



ScheduSmart

CS307 Design Document

February 9th, 2024

Team 30

Team Leader: Reece Ausmus

Reece Ausmus
Stanley Huang
Cassie Chang
Bradley Norris
Himanshu Sinha
Gloria Xu

1. Purpose.....	3
1.1 Functional Requirements.....	4
1.2 Non-functional Requirements...	5
2. Design Outline.....	7
2.1 High Level Overview.....	7
2.2 Sequence of Events Overview..	8
3. Design Issues.....	9
3.1 Functional Issues.....	9
3.2 Non-functional Issues.....	13
4. Design Details.....	16
4.1 Class Design.....	16
4.2 Descriptions of Classes and Interaction between Classes.....	17
4.3 Sequence Diagrams.....	20
4.4 UI Mockups.....	26

1 Purpose

In a rapidly evolving world where the demands of everyday life become increasingly complex, there arises an urgent need for a tool that unlocks our true potential for productivity. That is precisely what our innovative project aims to provide.

Introducing our groundbreaking solution that goes beyond conventional organization tools. Our project is designed to empower users in managing their lives effectively through the integration of assignment trackers and a comprehensive calendar system. But it doesn't stop there. We will use the power of generative AI to create an automatic schedule generation feature that optimizes efficiency like never before.

One of the key features of our project lies in its unparalleled level of customization. We understand that each user has unique preferences and requirements. With our user-friendly interface and extensive customization options, we offer an experience that is tailor-made for every individual and help to organize and manage his or her life an enjoyable journey.

The system we are designing has three main sections. Firstly, it will require a client-server system that allows users to access the program conveniently online, through a website. Secondly, it will have a well-designed calendar system that will allow users to effectively manage their weekly schedules, encompassing classes, social activities, work commitments, and more. At last, it will incorporate an assignment tracker allowing users to organize all of their assignments and will then sort and prioritize assignments based on workload and due date.

1.1 Functional Requirements

1.1.1 User account

As a user,

- a. I would like to create an account.
- b. I would like to retrieve my password or reset it when it is forgotten.
- c. I would like to update my profile for others to see.
- d. I would like to find other account and create a link between two account
- e. I would like to log in and log out of my account.
- f. I would like to turn on or off the event reminder.

1.1.2 Calendar system

As a user,

- a. I would like to add events and tasks on a specific date in my schedule.
- b. I would like to set a priority to my task.
- c. I would like to update my events and tasks when needed.
- d. I would like to add locations and times for events.
- e. I would like to add reminders or note on an event
- f. I would like to classify different events and tasks using different colors.
- g. I would like to add description to an event beyond name, date, time and location.
- h. I would like to delete events and tasks when needed.
- i. I would like to share my schedule with someone else.

1.1.3 Assignment tracker

As a user,

- a. I would like to track the progress of tasks.
- b. I would like to see deadlines for each assignment.
- c. I would include the workload for each assignment.
- d. I would like to share the progress with my teammates for course projects.
- e. I would like to drag and drop my tasks into my calendar to schedule time for them.

1.2 Non-functional Requirements

1.2.1 Performance

As a developer,

- a. I would like the website to run smoothly without crashing.
- b. I would like the website to support more than 30000 users.
- c. I would like the website to launch in no more than 5 secs.
- d. I would like the website to access data in no more than 7 secs.

1.2.2 Appearance

As a developer,

- a. I would like the website to have a visually appealing and modern design that aligns with current design trends
- b. I would like the users to change the website style according to their preferences.
- c. I would like the website to have a light and dark mode.

1.2.3 Security

As a developer,

- a. I would like the website to have two factor authorization.
- b. I would like the website to securely protect any personal information of users including location, email address and phone number.
- c. I would like the website to have a robust secure system that is hard to be hacked.

1.2.4 Usability

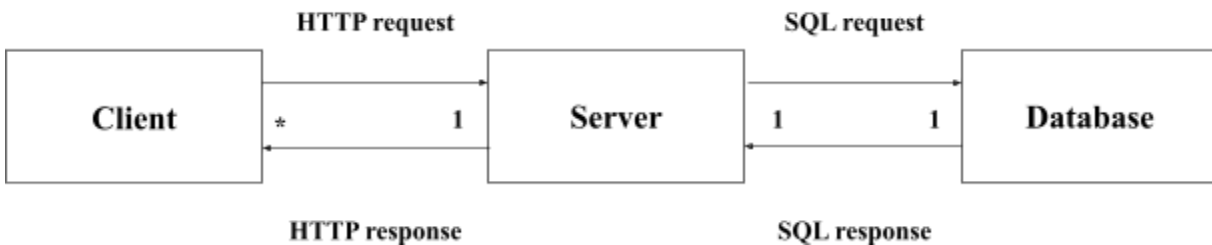
As a developer,

- a. I would like the website to be user-friendly and concise.
- b. I would like the website to provide a help feature for onboarding the user.
- c. I would like the website to support different languages.
- d. I would like the website to be able to run on any web browsers.

