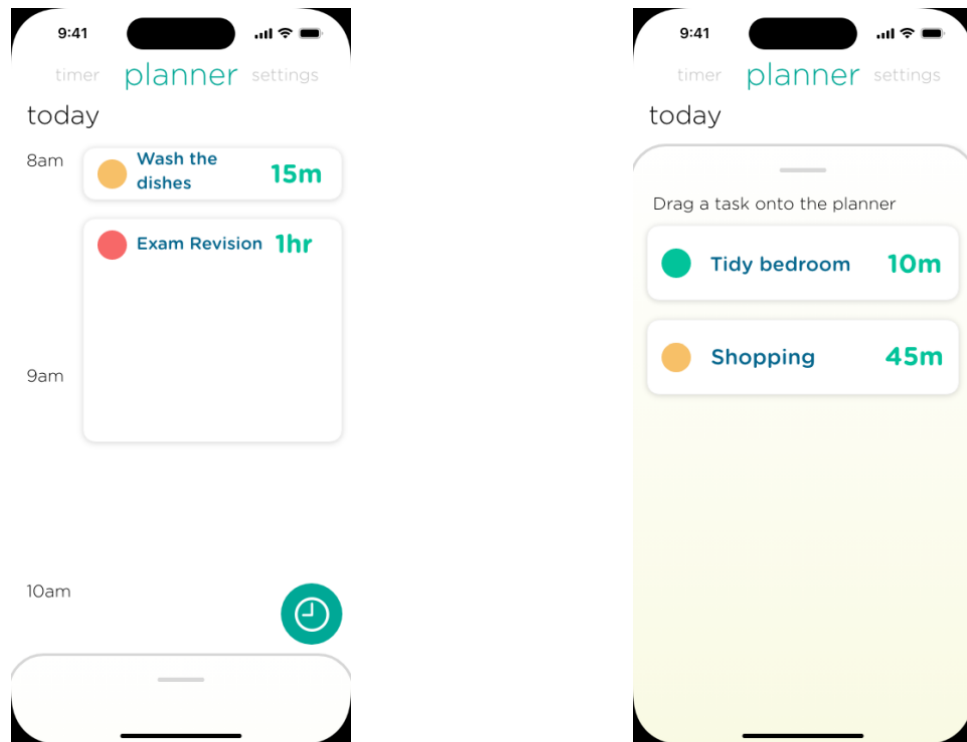
Figure 26: Planner Page Mock-ups

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.

4 Development

4.1 Development of the Back-end

In order to develop a functional back-end, a basic Node.js project was setup along with the Express framework as a starting point. This would act as the server for the API endpoints which the front-end will make requests to. The first functionality implemented was the ability to create, edit and delete tasks.

4.1.1 Task Methods

```
async function createTask(task, uid) {
    try {
        // Creating a reference to the specified user Tasks database.
        const tasksDbRef = collection(db, "Tasks", uid, "Tasks")
        await addDoc(tasksDbRef, {
            title: task.title,
            duration: parseInt(task.duration),
            priority: task.priority,
            completed: false
        });
    } catch (error) {
        console.log(error);
        throw new Error(error);
    }
}
```

Figure 28: createTask Back-end API Method

Figure 28 is a snippet of code used by the back-end to create a new task for the user in the tasks database table. By using try / catch blocks, errors can be handled effectively and reduce the chances of the server crashing if an error is encountered [20]. Try / catch blocks also allow for much improved debugging as they provide useful information about the error, making it easier to identify and fix issues within the code. The code for deleting and editing a task is based on the code in Figure 28 as both are manipulating data from the same database table.

After creating the methods for creating, editing and deleting tasks, an endpoint was created which will allow the front-end to access the method. Firstly, a controller was created for the tasks which is used to handle the HTTP request and response as well as call the relevant method:

```
class TasksApiController {
    createTask(req, res) {
        createTaskApi(req.body, req.authId)
            .then(uid => {
                res.status(200).send(uid)
            })
            .catch(err => {
                res.status(400).send(err)
            })
    }
}
```

Figure 29: TasksApiController

All methods used by the back-end require a controller and follow the same structure as Figure 29.

Followed by the routing endpoints:

```
const tasks_router = express.Router();
tasks_router.use(bodyParser.json());

const tasksApiController = require('./tasksApiController');

tasks_router.post(
    '/tasks/createtask',
    authenticate_token,
    tasksApiController['createTask']);

tasks_router.post(
    '/tasks/deletetask',
    authenticate_token,
    tasksApiController['deleteTask']);

tasks_router.post(
    '/tasks/edittask',
    authenticate_token,
    tasksApiController['editTask']);


module.exports = tasks_router;
```

Figure 30: Tasks Router and Endpoints

In addition to creating, updating and deleting a task, methods are required to mark a task as complete, as well as allow a user to set a weekly goal on the hours spent completing tasks:

```
async function markCompleted(task, uid) {
    try {
        const tasksDocRef = doc(db, "Tasks", uid);
        const tasksColRef = collection(tasksDocRef, "Tasks")
        await setDoc(doc(tasksColRef, task.id), {
            completed: true,
            completedTime: serverTimestamp(),
            startTime: null
        }, {
            merge: true
        });

    } catch (error) {
        throw new Error(error);
    }
}
```

Figure 31: markCompleted Back-end API Method

When markCompleted is called, the completed Boolean is updated to true and the completedTime field is set equal to a timestamp generated by the Firebase server. The startTime field is also updated to null, which is used by the planner function when determining where a task should be placed on the timeline. The merge flag is used by Firebase to determine whether to merge a document or overwrite it.

```
async function setGoal(body, uid) {
    try {
        const preferencesDocRef = doc(db, "UserPreferences", uid);
        await setDoc(preferencesDocRef, {
            goal: body.goal
        }, {
            merge: true
        });

    } catch (error) {
        throw new Error(error);
    }
}
```

Figure 32: setGoal Back-end API Method

The setGoal method simply updates the user's goal field in their associated UserPreferences document with the given value specified by the user.

4.1.2 Authentication Middleware

In order to reduce the risk of unauthorised changes and access to the database, the back-end will require some form of authentication. After some research, authentication tokens were chosen for this process [21]. Fortunately, Firebase has inbuilt authentication functionality that allows for the generation of authentication tokens for users. This token can then be included in the 'Authorization' header of each HTTP request. The back-end can then utilise the Firebase Admin SDK verifyIdToken method, validating any HTTP request by checking that the user's authentication token included in the header. As seen in the tasks_router post endpoints in Figure 30, 'auth_token' is referenced as a parameter, which means the authentication middleware will be executed before the relevant method is called.

```javascript
const getAuthToken = (req, res, next) => {
    if (
        req.headers.authorization &&
        req.headers.authorization.split(' ')[0] === 'Bearer'
    ) {
        req.authToken = req.headers.authorization.split(' ')[1];
    } else {
        req.authToken = null;
    }
    next();
}

module.exports = (req, res, next) => {
    getAuthToken(req, res, async () => {
        try {
            const { authToken } = req;
            const userInfo = await admin.auth().verifyIdToken(authToken);
            req.authId = userInfo.uid;
            return next();
        } catch (e) {
            return res.status(401).send(
                {
                    error: 'You are not authorized to make this request'
                });
        }
    });
}
```

Figure 33: Authentication Middleware

In Figure 33, the getAuthToken function checks if there is an 'authorization' field in the request headers and if the authorization scheme is 'Bearer'. If both conditions are met, it extracts the token and stores it in the 'req.authToken' property; otherwise, it sets 'req.authToken' to null.

The exported middleware function calls the getAuthToken function to extract the token and then moves on to the asynchronous callback. Inside the callback, it first seperates the authToken from the req object. Then, it tries to verify the token using Firebase Admin SDK's verifyIdToken method, which returns a decoded token containing user information if the token is valid. If the token is verified, the user's UID is extracted from the decoded token and stored in req.authId. The middleware then moves on to the next function in the chain using the next() call. If an error occurs during the verification process, the middleware sends a HTTP 401 Unauthorized response with an error message indicating that the user is not authorized to make the request. This response can then be used by the front-end to indicate an error has occurred.

4.1.3 User Authentication

Next, under user authentication a method for creating a new user was implemented. This code is also very similar to the createTask method however, it involves adding data to the AccountInformation table in the database. First, the Firebase method createUserWithEmailAndPassword to create a new user is called. Following this, a new entry is created for the user in AccountInformation and UserPreferences which is used to store save any user settings.

```javascript
async function createAccount(newUser) {
    try {
        const cred = await createUserWithEmailAndPassword(auth,
newUser.email, newUser.password);

        await setDoc(doc(db, "AccountInformation", cred.user.uid), {
            firstName: newUser.firstName
        });

        await setDoc(doc(db, "UserPreferences", cred.user.uid), {
            goal: 40
        });
    } catch (error) {
        throw new Error(error);
    }
}
```

Figure 34: Create Account Method

4.1.4 Planner Methods

In order to implement the planner functionality, the back-end will require the methods to set and remove the start time of a task. By giving a task a start time, the front-end will be able to render the task on a timeline at the correct time allocated by the user.

```javascript
async function setStartTime(task, uid) {
    try {
        const tasksDbRef = collection(db, "Tasks", uid, "Tasks")
        await setDoc(doc(tasksDbRef, task.id), {
            startTime: task.startTime
        }, {
            merge: true
        });

    } catch (error) {
        throw new Error(error);
    }
}
```

Figure 35: setStartTime Back-end API Method

And removing the start time:

```javascript
async function removeStartTime(task, uid) {
    try {
        const tasksDbRef = collection(db, "Tasks", uid, "Tasks")
        await setDoc(doc(tasksDbRef, task.id), {
            startTime: null
        }, {
            merge: true
        });

    } catch (error) {
        throw new Error(error);
    }
}
```

Figure 36: removeStartTime Back-end API Method

4.2 Development of the Front-end

After the foundations for the back-end had been developed, work on the front-end could begin.

4.2.1 Making Calls to the Back-end

In order to reduce code duplication and speed up development, the decision was made to create a single exported JavaScript module called 'API.js'. The benefit of this is that it allows for the implementation of the methods used to make the API calls before their full implementation is complete. All non-authentication API calls will use Axios, a promise-based HTTP library to make requests to the back-end and will use the following structure:

```javascript
const createTask = async (payload) => {
    const auth = getAuth(firebaseApp);
    const token = await getIdToken(auth.currentUser);
    const url = host + '/tasks/createtask/';

    try {
        await axios.post(url, payload, {headers:
                { authorization: `Bearer ${token}` }}).then(async (r) => {
            await responseHandler(r, "Task Created")
        })
    }
    catch (e) {
        await errorHandler(e)
    }
}
```

Figure 37: Front-end Create Task Method

Another benefit to using a single API module, is that it allows for the creation of a response handler, which can be used to humanise the HTTP responses for the user in the form of error messages and status messages:

```javascript
// Response handler used to display toast message
// if api call is successful / if there is an error
const responseHandler = async (r, okay) => {
    if (r.status === 200) {
        const toast = await toastController
            .create({
                message: okay,
                duration: 2000,
                color: 'dark',
                position: 'top'
            })
        return toast.present();
    } else {
        // If response has no status or the
        // response handler encounters an error
        const toast = await toastController
            .create({
                message: "Oops an error has occurred!",
                duration: 2000,
                color: 'dark',
                position: 'top'
            })
        return toast.present();
    }
}
```

Figure 38: Front-end HTTP Response Handler

4.2.2 User Authentication

Login Page

Authenticating users on the front-end requires more work than the back-end, as not only is a method of preventing unauthorised access to the methods required, but also a system to prevent users from accessing the main application without being signed in or registered. In order to keep the code clean and structured, a single API module was created which would contain the separate API calls the front-end will make to the back-end API. This also allows for reuse of any calls throughout the application if required, without code duplication.

