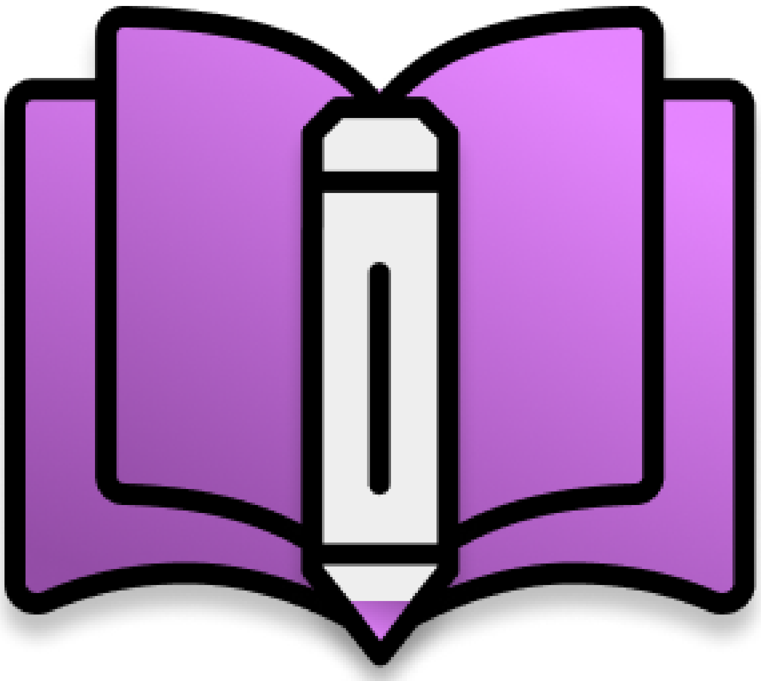


# Development and User Testing of an Academic Scheduling App for University Students

Degree Project Report in Computer Engineering

SIMON ANDERSSON

ERIK KARLSSON

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Degree Project Report In Computer Engineering 2024

# Development and User Testing of an Academic Scheduling App for University Students

SIMON ANDERSSON
ERIK KARLSSON

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2024

Development and User Testing of an Academic Scheduling App for University Students
Simon Andersson
Erik Karlsson

Supervisor: Jonas Almström Duregård, Dept. of Computer Science and Engineering
Examiner: Lars Svensson, Dept. of Microtechnology and Nanoscience

Degree Project Report in Computer Engineering 2024
Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The official logo of the app created in Figma.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Development and User Testing of an Academic Scheduling App for University Students
Simon Andersson
Erik Karlsson
Department of Computer Science and Engineering
Chalmers University of Technology

## Abstract

This project describes the development and user testing of a mobile application designed to assist university students in managing their academic schedules. The app, built using React-Native for cross-platform compatibility and NodeJS for backend efficiency, integrates the Canvas LMS API and a web scraper for TimeEdit to retrieve essential data.

User feedback was collected through a series of tests, starting with Figma prototypes and progressing to an emulator and real-time devices. The key findings highlighted the app's clean design and intuitive interface, with the calendar feature being particularly well-received.

Final user engagement scores ranged from 7/10 to 8/10, indicating strong approval of the app's concept and design. Areas for enhancement included customizable schedule options and the ability to import personal calendars. Despite challenges such as undetected bugs and the limitations of web scraping, the project successfully created a functional, user-friendly app.

This report emphasizes the importance of continuous user involvement and iterative testing in developing effective educational tools. Future research should focus on improving data integration methods, expanding the sample size for user testing, and exploring advanced algorithms for personalized study plans.

# Acknowledgements

First of all, we want to thank Jonas Duregård for helping and giving us valuable feedback during the project. We also want to thank Together Tech for allowing us to write our degree project there. A special thanks to our supervisors from Together Tech, Kevin Solovjov, and Håkan Rolin for guiding us through the technical side of the project.

Simon Andersson, Erik Karlsson, Gothenburg, Jun 2024

# List of Acronyms

Below is the list of acronyms that have been used throughout this thesis:

| | |
|---|---|
| API | Application Programming Interface |
| CID | Chalmers ID |
| CSS | Cascading Style Sheets |
| HTTP | Hypertext Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol Secure |
| I/O | Input/Output |
| JSON | JavaScript Object Notation |
| LMS | Learning Management System |
| NodeJS | JavaScript runtime |
| npm | Node Package Manager |
| OAuth2 | An authorization framework that enables applications to obtain limited access to user accounts on an HTTP service |
| REST | Representational State Transfer |
| UI | User Interaface |
| URL | Uniform Resource Locator |

# Contents

# List of Figures

# 1 Introduction

## 1.1 Background

The pressure to secure a spot in one's dream education program in Sweden has never been higher. However, around 30% of students drop out during the first term, leading to significant financial losses for the educational institutions, society, and the students themselves. Additionally, fewer than 50% of those who embark on a bachelor's degree program manage to graduate [1].

Several factors contribute to students' decisions to drop out, ranging from financial or family issues to difficulties in making friends or failing too many exams [2]. A study from Halmstad University highlights common reasons for students struggling with exams, including not studying enough, starting to study too close to the exam date, poor time management, etc. The study further outlines strategies for mitigating exam-related stress, recommending the elimination of disruptive elements, effective planning and allocation of time, and the utilization of a calendar. This aids students in visualizing their study schedule and incorporating necessary breaks [3].

In today's educational environment, various applications have been developed with the aim of supporting students' studies. A practical approach to reducing exam failure rates could be the creation of an app that offers a suite of tools. This app would help with organizing study schedules, reminding users of important deadlines and exam dates, and delivering daily motivation.

## 1.2 Purpose

The purpose of this project is to develop a planning app designed to assist university students in improving their academic success. The app aims to mitigate factors contributing to student dropouts by addressing issues such as poor time management, procrastination, and stress related to deadlines and exam preparations. By providing evidence-based time management features and personalized study schedules, the app seeks to reduce student stress during challenging times and meet diverse student needs. Initially targeting Chalmers students, the app will eventually expand to other universities.

## 1.3 Objectives

- Develop a user-friendly app with an intuitive interface through user research to ensure relevance and usability for the target audience.
- Implement features such as customizable study plans, deadline reminders, and easy integration with platforms like Canvas and TimeEdit.
- Involve students in the app development process to gather feedback and ensure the app addresses their needs effectively.
- Integrate evidence-based time management strategies to help students organize their schedules and tailor their study plans.
- Design the app to be scalable and adaptable, allowing it to serve a variety of users and expand beyond the initial target group.

## 1.4 Limitations

To make the project more narrow and not too complex, certain limitations have been established during the planning phase of the app. The app will be focused on Chalmers students in the first place, since creating a personalized app for every university in Sweden is way too time-consuming. While there is an intention to expand the app's reach to other universities in the future, such an expansion is beyond the scope of this project. The app will focus on a select number of features to ensure each one is high-quality and effective, rather than overwhelming users with too many options. Furthermore, the app will unfortunately not address the needs of students with disabilities. This arises from its broad audience focus, making it challenging and overly time-consuming to tailor the app for specific individuals.

# 2 Methods

## 2.1 Frontend Development

### 2.1.1 Framework Selection

The frontend of the proposed system will be developed using React-Native, a widely adopted and efficient cross-platform framework for building mobile applications. React-Native facilitates a streamlined development process by allowing developers to write a single codebase for both Android and iOS platforms, significantly reducing development time and resources [4]. Since the target was to develop an app for both Android and iPhone, the choice of framework had to have both a cohesive and responsive user interface that also supported cross-platform development. A few of the many benefits of choosing React-Native were its quick iterations and ability to make changes to both platforms simultaneously. The reason for choosing React-Native was also its big community and broad amount of libraries, which provide secure and stable solutions for potential future development challenges [5][6].

### 2.1.2 User Interface Design

The user interface will be designed following a user-centric approach to ensure a seamless and intuitive experience for end-users. Figma will be employed in the design process due to its powerful features that facilitate real-time collaboration among project members [7]. One of Figma's standout qualities is its ability to translate dynamic UI mockups directly into React-Native components [7]. The planned workflow involves using Figma to create an initial prototype of the application's interface, which will then be used to collect user feedback. This feedback will inform subsequent redesigns, enhancing the interface's usability and effectiveness. Utilizing React-Native's component-based architecture, the design will feature modular and reusable UI elements that ease maintenance and support future enhancements [4]. This methodical approach is expected to allow for rapid iterations in the design process while maintaining the application's scalability and ease of updates.

## 2.2 Backend Development

### 2.2.1 Technology Stack

The backend of the system will be implemented using NodeJS, a versatile and lightweight JavaScript runtime. NodeJS's non-blocking, event-driven architecture makes it well-suited for handling concurrent connections and executing server-side logic efficiently [8]. This is why it makes it a perfect fit for this project, anticipating and handling real-time data and multiple user requests without any problems performance-wise.

A key factor in selecting NodeJS as the backend platform was its robust support for third-party libraries accessible through npm, the Node Package Manager. Utilizing npm can significantly accelerate the development process by reducing time spent on custom solutions. However, the decision to choose NodeJS was based on more than just its package manager. NodeJS is highly user-friendly, with extensive documentation readily available online, enhancing its accessibility for developers. Its scalability is crucial for managing numerous connections simultaneously, ensuring the app can support multiple users as it grows, which is an essential feature for future high-demand scenarios. Additionally, NodeJS supports multi-platform development, making it an ideal choice for this project, which requires a versatile and adaptable backend solution [8].

### 2.2.2 API Development

To consolidate all the necessary information on a single platform, APIs are essential. Initially, three APIs were considered for integration into the project. However, after thorough research, only the Canvas LMS API and the TimeEdit API were deemed suitable for the project. The Canvas LMS API is critical as it enables the retrieval of course-related information, such as the courses a user is enrolled in and the corresponding assignments. The second API, TimeEdit, was intended to fetch the schedules for each course a user is attending.

## 2.3 User testing

Achieving the project's objectives effectively requires user testing, which will be strategically conducted throughout the project's duration. This approach ensures a comprehensive understanding of which aspects of the app meet user standards and which require enhancements. Each session will involve a single participant and include a series of questions both before and after the app usage. All feedback will be documented and utilized as a critical resource for iterative improvements to achieve optimal results. Each test will last between 15 to 20 minutes and will be conducted either face-to-face or online via platforms such as Discord or Zoom. Furthermore, these tests will serve as the primary focus of the report, highlighting the app's development and progress throughout the project. There were in total ten questions asked for the first round of user tests. The first two with the purpose of getting data on when and how students prefer to study and the third one is for pre-use expectations of the app:

- What time of day do you study best?
- How do you structure your study sessions, including intervals of activity and breaks?
- What are your expectations for our app?

These three questions were only asked for the first rounds of tests. The following seven questions was asked post-use of the app:

- What were your initial thoughts when you started using the app?
- What did you like or dislike about the app?
- Do you intend to use this app?
- Which parts of the app would you say are the most useful to you specifically?
- Is there anything in the app that bothers you?
- Is there anything you feel is missing from the app, and if so, what?
- On a scale of 1 to 10, how would you rate the overall experience?

# 3
# Technical and Scientific Background

## 3.1 Technologies

### 3.1.1 React-Native

Numerous mobile app frameworks exist, each offering unique benefits. Typically, mobile apps are categorized into three types: Native Apps, Web Apps, and Hybrid Apps. Among the various mobile app development frameworks available such as React Native, Flutter, and Xamarin, React Native is often chosen for developing native mobile apps for its simplicity and effectiveness [10].

React Native is a cross-platform framework that has gained popularity for its ease of use and efficiency. It enables the development of both Android and iOS applications using a single codebase. The primary advantages of React Native include the ability to write the app's user interface in JavaScript while components render with native code, enhancing performance and user experience. Furthermore, React Native supports hot reloading, allowing developers to instantly see changes without the need to recompile the entire app. This feature significantly speeds up the development process. Another significant benefit is the reusability of code across platforms; components designed for Android often require minimal adjustments to function on iOS, which streamlines the development cycle [11][12].

### 3.1.2 Redux

In JavaScript, Redux is a state container that predicts changes over time. Redux helps the user to write applications that will behave more consistently, run in different environments, and are easy to test. In scenarios like developing apps with React Native, using Redux is highly advantageous as applications can rapidly scale in complexity and size. Redux offers a suite of tools that aid developers in understanding when, where, why, and how the state in the app is being updated, promoting predictable and testable code to ensure the application functions as intended [13][14].

Determining when to use Redux can be challenging. It is particularly beneficial in the following scenarios [14]:

- Managing extensive state spread across various parts of the app.

- Handling states that require frequent updates.
- Simplifying complex logic involved in state updates.
- Collaborating on larger projects with multiple developers.

Key components of a Redux application include [14]:

- The store which is the core of every Redux application. The store is a container that holds the global state of the application. Direct changes to the store are prohibited. To make a change to the state an action must be sent to the store.

- To modify the state, developers must dispatch an action describing the desired change.

- Then the root reducers, which are functions, that calculate the new state based on the previous state, ensure the store is always up-to-date.

### 3.1.3 NodeJS

NodeJS is an asynchronous event-driven JavaScript runtime, which basically means it is a programming environment that executes JavaScript code on the server side [8]. NodeJS is open-source, supports cross-platform, and uses the V8 JavaScript engine, on which Google Chrome is built [15].

For building a scalable network application, it is beneficial to use NodeJS because of its characteristics to handle multiple connections concurrently, making sure that the network application is efficient and not reliant on thread-based networking. NodeJS does not create a new thread for every request, instead, it creates a single process and uses asynchronous I/O primitives to prevent code from blocking [15].

NodeJS can be used for a variety of applications due to its asynchronous, scalable, and fast nature. When building complex SPAs (Single Page Applications) such as Gmail, Netflix, Google Maps, and so on, it is ideal to use a lightweight and efficient runtime, such as NodeJS [16][17]. It also allows for faster data processing because of its non-blocking I/O model which improves the performance. However, NodeJS is not only beneficial to use for SPAs, it also has useful features for real-time chat applications, data streaming applications, embedded systems, and command-line applications. Overall, NodeJS is a valuable tool to use for its versatility and efficiency, which can be used for a wide range of applications [17].

### 3.1.4 API Development

The abbreviation API stands for Application Programming Interface, which is a commonly used word in the programming world. An API may be viewed as a communicator that enables two software components to send and receive requests between each other. The architecture of an API is usually divided into two parts, a client and a server. The client is the application sending the requests, and the server is the application sending the responses [18].

Generally, the four most popular types of API protocols or architectures are the REST, SOAP, RPC, and Websocketf APIs, which all have different use areas. The most common and popular one is the REST API. The abbreviation REST stands for Representational State Transfer. When using a REST API the client is presented with different functions such as GET, PUT, DELETE, and so on. These functions enable the possibility of accessing server data, and this data can be exchanged between clients and servers using HTTP. But, this is not the main feature of the REST API. The main feature of the REST API is statelessness, which means that between requests the server does not save client data. This can be desirable since it brings down the server's response time, hence improving performance. Also, the server does not need to manage any sessions, which eliminates the majority of problems associated with scalability [18][19].