2. Design Outline

2.1 High Level Overview

ScheduSmart is a web application that helps users arrange their time with our product. The project utilizes the client-server model where clients can interact with the server side by sending HTTP requests. Then, the server will request data from our database that clients require. Finally, client sides can have their desired results shown on the web page of ScheduSmart thanks to the server handling the data from the database and making it well displayed on the screen.



2.1.1 Client

- Be responsible for the user interface running on the devices of users
- Sends HTTP requests to the server to get information or update new information
- Process the returned data from the server and display it on web pages.

2.1.2 Server

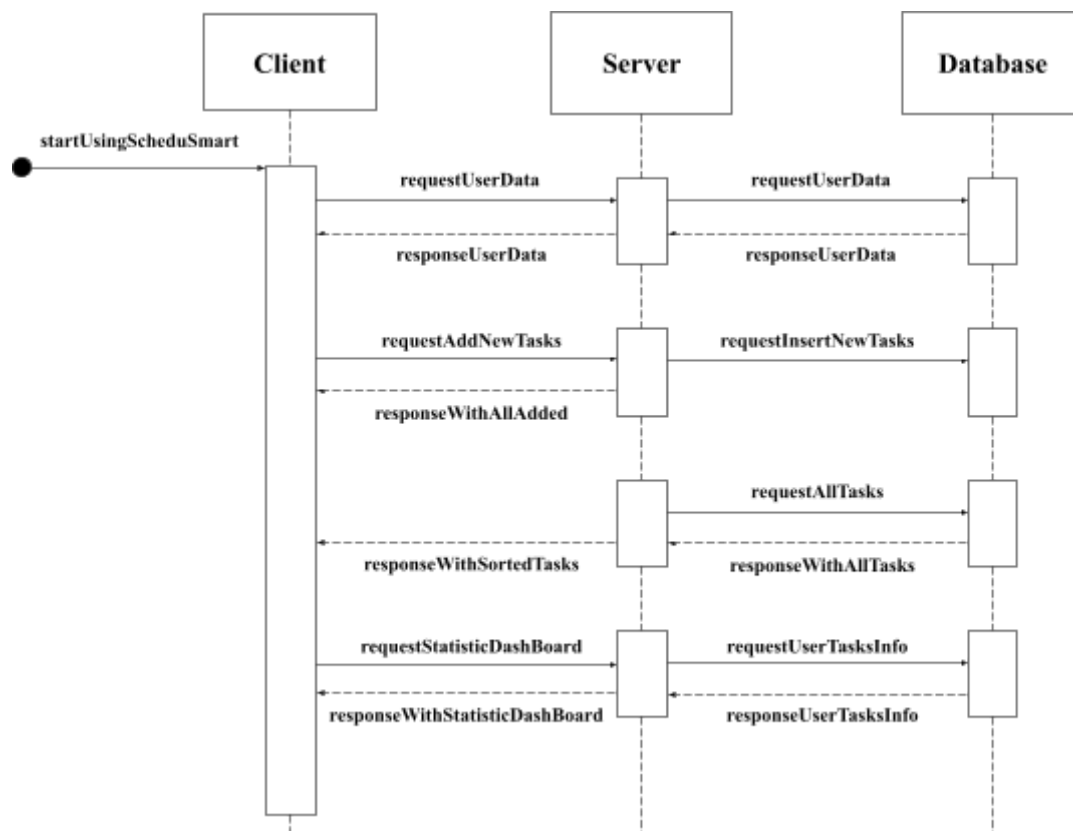
- Receives and handles HTTP requests from the client side.
- Sends query requests to the database to store, update, and retrieve data from the database
- Responses to clients with well-processed HTML pages.
- Generate dynamic changes by taking advantage of AJAX without reloading a new HTML page.

2.1.3 Database

- Stores and manages all the data claimed by our system in the NoSQL database on Firebase, such as user information, assignments and tasks of the users, calendar information, etc.
- Receives query from the server, then handling the request and response to the server with desired data.

2.2 Sequence of Events Overview

The sequence diagram below illustrates a flow that users use some features of ScheduSmart. For starters, a user logs in to our application so that the client side will send a request to the server for user information. Then, the server will send a query request to the database to get the pertinent data and response to the client side. After that, the user might want to add a new assignment to the calendar, which means that the client side has to request to add a piece of information to the database. After the server successfully inserts a new tuple into the database, it will show a new calendar with the new assignment on it. ScheduSmart can sort all the users' tasks, so the server will then request all tasks a user has and sort them for the user's reference. Finally, the user might want to check the statistical dashboard provided by our product. Accordingly, the client side will request the dashboard. Then, the server will send requests to the database for all the data related to the user and calculate and analyze them for the user.



3. Design Issues

3.1 Functional Issues

3.1.1 Comprehensive Features

Whether or not to provide comprehensive features of the assignment tracker?

- Option 1: Yes
- Option 2: No

Choice: Option 1

Recognizing the evolving needs of users, it is important for our system to go beyond basic task management features such as setting due dates, adding descriptions, marking tasks as complete and so on. Some additional features including priority levels, categorization and progress tracking could assist users in managing their tasks more effectively and stay organized. Although it is more complicated to implement. We aim to provide the best experience for all of users.