The first step was to begin work on the user interface, making use of and customising Ionic's inbuilt components such as the input fields and buttons in order to create a login screen:

```
<template>
  <ion-page>
    <ion-content fullscreen class="ion-padding" scroll-x="false" scroll-y="false">
      <div class="container" style="top: 20%;">
        <img src="../../assets/Logo.png" class="logo">
      </div>
      <div class="container ion-padding-horizontal" style="top: 50%;">
        <ion-item fill="outline" mode="md" style="min-width: 80%; padding-bottom:
20px;">
          <ion-input label="email" label-placement="floating" placeholder="email" v-
model="payload.email"></ion-input>
        </ion-item>
        <ion-item fill="outline" mode="md" style="min-width: 80%; padding-bottom:
20px;">
          <ion-input label="password"
                     type="password"
                     label-placement="floating"
                     placeholder="password"
                     v-model="payload.password">
          </ion-input>
        </ion-item>
      </div>
    </ion-content>

    <ion-footer class="ion-no-border ion-padding-bottom" translucent="true">
      <div>
        <ion-button expand="block"
                    class="ion-padding-horizontal"
                    size="large"
                    style="color: white; --border-radius: 20px; font-size: 25px;"
                    @click="doLogin">login
        </ion-button>
        <br>
        <ion-button expand="block"
                    class="ion-padding-horizontal"
                    size="large"
                    style="color: white; --border-radius: 20px; font-size: 25px; --
background: #02C39A;"
                    @click="signUp()">create account
        </ion-button>
      </div>
    </ion-footer>
  </ion-page>
</template>
```

Figure 39: Snippet of the Front-end Login Page

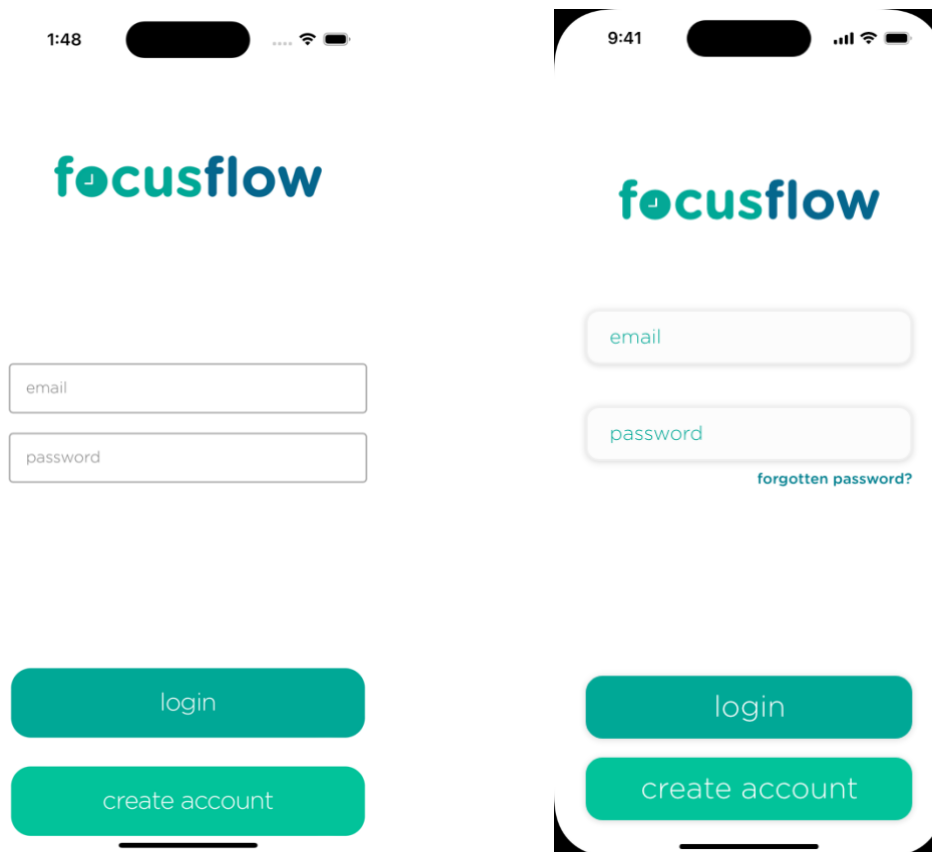This produces the following user interface:

Figure 40: Login page prototype (left) compared with the mockup (right)

Next, the login method can be created allowing the users to login. The login method was created using built in Firebase authentication methods. As previously mentioned, all back-end API calls and authentication methods will be contained within a single exported module called 'API.js'. The reason for this is to improve maintainability and reduce code duplication, as there may be a need for a single method to be used by multiple components.

```javascript
const login = async (payload) => {
    const auth = getAuth(firebaseApp);
    try {
        await signInWithEmailAndPassword(auth,
            payload.email,
            payload.password).then(async r => {
                r["status"] = 200
                await responseHandler(r,"Logged in successfully")
        })
    }
    catch (e) {
        e["status"] = 200
        await responseHandler(e, e.message)
    }
}
```

Figure 41: Front-end Login Method

Navigation Guard

In order to proceed, the application requires a Vue.js router. A router allows the application to be made up of multiple pages by mapping a URL endpoint to specific Vue.js components [22]. This allows for navigation through the app by changing the URL either manually or programmatically. The ability to change the URL programmatically allows for the use of a navigation guard. Navigation guards are used to guard navigations either by redirecting it or cancelling it [22]. This is required to prevent unauthorised users from accessing pages other than the login or registration page.

```typescript
import { createRouter, createWebHistory } from '@ionic/vue-router';
import { RouteRecordRaw } from 'vue-router';
import api from "@/api/api";

const { getCurrentUser } = api();

const routes: Array<RouteRecordRaw> = [
  {
    path: '/',
    redirect: '/login'
  },
  {
    path: '/login',
    name: 'login',
    component: () => import('@/views/auth/LoginView.vue'),
  },
  {
    path: '/tasks',
    name: 'tasks',
    component: () => import('@/views/HomeView.vue'),
    meta: {
      requiresAuth: true
    }
  }
]

const router = createRouter({
  history: createWebHistory(process.env.BASE_URL),
  routes
})

// Before each route transition:
router.beforeEach(async (to) => {
  // Check if the target route requires authentication.
  const requiresAuth = to.matched.some((record) => record.meta.requiresAuth)

  // If it requires authentication and no user is currently logged in,
redirect to '/login'.
  if (requiresAuth && !(await getCurrentUser())) {
    return '/login'
  }
  // Otherwise, proceed as normal.
})

export default router
```

Figure 42: Front-end Vue.js router

4.2.3 Navigation

Currently after logging in, users directed to '/tasks' will be met with a blank screen due to HomeScreen.vue – the main component for the tasks URL endpoint containing no elements. Following the results from the design survey, the decision was made to attempt to combine swipe-based page navigation with a navigation bar. This allows for increased flexibility as not only will it serve as a standard navigation bar which can be clicked for navigation, users will be able to swipe between pages.

In order to implement this feature, a library called Swiper.js has been utilised. Swiper.js is a JavaScript framework used for creating a touch-based slides system [23]. The following placeholder components were also created: Progress tab, Planner tab, Tasks tab, Timer tab and Settings tab. This allows a Swiper.js 'slide' component to be created for each placeholder:

```
<template>
  <ion-page>
    <ion-content scroll-y="false">
      <swiper :modules="modules"
              @swiper="setSwiperInstance"
              :slides-per-view="1"
              :space-between="10"
              class="swiper"
              @slideChange="onSlideChange"
              @slideChangeTransitionStart="destroyPanes"
              @slideChangeTransitionEnd="notDestroyPanes"
              @afterInit="notDestroyPanes">
        <swiper-slide>
          <progress-tab></progress-tab>
        </swiper-slide>
        <swiper-slide>
          <planner-tab @lock-slide="lockSlide"
                       @unlock-slide="unlockSlide"
                       :destroy="destroy"
                       v-if="activeIndex === 1"
                       :tab="activeIndex === 1">
          </planner-tab>
        </swiper-slide>
        <swiper-slide>
          <tasks-tab @view-timer-parent="viewTimer"
                     v-if="activeIndex === 2"
                     :tab="activeIndex === 2">
          </tasks-tab>
        </swiper-slide>
        <swiper-slide>
          <timer-tab :task="data.task"></timer-tab>
        </swiper-slide>
        <swiper-slide>
          <settings-tab @lock-slide="lockSlide"
                        @unlock-slide="unlockSlide">
          </settings-tab>
        </swiper-slide>
      </swiper>
    </ion-content>
  </ion-page>
</template>
```

Figure 43: Front-end Navigation

Implementing the navigation bar proved much more challenging. In order to achieve this, <span> tags were used in the header with custom CSS styling and classes to allow the name of the current page to be displayed at the top of the application. This CSS styling allowed for the implementation of the opacity and scrolling effect on the different page titles from the mock-ups. Finally, the @click attribute allows the user to click on each page title allowing for navigation bar style navigation:

```
<ion-header>
  <ion-toolbar style="min-height: 70px; --background: #ffffff; --border-
color: #ffffff;">
    <div class="swiper-pagination" slot="end">
      <span
          v-for="(slide, index) in tabs"
          :key="index"
          :class="[
          'custom-bullet',
          { 'swiper-pagination-bullet-active': index === activeIndex }
        ]"
          @click="handlePaginationClick(index)"
          :style="{
          transform: `translateX(${
            (index - activeIndex) * (
              index === activeIndex - 1 || index === activeIndex + 1
                ? 180
                : 180
            )
          }%)`,
          fontSize: index === activeIndex ? '2rem' : '1.2rem',
          opacity: index === activeIndex ? 1 : 0.5
        }"
      >
        {{ slide }}
      </span>
    </div>
  </ion-toolbar>
</ion-header>
```
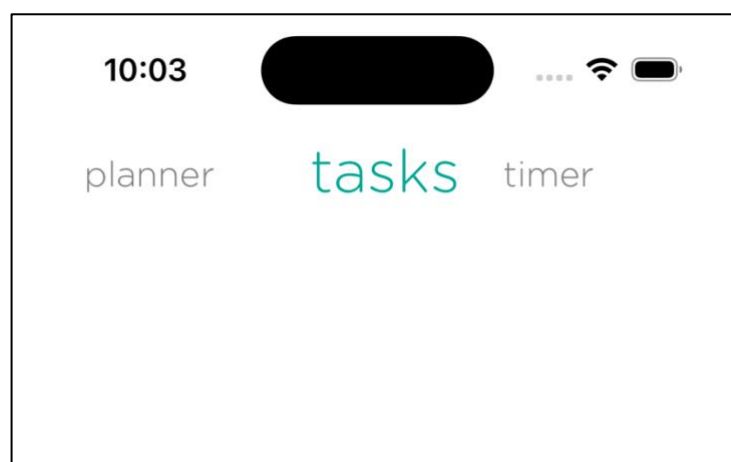
Figure 44: Front-end Navigation Bar

Figure 45: Navigation Bar Prototype

4.2.4 Tasks Page

Viewing Tasks

With navigation complete, development on implementing the user interface for viewing, managing and creating tasks could begin. In order to read the Firestore database, a library called Vuefire was used. Vuefire provides bindings for Vue.js allowing for real-time syncing between the front-end and the database [24]. This reduces the need to send GET requests to the back-end, reducing server load as it is not required to serve data. One benefit of using Vue.js is that it allows for components to be nested, also known as parent components and child components. This can reduce code duplication whilst also resulting in much more readable, tidy code. This was utilised for the tasks page to create a component called "TaskListItem". The task list item component is the template for an individual task as displayed on the tasks page. As the parent tasks page iterates through the tasks, each task will be displayed in accordance with the TaskListItem template:

```
<template>
  <ion-card>
    <ion-card-content>
      <ion-grid>
        <ion-row>
          <ion-col size="auto">
            <ion-avatar :class="priorityColour"></ion-avatar>
          </ion-col>
          <ion-col>
            <h1>{{ task.title }}</h1>
          </ion-col>
          <ion-col size="auto" class="right-align">
            <ion-card-title color="primary"
                            style="font-size: 25px">{{ duration }}</ion-card-title>
          </ion-col>
        </ion-row>
      </ion-grid>
    </ion-card-content>
  </ion-card>
</template>
```