So what are the key steps of using an REST API? It is important to know that REST APIs are very secure with proper authentication. The two most common ways of securing an REST API are with authentication tokens and API keys. The authentication tokens make sure that users are who they claim to be before they are authorized to make API calls. An API key identifies the application that wants to make the API call and checks if it has the access rights necessary to make the call [18].

#### 3.1.4.1 Canvas LMS API

The Canvas LMS API, a REST API, provides detailed information on how to access and modify the data externally. The API uses OAuth2 for authentication, which is a protocol that enables third-party applications such as Google to authenticate and perform actions as a user. This approach is favorable since the use of a third-party application does not require the user's password. The Canvas LMS API is HTTPS-based, which ensures secure data transfer and that sensitive information is well protected through encryption [20].

#### 3.1.4.2 Endpoints

In the realm of mobile application development, the concept of "endpoints" serves as the backbone for seamless communication between the app and the server. These endpoints facilitate the exchange of data and enable various functionalities within the application [21].

Endpoints in the context of a mobile application can be likened to designated pathways or channels through which the app interacts with the backend server. They represent specific URLs that mobile devices use to send requests for data retrieval, storage, or other operations [21].

Endpoints play a pivotal role in the development of mobile applications by enabling them to fetch data from the server, submit user inputs, authenticate users, and perform other essential tasks. Essentially, they serve as the conduits through which the mobile app accesses and interacts with backend services [21].

### 3.1.5 Server Implementation

In the landscape of mobile application development, a robust backend infrastructure is vital for seamless communication between the app and external services. Google Cloud Functions offers a serverless computing solution that allows developers to deploy and scale backend logic without managing server infrastructure directly [22][23].

Google Cloud Functions represent small, single-purpose functions that are triggered by various events, such as HTTP requests, database changes, or file uploads. These functions execute in a managed environment without the need for provisioning or managing servers, making them an excellent choice for implementing backend logic for mobile applications [22][23].

### 3.1.6 Web scraping

Web scraping is an automated method for extracting large amounts of data from websites, converting unstructured HTML data into structured formats like spreadsheets or databases. It's achieved through various means, including online services, web scraping APIs, or custom code. While some websites offer structured data access through APIs, others require scraping due to limitations or lack of technological advancement [24].

The process involves two components: the crawler and the scraper. The crawler browses the web to locate desired data by following links, while the scraper is a tool designed to extract data from specific websites. Scrapers can target either all data on a site or specific information based on user preferences [24].

Web scrapers work by fetching specified URLs, loading HTML, and optionally extracting CSS and JavaScript elements. They then extract the desired data and output it in various formats like Excel, CSV, or JSON. Customizing the scraper ensures efficient and accurate data extraction tailored to project complexity and scope [24].

It is important to acknowledge that web scraping raises ethical and legal considerations. The legality of employing bots to gather data from public websites is still unclear. This issue falls into a grey area because many relevant laws were established long before the internet became widespread or generative AI was developed. Prioritizing which laws apply in such cases has yet to be fully resolved [25].

### 3.1.7 Algorithm

There are few things an algorithm can not do, so sorting sets of numbers, recommending user content, or auto-generating study schedules are often not problems at all. The basic structure of an algorithm is that it typically starts with an initial input of some instructions or rules depending on the task, that are needed for the computation. When the computation is done, the algorithm produces an output [26].

There are several types of algorithms out there, all having a specific purpose and design to complete a certain type of task. There are, for example, search engine algorithms that find keywords in search strings, set them as input, and the algorithm searches its associated database for relevant information to later return the results. This is just an example of an algorithm, and as mentioned there are a lot of different algorithms. Common areas where algorithms are implemented in day-to-day life are for example, in the kitchen and instructions for a recipe, or in the traffic where the traffic signals use algorithms to predict and manage traffic flow [26].

## 3.2 Time management

### 3.2.1 When is the best time to study during the day?

The study of the human brain spans centuries, yet much remains to be understood. The brain is broadly divided into three main parts: the cerebrum, the brainstem, and the cerebellum. The cerebrum, the largest section, houses the hippocampus, a critical area for processing information and learning [27][28].

Research suggests that the timing of learning varies significantly among individuals. Scientific studies indicate that the optimal period for natural learning generally occurs in the late morning, from approximately 9:00 AM to noon. This timing coincides with physiological changes in the body; during sleep, body temperature drops and begins to rise just before waking. This increase in temperature activates various bodily functions, including working memory, alertness, and concentration, thus facilitating peak learning conditions during late morning hours [29].

Despite a decline in alertness post-noon, a 2011 study involving 428 students found that midday fatigue might enhance creative thinking, benefiting those engaged in study or learning activities. Contrarily, other studies suggest that deep focus and learning are most effective from 4:00 AM to 7:00 AM [29][30].

The question of when best to study remains subjective. Individual preferences play a significant role; early risers may find morning hours more productive, whereas others might prefer studying later in the day. Currently, there is no conclusive evidence to support a universally optimal time for learning. As such, both the brain and the body's unique rhythms should guide individuals in choosing their study schedules.

### 3.2.2 Active study time versus breaks

Research indicates that effective breaks can enhance information retention and increase productivity during study sessions. These studies have also highlighted additional benefits of taking breaks, including improved mood, better overall health, and enhanced performance. Breaks provide essential downtime for the brain and body to recuperate and recharge energy. However, opinions vary on the optimal duration of study sessions. Recommendations typically range from 30 to 60 minutes of focused study followed by 10 to 20 minutes of rest. For more extended sessions, such as final exam preparations, longer breaks may be beneficial [31].

Various studies advocate different optimal ratios of study to break times. Some suggest a 90:15 ratio, while others find a 60:15 or 45:10 ratio more effective. According to certain scientific findings, the best balance might be 52 minutes of study followed by a 17-minute break, which has been shown to yield excellent results. Additionally, the Pomodoro Technique—25 minutes of concentrated study followed by a five-minute break—is favored by many for maintaining focus and energy. Ultimately, the choice of study and break intervals depends on individual preferences and what proves most effective for the learner [32].

While numerous study techniques exist, they all agree on the importance of active breaks. How these breaks are spent can vary; some individuals might engage in a brief physical activity to rejuvenate, while others may prefer a more relaxing break to minimize stress. It's crucial to use a timer to manage break durations effectively, ensuring readiness to resume studying [31].

Like choosing study times and break intervals, determining the best time of day for study is highly individual. Personal preference plays a significant role in deciding when one is most receptive and alert for learning.

# 4
# Development Process

## 4.1 Planning Phase

Considerable time in the early stages of the project was dedicated to planning, ensuring the project had a robust and stable starting point. Initially, it was crucial to determine which app features were necessary to meet the project objectives. The initial selection of features planned for implementation at various stages was overwhelming. Understanding what could be achieved within two quarters is challenging. Consequently, the company's supervisor suggested the idea of identifying essential "must-have" features and desirable "nice-to-have" features. This prioritization ensured that the project initially focused on the critical features, allowing for later incorporation of the "nice-to-have" features.

Discussions also revolved around choosing appropriate frameworks, available APIs, and clarifying the project's primary goal. Selecting the framework was straightforward; Kevin, the supervisor, had extensive experience in app development, making React Native the obvious choice due to its ease of use. Decisions were also made for the programming languages: TypeScript with CSS for frontend styling, and Node.js for the backend, which was an easy choice given Kevin's prior experience with Node.

Identifying necessary APIs was relatively simple. Initially, three APIs were selected: the Canvas LMS API, the TimeEdit API, and the LADOK API. However, subsequent research revealed that the LADOK API was outdated and therefore unusable. As a result, only the Canvas and TimeEdit APIs were chosen for the project. No further API research was conducted during the planning phase.

It was important for both the company and the Chalmers supervisor that there was a clear objective with the project. In the beginning, the objective was vague and overly broad. Thus, it became necessary to define a more targeted and purposeful objective for both the development process and the report. The agreed prime objective was to create a user-friendly app with relevant features. To achieve this, user testing was deemed essential; a series of questions were prepared to gather meaningful and actionable feedback from university students. This feedback would later inform improvements to the app.

Establishing effective communication with both the company and Chalmers was crucial. Weekly meetings with the company were scheduled to review the project's progress. Due to initial delays caused by the awaited exam results, the assignment of the Chalmers supervisor was postponed. Consequently, regular interactions with the Chalmers supervisor commenced a few weeks after the project had already begun. These sessions, held once per week, primarily centered on the development of the report.

## 4.2 Design

The main aim of the designing phase was to create a user-friendly experience that did not overwhelm the user with unnecessary information. Careful planning was done to ensure that the app's design prioritized simplicity, intuitiveness, and functionality. Every element of the user interface was meticulously crafted to facilitate easy navigation and comprehension. Extensive research and analysis were conducted to understand the target audience's preferences, behaviors, and pain points, allowing for a tailored design approach.

Usability testing was conducted throughout the design process to gather feedback and identify areas for improvement. Multiple rounds of user testing sessions helped refine the app's design based on real-world usage patterns and user feedback.

Ultimately, the goal was to create an app that users could navigate effortlessly, accomplish tasks efficiently, and enjoy using on a daily basis. By prioritizing user-centric design principles and conducting thorough testing, an app was developed that not only meets the needs of the target audience but also delights them with its intuitive and user-friendly interface.

The app's design needed to be both robust and modular to facilitate quick fixes when changes were required. The absence of major alterations to the preliminary design suggested that users appreciated the initial layout. The app's simplicity and effectiveness from the outset likely contributed to this satisfaction. While the design was well-received, it was not without imperfections. However, the combination of its modular and robust structure, along with user feedback, brought the desired design closer to fruition for both the development team and users.

## 4.3 Implementation

### 4.3.1 Frontend Implementation

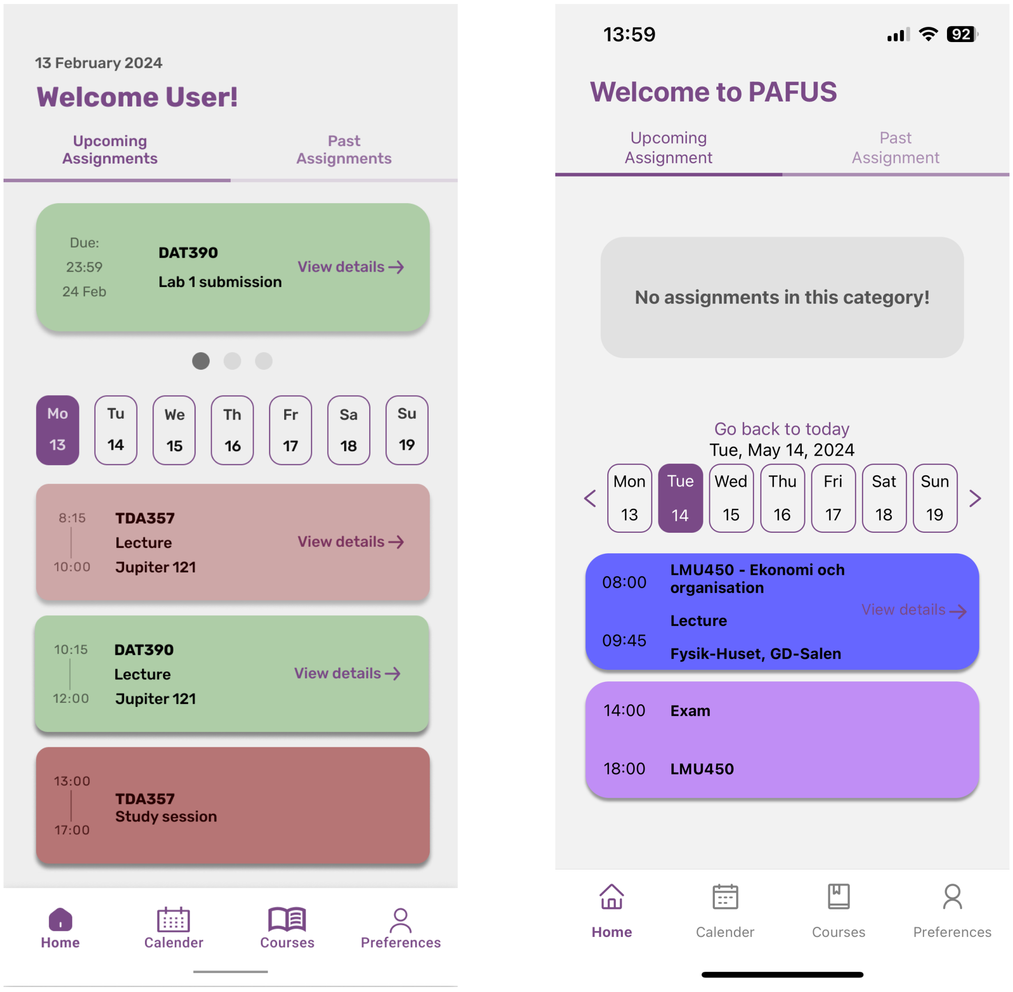
It was essential to begin by familiarizing with the development tools used for the app, specifically React Native and Node. These tools proved straightforward to learn, which facilitated a prompt start to the app development. Prior to programming, it was critical to plan the appearance of the app screens. As noted in the design section, the initial screens were crafted using Figma. Starting with Figma enabled a smoother transition of these designs into the app later on. Given Figma's user-friendly nature, significant time was not required for this stage, allowing the actual development of the app to commence within the first few weeks of the project.

The initial key step involved establishing a navigation bar to facilitate easy movement between the app's main screens. This task was challenging initially, due to a lack of experience within the group of how integration between various components works. Given the app's focus on simplicity and functionality, it was crucial for the navigation bar to be both straightforward and representative of the app's core functionalities. The first four screens implemented were the home, calendar, course, and preferences screens, each offering distinct features but interconnected in functionality.

### 4.3.2 Home screen

The design of the home screen aimed to be both informative and simplistic, serving as the initial interface that users encounter. It was essential to select features that effectively represent this screen, including upcoming deadlines, past deadlines, a weekly calendar, and a daily overview of current lectures and other activities. Additional information about a lecture or an assignment could be viewed on the home screen with just a simple click on the activity. The information for a lecture is similar to the information provided in TimeEdit, and the additional information for an assignment is mainly a description of the assignment. Initially, 'dummy data' was used to perfect the design by identifying aspects that required refinement. This temporary data was later replaced with actual data obtained from API calls and integrated app logic.

The home screen was designed to aggregate and display the most crucial information from other parts of the app. Although it was the first screen implemented, the full integration of its functionality was contingent upon the completion of the other screens' designs and logic. This strategic approach ensured that the home screen would effectively synthesize and present the most important information to users. As such, while the home screen's UI was developed early, its final completion and logic integration were the last steps in the process, ensuring all elements were cohesive and functional.