3.1.2 Account Information Needed

What kind of information is needed for creating an account?

- Option 1: Preferred username, password, email address, address, age, phone number
- Option 2: Preferred username, password, email address

Choice: Option 2

For an account creation, username, password and email address are required. The username and email address are used for identity verification, while the password ensures secure login. Some other information such as age and phone number etc. are optional to provide or they can be updated after creating an account in their profiles.

3.1.3 Two Factor Authentication

Is two factor authentication provided or not?

- Option 1: Yes, we plan to implement two-factor authentication
- Option 2: No, we do not plan to implement two-factor authentication

Choice: Option 1

We plan on including two factor authentication. Any form of MFA is necessary in today's modern technological landscape. Systems that do not include some form of MFA are highly susceptible to those with malicious intentions. We will implement two factor authentication through a simple email system. When a user attempts to login-in, they will be required to provide their username and password. Once the system confirms that their username and password are correct, it will then send an email to their account's provided address. This email will include a randomly-generated, secure code that will then be needed to finish login. Once the user provides the correct code, they will be logged in successfully.

3.1.4 Password Retrieval Mechanism

How to design a password retrieval mechanism?

- Option 1: By sending an email that allows users to reset passwords
- Option 2: Answer predefined password retrieval questions

Choice: Option 1 and 2

We plan to design both. Those two options have different functionalities and can work together to assist users in retrieving passwords. During the account creation process, we will provide a list of predefined password retrieval questions. Users can select one question and provide an answer, which will be securely stored. In the event that users forget their password, they can use this

answer to recover it. However, if users forget the answer to their retrieval question, they will have an alternative option. They can choose to receive an email that will allow them to reset their password. This additional measure ensures that users have a reliable method to regain access to their accounts, even if they cannot recall the answer to their retrieval question.

3.1.5 Duplicate Usernames

Whether or not the system allows duplicate usernames?

- Option 1: Yes
- Option 2: No

Choice: Option 2

According to our design, we decide to use username to verify identity. Hence, unique usernames are needed. In addition, non-duplicate usernames are easy to store in a database which helps to simplify database management.

3.1.6 Shared Schedules and Tasks

How does an individual share schedules and tasks with other users?

- Option 1: Use email address as the identifier
- Option 2: Use username as the identifier

Choice: Option 1 and 2

In fact, both of them can serve as an identifier when a user wants to share schedules and tasks with other people. But email addresses are lengthy and usernames are more user-friendly. As a result, we plan to design a system that primarily utilizes usernames as the preferred identifier. However, email addresses will also be incorporated as an auxiliary option, providing additional

assistance when needed. This approach aims to optimize user-friendliness while ensuring flexibility in the sharing and collaboration features of our system.

3.1.7 Time Zone Differences

How to handle time zone differences in scheduling an event?

- Option 1: Allow users to set their preferred time zone.
- Option 2: Automatically detect and adjust for the user's local time zone.

Choice: Option 2

Automatically detecting and then adjusting for the user's local time zone will reduce the likelihood of any scheduling errors due to human misinput. Also, this functionality enhances user convenience by minimizing the amount of input requirements to schedule an event.

3.1.8 Notifications

How to handle notifications for upcoming tasks and events?

- Option 1: Implement automatic push notifications and email notifications for upcoming events.
- Option 2: Create reminders for events inside of the interface that pop-up onto the screen of the user.

Choice: Option 1

Implementing push and email notifications will ensure that users receive timely reminders, even when they are not using the service.

3.2 Non-functional Issues

3.2.1 Web Server

What web server should we use?

- Option 1: Apache
- Option 2: Nginx

Choice: Nginx

Instead of utilizing process-driven architecture like Apache, Nginx uses asynchronous, non-blocking, event-driven architecture so that its web server has the ability to handle tons of client requests efficiently with minimal resources. Furthermore, Nginx can provide us with several useful functions, such as reverse proxy, load balancer, and caching solution. In this way, ScheduSmart can have high performance and high security thanks to these functions.

3.2.2 Backend Programming Language

What backend programming language should we use?

- Option 1: PHP
- Option 2: Flask (Python)
- Option 3: Golang
- Option 4: Perl

Choice: Flask (Python)

There are many reasons for us to choose Flask. In contrast with PHP, Flask is simple and lightweight, so we can easily understand it and do not need to have too much cost on our memory to use it. Moreover, since Flask offers an integrated development server, developers can test their code easily without the need to construct an additional server, which is good for Scrum

methodology. Compared with Golang, Flask is more flexible in choosing different libraries to use, while Golang does not have such high flexibility. For Perl, it is not active now. Few people keep using it, so it might be difficult to find someone to maintain our ScheduSmart in the future.

3.2.3 Frontend Programming Language

What frontend programming language should we use?

- Option 1: HTML + JavaScript
- Option 2: React
- Option 3: Vue
- Option 4: Swift

Choice: React

Since React applies component-based architecture, it enables developers to reuse code easily and write code efficiently. Additionally, React can have higher performance when it comes to large and dynamic data due to virtual DOM. Last but not least, React has lots of libraries to import so that developers can use them to save time.