Figure 46: Front-end TaskListItem

The TaskListItem template was constructed using an Ionic card component – a basic container element that can be customised using a variety of properties and CSS, along with an avatar component, typically used to display circular profile pictures. This avatar component worked perfectly for displaying the task priority colour as it can be easily changed by altering the background colour, resulting in a coloured circle:
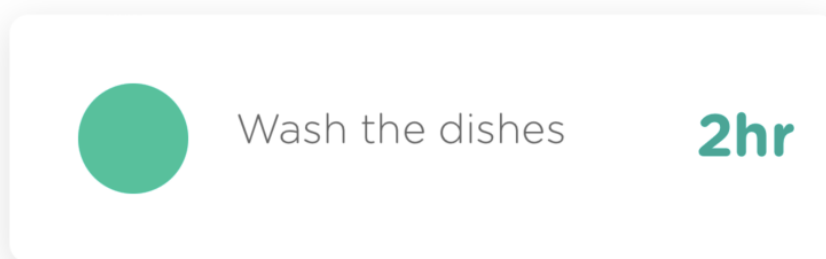

Figure 47: Front-end Individual Task

Then, to use TaskListItem on the tasks page, the tasks need to be iterated through with v-for, passing each task to the TaskListItem component:

```
<template>
  <ion-page>
    <ion-content :scroll-y="draggable">
      <div v-if="placeholder">
        <task-list-item v-for="task in tasks" :task="task"></task-list-item>
      </div>
      <ion-card v-else>
        <ion-card-content style="font-size: 1.4rem; padding-bottom: 30px">
          swipe up to create a task!
        </ion-card-content>
      </ion-card>
    </ion-content>
  </ion-page>
</template>
```

Figure 48: Front-end Tasks Page Template

In Figure 48, a basic placeholder card component was also created. This placeholder is displayed when a user has no tasks in order to avoid a blank screen which may be confusing to them:
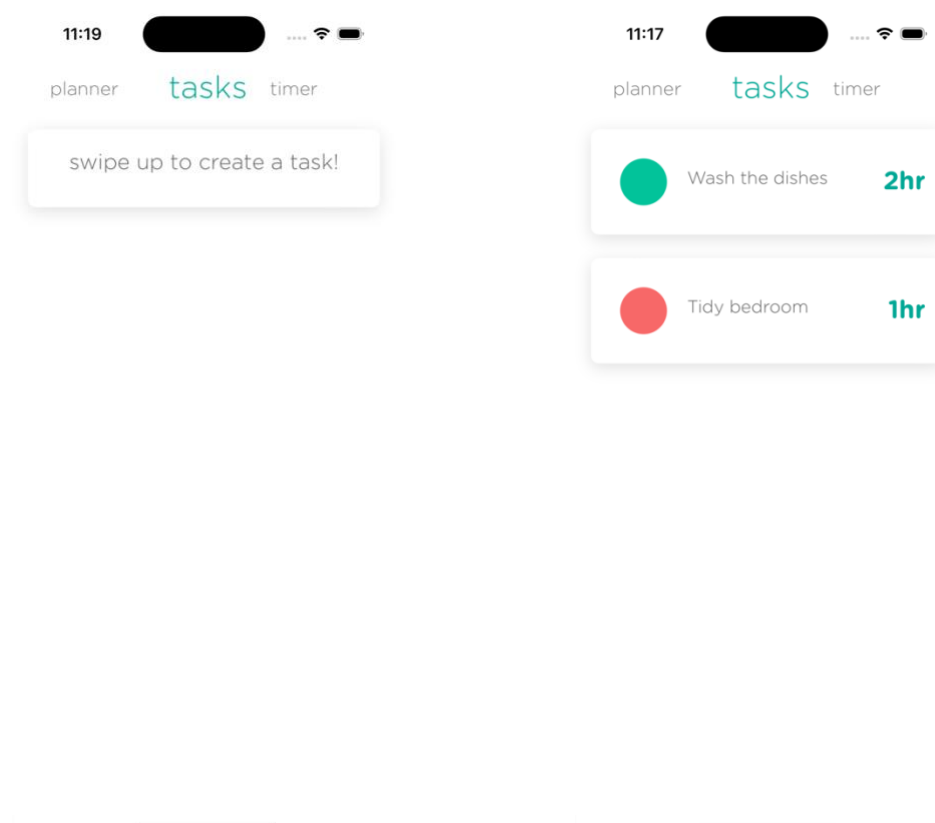
Figure 49: Front-end Tasks Page

Creating Tasks

In the mock-up designs in section 3.8, the tasks page utilises a draggable component which users can drag up from the bottom of the screen to reveal an input form to create a task. In order to implement this feature, a library called v-cupertino was used. V-cupertino provides a draggable pane component [25] that can be customised to the requirements of this project. The create task component also uses the input, button and picker Ionic components in order to create the input form:

```html
<template>
  <v-cupertino :drawerOptions="options" ref="bottomSheet">
    <div>
      <h1 style="padding-bottom: 40px"> create a task </h1>
    </div>
    <div>
      <ion-list>
        <ion-item style="min-width: 80%; padding-bottom: 20px;">
          <ion-input label="title" v-model="payload.title"></ion-input>
        </ion-item>

        <ion-item style="min-width: 80%; padding-bottom: 20px;">
          <ion-input label="priority" v-model="payload.priority"></ion-input>
          <ion-picker
              :isOpen="pPicker"
              :columns="priorityOptions"
              :buttons="priorityButtons"
              @did-dismiss="pPicker = false"
          ></ion-picker>
        </ion-item>

        <ion-item style="min-width: 80%; padding-bottom: 20px;">
          <ion-input label="duration" v-model="displayedDuration"></ion-input>
          <ion-picker
              :isOpen="dPicker"
              :columns="durationOptions"
              :buttons="durationButtons"
              @did-dismiss="dPicker = false"
          ></ion-picker>
        </ion-item>

        <ion-button id="createtask" :disabled="valid" style="min-width: 80%;"
@click="callCreateTask">Submit</ion-button>
        <ion-loading :is-open="loading" trigger="createtask" message="Creating
task..."> </ion-loading>
      </ion-list>
    </div>
  </v-cupertino>
</template>
```

Figure 50: Front-end Create Task Component Snippet

When the Submit button is clicked, the method referenced in Figure 37 from API.js to create a task is called, submitting the payload from the form to the back-end where it is handled. The form also has validation methods implemented, to prevent invalid tasks from being submitted. The current validation implementation only checks for blank fields and a task duration of 0, but can be further expanded in the future if required. When added to the tasks page, the create task component can be seen in Figure 51.
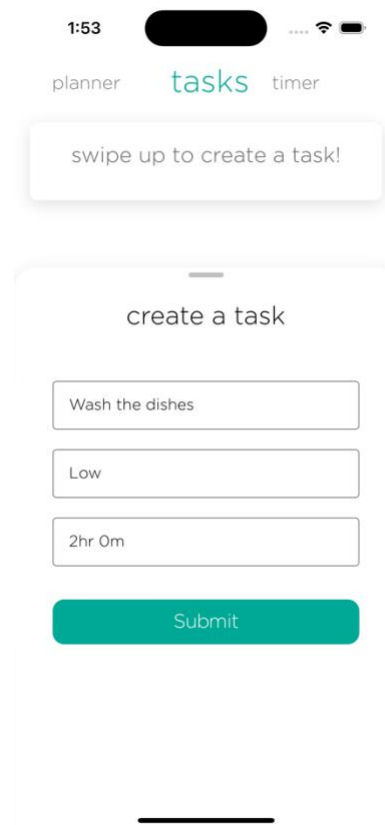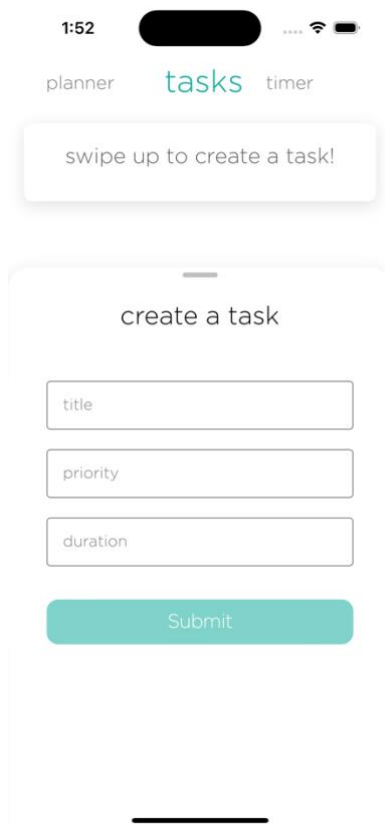
Figure 51: Front-end Create Task Dialogue

Deleting Tasks

In order to implement the functionality to delete a task, a UI component suitable for this needed to be used. This was done with an Ionic component known as a popover, which can be used to reveal a context menu to the user when tapping on a task. This menu can be used to include access to other functions or features should it become necessary. The code below for the popover component was added to the code for TaskListItem (Figure 46):

```
<ion-popover :trigger="task.id" trigger-action="click" :dismiss-on-select="true">
  <ion-list>
    <ion-item :button="true" @click="viewTimer(task)">Start Timer</ion-item>
    <ion-item :button="true" @click="callDeleteTask(task)">Delete Task</ion-item>
  </ion-list>
</ion-popover>
```

Figure 52: Front-end Context Menu Snippet

Upon implementation, the decision was made to also add the option to view the associated timer for that task. The method 'callDeleteTask()' is also contained within the API.js module and is identical in structure to the method used (Figure 37) to call the create task function on the back-end.
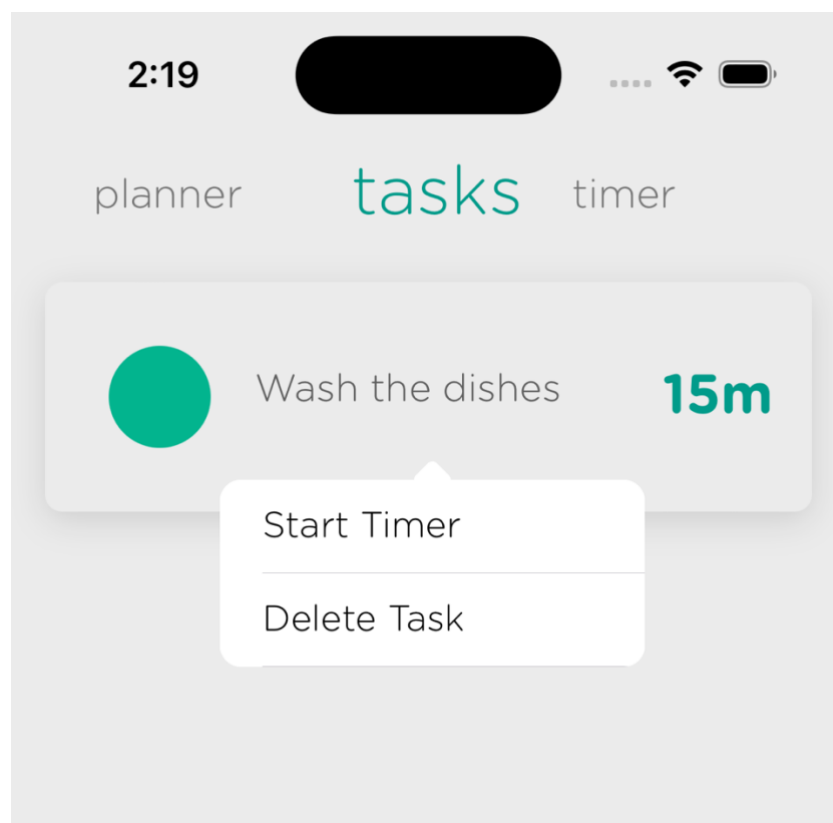