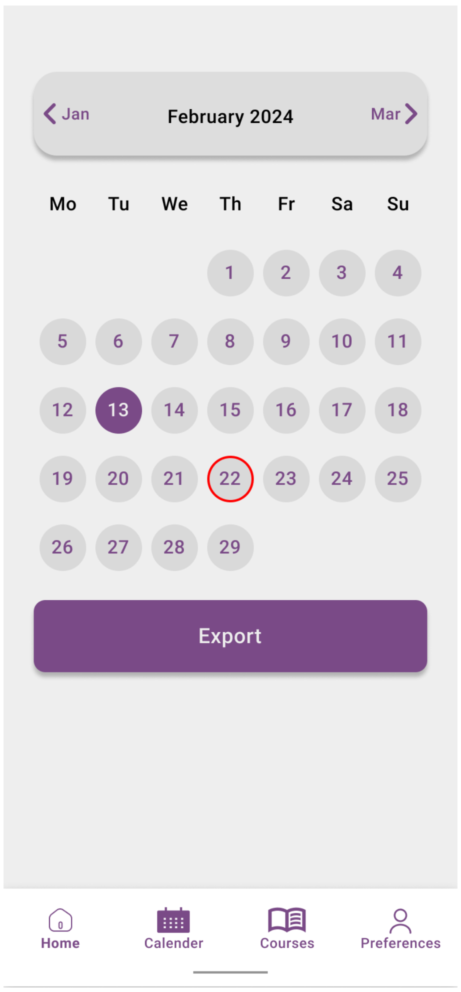


(a) Preliminary Design (b) Final Design

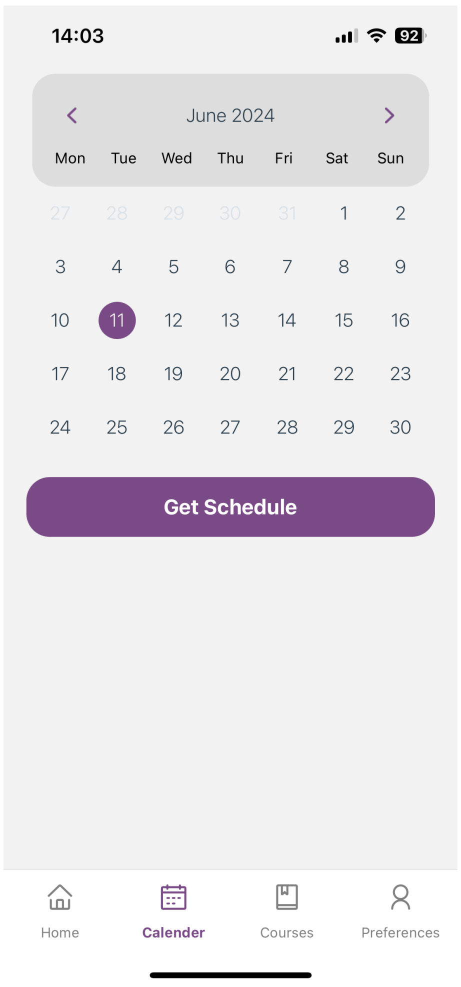
**Figure 4.1:** Comparison of the final and preliminary design of the home screen

### 4.3.3 Calendar screen

The calendar design was inspired by the iPhone's interface. To avoid replicating Apple's exact calendar design, the calendar screen was initially prototyped in Figma. As implementation commenced, it became apparent that designing both the aesthetics and logic of the calendar would be time-consuming. Consequently, research was conducted to find a complete calendar solution that matched the preconceived Figma design. It was discovered that React Native includes a calendar component that is closely aligned with this vision. As a result, much of the time originally allocated for designing the calendar screen was redirected towards understanding and adjusting the component's various parameters to achieve the desired look.



(a) Preliminary Design

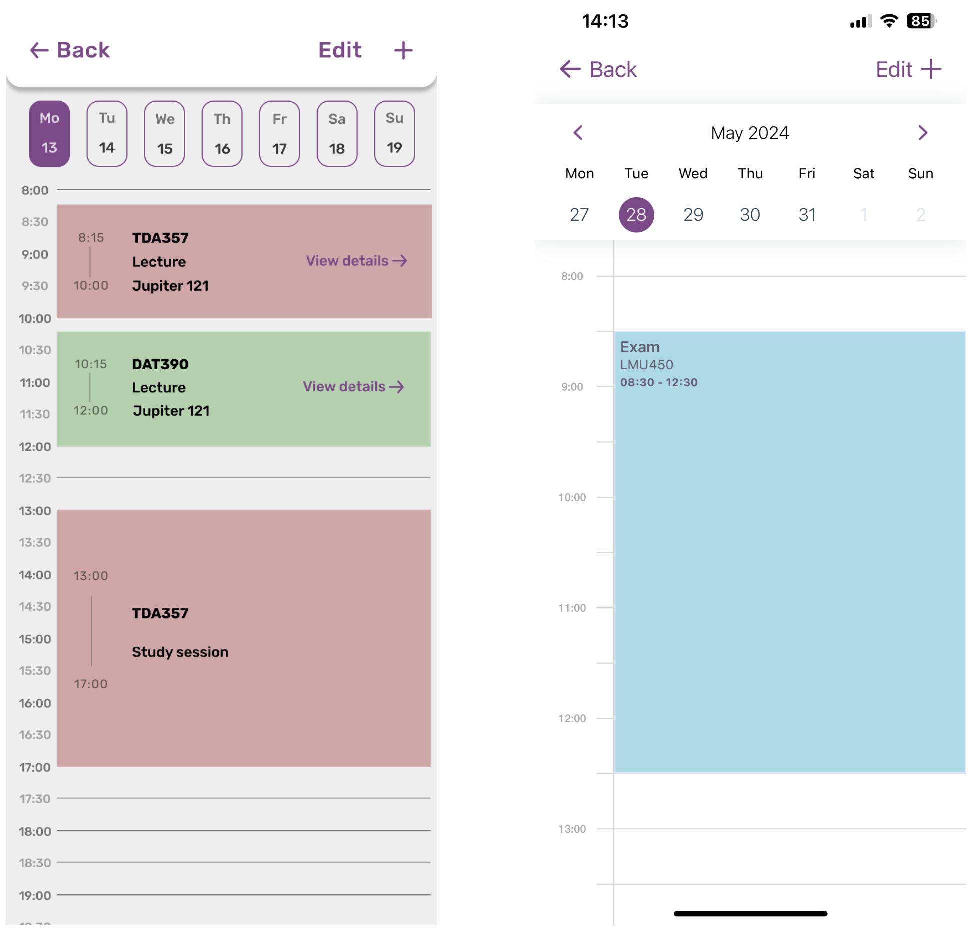


(b) Final Design

**Figure 4.2:** Comparison of the final and preliminary design of the calendar screen

#### 4.3.3.1 Timeline calendar

Following the main calendar interface's completion, the focus shifted to designing a timeline calendar. This feature, which displays all events for a selected day, also drew inspiration from the iPhone's calendar. Since most designs were already finalized in Figma, the timeline calendar had a clear direction from the start. However, considering the availability of an existing library for the main calendar, exploration of similar libraries for the timeline calendar was deemed necessary. Like the main calendar, existing libraries were used to replicate the intended design and user interactions, dedicating significant time to mastering these libraries' parameters and integration. The timeline calendar's primary function was to display the schedule of actively selected courses, similar to TimeEdit. It also needed to display lectures and various activities for different courses and provide a feature for adding events not included in the schedule sourced from the web scraper. This capability was incorporated in response to feedback from user testing. The integration of an edit button for custom events was later implemented. The edit button makes sure that the user can edit or remove the custom events set by them.

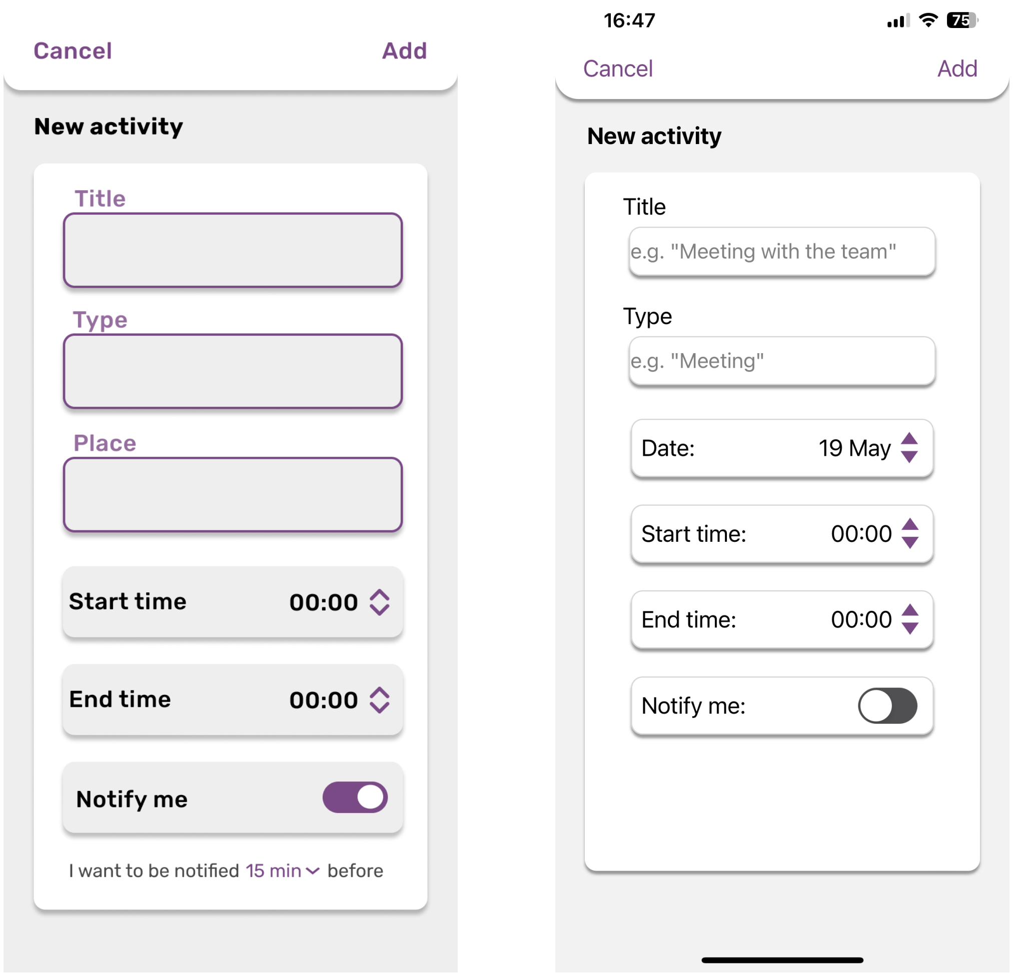


(a) Preliminary Design (b) Final Design

**Figure 4.3:** Comparison of the final and preliminary design of the timeline calendar

#### 4.3.3.2 New activity

The option to add new activities was not initially planned as a core feature. However, this feature became essential after receiving feedback during the first round of user testing, before any major implementations were made to the app. The design of this screen included options for setting the title, type, place, start, and end time of the new activity and whether it should trigger notifications. Implementation of this logic was initially deferred as it was not deemed critical. However, when it became necessary to integrate this feature into the timeline calendar, challenges arose with the activities not being placed on the desired dates. After some redesign and discussion, the desired behavior was achieved by adding a date picker which placed the activity at the wanted date.



Start time 00:00

End time 00:00

Notify me

I want to be notified 15 min before

e.g. "Meeting with the team"

e.g. "Meeting"

Date: 19 May

Start time: 00:00

End time: 00:00

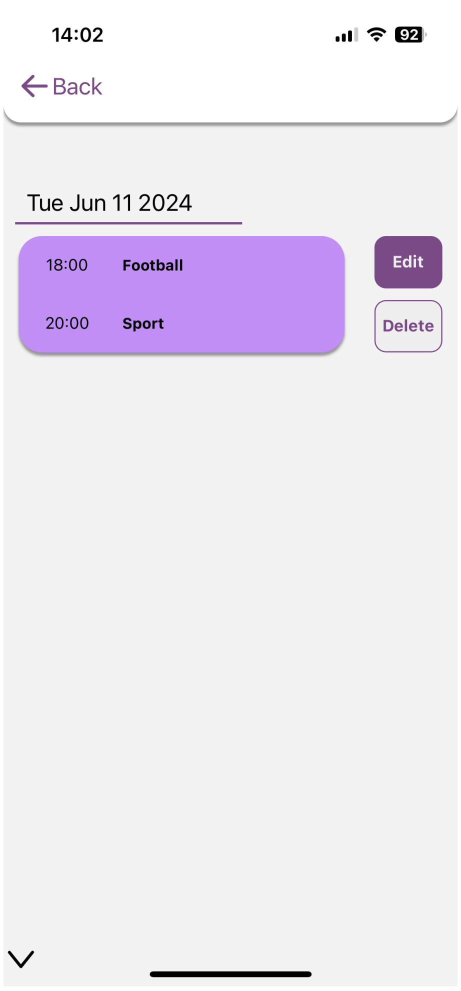
Notify me:

(a) Preliminary Design (b) Final Design

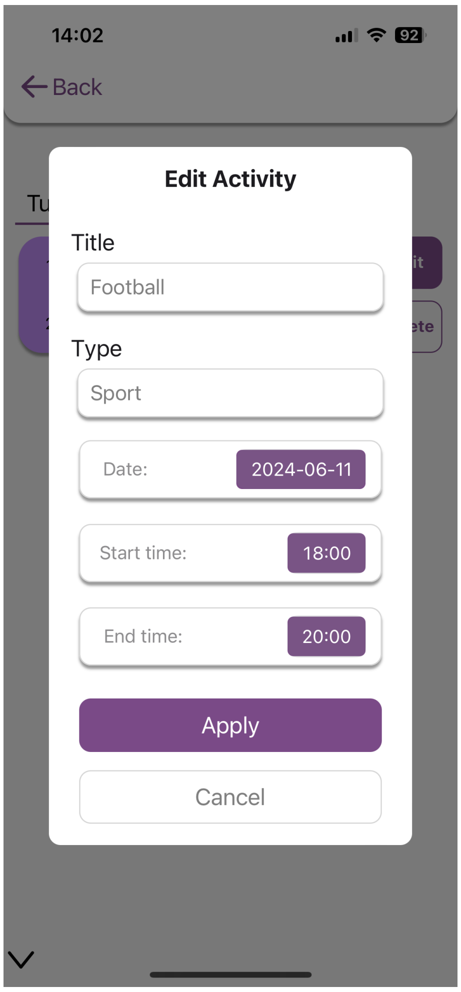
**Figure 4.4:** Comparison of the final and preliminary design of the new activity screen

### 4.3.4 Edit activity

With the introduction of the last feature, it became quite clear that a feature to edit activities needed to be implemented. This could be because a user accidentally enters the wrong information when creating a new activity, an activity has been moved or an activity needs to be deleted. The edit activity feature, therefore, allows the user to change all the information about the activities and also delete them.