3.2.4 Database

What database should we use?

- Option 1: MySQL
- Option 2: FireBase (NoSQL)
- Option 3: MongoDB

Choice: FireBase (NoSQL)

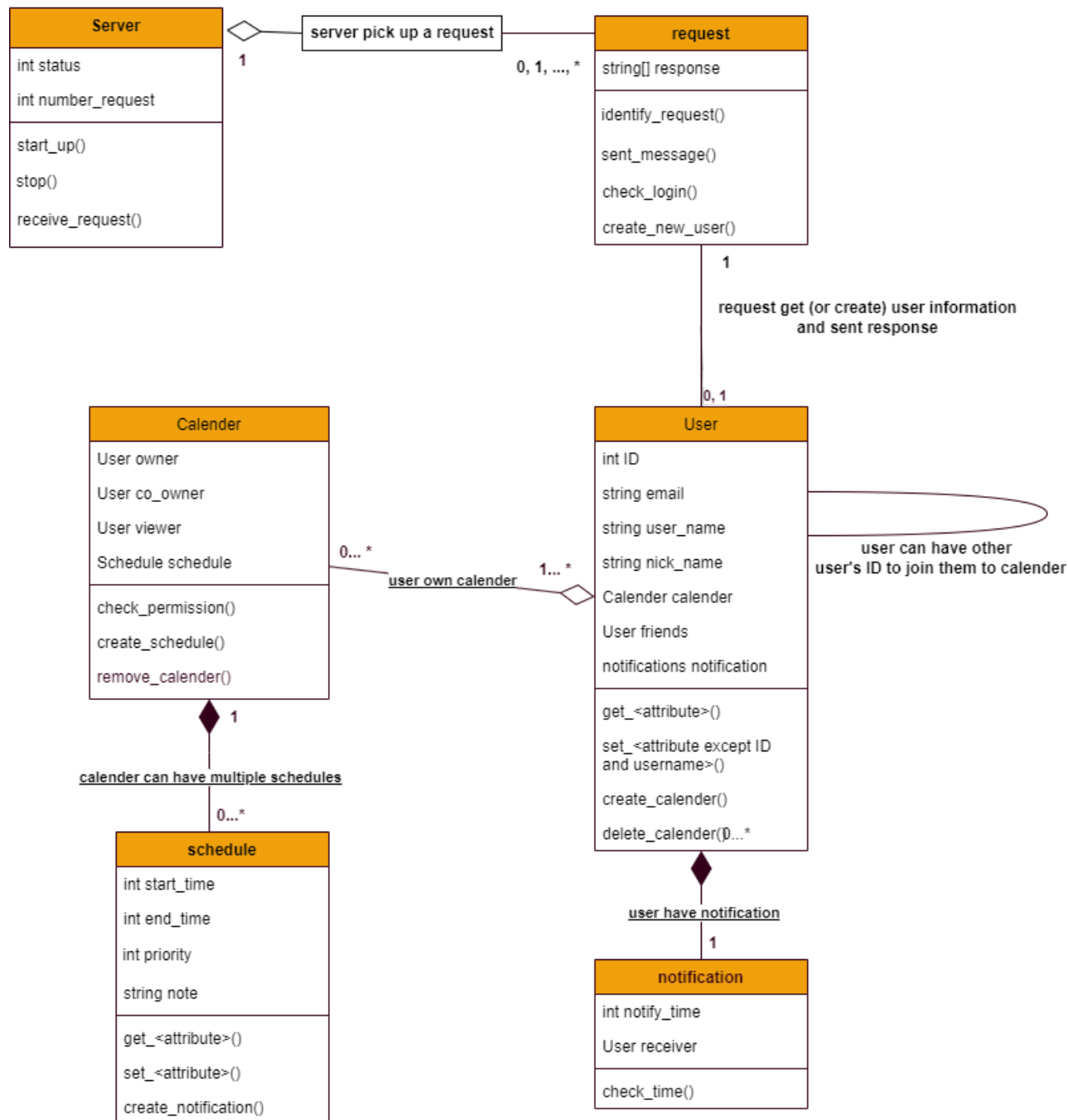
We choose FireBase as our database for the following reasons. The main reason is that it can handle authentication and authorization for our application so that users can log in through email

or Facebook. In addition, it offers analytics tools, enabling us to analyze user activities easily.

Last but not least, because FireBase is built on the Google Cloud Platform, it has the scalability to manage huge amounts of user requests.

4. Design Details

4.1 Class Design



4.2 Descriptions of Classes and Interaction between Classes

The program has 6 classes: Server, Request, User, Calendar, Schedule, and Notification.

4.2.1 Server

The server class handles all server side functionality. It is responsible for creating, managing a server online, and receiving connection requests to produce and hold multiple **Request**.

Member Variables:

- int status
- int number_request

Methods:

- start_up()
- stop()
- receive_request()

4.2.2 Request

The request class handles the other end of the server-side functionality. It takes connection requests information (HTTP) from users, parses the message, and sends the message back to the user. Effectively, it is a one to one client serving an user.