Figure 53: Front-end Context Menu

Completing a Task

The intention is for the user to be able to mark a task as completed quickly and easily, without having to access any menu. This was achieved by adding click functionality to the avatar circle, which then calls the API method from API.js, in order for the back-end mark a task as completed. Functionality was also added to include an animation with CSS when a task is marked as complete for improved user feedback and experience:

```
<ion-avatar :class="[priorityColour, { 'spin-once': spin }]"
            @click="markTaskCompleted(task)"
            ref="completedCircleRef"
            @mousedown="press"
            @mouseup="release"
            @onmouseleave="release">
  <ion-icon v-if="spin" style="color: #ffffff; font-size: 1.6rem"
:icon="checkmark"></ion-icon>
</ion-avatar>
```

Figure 54: Complete Task HTML Snippet

```
@keyframes spin {
  0% {
    transform: rotate(0deg);
  }
  100% {
    transform: rotate(360deg);
  }
}

.spin-once {
  animation: spin 0.5s ease-in-out;
}

@keyframes slide-off {
  0% {
    transform: translateY(0);
    opacity: 1;
  }
  100% {
    transform: translateY(-100%);
    opacity: 0;
  }
}

.slide-off {
  animation: slide-off 0.5s ease-in-out forwards;
}
```

Figure 55: Complete Task CSS Animation Snippet

The animation causes a white checkmark to appear within the priority circle of a task, followed by a spin animation after which the task slides upwards and out of view.

4.2.5 Timer Page

The timer page is designed to act as a simple focus timer, which can be expanded and worked upon in the future. Based on the design from Section 3.8, the timer component would be made up of a circular progress bar, text to display the remaining duration and buttons to start and pause the timer as well as add an extra five minutes if required. To create the circular progress bar, a Vue.js compatible library called vue-ellipse-progress [26], which contains customisable, animated circle progress bars was used.

```html
<template>
  <div>
    <ve-progress v-if="mounted"
                 thickness="10%"
                 color="#00A896"
                 :progress="progress"
                 :size="300"
                 animation="loop"
                 font-size="2rem">
      <span slot="legend-value">{{ remainingDuration }}</span>
    </ve-progress>

    <ion-alert
        :is-open="alert"
        header="Alert"
        message="Timer Complete!"
        :buttons="alertButtons"
    ></ion-alert>
  </div>

  <div class="button">
    <ion-button size="large" @click="addFive">add 5m</ion-button>
  </div>

  <div class="floatingbutton">
    <ion-fab>
      <ion-fab-button>
        <ion-icon :src="playPause" @click="toggleTimer"></ion-icon>
      </ion-fab-button>
    </ion-fab>
  </div>
</template>
```

Figure 56: Front-end Timer Page Snippet

During development, various techniques were experimented with to manage the timer's state and update the UI accordingly. Initially, an attempt was made to use the built-in JavaScript setInterval and clearInterval functions, but it was discovered that the "vue-timer-hook" library provided a more efficient and simple approach to handling timers in Vue.js.

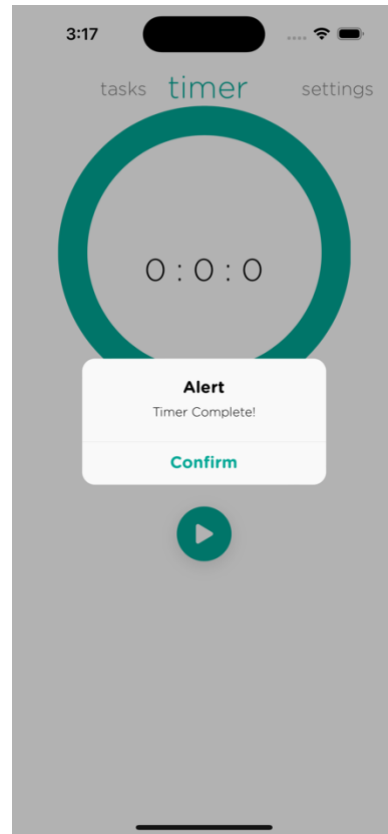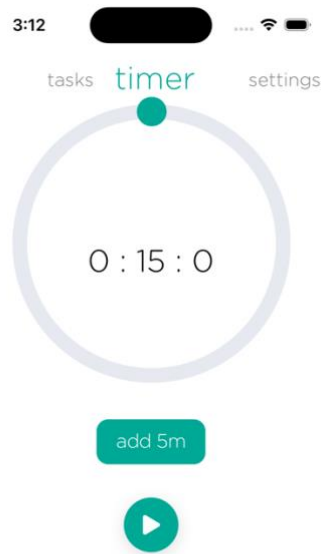Also included is the ability for an alert to appear when the timer is complete:

Figure 57: Front-end Timer Page

4.2.6 Planner Page

The planner page is intended to be used as a schedule / timetable to allow students with ADHD to plan their day. Based on the designs from Section 3.8, the planner page will utilise a draggable pane component – similar to the one used to create a task, from which users can drag any tasks that they have created onto the timetable to schedule them. Development first started with creating the draggable pane component using v-cupertino. In order to add the drag and drop functionality, a library called vue.draggable [27] was used. This adds a draggable container element which allows components within to be made draggable. The reason this library was chosen is because it is well supported due to being based upon Sortable.js and compatible with touch screens.

Creating the planner page proved more challenging than expected and a number of issues were encountered. The first limitation is that due to the way in which the timeline is calculated when a task is added, tasks can only be created in increments of 15 minutes. It is possible that this can be resolved, and more flexibility added in the future, but this will be discussed later in the evaluation. The draggable tasks feature was another area that required several iterations. Initially, the drag and drop feature experienced issues with tasks getting stuck or overlapping. To resolve this, a function called checkFull was implemented to check if the position that a task is being dropped on to already has a scheduled task or not. This function will return either true or false depending on this condition and will either allow or reject the drag operation.
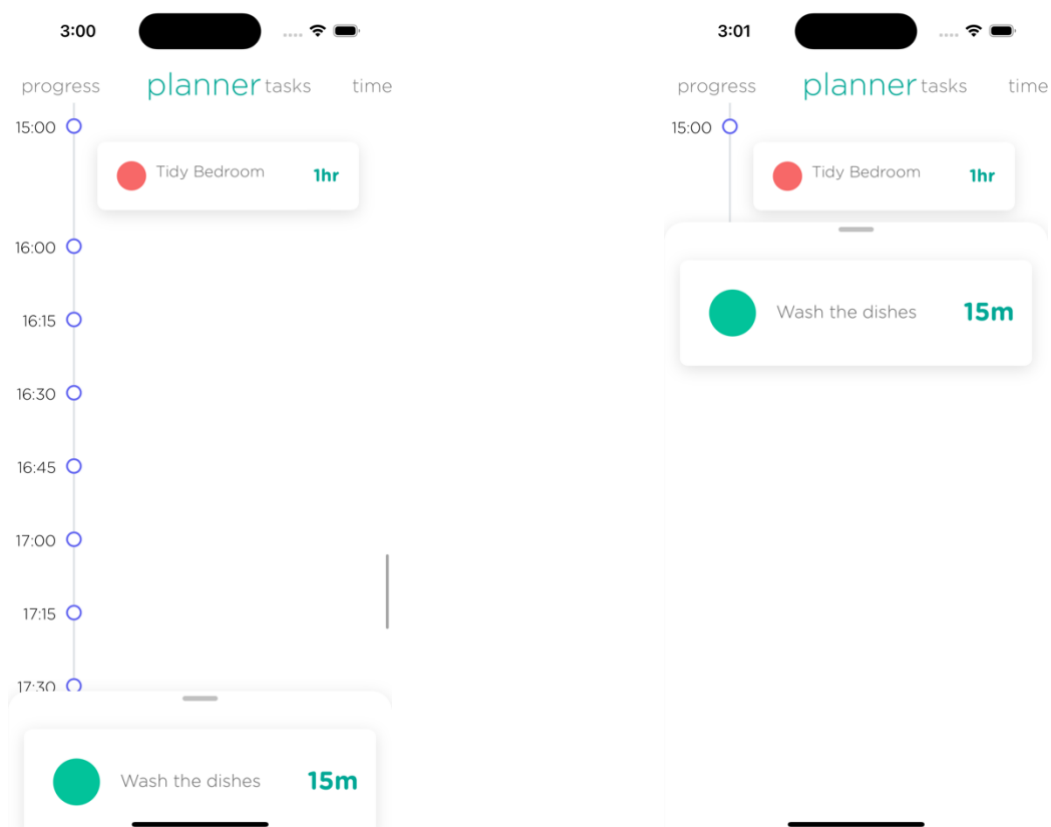

Figure 58: Front-end Planner Page

4.2.7 Progress Page

In order to implement <mark>FR06</mark> and using the mock-ups <mark>from Section 3.8,</mark> the progress page would be made up of two main components: a chart displaying the daily number of tasks completed in the current week, and a circular progress bar to display the number of hours spent completing tasks against a target set by the user. This feature is intended to provide the user with a breakdown of their progress and has room for expansion in the future.

The tasks chart proves an intuitive and visual representation of the user's number of tasks completed over a week. The component was developed using vue-chartjs, a library based on Chart.js, which provide versatile capabilities for creating attractive and responsive charts [28]. Each column represents a day of the week, and its height corresponds to the number of tasks completed on that day. This allows users to instantly assess their productivity levels and patterns throughout the week.

In addition to the tasks chart, a circular progress bar, commonly known as a radial progress indicator is used to display the number of hours worked along with a goal that can be set by the user. This indicator provides a user-friendly and visually compelling way of displaying progress towards a goal. The circular design is particularly effective for representing a ratio or percentage, as it intuitively implies a 'whole' or 'complete' state when the bar is fully filled.

Figure 59: Front-end Progress Page

5 Results and Evaluation

This section will assess the prototype using an HCI metric, in addition to examining its alignment with the functional and non-functional requirements.

5.1 Heuristic Evaluation

In order to meet the Non-Functional Requirement NFR01, which states the importance of following Jakob Nielsen's Usability Heuristics, this section will assess the graphical user interface (GUI) of the final product in light of Neilson's Usability Heuristics [8]. This comparison is essential to ensure that the application aligns with established HCI standards.

5.1.1 Visibility of System Status

The design should always keep users informed about what is going on, through appropriate feedback within a reasonable amount of time [8].