(a) Edit activity screen

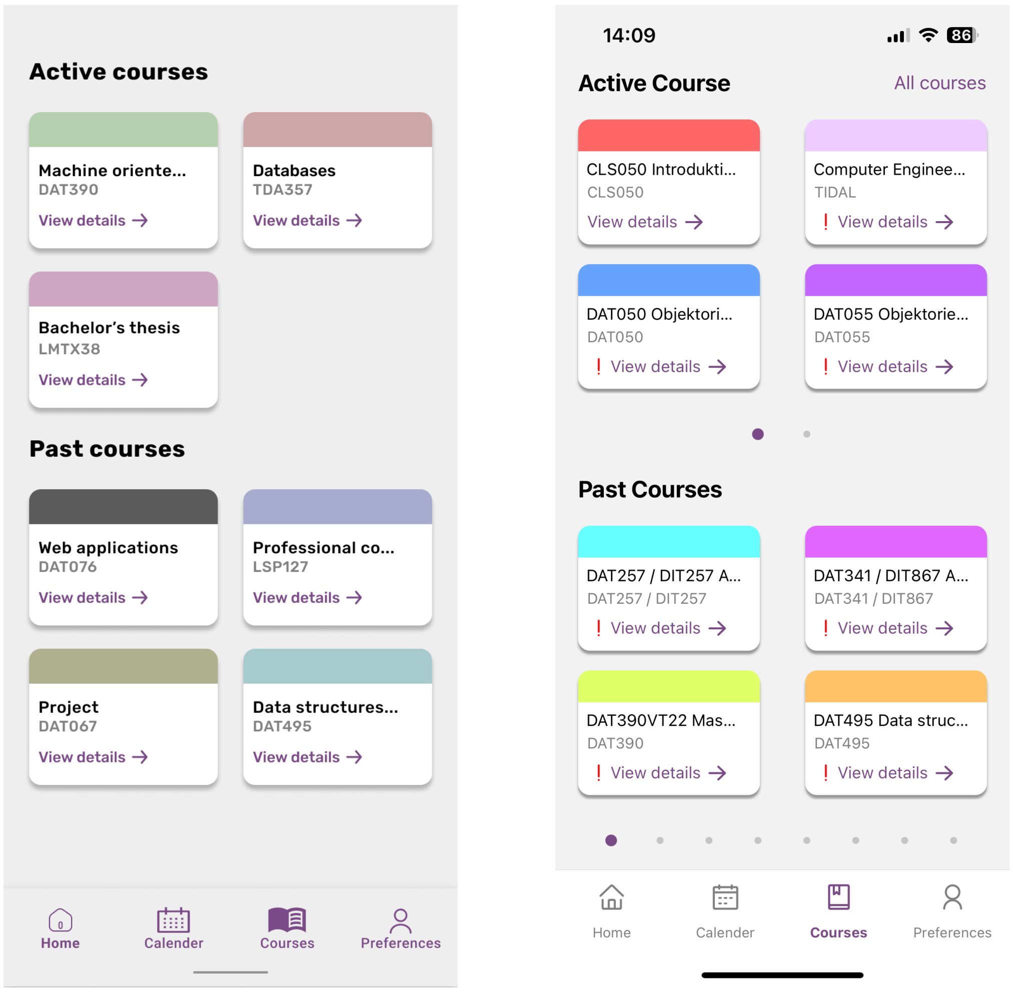


(b) Edit activity modal

**Figure 4.5:** Design for the edit activity screen and edit activity modal

### 4.3.5 Course screen

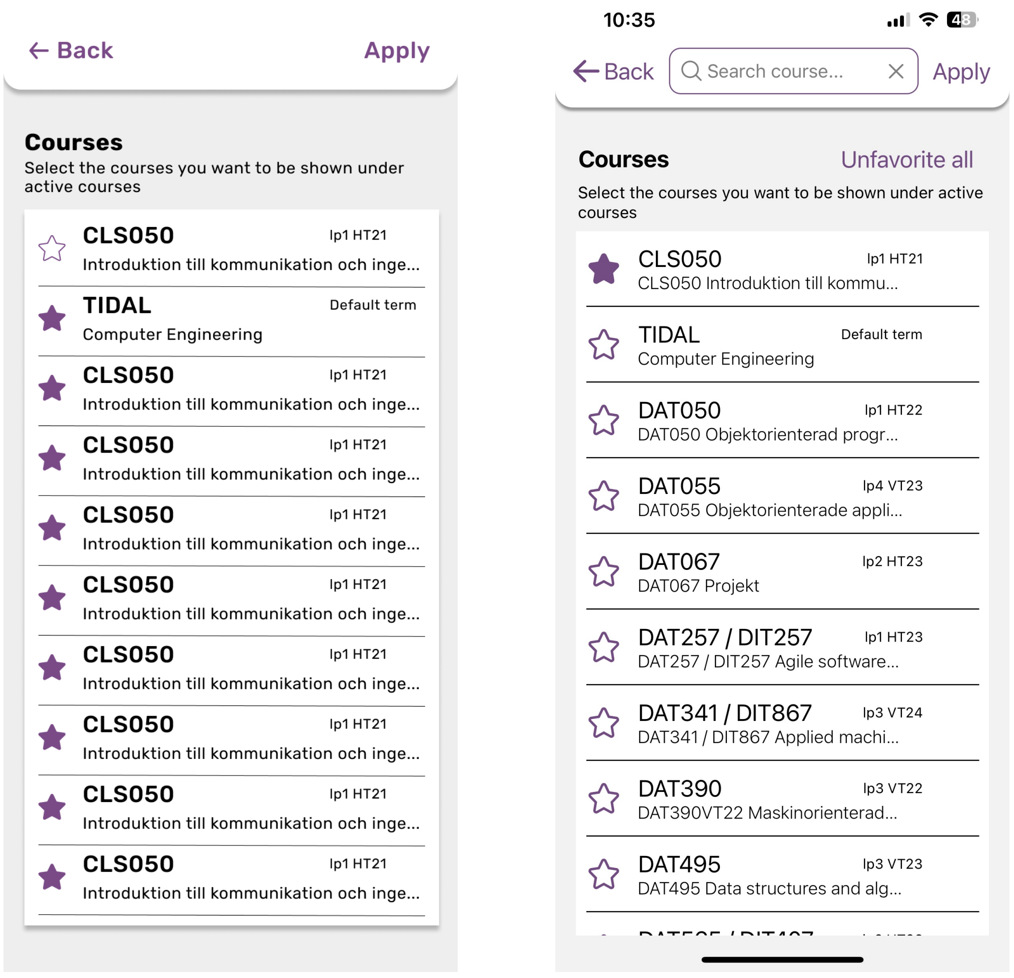
The course page drew significant inspiration from the Canvas UI, albeit with modifications to display both active and past courses for the user. As for the other screens, this screen also needed to be straightforward and user-friendly. Users needed to be able to easily select and view both active and past courses, with additional course information accessible through a simple click. Initial efforts were focused on designing and finalizing the UI, with plans to replace all placeholder data with actual data later in the development process. Additional information displayed upon selecting a course includes deadlines/assignments, course points, and the exam date. With the course tab divided into active and past courses, a straightforward method was required to designate which courses were active and which were past. The solution implemented was the addition of an all-courses screen, displaying all possible courses, akin to the layout found in Canvas. This screen enhances the user experience by including a search bar to easily locate specific courses and a button that unfavorites all courses, eliminating the need for manual adjustments for each course.



(a) Preliminary Design (b) Final Design

**Figure 4.6:** Comparison of the final and preliminary design of the course tab screen

Originally, each course was assigned a specific color as a visual identifier. However, challenges arose when transitioning from dummy data to real data retrieved via API calls, complicating the color assignment for each course. Despite the importance of color coding for aesthetic and user-friendly design, prioritizing the accuracy of the logic took precedence over immediate design adjustments. A significant challenge was accurately displaying courses designated as favorites on the "all-courses" screen in the active section, and others in the past section. Once this logic was established, only minor design adjustments were needed, as most of the design had already been finalized before integrating the actual data. Upon completing all design aspects, the task of assigning a unique color to each course was undertaken.



← Back Apply

**Courses**
Select the courses you want to be shown under active courses

| | | | |
|---|---|---|---|
| ☆ | **CLS050** | Introduktion till kommunikation och inge... | lp1 HT21 |
| ★ | **TIDAL** | Computer Engineering | Default term |
| ★ | **CLS050** | Introduktion till kommunikation och inge... | lp1 HT21 |
| ★ | **CLS050** | Introduktion till kommunikation och inge... | lp1 HT21 |
| ★ | **CLS050** | Introduktion till kommunikation och inge... | lp1 HT21 |
| ★ | **CLS050** | Introduktion till kommunikation och inge... | lp1 HT21 |
| ★ | **CLS050** | Introduktion till kommunikation och inge... | lp1 HT21 |
| ★ | **CLS050** | Introduktion till kommunikation och inge... | lp1 HT21 |
| ★ | **CLS050** | Introduktion till kommunikation och inge... | lp1 HT21 |
| ★ | **CLS050** | Introduktion till kommunikation och inge... | lp1 HT21 |

(a) Preliminary Design

10:35

← Back Search course... Apply

**Courses** Unfavorite all
Select the courses you want to be shown under active courses

| | | | |
|---|---|---|---|
| ★ | CLS050 | CLS050 Introduktion till kommu... | lp1 HT21 |
| ☆ | TIDAL | Computer Engineering | Default term |
| ☆ | DAT050 | DAT050 Objektorienterad progr... | lp1 HT22 |
| ☆ | DAT055 | DAT055 Objektorienterade appli... | lp4 VT23 |
| ☆ | DAT067 | DAT067 Projekt | lp2 HT23 |
| ☆ | DAT257 / DIT257 | DAT257 / DIT257 Agile software... | lp1 HT23 |
| ☆ | DAT341 / DIT867 | DAT341 / DIT867 Applied machi... | lp3 VT24 |
| ☆ | DAT390 | DAT390VT22 Maskinorienterad... | lp3 VT22 |
| ☆ | DAT495 | DAT495 Data structures and alg... | lp3 VT23 |

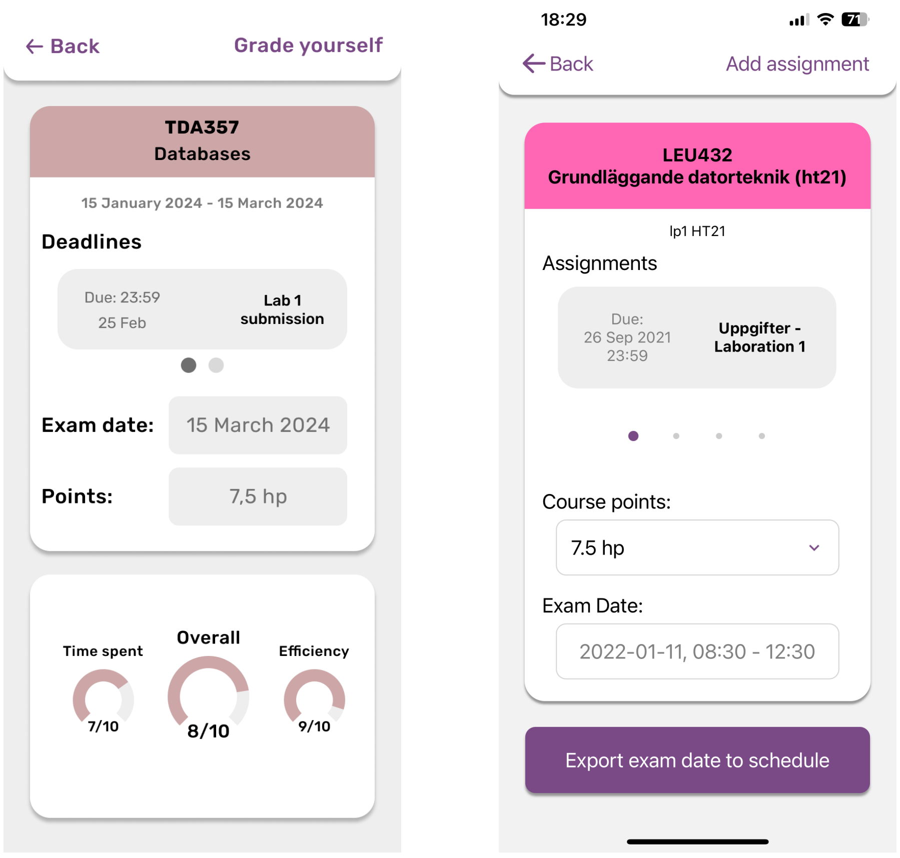
(b) Final Design

**Figure 4.7:** Comparison of the final and preliminary design of the all courses screen

#### 4.3.5.1 Course details

It was important for each course to display more detailed information than just its name and course code. The initial plan for this screen included displaying assignments and deadlines related to the course, the course points, and the exam date. During the design and feature planning stage, there were discussions about including progress tracking for each course, allowing users to see their progress throughout the course duration. This feature was considered in the initial design but was ultimately deemed non-essential and removed from the final design.

The idea to revise the course details screen emerged from user feedback during app testing. A user requested the ability to see the exam date in the calendar, leading to the addition of an export button for the exam date on the course details screen. Regarding assignments and deadlines, an endpoint was set up on the server for the API call handling assignments for each course. However, it was quickly realized that the API call did not retrieve assignments for all courses, as some course setups do not include the assignment tab on the Canvas page. To address this, an add assignment button was added to the course details screen, allowing users to manually add missing assignments for the course.