Member Variables:

- string[] response

Methods:

- identify_request()
- sent_message()
- check_login()

- `create_new_user()`

4.2.3 User

This class is effectively the central component to the whole program. It not only stores user data, including the calendar and assignment tracker, but also allows the user to edit their own information, send requests to the **server**, and update the **calendar** and assignments.

Member Variables:

- `int ID`
- `string email`
- `string user_name`
- `string nick_name`
- `Calendar calendar`
- `User friends`
- `Notification notification`

Methods:

- `get_<attribute>()`
- `set_<attribute>()`
- `create_calendar()`
- `delete_calendar()`

4.2.4 Calendar

This method holds the calendar's base information. It stores permissions, such as owner, co-owner, and those with view permissions only. It also allows you to create new schedules, and change permissions on the calendar.

Member Variables:

- User owner
- User co_owner
- User viewer
- Schedule schedule

Methods:

- check_permission()
- create_schedule()
- remove_calender()

4.2.5 Schedule

Schedule is an object that stores information about an event, including starting time, ending time, priority, and a small note if it exists. It also has a method to change that data or create notification for the user to hold.

Member Variables:

- int start_time
- int end_time
- int priority
- String note

Methods:

- get_<attribute>()
- set_<attribute>()
- create_notification()

4.2.6 Notification

The notification function interacts with the **User** and the **schedule**. It reminds the user when there is an event upcoming. It will send notifications to the user if need to.

Member Variables:

- int notify_time
- User receiver

Methods:

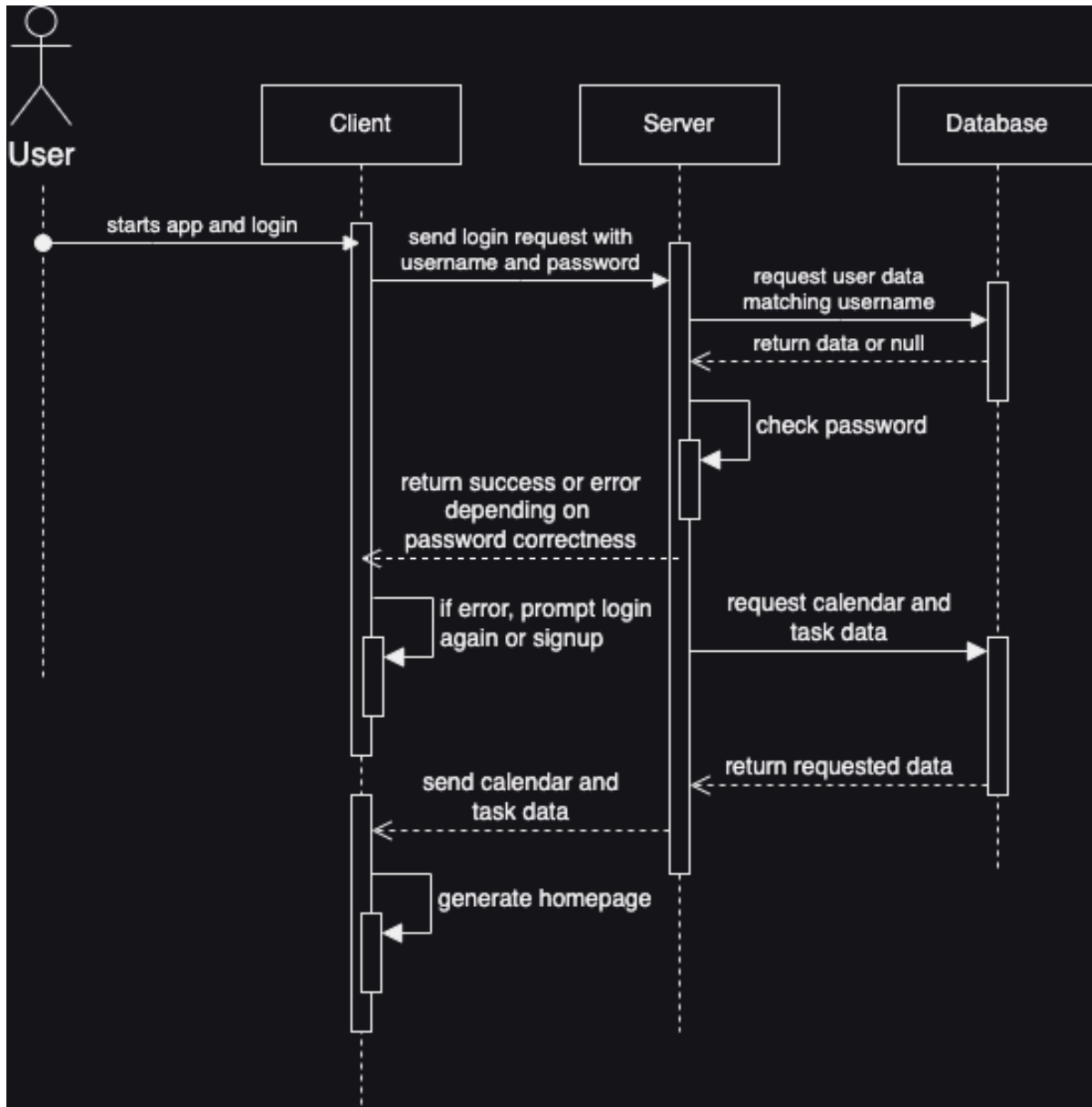
- check_time()