The principle of visibility of system status states that users should be continuously informed about what is happening within a reasonable time frame through feedback. In order to fulfil these criteria, certain design decisions were taken into account during development.
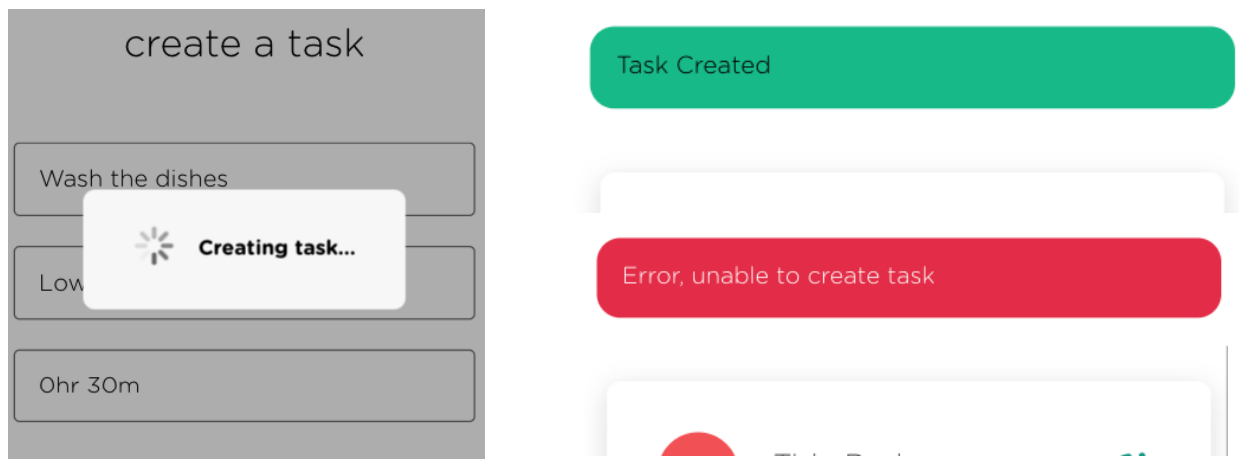

Figure 60: Loading Animations and Success/Error Toasts

Firstly, every time a user performs a major action, they are provided with clear feedback in the form of loading animations and success/error toasts (Figure 60). This immediate response to any action reduces the ambiguity surrounding the result of the action and aligns with user expectations.
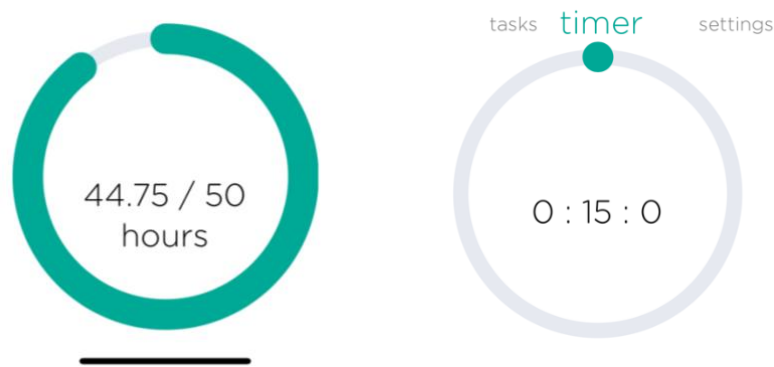
Figure 61: Progress Circles For Tracking Hours and Focus Timer

The application also employs progress circles in two key features - tracking hours spent on tasks and the focus timer (Figure 61). These visual indicators keep the user informed about their progress in real-time, helping them manage their tasks and time effectively.

Further, when the focus timer concludes, an alert is triggered, informing the user that their focus period has ended. This proactive communication prevents overwork and encourages breaks, which is especially helpful for students diagnosed with ADHD.

Haptic feedback is another crucial element of the application's design that ensures the user is aware of their interactions with the system using vibrations.
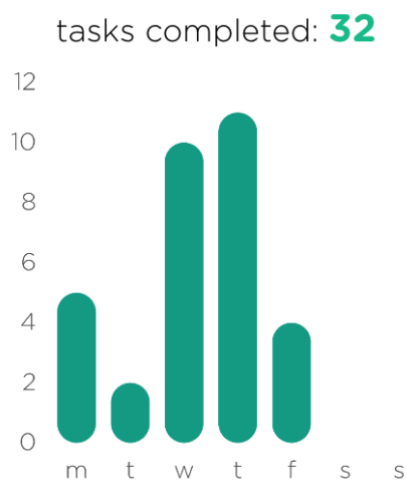

Figure 62: Weekly Task Completion Chart

Lastly, the weekly task completion chart (Figure 62) offers a visual representation of the user's productivity, keeping them aware of their performance trend. This feature allows users to evaluate their productivity over time, thereby motivating them to improve.

All these design elements align with the visibility of system status principle, ensuring the user is consistently informed about the application's status and their interaction results.

5.1.2 Match between System and the Real World

The design should speak the users' language. Use words, phrases, and concepts familiar to the user, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order [8].

Adherence to the principle of matching the system with the real world is evident in the application's design, ensuring that the interface communicates with the user in a familiar and clear manner.
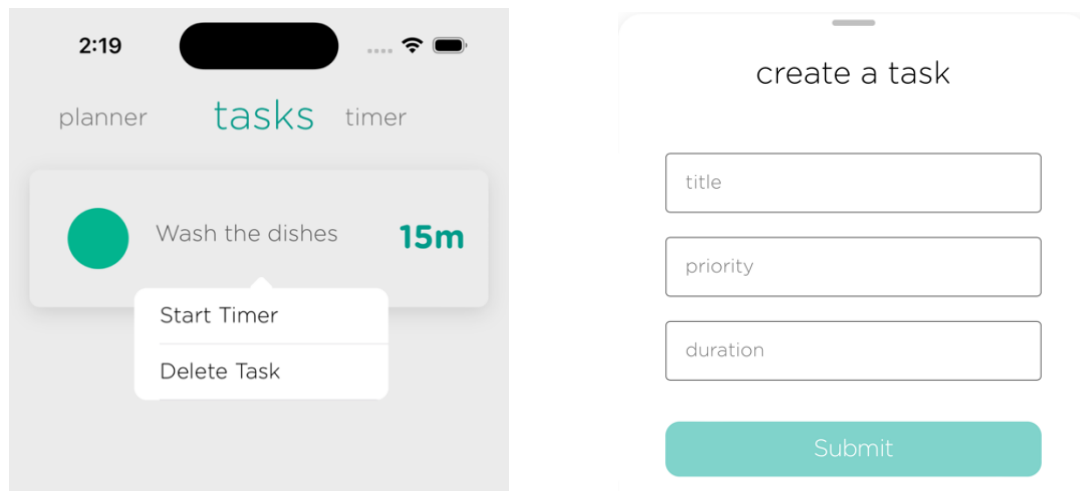

Figure 63: Front-end User Interface

The application uses language that aligns with the user's perspective rather than system-oriented terms. For instance, the application uses terminologies like "tasks," "planner," and "timer," which naturally correspond with the typical vocabulary of a student managing their tasks (Figure 63).
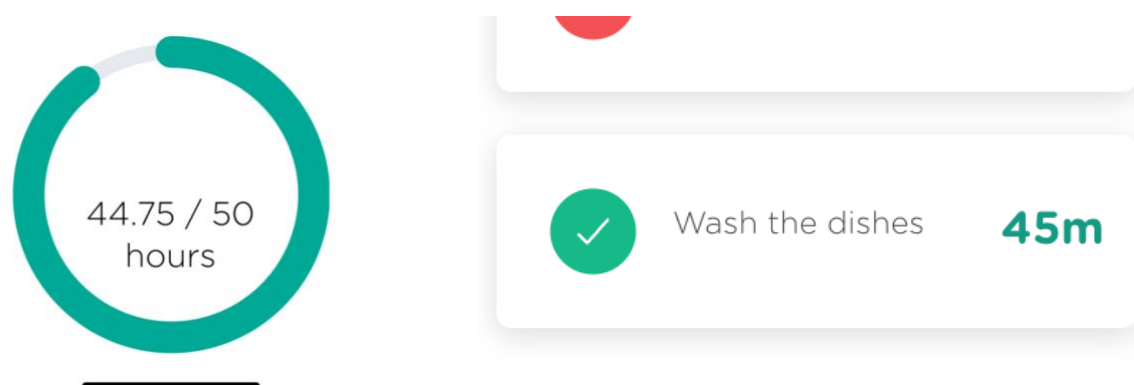

Figure 64: Use of Familiar Icons and Symbols

The application employs familiar icons and symbols that are prevalent in many other applications, such as a tick for task completion and a circular progress bar for time tracking (Figure 64). These recognisable visual cues encourage intuitive interaction, making it easier for users to navigate and use the application.

Overall, the application's design follows the real-world conventions, which enhances the user's comprehension and ease of use. The familiar elements help to minimise the learning curve, allowing the user to efficiently operate the application from the very beginning.

5.1.3 User Control and Freedom

Users often perform actions by mistake. They need a clearly marked "emergency exit" to leave the unwanted action without having to go through an extended process [8].

One of the crucial aspects of Jakob Nielsen's usability principles is to provide users with a sense of control and freedom while navigating the system. While the application excels in many usability principles, it does fall short in fully aligning with this specific principle.

Figure 65: Lack of Cancel When Creating Tasks

Though the application provides a user-friendly interface, it lacks certain elements that would empower the user with greater control. A noticeable omission is the absence of an "cancel" feature, which prevents the users from easily cancelling their actions if mistakes are made while entering a task or wish to block the action. (Figure 65).

Furthermore, the inability to customise the duration of the focus timer or break intervals constrains the user's freedom to adjust the system according to their personal preferences and requirements. This rigid structure may not cater to the diverse needs of all ADHD students, who might benefit from personalized focus periods and breaks (this can be seen in Figure 57).

Unfortunately, the application's design does not entirely support the user control and freedom principle. Future iterations of the application could benefit from incorporating features such as "cancel" or "undo" buttons, that allow greater flexibility and control to cater to individual user preferences.

5.1.4 Consistency and Standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform and industry conventions [8].

The principle of consistency and standards suggests that applications should follow established rules and incorporate consistent elements to prevent user confusion. The application design effectively demonstrates this principle through various means.
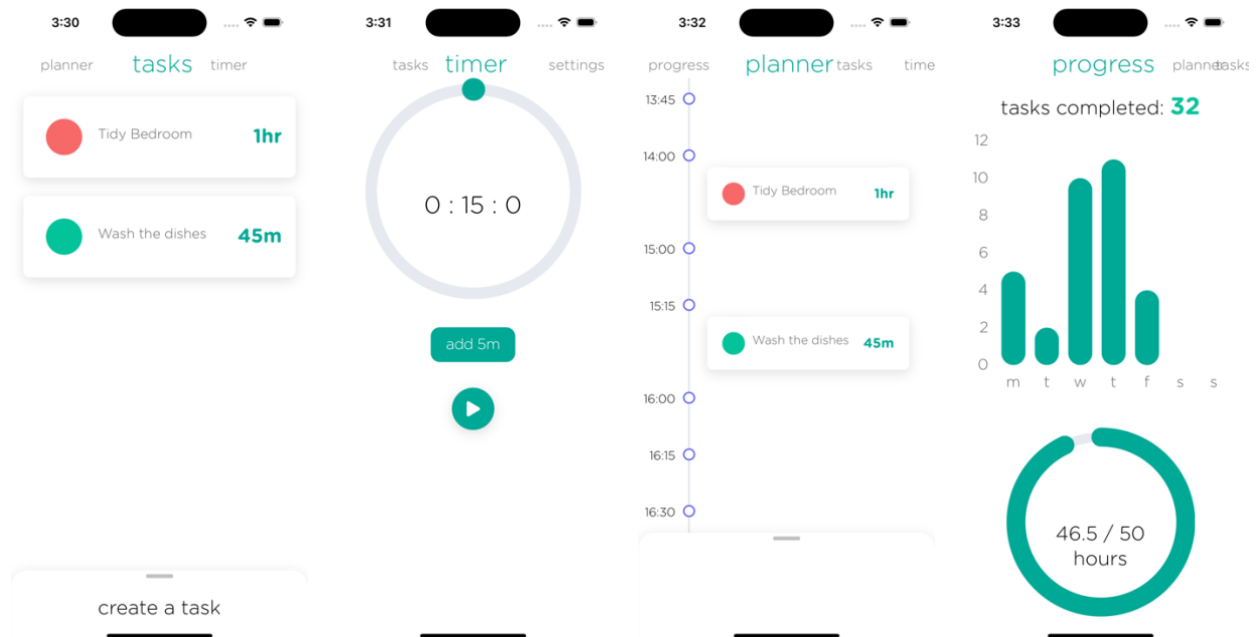