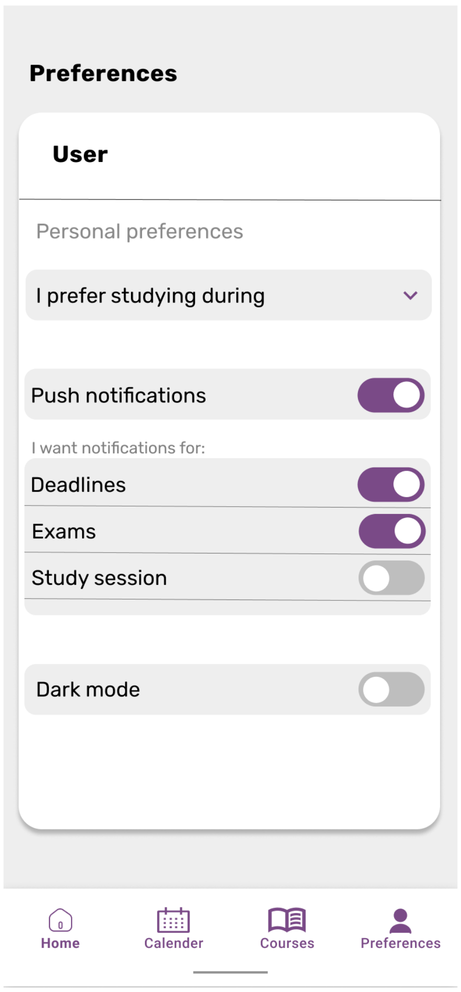


(a) Preliminary Design (b) Final Design

**Figure 4.8:** Comparison of the final and preliminary design of the course details screen

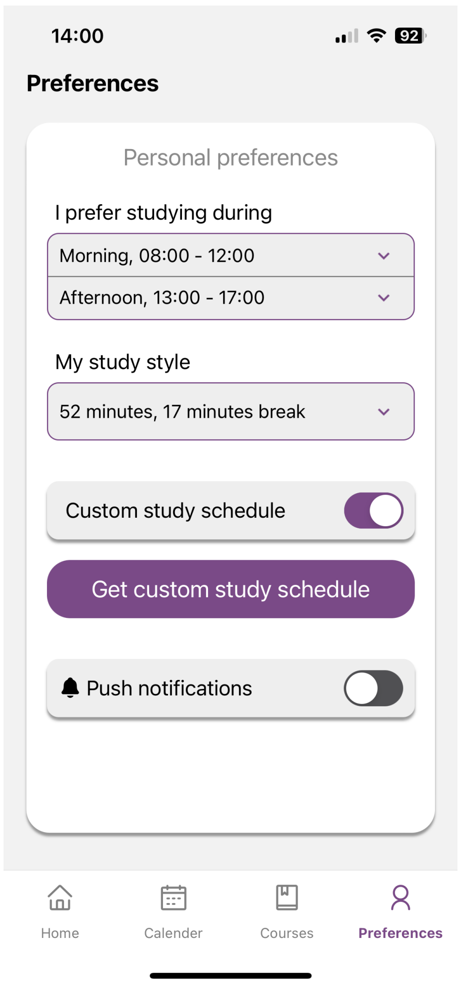
### 4.3.6 Preferences screen

The preferences screen received the least priority. Initially conceived as a settings page for user account modifications, it soon became apparent that such a screen was unnecessary. Instead, a preferences screen was developed, allowing users to set personal preferences for the app, such as preferred study times, preferred study style, and notification settings. Preferred study times and styles offer various alternatives, each based on user preferences. These alternatives were carefully selected through research conducted online and through user tests.



Push notifications: on
Deadlines: on
Exams: on
Study session: off
Dark mode: off

(a) Preliminary Design



I prefer studying during: Morning, 08:00 - 12:00; Afternoon, 13:00 - 17:00
My study style: 52 minutes, 17 minutes break
Custom study schedule: on
Push notifications: off

(b) Final Design

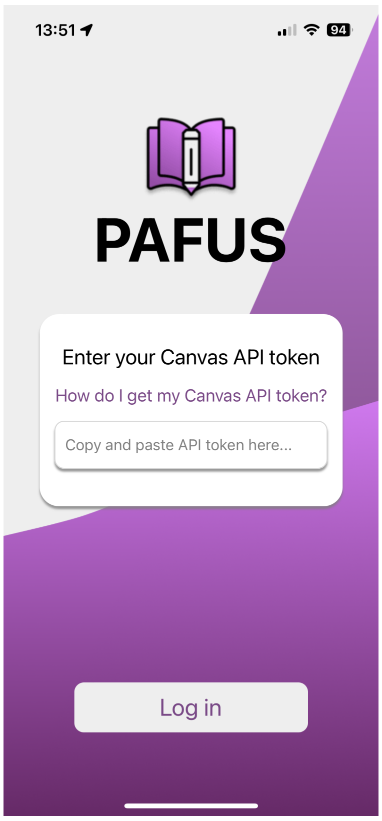
**Figure 4.9:** Comparison of the final and preliminary design of the preferences screen

### 4.3.7 Setup screens

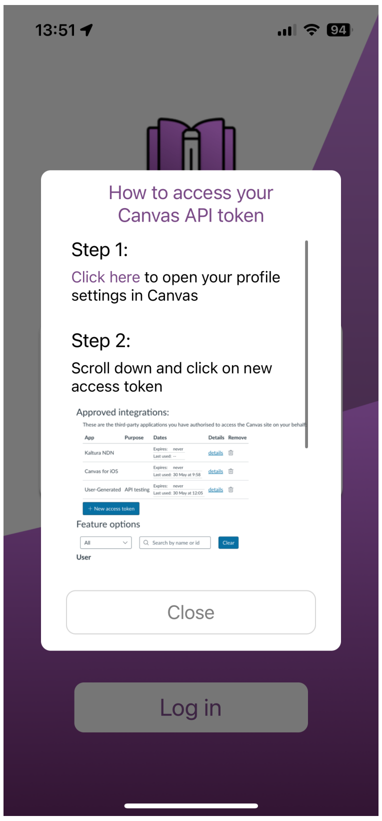
After the preliminary design for the four main screens was completed, development proceeded to the setup screens, which were essential for configuring the app correctly. These setup screens were initially designed in Figma, similar to the other screens, ensuring a cohesive design before the implementation of the code.

#### 4.3.7.1 Login screen

The first screen that users encounter is the login screen, designed to be straightforward yet welcoming. Initially, clicking the login button was intended to redirect users to the Canvas login page to log in with their CID and password. However, due to complications with CID access, an alternative solution requires users to enter an access token generated from their Canvas profile. This token ensures users receive the appropriate API information from their Canvas page. While this workaround is functional, it is neither sustainable nor user-friendly and remains a temporary solution. If the user does not know how to find their Canvas API token, a link has been provided to show the user how to retrieve it.


13:51
PAFUS
Enter your Canvas API token
How do I get my Canvas API token?
Copy and paste API token here...
Log in

(a) Design login screen



How to access your Canvas API token

Step 1:
Click here to open your profile settings in Canvas

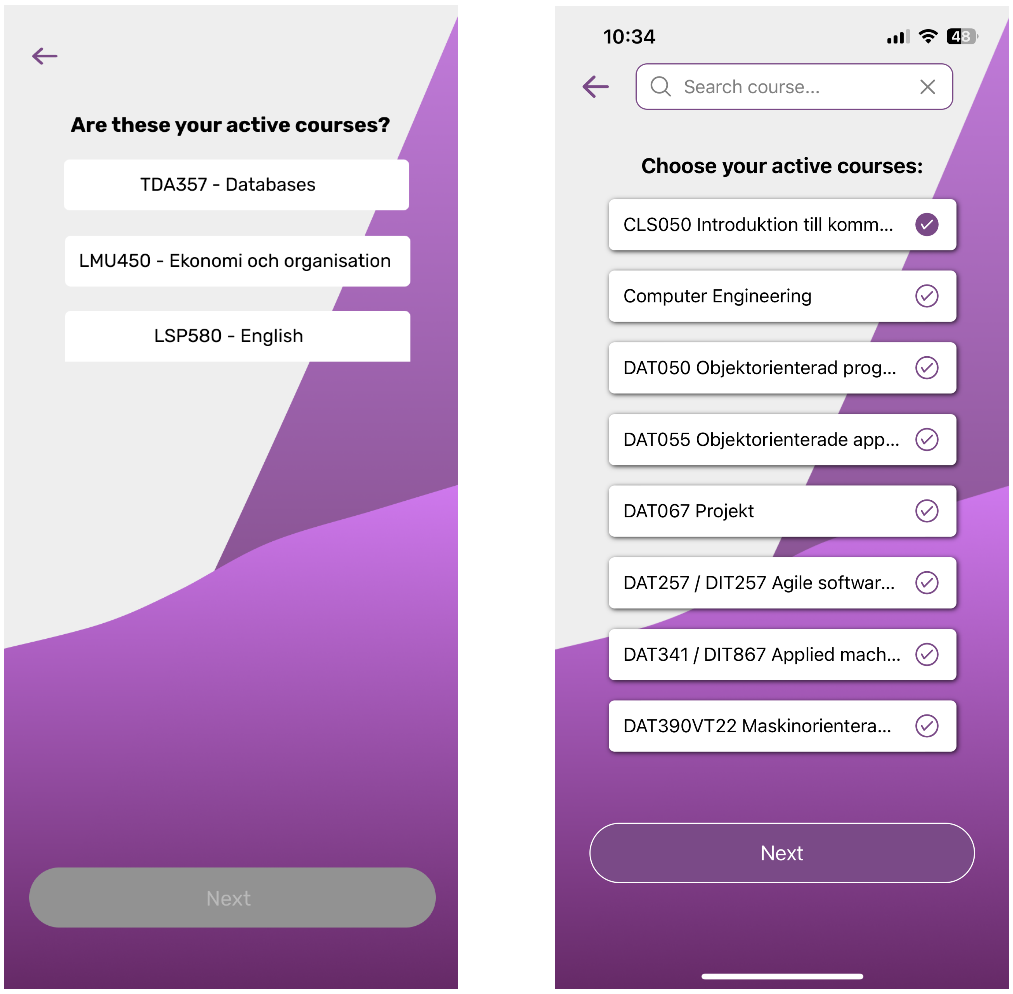
Step 2:
Scroll down and click on new access token

(b) Guide for retrieving Canvas API token

**Figure 4.10:** Design of the login and guide for retrieving Canvas API token

#### 4.3.7.2 Setup active courses

Following login, the next four screens guide users through the initial setup. The first setup screen displays all courses registered to the user. This screen allows users to select which courses are currently active and which should be categorized as past. To enhance usability, a search bar has been incorporated, similar to that on the all-courses screen, making it easier to locate specific courses.

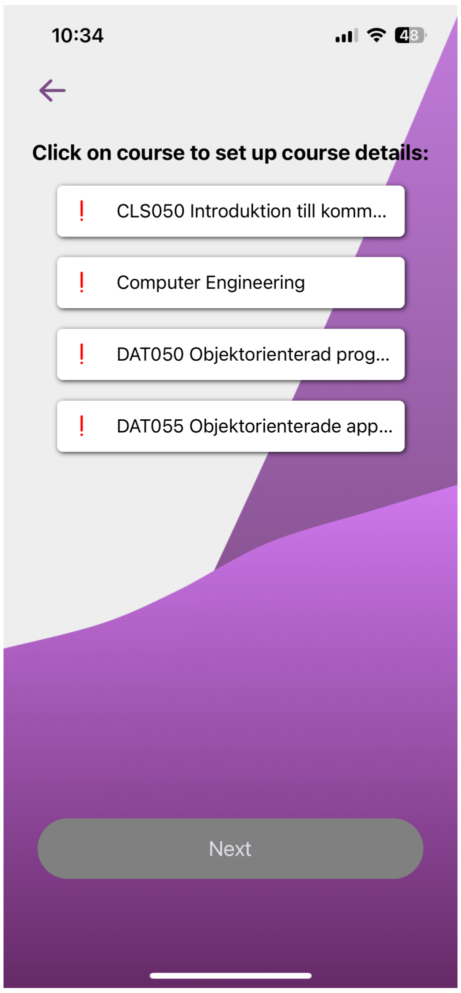

Are these your active courses?
TDA357 - Databases
LMU450 - Ekonomi och organisation
LSP580 - English
Next
10:34
Search course...
Choose your active courses:
CLS050 Introduktion till komm...
Computer Engineering
DAT050 Objektorienterad prog...
DAT055 Objektorienterade app...
DAT067 Projekt
DAT257 / DIT257 Agile softwar...
DAT341 / DIT867 Applied mach...
DAT390VT22 Maskinorientera...
Next

(a) Preliminary Design (b) Final Design

**Figure 4.11:** Comparison of the final and preliminary design of the setup screen for choosing active courses

#### 4.3.7.3 Setup course details

After selecting active courses, users proceed to the next screen to enter essential course details, such as course points, which are critical for the study session algorithm. Users can also enter the exam date, including its start and end times, to enable exporting the exam date to the calendar. If a course does not have an exam, users can leave the fields associated with the exam date empty. The design for the this screen can be seen in Figure 4.12.



10:34

Click on course to set up course details:

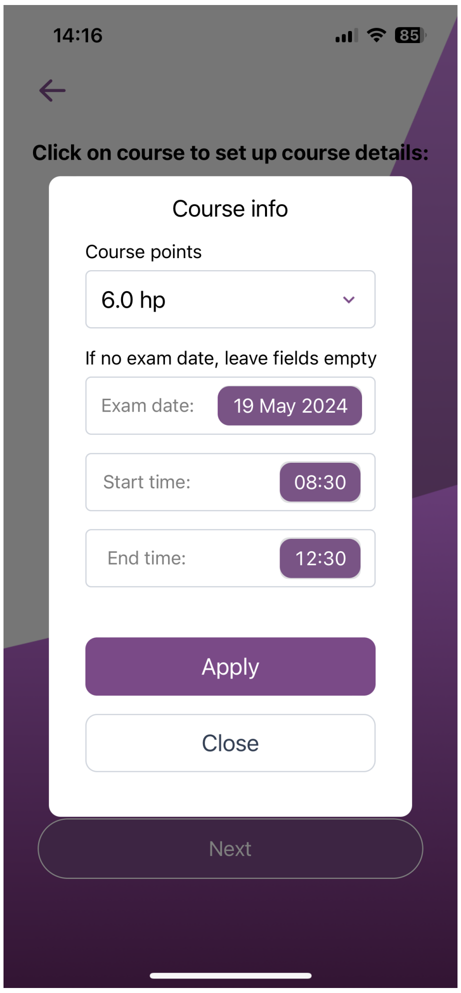
! CLS050 Introduktion till komm...

! Computer Engineering

! DAT050 Objektorienterad prog...

! DAT055 Objektorienterade app...