4.3 Sequence Diagrams

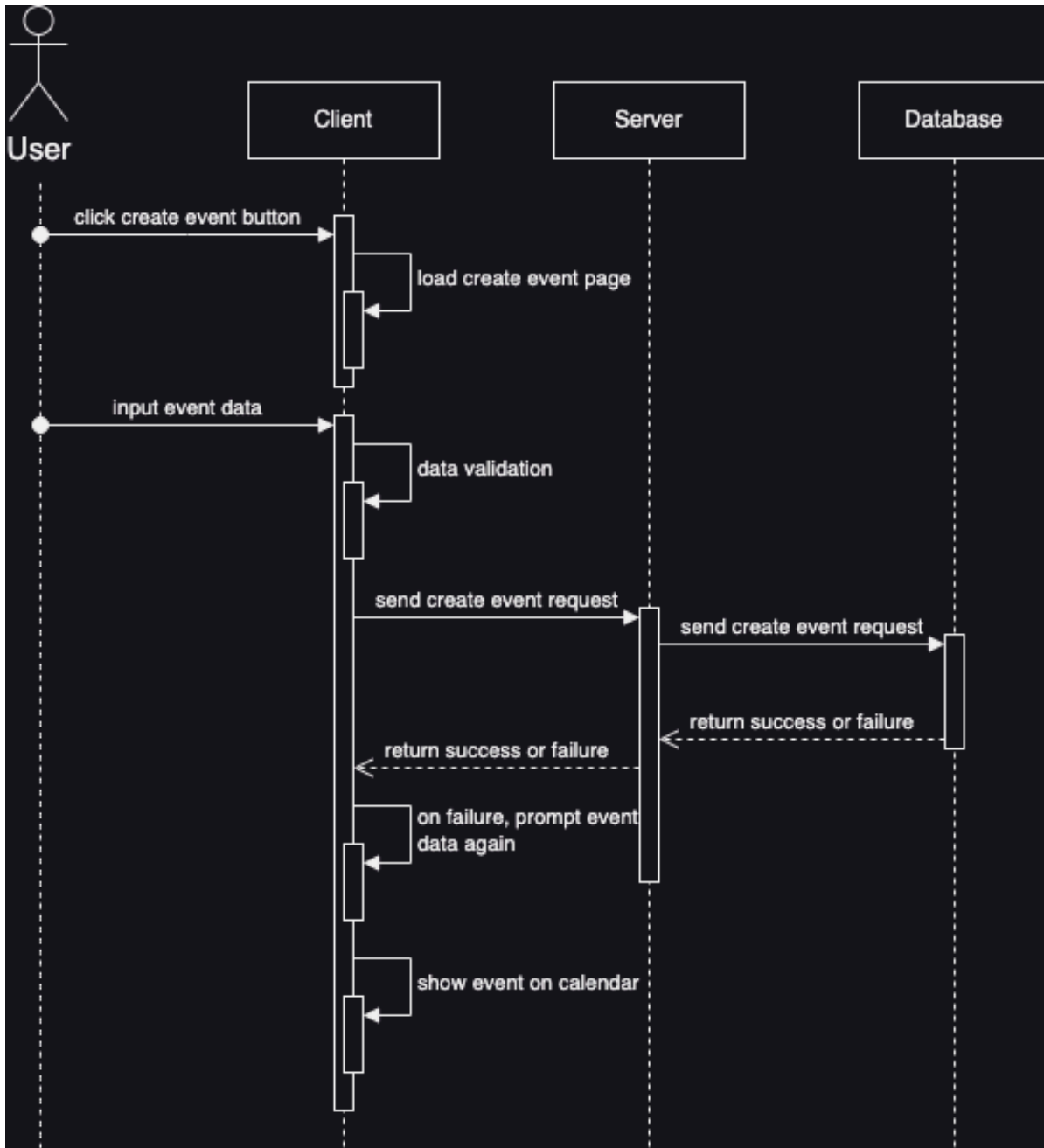
The following sequence diagrams show some of the major sequence of events of the application.

The diagrams will show the process of logging in, creating an event, updating an event, planning a task, and generating a schedule with AI. In this application, the user will do something on the UI (client), the client will send a request to the server, the server will send requests to the database as necessary, and the server will return to the client. The client will handle data validation and loading the UI.