Figure 66: Consistent Design Throughout the Application

All pages within the application employ a uniform design pattern. For instance, the Tasks, Timer, Planner, Progress, and Settings pages all feature consistent typography and colour scheme, facilitating a smooth and predictable navigation experience (Figure 66).
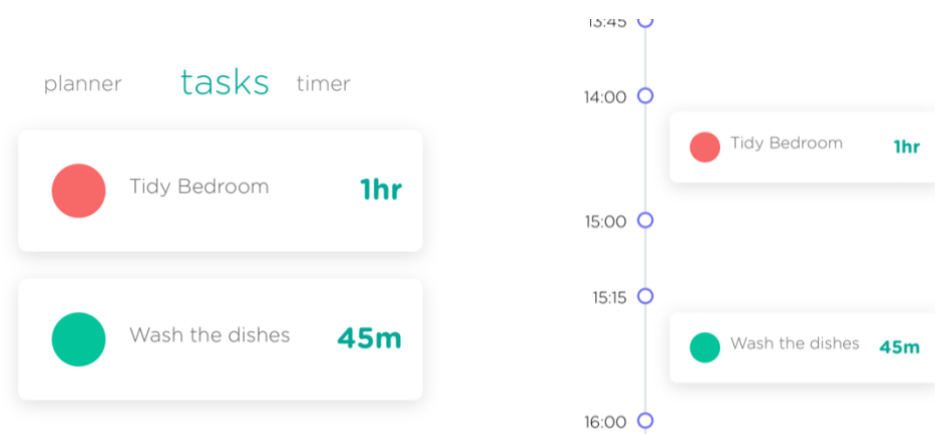

Figure 67: Task Design

The application also upholds consistency in interactive components. The card design used in the tasks, for instance, uses a priority circle, title, and duration. This design is used on the tasks page and carries over to the Planner page, where tasks from the bottom sheet can be dragged and dropped onto the timeline, maintaining familiarity and coherence (Figure 67).

Overall, the application abides by the principle of consistency and standards largely, ensuring a user-friendly and intuitive experience.

5.1.5 Error Prevention

Good error messages are important, but the best designs carefully prevent problems from occurring in the first place. Either eliminate error-prone conditions, or check for them and present users with a confirmation option before they commit to the action [8].

The principle of error prevention stresses the importance of system design that mitigates the risk of user errors. The application incorporates a number of features to prevent errors, although some areas could be further enhanced.

Figure 68: Front-end Create Task Form

The application's design makes use of a simple and intuitive interface, reducing the potential for user error. For example, the task creation form on the Tasks page explicitly asks for the title, priority, and duration of a task, guiding the user to input necessary information (Figure 68).

However, the application could improve in providing input validation to prevent errors. For instance, it could add form validation to ensure the user doesn't leave any fields empty or input inappropriate values when creating a task.

The timer component on the Timer page, with its ability to add five minutes and a play/pause button, simplifies interaction and avoids errors related to time management. And on the Planner page, tasks can be moved and adjusted on the timeline, allowing for corrections and preventing scheduling errors (see Figure 66).

Despite these features, the application could enhance error prevention through incorporating features like confirmations for destructive actions. For example, asking for user confirmation before deleting a task could prevent accidental removals.

In conclusion, while the application incorporates certain elements to prevent errors, it could potentially be improved in areas such as input validation and confirmations for destructive actions to fully align with the error prevention principle.

5.1.6   Recognition Rather Than Recall

Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design (e.g. field labels or menu items) should be visible or easily retrievable when needed [8].

The principle of recognition rather than recall emphasises the importance of making objects, actions, and options visible to users. By employing this principle, the application supports users by not burdening them with remembering information from one part of the interface to another.

The tasks page is a prime example of this principle in action, with each task item presented in a card design that includes the priority circle, title, and duration (see Figure 67). This layout allows users to recognise each task's status without having to recall specific information.

Figure 69: Timer Play/Pause Button

The timer page aids recognition through the use of universally accepted symbols for the play/pause button (Figure 69). These icons, familiar to most users, enable instant recognition, simplifying interaction.
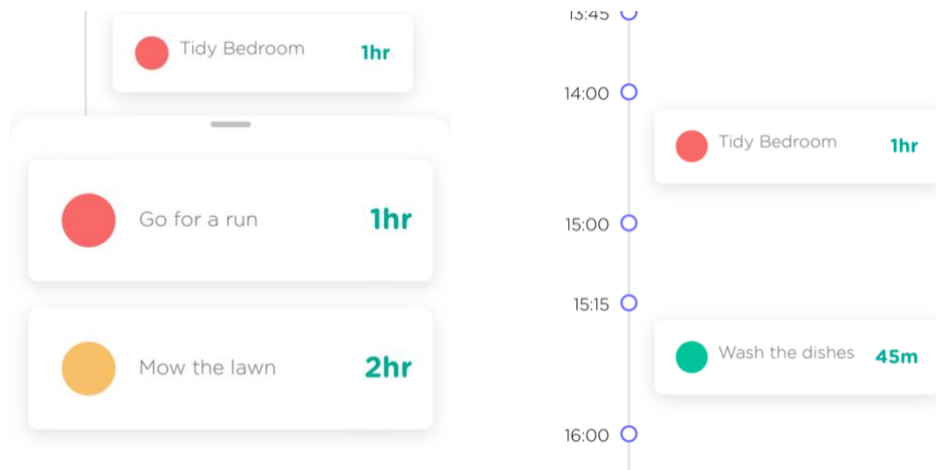
Figure 70: Planner Page

Similarly, the planner page displays tasks from the tasks page in a draggable bottom sheet, helping users recognise and schedule their tasks (Figure 70). The timeline representation with time intervals supports users in recognising their schedule without needing to recall task timings.

In summary, the application largely employs the principle of recognition rather than recall, offering an intuitive, user-friendly interface that reduces the cognitive load on users.

5.1.7 Flexibility and Efficiency of Use

Shortcuts — hidden from novice users — may speed up the interaction for the expert user so that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions [8].

The principle of flexibility and efficiency of use suggests that an application should cater to both novice and experienced users by allowing customisation and providing accelerators to speed up interaction. The application demonstrates this principle to some extent, but there is room for improvement.

The Tasks page offers an intuitive way to create tasks and mark them as complete, catering to both novice and experienced users (see Figure 63). Yet, the application could provide more flexibility by allowing users to edit tasks after creation, which could enhance efficiency for more advanced users.
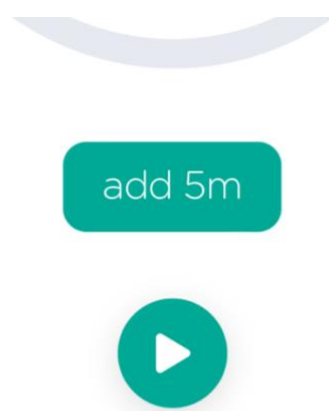

Figure 71: Add Five Minutes Timer Button

The Timer page includes a button to add 5 minutes to the timer, enabling users to extend their focus time as needed (Figure 71). However, offering more customisation options for the duration of the timer and the break intervals could better accommodate diverse user preferences and habits.

On the Planner page, the ability to drag and drop tasks onto the timeline and adjust their duration or timing provides flexibility and ease of use (Figure 70). Still, including a feature for recurring tasks or templates for common schedules could significantly improve efficiency for frequent users.
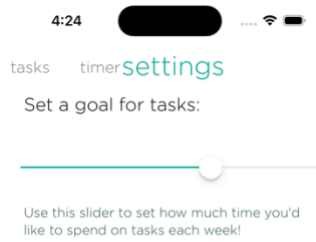
Figure 72: Front-end Settings Page

The Settings page allows users to set a goal for the number of hours they aim to spend completing tasks each week, offering a degree of customisation (Figure 72). However, providing more options for personalisation could enhance the app's adaptability to individual user needs.

In summary, while the application incorporates certain elements of flexibility and efficiency of use, greater personalisation and advanced features could be incorporated to cater to the needs of both novice and more experienced users, better aligning with this usability principle.

5.1.8 Aesthetic and Minimalist Design

Interfaces should not contain information that is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility [8]*.*

The principle of aesthetic and minimalist design promotes simplicity, where unneeded information is reduced, allowing relevant content to stand out. The application demonstrates this principle quite effectively in its design.
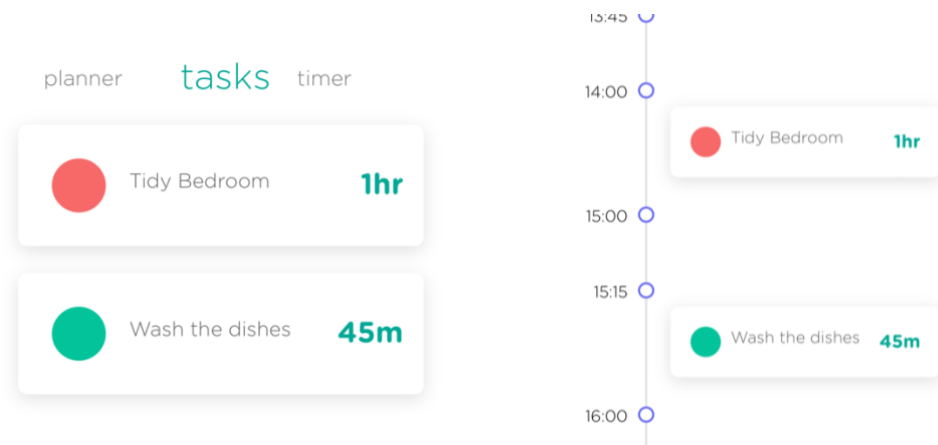

Figure 73: Task Design

The Tasks page illustrates this principle through a clean card design, with each task card featuring only essential information: a colour-coded priority circle, the task title, and the task duration (Figure 73). This minimalistic approach makes it easy for users to quickly scan and understand their tasks.
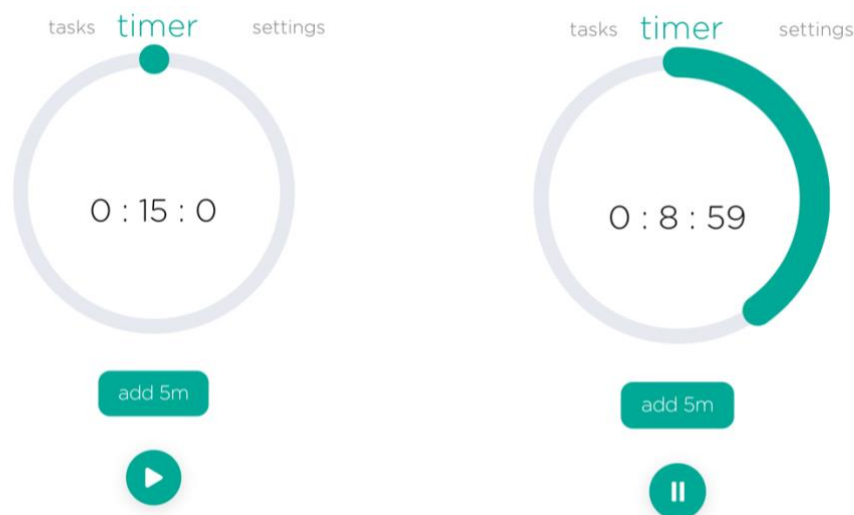