(a) Setup course details design



14:16

Click on course to set up course details:

Course info

Course points: 6.0 hp

If no exam date, leave fields empty

Exam date: 19 May 2024

Start time: 08:30

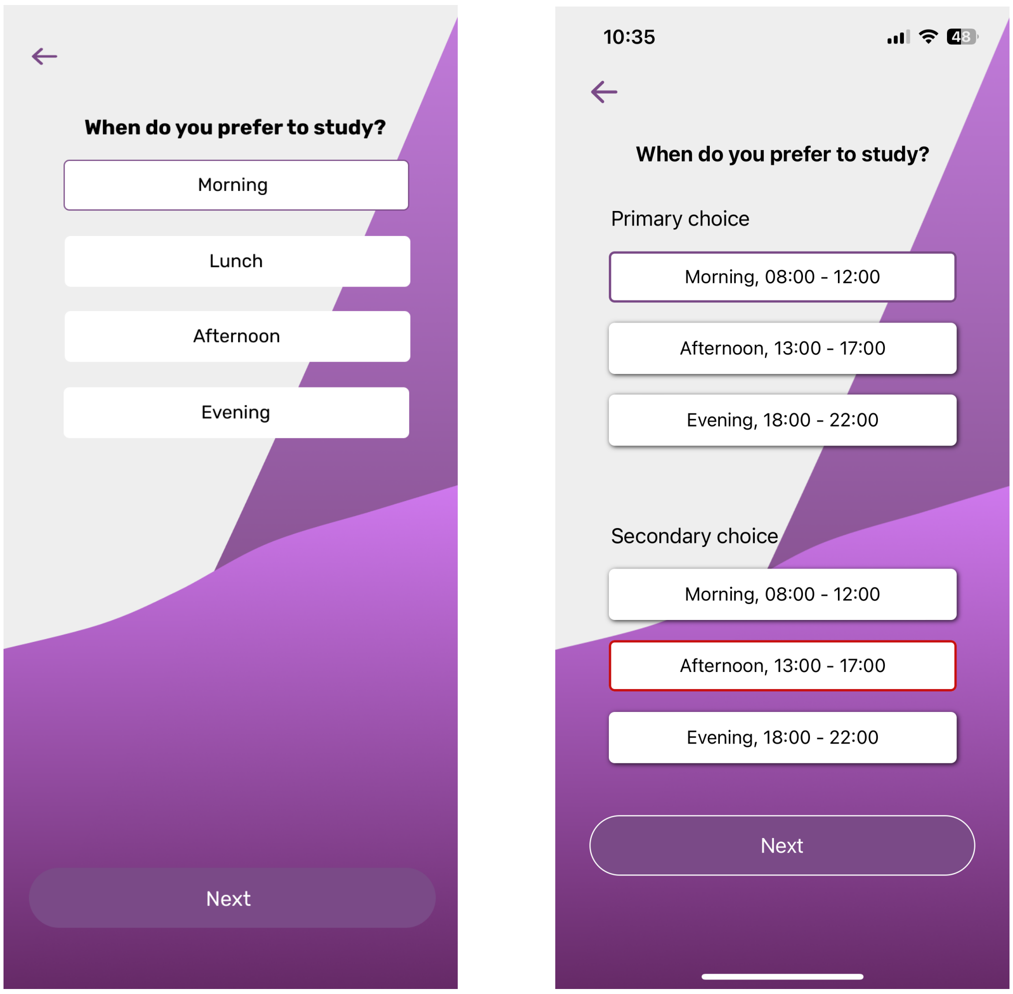
End time: 12:30

(b) Design for entering information about the course

**Figure 4.12:** Here is the design for setting up the course details for each course chosen as active

### 4.3.7.4 Setup user preferences when to study

Once the course setup is complete, users advance to the final two screens. The first screen allows users to choose their preferred study times, requiring them to select two study times—one primary and one secondary—to generate an eight-hour work-day schedule.

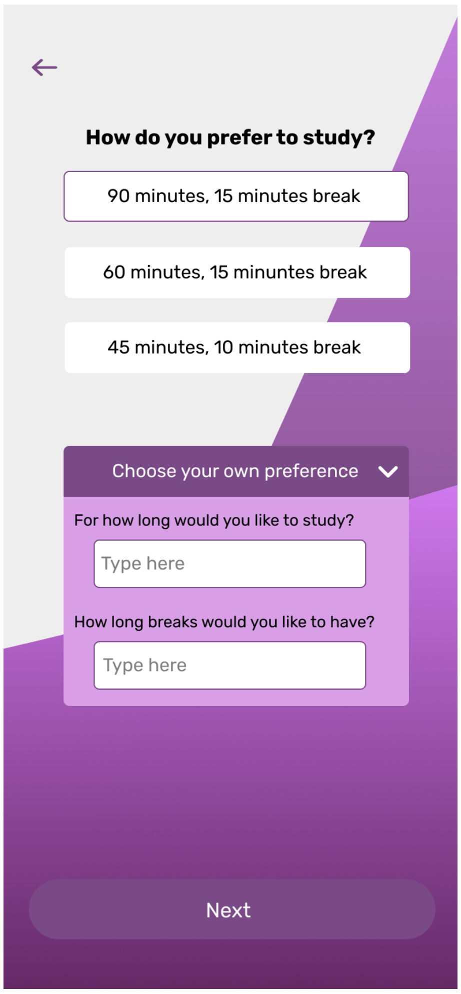


(a) Preliminary Design (b) Final Design

**Figure 4.13:** Comparison of the final and preliminary design of the setup screen for choosing the preferred time when to study

#### 4.3.7.5 Setup user preference how to study

The final screen presents study method options based on research-supported calculations. These alternatives are designed for four-hour-long sessions, ensuring a balanced mix of study and break times. Once the setup is complete, the app processes the information and presents the finalized setup to the user within a few seconds. As shown in Figure 4.14 the preliminary design had the choice for choosing a unique preference on how to study. This feature was later removed as seen in the new design due to complications with the algorithm.



How do you prefer to study?

90 minutes, 15 minutes break

60 minutes, 15 minuntes break

45 minutes, 10 minutes break

Choose your own preference

For how long would you like to study?

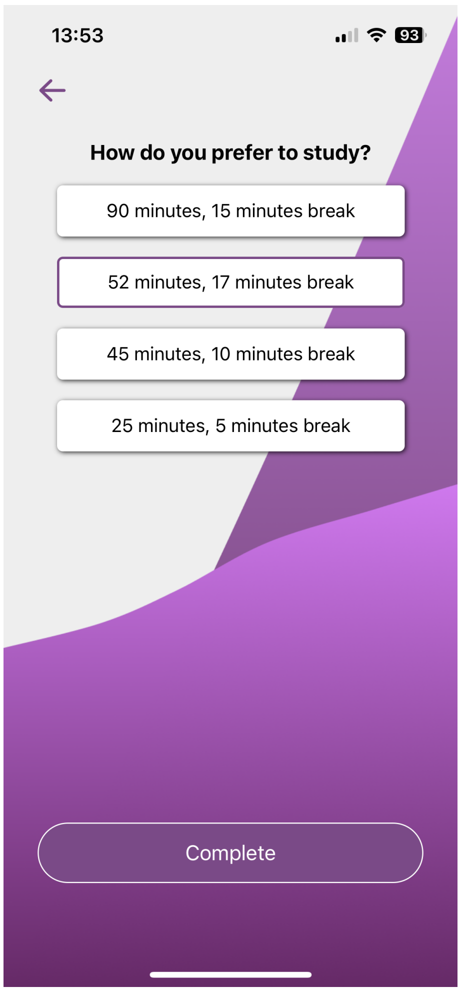
Type here

How long breaks would you like to have?

Type here

Next

(a) Preliminary Design



13:53

How do you prefer to study?

90 minutes, 15 minutes break

52 minutes, 17 minutes break

45 minutes, 10 minutes break

25 minutes, 5 minutes break

Complete

(b) Final Design

**Figure 4.14:** Comparison of the final and preliminary design of the setup screen for choosing the preferred alternative how to study

### 4.3.8 Backend Implementation

#### 4.3.8.1 Google Cloud Hosting

The backend of the application was established using Google Cloud Functions, a serverless computing service provided by Google Cloud. This setup allowed for the deployment and management of backend logic without the need to handle server infrastructure directly. Google Cloud Functions efficiently managed the execution of server code in response to HTTP requests and other events, ensuring scalability and reliability.

In addition to running the server, Google Cloud Storage buckets were utilized to store schedule data. This integration enabled the secure saving and accessing of user schedule information, leveraging the scalability and accessibility of Google's cloud storage solutions. By combining Google Cloud Functions with Google Cloud Storage, a robust and flexible backend infrastructure capable of meeting the application's demands was ensured.

#### 4.3.8.2 Web Scraper

The web scraper was developed to obtain the schedule for specific courses from TimeEdit, as direct access to their API was not available. An alternative solution that was brought up apart from the web scraper, was to use the schedule that is provided on Canvas. This would be done using the Canvas LMS API to access the schedule for a specific course. This idea was quickly erased due to all courses did not upload their schedule on Canvas, and was therefore deemed as an unreliable solution.

The process initiates when the TimeEdit endpoint is invoked this will be explained later. Through this endpoint, the scraper retrieves a list of course codes for which the user seeks scheduling information. Once the list is acquired, the scraper opens a browser window with the TimeEdit URL.

Upon the page loading successfully, the scraper inputs each course code into the designated input field and selects the corresponding course iteratively. After all course codes are entered, the scraper activates the button to display the schedule. Once the new page is loaded, the scraper locates and selects the appropriate button to download the schedule in the desired format.

#### 4.3.8.3 Schedule Extraction

The web scraper downloads the schedule in the ICS (iCalendar) format, which is the standard file format for calendar data exchange. Each event within the ICS file, such as lectures and exercises, is then parsed into objects containing specific details: title, summary, start time, end time, room, and building. These objects are compiled into a list.

Once all events have been converted into objects, the original ICS file is deleted. The list of event objects is then serialized into a JSON file, which is subsequently uploaded to a Google Cloud Bucket. This allows the schedule data to be accessible to all users via the cloud storage service.

#### 4.3.8.4 Endpoints

To access the endpoints, a URL is needed. This is the application's base URL:

**https://us-central1-pafus-421113.cloudfunctions.net/gcp-func-pafus**

To access an endpoint the base URL is combined with a fetch request to retrieve the information or data that the user has requested.

The TimeEdit endpoint is pivotal for the application to work. It has the task of retrieving the user's schedule from TimeEdit for the application to display. The TimeEdit endpoint looks like this:

**GET /events?code=*ABC123*&code=*DEF456...***

In the provided URL, the endpoint "/events" signifies the server location where data related to events is stored and accessed. The question mark (?) acts as a delimiter, separating the URL path from the parameters passed to the server via the fetch call. Within the query string, the parameter named "code" is specified, allowing one or more course codes to be transmitted to the server. The ampersand (&) determines how many parameters are sent by separating them. These course codes, exemplified as "ABC123" and "DEF456," serve as placeholders in this illustration and represent identifiers for specific courses selected by the user. Through this URL structure, the fetch call retrieves the schedule information corresponding to the courses specified by the user.

The core functionality of the TimeEdit schedule retrieval endpoint revolves around efficiently fetching and storing schedule data for the user's courses. When called, the endpoint first checks if the schedules for the specified courses have already been scraped and stored. If the data is available, it retrieves and returns the schedules without the need for further scraping, thus optimizing performance and reducing unnecessary workload on the backend.

The first Canvas endpoint was created to allow the user to fetch their courses from the Canvas API. The application does a fetch request to the backend that looks like this:

**GET /api/courses?token=*<ACCSESS TOKEN>***

The endpoint "/api/courses" is the designated location on the server where course-related data is managed and accessed. Within the fetch call, the query string includes a crucial parameter, the user's access token, required for authentication with the Canvas API. Upon receiving the access token, the server initiates this subsequent GET request:

**GET /api/v1/courses?access_token=<*ACCESS TOKEN*>**

This secondary request retrieves a list of courses associated with the user, encompassing both current and past enrollments.

The Canvas endpoint for assignments functions similarly to the course endpoint. It is structured as follows:

**GET /api/assignments?token=<*ACCESS TOKEN*>&courseId=<*courseId*>**

The primary distinction between this endpoint and the course endpoint lies in its focus on the assignments section of the server rather than the courses section. Additionally, this endpoint includes an extra parameter, the courseId. Each courseId is unique to every course within the Canvas API, ensuring that the assignments retrieved are specific to the correct course. This parameter is essential for accurately identifying and gathering assignments relevant to the designated course. When this request is made to the backend, it initiates another request to the Canvas API formatted as:

**GET /api/v1/courses/<*courseId*>/assignments?access_token=<*ACCESS TOKEN*>**

The Canvas API responds with a list of all assignments for the specified course. This integration allows the backend to fetch and provide comprehensive assignment information for the user's courses.

#### 4.3.8.5 Algorithm for study sessions

During the planning phase of the project, it became evident that a bespoke algorithm was essential for generating custom study schedules for students, specifically tailored to their preferred study times and intensity. This algorithm was designed to stand out from typical school scheduling systems that merely display course schedules, by incorporating a more personalized and dynamic approach. Ensuring the algorithm was well-structured and modular was important, particularly to facilitate future enhancements.

The implementation of the algorithm started later in the project timeline than anticipated, primarily because it was dependent on the completion of other essential features that were necessary for its accurate deployment and functioning.

Two critical parameters were central to the algorithm's effectiveness: the course points and the corresponding weekly workload. For example, 1.5 course points would equate to four hours of workload per week, which includes lectures and related activities, based on a term length of ten weeks. The algorithm was designed to accommodate multiple course point options: 1.5, 3.0, 4.5, 6.0, or 7.5.

Another pivotal factor was determining the students' preferred study period. Based on research and feedback from the initial tests, it was decided to offer three potential study times: morning (08:00 - 12:00), afternoon (13:00 - 17:00), or evening (18:00 - 22:00). The algorithm was tailored to generate schedules within these specific periods. For students managing heavy course loads, such as two courses at 7.5 points each, the algorithm allowed for scheduling up to eight hours a day to minimize the number of study hours during the weekend effectively.

The complexity of the algorithm extended beyond merely allocating study hours; it also had to accurately account for the actual hours spent on lectures and other course-related activities to calculate the total required study time appropriately. Moreover, challenges arose in formatting dates correctly for seamless integration with the calendar, which demanded a specific format to display the scheduled events effectively.

#### 4.3.8.6 State management with Redux

Before the implementation of the app began, there was a discussion about how the app should manage states and how different components would need to access the same data. After consulting with the company's supervisor, the decision was made to use Redux as the solution to this problem.

Instead of passing data through props from parent to child components, which can become confusing and hard to debug, Redux provides a centralized place called the store where all the app's state/data is stored. Now, any component that needs to share data can directly access the state from the store.

To manage state changes in Redux, actions, and reducers are defined. Actions describe what happened, and reducers specify how the state should change in response to those actions. This structure keeps state management clean and predictable, making it easier to track bugs and test the app, especially as it grows and becomes more complex. Down below in Figure 4.15 is basic a illustration of how state management works with Redux.

For example, some screens in the app needed to share data about the courses. Since course data is fetched from an API, the first screen that encounters this data dispatches it to the Redux store. This allows other parts of the app to retrieve the same data. When handling different types of data, such as courses and events, it is preferable to have separate slices to structure the data.

Each slice contains parameters, reducers, and actions related to its specific data. For instance, an event slice might have parameters like start time, end time, title, and summary, and reducers such as setEvent, addEvent, or removeEvent. Each slice is connected to the store, ensuring that states update correctly. When different parts of the app share data, they both dispatch and retrieve information from the store, ensuring consistent state updates throughout the app.