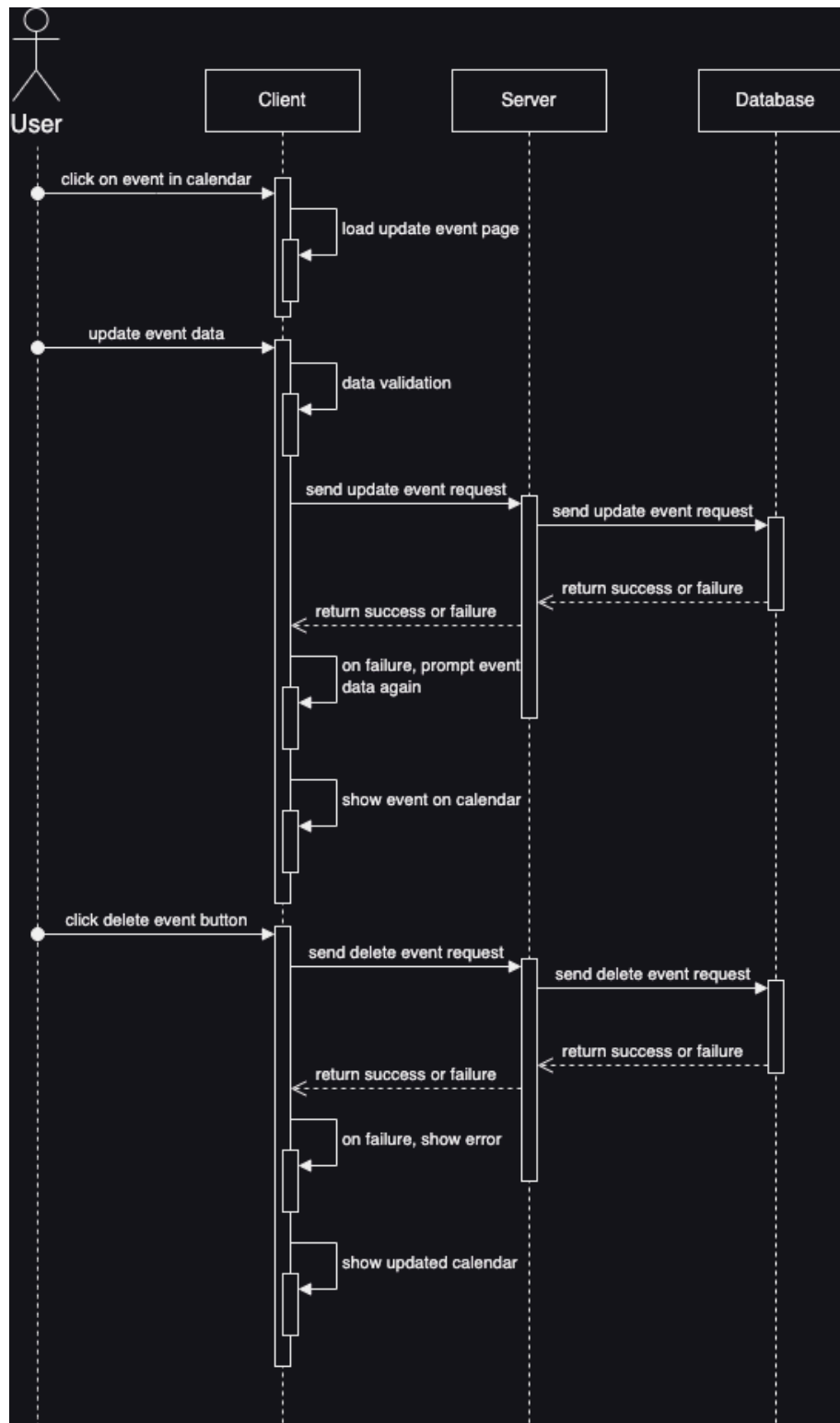
4.3.1 User Login



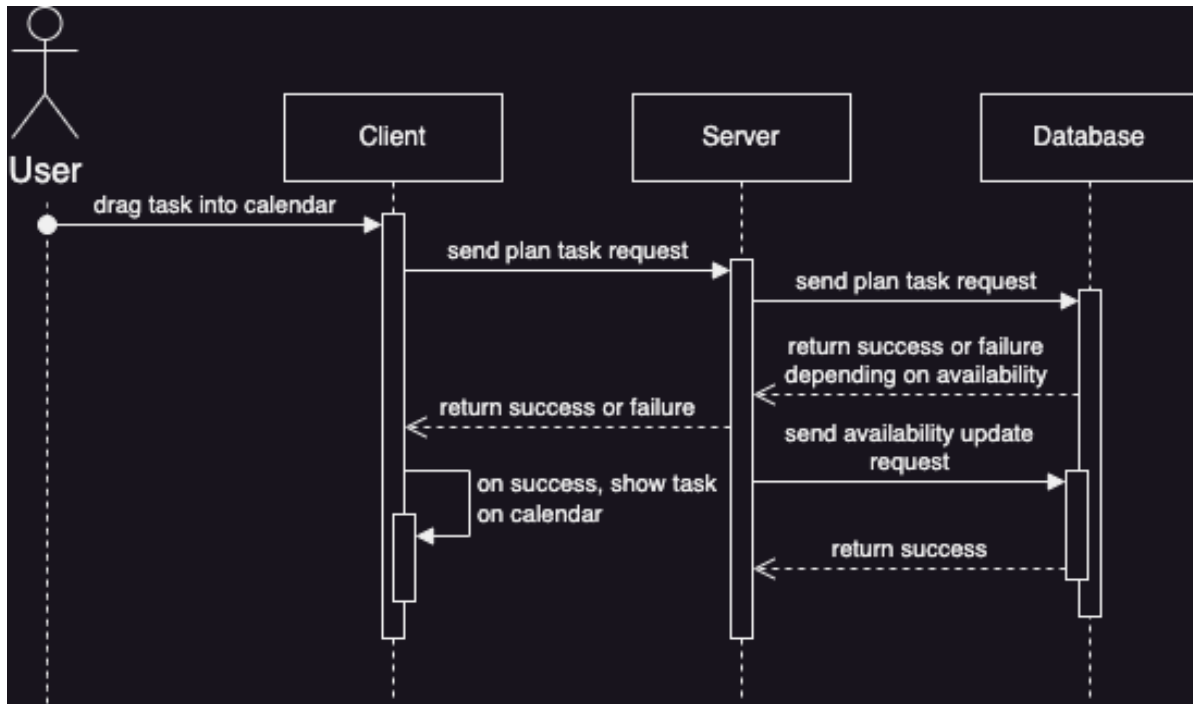