Figure 74: Front-end Timer Page

The Timer page illustrates simplicity with a straightforward progress bar and essential buttons for controlling the timer (Figure 74). The absence of unnecessary information or options reduces distractions and makes the page efficient for its purpose.
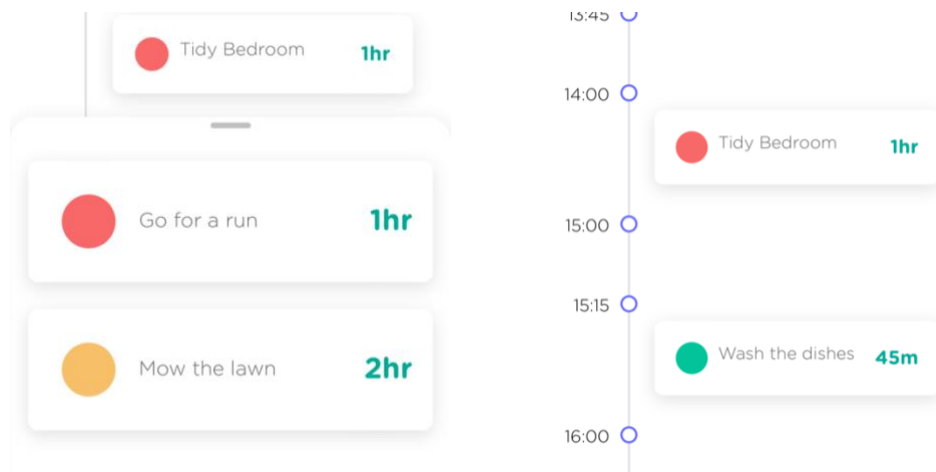
Figure 75: Front-end Planner Page

The Planner page also adheres to a minimalist design, with a clear timeline and drag-and-drop tasks from a bottom sheet. This approach simplifies scheduling tasks without clutter (Figure 75).
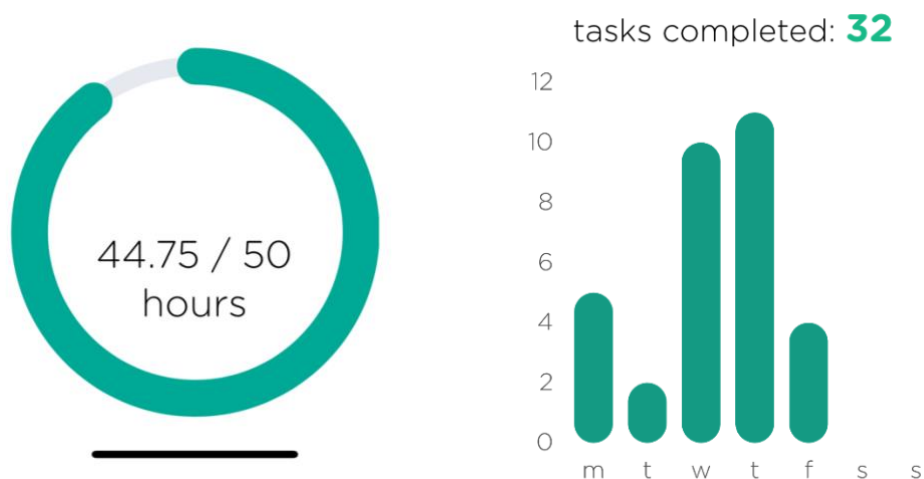

Figure 76: Front-end Progress Page Components

Similarly, the Progress page employs a straightforward layout with a weekly task completion chart and a circular progress bar to represent hours spent on tasks, providing a quick overview of the user's progress without excess information (Figure 76).

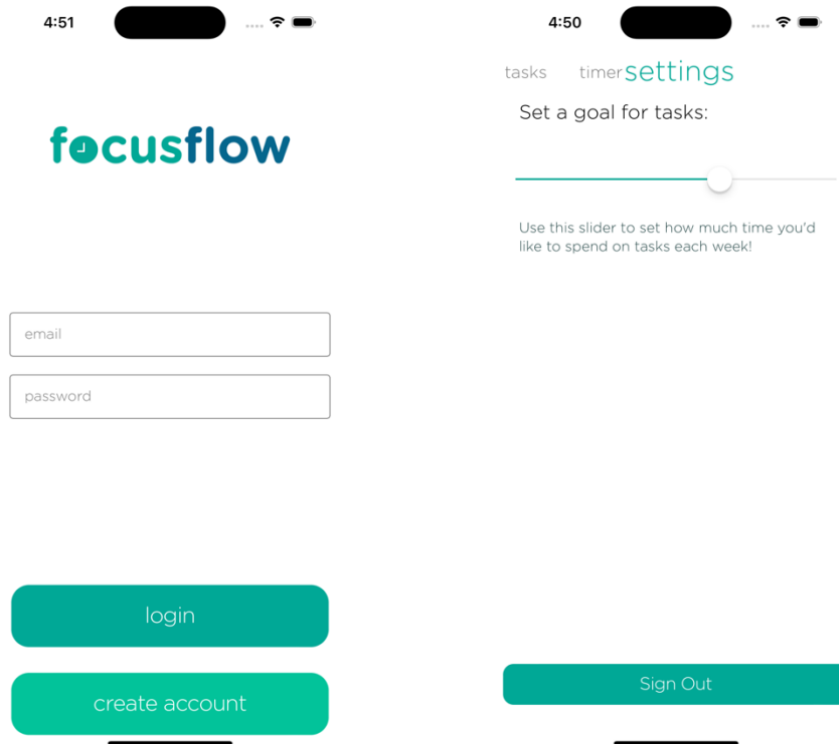*Figure 77: Front-end Login and Settings Page*

Lastly, the application's login and settings pages contain only the necessary fields and options, making them easy to navigate and use (Figure 77).

In essence, the application aligns well with the principle of aesthetic and minimalist design, presenting only necessary information and functionality while maintaining an attractive and easy-to-navigate interface.

5.1.9 Help users recognise, diagnose, and recover from errors

Error messages should be expressed in plain language (no error codes), precisely indicate the problem, and constructively suggest a solution [8].

The principle of helping users recognise, diagnose, and recover from errors advocates for clear and simple error messages that guide users towards solutions. It appears that this area is where the application could improve.
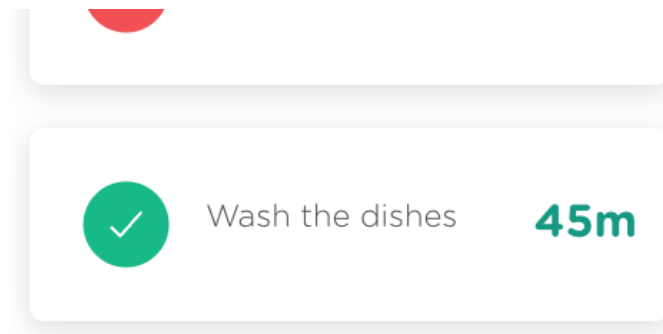

Figure 78: Task Completion Checkmark Animation

The application includes animations and success/error toasts to communicate the system status. For instance, when a task is marked as complete, an animation plays, and the task moves upwards off the screen, clearly indicating the successful completion of the task (Figure 78).

However, the application seems to lack clear and explanatory error messages when things go wrong. If an error occurs, such as failure to load data or unsuccessful task creation, the application could be more explicit in communicating these errors to the user, along with steps for recovery.

Figure 79: Empty Create Task Form With Button Disabled

In the current design, if users make an error, such as leaving a field empty while creating a task or entering an incorrect login credential, the application does not provide an explicit error message guiding them to rectify the problem (Figure 79).

The application could benefit from incorporating detailed and easy-to-understand error messages that not only indicate that an error has occurred but also guide users on how to correct the issue.

In conclusion, while the application does well in various aspects of usability, further improvement in helping users recognise, diagnose, and recover from errors would enhance its overall usability and user experience.

5.1.10 Help and Documentation

It's best if the system doesn't need any additional explanation. However, it may be necessary to provide documentation to help users understand how to complete their tasks [8].

The last of Nielsen's usability principles suggests that it can be beneficial to provide help and documentation. While users should not have to resort to a help manual to use the system, having easily accessible, searchable, and beneficial documentation can be of significant assistance when users are stuck or confused.

In its current version, it seems the application could be improved to strengthen its alignment with this principle. Currently, the application does not provide an in-built help or documentation system to assist users when they encounter issues. Due to limited time, this was a feature that had to be missed however, it could be easily added in the future.

This lack of available help could be fixed with the introduction of a dedicated 'Help' or 'FAQ' section on the settings page, where users could find answers to commonly asked questions and documentation of the application's features and functions. This section could include helpful tips for getting the most out of the application's capabilities, troubleshooting advice, and guidelines for managing tasks more effectively.

Furthermore, incorporating in-context help or tips could be advantageous. For instance, the application could offer tooltips when users press and hold a button or feature, explaining its function. This assistance could reduce confusion and enhance usability.

In conclusion, while the application demonstrates many strengths in its design and functionality, the addition of easily accessible help and documentation could further improve the user experience and adherence to Nielsen's usability principles.

5.2 User Evaluation

The User Evaluation section presents a brief analysis of feedback gathered from a group of users who have been previously diagnosed with ADHD. These users were recruited through an Instagram story post seeking students diagnosed with ADHD who were willing to participate in a brief user study survey. The purpose of this evaluation was to assess the usability of the application and identify areas for improvement. The evaluation was structured around a series of questions designed to gather insights into various aspects of the user experience, including learnability, feedback and confirmation, consistency, navigability, error handling, efficiency, and simplicity. Each question is followed by a rationale explaining its purpose and an evaluation of the application based on the users' responses.

5.2.1 Question 1: How easy was it for you to learn how to use the app?

Answer options: Extremely easy, Somewhat easy, Neutral, Somewhat Difficult and Extremely difficult.

🔵 Extremely easy	1	
🟠 Somewhat easy	2	
🟢 Neutral	1	
🔴 Somewhat difficult	1	
🟣 Extremely difficult	0	

Figure 80: User Evaluation Question 1 Results

<u>Rationale</u>
This question was designed to assess the app's learnability, which is a crucial aspect of usability. A user-friendly app should be easy to learn and use, even for first-time users.

<u>Evaluation</u>
The majority of users found the app either extremely easy or somewhat easy to learn, indicating that the app's design and interface are intuitive and user-friendly. However, one user found it somewhat difficult, suggesting that there may be room for improvement in the app's onboarding process or in-app guidance.

5.2.2   Question 2: Did the app provide appropriate feedback and confirmation for your actions?

Answer options: Yes, No, Sometimes.

● Yes	4	
● No	0	
● Sometimes	1	

Figure 81: User Evaluation Question 2 Results

Rationale
Feedback and confirmation are essential for a positive user experience. They reassure users that their actions have been successful or alert them to any errors.

Evaluation
Most users felt that the app provided appropriate feedback and confirmation for their actions. This suggests that the app effectively communicates with users about the results of their actions, contributing to a smooth and intuitive user experience.

5.2.3   Question 3: Were there any inconsistencies in the app's interface or design that caused confusion?

Answer options: Yes, No.

● Yes	0	
● No	5	

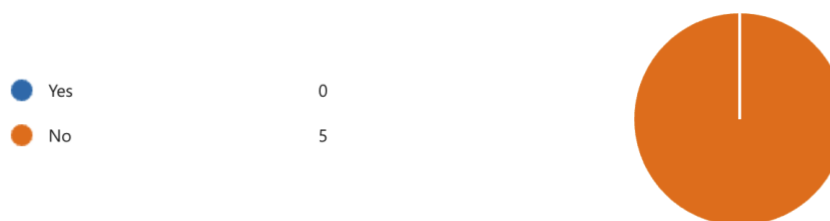*Figure 82: User Evaluation Question 3 Results*

Rationale
Consistency in an app's interface and design can significantly enhance usability by making the app easier to learn and use.

Evaluation
All users reported no inconsistencies in the app's interface or design, indicating that the app maintains a consistent look and feel throughout, which can help users learn how to use it more quickly and easily.

5.2.4 Question 4: Were there any instances where you had difficulty locating specific features or options within the app?