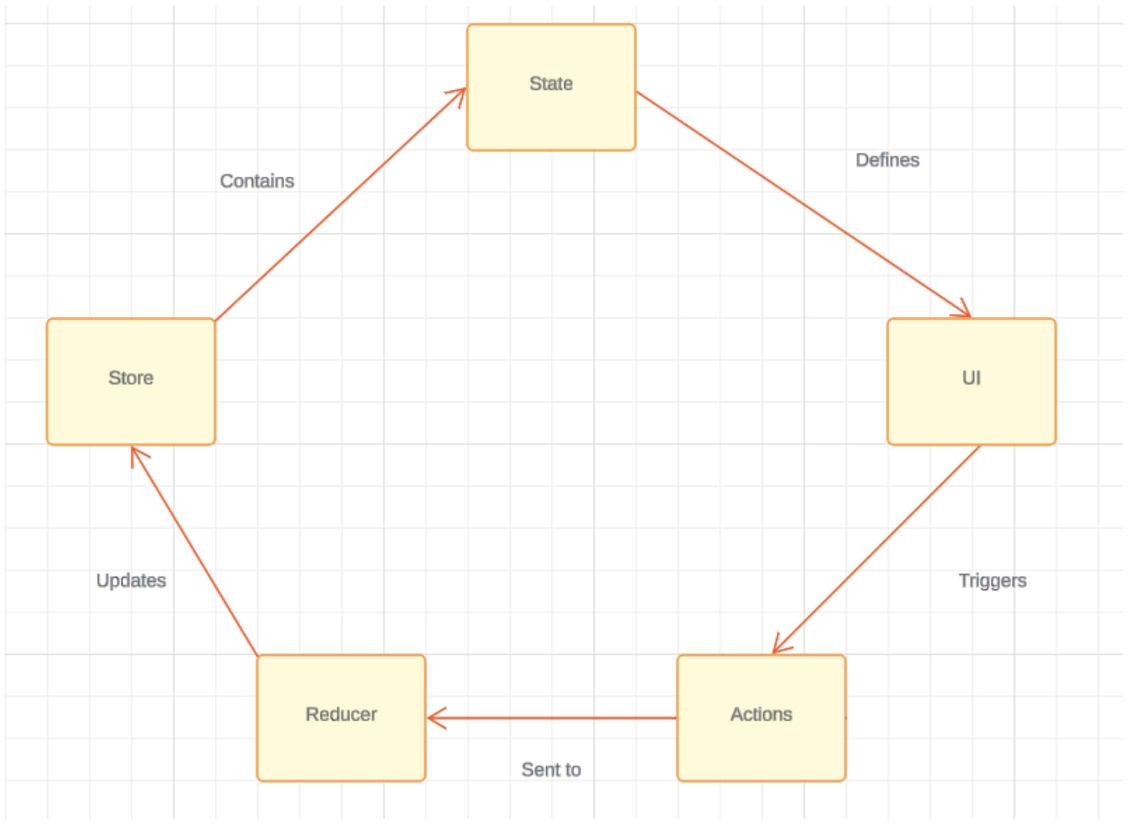


**Figure 4.15:** Illustration of how state management works in React-Native with Redux

#### 4.3.8.7 Redux Slices

The Course Slice stores all course-related information. Key parameters include course ID, course name, course code, term, favorite status (indicating if the course is active), course points, exam date, and the start and end times for exams. Actions are provided for setting courses, updating course details, and toggling the favorite status, ensuring accurate and up-to-date course data.

The Event Slice handles information fetched from the TimeEdit web scraper. Parameters include the start and end times for the event, title/course code, summary/course name, room, house, and location, which specify where the event is taking place. An action for setting events is included, facilitating the display of events in the calendar.

Custom Event Slice manages data related to user-added activities and study sessions suggested by the algorithm. It includes four parameters: start and end times for the custom event, title, and summary. Actions for adding new custom events and clearing all custom events are provided to keep the custom event list current.

This slice manages exam event data for calendar display. The main parameters are the start and end times for exams, formatted as YYYY-MM-DDTHH:MM (e.g., 2024-05-10, 08:00 - 12:00), along with the title and summary for each exam event. An action to add new exam events is included, allowing courses with exam dates to be added to the calendar.

The Study Time Slice oversees user preferences for study times. It contains two parameters: one for the primary study time choice and one for the secondary choice. Two actions are available, one for setting the primary time and another for setting the secondary time, enabling users to update their study preferences independently.

This slice handles user preferences for study styles. It has a single parameter to store the user's preferred study method. An action for setting the preferred value is included, ensuring the user's study style choice is recorded accurately.

## 4.4 Testing

### 4.4.1 User testing

At the onset of the project, establishing an effective testing protocol was challenging due to initial ambiguities concerning the project's objectives and a clear development roadmap. With the clarification that the primary goal was to create a user-friendly application, the role of user testing was prioritized to ensure the app met user needs effectively.

User tests were planned and executed to gather initial reactions and detailed feedback on the app's design and functionality. These sessions involved participants using the app prototype, navigating between different screens, and verbally expressing their thought processes. This approach was instrumental in identifying immediate user reactions and areas needing enhancement. Key questions included:

- Pre-use expectations: What are your study habits and expectations from an educational app?
- Post-use impressions: What did you like or dislike about the app? Which features were most useful?

The initial tests were conducted using a Figma prototype to overcome the unavailability of a fully functional platform. This phase was great for gathering early feedback despite the limited functionality. The second rounds of tests were carried out using an iPhone emulator, accommodating the practical constraints of device availability. Although testing with a real device would have been ideal for the best possible user experience, each session nonetheless provided vital insights that were instrumental in progressively refining the app.

# 5 Results

## 5.1 Overview of Testing

Following the initial setup and planning described in the Development Process, the user testing sessions generated substantial feedback that shaped the evolution of the application. The primary objective of these tests was to refine the application based on user feedback and ensure the functionality and design aligned with user expectations.

User feedback was integral from the early stages using a basic Figma prototype to later stages with a more developed emulator version. Participants provided valuable insights into the app's usability and design, which were critical in directing the development efforts. Feedback highlights included:

- Design adjustments made in response to user preferences.
- Functional enhancements inspired by user difficulties and suggestions.
- Overall satisfaction ratings that informed the final adjustments before completion.

The testing progressed through several phases, each marked by an increase in the application's complexity and a closer approximation to the final product. By the final rounds of testing, the application had incorporated significant improvements based on earlier feedback, offering a nearly finalized version for user evaluation. This iterative process ensured that the application met the high standards expected by both the development team and the end users.

For all user testing rounds, five to six participants were engaged, with sessions conducted individually to ensure feedback was unbiased and reflective solely of each user's personal experience. A safe and quiet environment was maintained for each session, allowing participants to engage with the testing process without feeling rushed. This approach was consistent across both in-person and online formats. Initially, in-person tests utilized a Figma-based prototype to gather early-stage feedback. As development progressed, these sessions transitioned to using a real device to enhance the authenticity of the user experience. For online sessions, an iPhone emulator was employed, providing a close approximation to real device interaction, thereby ensuring the feedback remained highly relevant and useful for refining the application.

## 5.2 Ease of Use and Interface Usability

The first two rounds of user tests revealed several insights into the app's usability and interface design. Users universally praised the design for its clean aesthetic and relevance. However, concerns were raised about the color scheme and certain layout choices, which could impact the app's overall accessibility. For instance, the homepage layout confused some users, complicating navigation between different screens. Although the interface was considered straightforward by most, the feedback indicated a desire for customizable options to accommodate varied user preferences.

For the final rounds of testing, some bugs were initially discovered that had not been spotted before. However, users appreciated the app's overall design, finding it user-friendly and smooth. The issues regarding color schema and certain layout choices were no longer a problem since these issues were resolved after receiving feedback from the second round of tests, which boosted the overall user experience.

## 5.3 Feature Relevance and Functionality

During the initial testing phases, despite missing features, users responded favorably to existing functionalities such as the study planner and the homepage's capacity to display significant amounts of relevant information. There were concerns about the integration of TimeEdit and the Canvas API, which were not demonstrated in these rounds. The ability to tailor schedules to personal preferences was well-received. Additionally, there was interest in features that would allow schedule adjustments in response to conflicting activities. When asked which parts of the app were most effective, responses varied: some preferred the calendar for its ease of use and customization options, while others favored the homepage for its relevant features and straightforward design. The clear display of deadlines and important dates was also noted as particularly beneficial.

The most praised feature of the final version of the app was the calendar. Users appreciated the overall design and the features associated with the calendar, noting that its familiarity with the standard iPhone calendar made navigation simple and smooth. There were no longer any concerns about the integration of TimeEdit and the Canvas API since all of these features now worked as expected. Users also appreciated the ability to edit custom events in the calendar, enhancing the user experience and the customization of the calendar.

## 5.4 User Engagement and Satisfaction

Feedback from user testing indicated that the app was moderately engaging, which was anticipated given its ongoing development and incomplete state. As depicted in the accompanying graph, user engagement scores ranged from 5/10 to 8/10. Although an 8/10 score is commendable at such an early stage, there is still ample opportunity for enhancement to increase engagement across all users. The feedback highlighted the app's clear and effective user interface and the utility of the homepage as positive features. However, issues such as an unsatisfactory color scheme and confusion in navigating between screens detracted from the overall experience. Despite these areas for improvement, the feedback from the initial testing rounds was generally positive, with many favorable ratings.

The overall ratings from the final rounds of testing were around 7/10 and 8/10. For an app that is still in beta and far from perfect, these scores are fantastic. They indicate that users appreciate the overall concept and design of the app. However, the ratings also highlight areas for improvement in user experience and functionality, which could lead to higher satisfaction scores in the future. Since issues such as bad color scheme and confusing navigation were now resolved, the feedback from the users was very positive which the overall score for the app indicates.

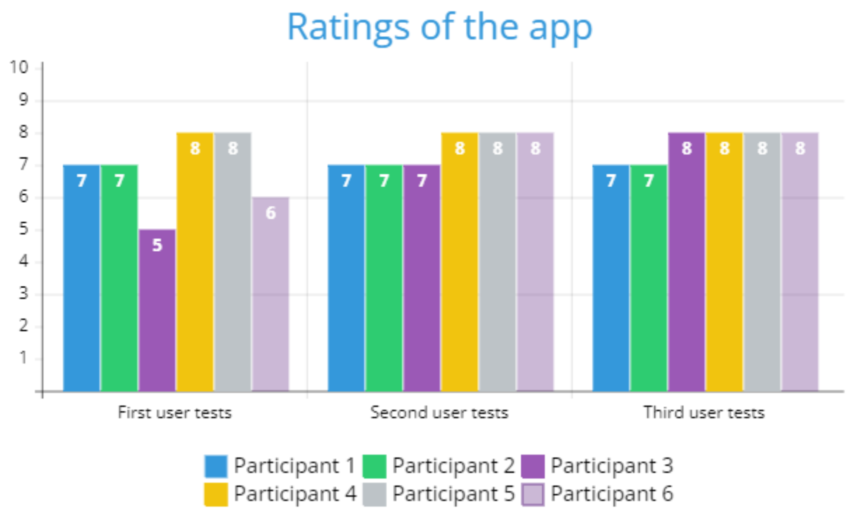


**Figure 5.1:** The overall ratings of the app from participants for each test round

## 5.5 Challenges and Issues Identified

Given that the app was still in its early development stages during the initial two test rounds, it was evident there was substantial room for improvement. The tests uncovered several challenges and issues. Immediate visual concerns were noted with the aesthetics of certain components and their color choices. Users also identified functionality issues, particularly with the calendar; missing features hindered customization according to user preferences. Additionally, the assignment section was criticized for lacking crucial features, such as a detailed view that provides more information like assignment descriptions. Feedback from the homepage revealed problems with the schema presented, including the absence of a swipe function to dismiss past activities. Despite these numerous issues, the challenges identified were invaluable, providing critical feedback and highlighting opportunities for significant enhancements, aligning with the core objectives of these tests.

Despite testing the final version of the app, there were still some issues and areas for improvement. Users identified some undetected bugs that impacted their experience. They also expressed a desire for custom schedule options and suggested making the button for receiving a custom study schedule optional. Additionally, users recommended a feature to import personal calendars into the app, ensuring all planned events are consolidated in one calendar.

# 6 Conclusion/Discussion

## 6.1 Summary

The main objective of this project was to develop a user-friendly mobile application tailored to the needs of students. The aim was to ensure that the app was relevant to the target audience by incorporating user research and feedback, focusing on evidence-based features for effective time management, and maintaining scalability and adaptability for diverse users.

For the frontend development, React-Native was selected for its cross-platform capabilities. The design process utilized Figma for creating the app's design, ensuring a user-centric interface.

The backend development used NodeJS for its efficient, non-blocking architecture, suitable for handling concurrent connections. The backend implementation also involved the Canvas LMS API for retrieving information about courses and assignments for users, and a web scraper to retrieve scheduling information from TimeEdit.

User testing was conducted to gather feedback and refine the app. Initial prototypes were tested using Figma, which later on progressed to an emulator or a real-time device. The feedback was collected through pre- and post-use questions, ensuring that the app progressively improved.

The key findings from the user testing sessions highlighted that the app's design was generally appreciated for its clean and intuitive interface. However, early feedback indicated issues regarding the color schema and layout choice for some components. The main features that were well-received were the study planner and homepage. However, there were concerns about API integration and missing functionalities during the first test rounds, but these concerns vanished later on together with the concerns regarding the color schema and layout choices. The lowest score the app received was a 5/10, and the highest was an 8/10.

Through iterative testing, the project successfully created a mobile application that meets students' needs through a user-centered design. While the overall feedback received from users was positive, several areas were identified that needed improvement. By taking in all the feedback, the project underscored the importance of frequent user testing, to create a relevant and effective application.

## 6.2 Critical Discussion

### 6.2.1 Strengths and Weaknesses

One of the main strengths of this project lies in its design and features. By involving students in the development process and having iterative user testing, the project ensured that the application was tailored to meet its target audience's specific needs and preferences, ensuring that the app has a user-centered design approach. Using React-Native as the primary frontend technology ensured a cohesive and responsive user interface across both Android and iOS platforms, significantly reducing development time. To further enhance the app's robustness and potential long-term use, the app uses evidence-based features for time management and its structure makes sure that the app is scaleable and adaptable for potential future change. Additionally, the backend development uses NodeJS which offers efficient handling of multiple concurrent connections and smooth integration of third-party libraries, ensuring a stable and scaleable system.

Despite the strengths of the app, the project faced several limitations. First, the sample size for the user testing could have been larger, with only five to six participants per testing round. A larger sample size could have increased the diversity and uniqueness of the feedback and may represent the opinions of a broader student population. Second, the methodology faced some constraints regarding the integration of the TimeEdit API, leading to the use of web scraping as an alternative solution, which long term may not be the most suitable solution. The best-case scenario is to use the TimeEdit API and access the relevant data. Additionally, some potential biases may have inflicted the feedback since it was the same participants for each test round. Although the algorithm largely functioned as intended, there were some flaws that the development team did not anticipate and did not have time to address. For instance, the algorithm does not account for courses that span two quarters. The algorithm treats these courses as if they last only one quarter, which affects the study schedule it generates. Consequently, a course like LEU432 (Introduction to Computer Engineering), which is worth 7.5 course points and is intended to span two quarters, is incorrectly treated as a 7.5 course point course for each quarter. By addressing these limitations in the future, the overall effectiveness and user satisfaction will hopefully be improved.

## 6.3 Evaluation of Results and Approach

### 6.3.1 Results Evaluation

The project achieved its primary objective of developing a user-friendly app with features like customizable study plans, assignment reminders, and integration with Canvas and TimeEdit. User feedback confirmed the usefulness and relevance of these features, particularly appreciating the study planner and the interface designed with React-Native.

Despite these successes, there were some deviations from the expected outcomes. While the Canvas LMS API was successfully integrated, issues with the TimeEdit API led to the adoption of web scraping as an alternative solution. This approach, though effective in the short term, is less reliable than direct API integration. Additionally, the project experienced a time management challenge, leading to only three rounds of user testing instead of the planned four. Nevertheless, the feedback from these tests provided valuable insights that significantly influenced the app's development.

### 6.3.2 Planning and Approach Evaluation

When reflecting on the overall research process for the project, a lot of aspects worked well, while others presented some difficulties and needed further research. The overall design of the app which focused on a user-centered design worked well and was highly effective. The involvement of students in the development process and having user testing rounds were deemed to be efficient, ensuring the app met the users' needs and preferences. The proposal from the company supervisor of using React-Native for frontend development and NodeJS for backend development proved to be robust and efficient, making sure that the project is scalable and adaptable, aligning with the project goals.

However, there were areas that presented some issues and did not work as expected. The initial plan underestimated the integration of some APIs, discovering the unavailability and complexity that was not anticipated at the start. Therefore, alternative solutions had to be implemented such as web scraping, to ensure the app worked as intended. This highlights the importance of thorough preliminary research, making sure that alternative solutions can be discovered at the beginning of the project and not during the development process.

Yet another challenge that arose was the limited sample size of users who tested the app, which may have resulted in not capturing the full diversity of user needs and preferences. To counteract this in future projects, expanding the sample size pool would be ideal, ensuring that the feedback and insights are reflected from a broader range of participants.

Overall, to improve the approach in future projects, it would be beneficial to:

- Conduct more thorough and extensive studies for all planned integrations and technologies.
- Increasing the sample size of participants testing the app.
- Having contingency plans in case of potential challenges, such as alternative methods for data integration.

Although the project achieved its primary objectives and produced a functional and user-friendly app with relevant features, these assessments gave a valuable lesson for improving the planning and development process in future projects.

## 6.4 Future Research

### 6.4.1 Recommendations for Further Research

Here are some areas that could be further investigated to build upon this project's findings:

1. Move on from web scraping as the main methodology for receiving schedule information from TimeEdit. The most ideal scenario is to convince Chalmers to get access to the TimeEdit API ensuring a permanent, stable, and reliable solution for retrieving data from TimeEdit.

2. Instead of having a rule-based algorithm, investigate the potential of a machine learning algorithm to provide personalized study plans. This could potentially significantly enhance the app's ability to meet a variety of user needs and preferences, improving the overall educational outcome.

3. Expanding the sample size for user testing would definitely help to capture a wider range of user preferences and hopefully receive valuable feedback. If the design and functionality are still in the future scope, this would definitely help the project in the future.

These areas address the core challenges and opportunities identified during the project, and by addressing these areas the app's effectiveness, satisfaction, and overall impact will be improved.

# Bibliography

[1] allastudier.se, “Här hoppar flest av,” *allastudier.se*, Jul. 10, 2021. [Online]. Available: `https://allastudier.se/tips-o-fakta/h%C3%A4r-hoppar-flest-av-16910` (Retrieved 2024-02-16)

[2] U. Jönsson and B.-M. Johansson, “Varför hoppar vissa studenter av sin högskoleutbildning” *Malmö högskola*, [Online]. Avaliable: `https://www.diva-portal.org/smash/get/diva2:1490537/FULLTEXT01.pdf`. [Accessed: 2024-02-16]

[3] Högskolan i Halmstad, "Så hanterar du tentastress och hittar balans i pluggandet," 2021. [Online]. Available: `https://www.hh.se/student/studentnytt/studentnytt/2021-10-04-sa-hanterar-du-tentastress-och-hittar-balans-i-pluggandet.html`. [Accessed: 2024-02-16]

[4] React Native, "React Native · A framework for building native apps using React," *reactnative.dev*, 2022. [Online]. Available: `https://reactnative.dev/`. [Accessed: 2024-02-18]

[5] Better React, "React Native: Building Cross-Platform Mobile Apps with React," *Better React*, 1 Dec. 2023. [Online]. Available: `https://betterreact.dev/insights/react-native-building-cross-platform-mobile-apps-with-react/`. [Accessed: 2024-04-28]

[6] AppFirmsReview, "Benefits of Choosing React Native for Cross-Platform App Development," *AppFirmsReview*, 15 June 2022. [Online]. Available: `https://appfirmsreview.com/benefits-choosing-react-native-cross-platform-app-development/`. [Accessed: 2024-04-28]

[7] Figma Learn, "What is Figma?," [Online]. Available: `https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma`. [Accessed: 2024-02-18]

[8] Node.js, "About," *Node.js*, [Online]. Available: `https://nodejs.org/en/about`. [Accessed: 2024-02-18]

[9] Kinsta, “What Is Node.js and Why You Should Use It,” *Kinsta*, 7 Nov.2023.[Online]. Available: `https://kinsta.com/knowledgebase/what-is-node-js/`. [Accessed: 2024-05-08]

[10] Technostacks, "Top Mobile App Development Frameworks," *Technostacks*, 2024. [Online]. Available: `https://technostacks.com/blog/mobile-app-development-frameworks/`. [Accessed: 2024-05-08]

[11] Meta Platforms, Inc. "Getting Started with React Native," *React Native Documentation*, 15-May-2024. [Online]. Available: `https://reactnative.dev/docs/getting-started`. [Accessed: 2024-05-08]

[12] Meta Platforms, Inc. "Learn the Basics of React Native," *React Native Documentation*, 22-Apr-2024. [Online]. Available: `https://reactnative.dev/docs/tutorial`. [Accessed: 2024-05-08]

[13] Redux.js, "Getting Started with Redux," *Redux Documentation*, 31-Mar-2024. [Online]. Available: `https://redux.js.org/introduction/getting-started`. [Accessed: 2024-05-08]

[14] Redux.js, "Redux Fundamentals, Part 1: Redux Overview," *Redux Documentation*, 25-Nov-2023. [Online]. Available: `https://redux.js.org/tutorials/fundamentals/part-1-overview`. [Accessed: 2024-05-08]

[15] Node.js, "Introduction to Node.js," *Node.js Documentation*, 2024. [Online]. Available: `https://nodejs.org/en/learn/getting-started/introduction-to-nodejs`. [Accessed: 2024-05-09]

[16] Bloomreach, "What Is a Single Page Application?," *Bloomreach Blog*, 16-Sep-2022. [Online]. Available: `https://www.bloomreach.com/en/blog/2018/what-is-a-single-page-application`. [Accessed: 2024-05-09]

[17] J. McMinn, "Top Use Cases for Node.js: Unlocking Its Full Potential," *Matchbox Design Group Blog*, 30-Mar-2023. [Online]. Available: `https://matchboxdesigngroup.com/blog/use-cases-for-nodejs/`. [Accessed: 2024-05-09]

[18] Amazon Web Services, Inc, "What is an API?," *Amazon Web Services Documentation*, 2024. [Online]. Available: `https://aws.amazon.com/what-is/api/`. [Accessed: 2024-05-12]

[19] H. Subramanian and P. Raj, "Advantages and disadvantages of statelessness." *Hands-On RESTful API Design Patterns and Best Practices*, O’Reilly Media, 2018. [Online]. Available: `https://www.oreilly.com/library/view/hands-on-restful-api/9781788992664/`

fff729a2-d426-4c0f-96a1-0c2a181941de.xhtml. [Accessed: 2024-05-12]

[20] Instructure, Inc, "Canvas LMS REST API Documentation," *Canvas Documentation*, 2024. [Online]. Available: https://canvas.instructure.com/doc/api/. [Accessed: 2024-05-12]

[21] Dotcom-Monitor, "What Are API Endpoints? Why Are They Important?," *Dotcom-Monitor Blog*, 2023. [Online]. Available: https://www.dotcom-monitor.com/learn/what-are-api-endpoints/. [Accessed: 2024-05-12]

[22] Prisma Data Guide, "Google Cloud Functions," *Prisma Data Guide*, 2024. [Online]. Available: https://www.prisma.io/dataguide/serverless/serverless-comparison#google-cloud-functions. [Accessed: 2024-05-12]

[23] A. Khalfe, "Serverless Computing: AWS Lambda, Azure Functions, and Google Cloud Functions," *Talent500 Blog*, 30-Nov-2023. [Online]. Available: https://talent500.co/blog/serverless-computing-aws-lambda-azure-functions-and-google-cloud-functions/. [Accessed: 2024-05-12]

[24] GeeksForGeeks, “What is Web Scraping and How to Use It?,” *GeeksforGeeks*, Jun. 22, 2020. [Online]. Available: https://www.geeksforgeeks.org/what-is-web-scraping-and-how-to-use-it/. [Accessed: 2024-05-12]

[25] E. Hasson, "Is Web Scraping Illegal? Depends on Who You Ask," *Imperva Blog*, 7-Dec-2023. [Online]. Available: https://www.imperva.com/blog/is-web-scraping-illegal/. [Accessed: 2024-05-12]

[26] A. S. Gillis, "What is an algorithm?," *TechTarget WhatIs*, 2023. [Online]. Available: https://www.techtarget.com/whatis/definition/algorithm. [Accessed: 2024-05-15]

[27] Johns Hopkins Medicine, "Brain Anatomy and How the Brain Works," *Johns Hopkins Medicine Health Library*, 2024. [Online]. Available: https://www.hopkinsmedicine.org/health/conditions-and-diseases/anatomy-of-the-brain. [Accessed: 2024-05-03]

[28] R. Hirsh, “The hippocampus and contextual retrieval of information from memory: A theory,” *Behavioral Biology*, vol. 12, no. 4, pp. 421–444, Dec. 1974, doi: https://doi.org/10.1016/s0091-6773(74)92231-7. [Accessed: 2024-05-03]

[29] D. Dovey. "When Does The Brain Work Best? The Peak Times And Ages For Learning." *Medical Daily*, 8-Aug-2016. [Online]. Available: `https://www.medicaldaily.com/when-does-brain-work-best-peak-times-and-ages-learning-394153`. [Accessed: 2024-05-03]

[30] LSBF, "What is the best time for studying—day or night?," *LSBF Blog*, 11-Jan-2024. [Online]. Available: `https://www.lsbf.edu.sg/blog/motivation-and-inspiration/what-is-the-best-time-for-studying-day-or-night`. [Accessed: 2024-05-03]

[31] K. K. Randolph, "Energizing Study Break Ideas & What to Avoid," *Fastweb Blog*, 11-Jan-2023. [Online]. Available: `https://www.fastweb.com/student-life/articles/energizing-study-break-ideas-what-to-avoid`. [Accessed: 2024-05-03]

[32] StudySmarter, "Study Breaks: Importance, Productivity, Ideas," *StudySmarter Magazine*, 2024. [Online]. Available: `https://www.studysmarter.co.uk/magazine/study-breaks/`. [Accessed: 2024-05-03]

# A

# Appendix 1

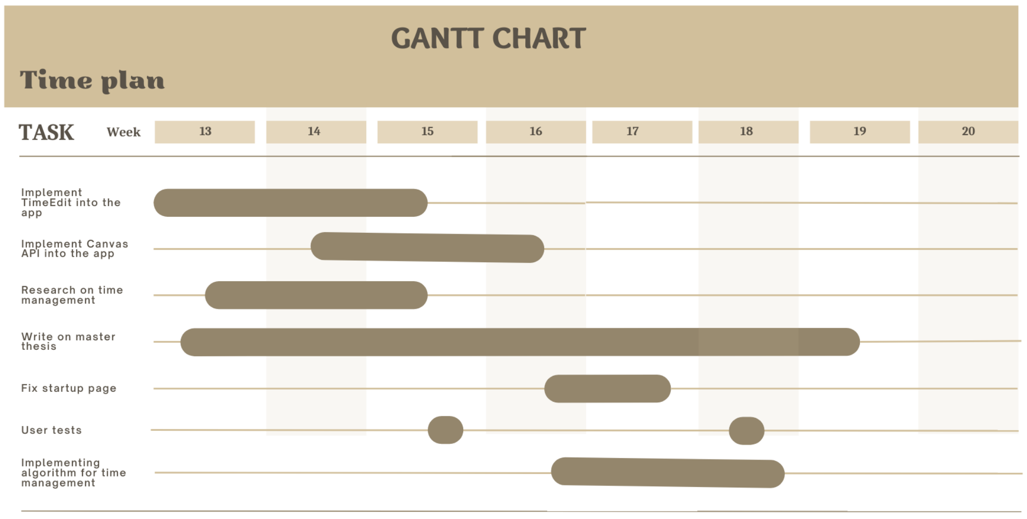
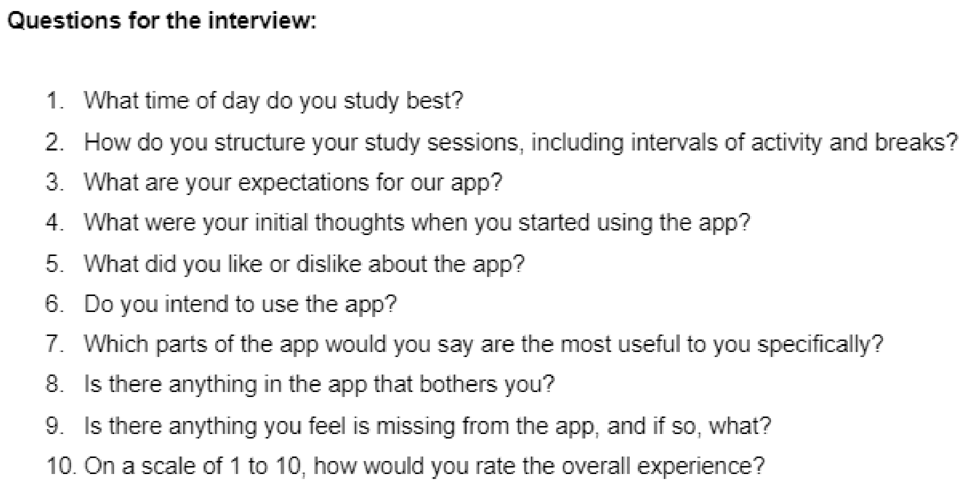


**Figure A.1:** Time plan for the project



**Questions for the interview:**

1. What time of day do you study best?
2. How do you structure your study sessions, including intervals of activity and breaks?
3. What are your expectations for our app?
4. What were your initial thoughts when you started using the app?
5. What did you like or dislike about the app?
6. Do you intend to use the app?
7. Which parts of the app would you say are the most useful to you specifically?
8. Is there anything in the app that bothers you?
9. Is there anything you feel is missing from the app, and if so, what?
10. On a scale of 1 to 10, how would you rate the overall experience?

**Figure A.2:** The questions asked under the user test interview