Answer options: No, Other (please specify).

●	No	2
●	Other	3

Figure 83: User Evaluation Question 4 Results

Rationale

This question was designed to assess the app's navigability. Users should be able to easily locate and access all features and options within the app.

Evaluation

While some users had no difficulty locating specific features or options, others reported minor issues, particularly with the task creation and planner features. This feedback suggests that these features may need to be made more intuitive or that additional guidance may be needed to help users understand how to use them.

5.2.5 Question 5: Did the app provide appropriate error handling and help you recover from any mistakes?

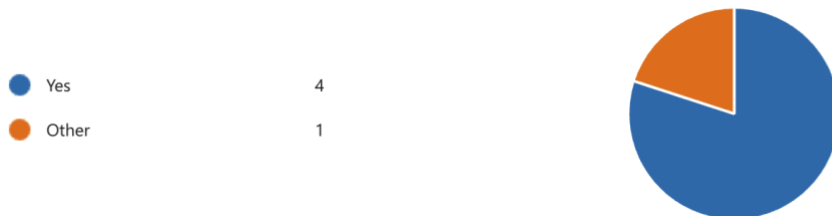Answer options: Yes, Other (please specify).

●	Yes	4
●	Other	1

Figure 84: User Evaluation Question 5 Results

Rationale

Good error handling is crucial for a positive user experience. The app should not only prevent errors where possible but also provide clear and helpful error messages when errors do occur.

Evaluation

Most users felt that the app provided appropriate error handling. However, one user reported an issue with the task creation form validation, indicating that the app may need to improve its error handling in this area.

5.2.6 Question 6: Were the interaction patterns (e.g., swiping, tapping, dragging) consistent and predictable throughout the app?

Answer options: Yes, Other (please specify).

| Yes | 4 |
| Other | 1 |

Figure 85: User Evaluation Question 6 Results

Rationale
Consistent and predictable interaction patterns can enhance usability by making the app easier to learn and use.

Evaluation
Most users found the app's interaction patterns consistent and predictable. However, one user reported occasional glitches when dragging tasks on the planner screen, suggesting that this feature may need to be refined to ensure a smoother user experience.

5.2.7 Question 7: How quickly were you able to accomplish tasks using the app?

Answer options: Extremely quickly, Somewhat quickly, Neutral, Somewhat slowly, Extremely slowly.

Extremely quickly	1
Somewhat quickly	4
Neutral	0
Somewhat slowly	0
Extremely slowly	0

Figure 86: User Evaluation Question 7 Results

Rationale
This question was designed to assess the app's efficiency. Users should be able to accomplish tasks quickly and easily using the app.

Evaluation
All users were able to accomplish tasks either extremely quickly or somewhat quickly, indicating that the app is efficient and allows users to complete tasks with minimal effort.

5.2.8 Question 8: Were there any instances where you felt overwhelmed by the amount of information or options presented?
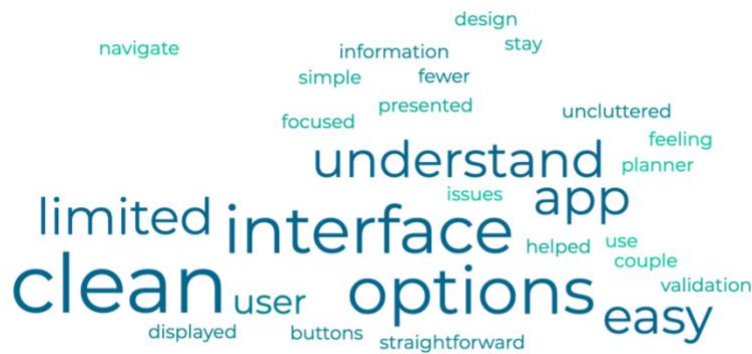

Figure 87: User Evaluation Question 8 Word Cloud

Answer options: Open text answer.

<u>Rationale</u>
This question was designed to assess the app's simplicity and clarity. Users should not feel overwhelmed by the amount of information or options presented in the app.

<u>Evaluation</u>
All users reported that they did not feel overwhelmed by the amount of information or options presented in the app. This feedback suggests that the app's design is clear and uncluttered, which can help users focus and avoid feeling overwhelmed.

5.2.9 Question 9: Any other feedback you could provide on your experience using the application?

Answer options: Open text answer.

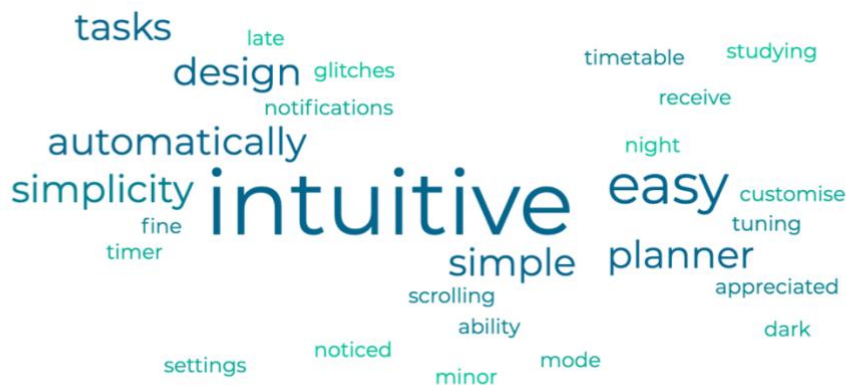*Figure 88: User Evaluation Question 9 Word Cloud*

<u>Rationale</u>
This open-ended question was included to capture any additional feedback or observations from users that may not have been covered by the previous questions.

<u>Evaluation</u>
Users provided a range of feedback, highlighting both the strengths and areas for improvement in the app. Positive feedback included the app's simplicity, easy navigation, and clean user interface. Areas for improvement included the need for more flexibility with task durations, improvements to the planner feature, and the addition of features like reminder notifications, task editing, and timer customisation.

5.2.10 Summary of User Evaluation

The user evaluation provided valuable insights into the app's usability and user experience. Overall, users found the app easy to learn and use, with consistent and predictable interaction patterns and appropriate feedback and confirmation for their actions. The app's simplicity and uncluttered design were also appreciated, with users reporting that they did not feel overwhelmed by the amount of information or options presented.

However, the evaluation also highlighted areas for improvement. Some users reported minor issues with locating specific features or options and with the task creation and planner features. Feedback also suggested that the app could benefit from more flexibility with task durations, improvements to the planner feature, and the addition of features like reminder notifications, task editing, and timer customisation.

This feedback will be invaluable in guiding future improvements to the app, with the aim of enhancing its usability and user experience for individuals with ADHD. By addressing the identified issues and incorporating users' suggestions, the app can better meet the needs and preferences of its target users.

6 Conclusion

7 References

[1] P. Song, M. Zha, Q. Yang, Y. Zhang, X. Li, I. Rudan and , "The prevalence of adult attention-deficit hyperactivity disorder: A global systematic review and meta-analysis," 11 February 2021. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7916320/. [Accessed February 2023].

[2] NICE, "Attention deficit hyperactivity disorder: How common is it?," [Online]. Available: https://cks.nice.org.uk/topics/attention-deficit-hyperactivity-disorder/background-information/prevalence/. [Accessed 7 February 2023].

[3] K. Spiel, E. Hornecker, R. M. Williams and J. Good, "ADHD and Technology Research – Investigated by Neurodivergent Readers," 29 April 2022. [Online]. Available: https://doi.org/10.1145/3491102.3517592. [Accessed 14 February 2023].

[4] unorderly GmbH, "Structured - Daily Planner," 9 April 2020. [Online]. Available: https://structured.app/. [Accessed 9 February 2023].

[5] NHS, "Attention deficit hyperactivity disorder (ADHD) - Symptoms," 20 October 2017. [Online]. Available: https://www.nhs.uk/conditions/attention-deficit-hyperactivity-disorder-adhd/symptoms/. [Accessed 7 February 2023].

[6] B. Desrochers, E. Tuson and J. Magee, "Evaluation of Why Individuals with ADHD Struggle to Find Effective Digital Time Management Tools," 24 October 2019. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3308561.3354622. [Accessed 14 February 2023].

[7] M. Olan, H. Medrano, D. Lopez and L. Escobedo, "SmarTasko: Supporting short and spontaneous activities of daily living of ADHD individuals," 26 December 2022. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3565494.3565500. [Accessed 14 February 2023].

[8] J. Nielsen, "10 Usability Heuristics for User Interface Design," 24 April 1994. [Online]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/. [Accessed 23 February 2023].

[9] Todoist, "Todoist - A To-Do List to Organize Your Work & Life," 28 January 2007. [Online]. Available: https://todoist.com/. [Accessed 10 February 2023].

[10] Tiimo, "Tiimo - Visual Daily Planner," 8 November 2019. [Online]. Available: https://www.tiimoapp.com/. [Accessed 10 February 2023].

[11] D. T. F. Burgess, "Guide to the Design of Questionnaires," May 2001. [Online]. Available: https://nats-www.informatik.uni-hamburg.de/pub/User/InterculturalCommunication/top2.pdf. [Accessed 8 March 2023].

[12] J. Nielsen, Usability Engineering, San Francisco: Morgan Kaufmann, 1993.

[13] D. Norman, The Design of Everyday Things: Revised and Expanded Edition, New York City: Basic Books, 2013.

[14] D. Norman, Emotional Design: Why We Love (or Hate) Everyday Things, New York City: Basic Books, 2004.

[15] D. Norman, User Centered System Design, CRC Press, 1986.

[16] Vue.js, "Introduction," [Online]. Available: https://v3.vuejs.org/guide/introduction.html. [Accessed 31 03 2023].

[17] Node.js, "About," [Online]. Available: https://nodejs.org/en/about. [Accessed 05 04 2023].

[18] E. Wong, "Shneiderman's Eight Golden Rules Will Help You Design Better Interfaces," 31 07 2020. [Online]. Available: https://www.interaction-design.org/literature/article/shneiderman-s-eight-golden-rules-will- help-you-design-better-interfaces. [Accessed 2023 04 02].

[19] J. Hannah, "An Introduction to Color Theory and Color Palettes," 06 12 2022. [Online]. Available: https://careerfoundry.com/en/blog/ui-design/introduction-to-color-theory-and-color-palettes/. [Accessed 03 04 2023].

[20] MDN Contributors, "try...catch," 21 02 2023. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/try...catch. [Accessed 10 04 2023].

[21] MDN Contributors, "HTTP Authentication," 10 04 2023. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication. [Accessed 10 04 2023].

[22] Vue.js, "Vue Router," 2021. [Online]. Available: https://router.vuejs.org/. [Accessed 15 04 2023].

[23] Swiper.js, "Swiper: The Most Modern Mobile Touch Slider," 11 Februrary 2015. [Online]. Available: https://swiperjs.com/. [Accessed 16 April 2023].

[24] E. S. M. Morote, "Vuefire," 13 April 2016. [Online]. Available: https://vuefire.vuejs.org/. [Accessed 17 April 2023].

[25] R. Mejia, "V-Cupertino," 19 November 2020. [Online]. Available: https://github.com/Devrax/v-cupertino. [Accessed 18 April 2023].

[26] S. Atamantschuk, "Vue-Ellipse-Progress," 17 June 2019. [Online]. Available: https://github.com/setaman/vue-ellipse-progress/tree/v2-dev. [Accessed 20 April 2023].

[27] D. Desmaisons, "Vue.Draggable," 30 June 2016. [Online]. Available: https://github.com/SortableJS/vue.draggable.next. [Accessed 20 April 2023].

[28] J. Juszczak, "vue-chartjs," 27 July 2016. [Online]. Available: https://vue-chartjs.org/. [Accessed 22 April 2023].

[29] R. Sherman, Business Intelligence Guidebook, Morgan Kaufmann, 2015.