4.3.2 Create an Event



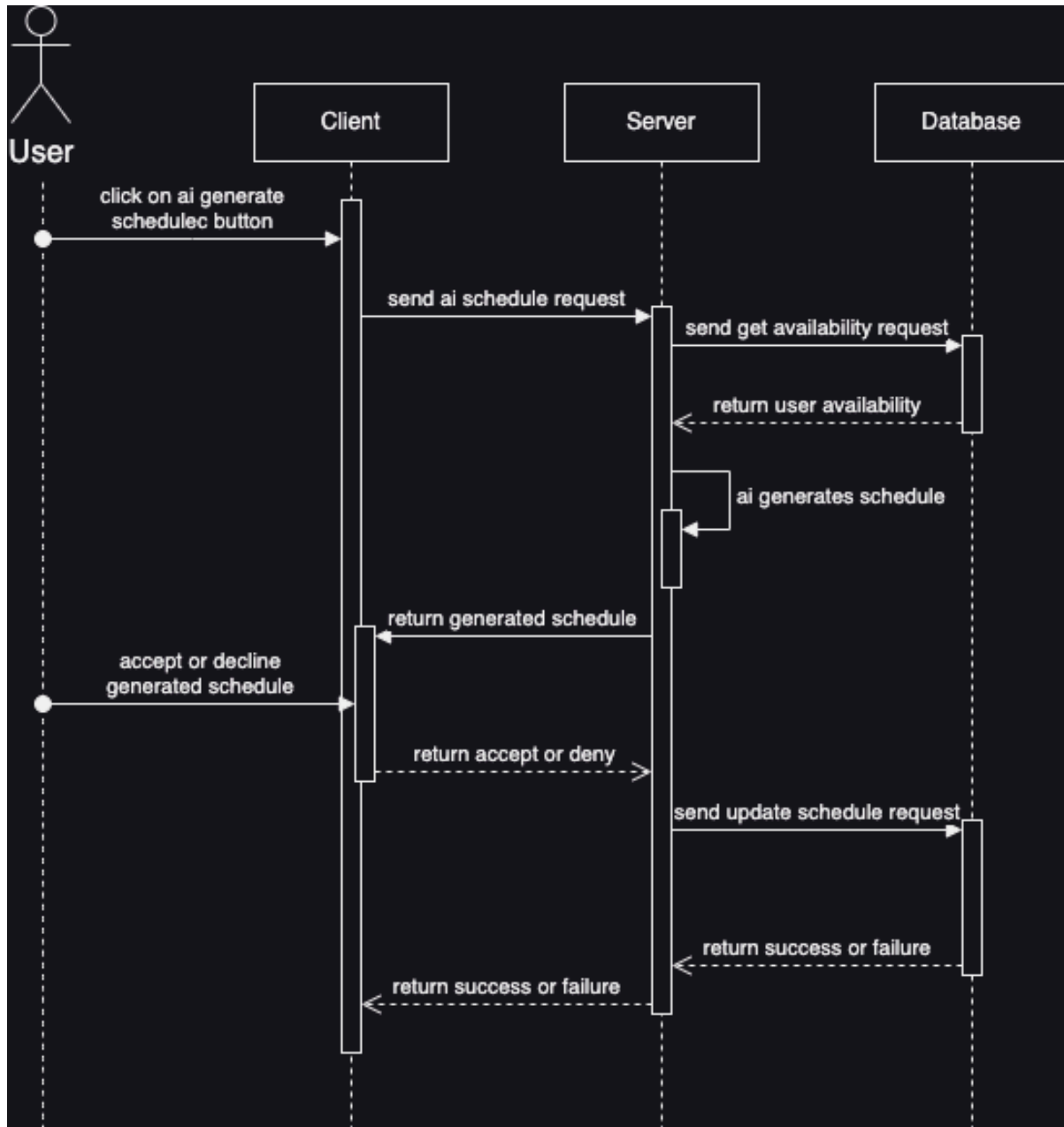
4.3.3 Update an Event



4.3.4 Plan a Task



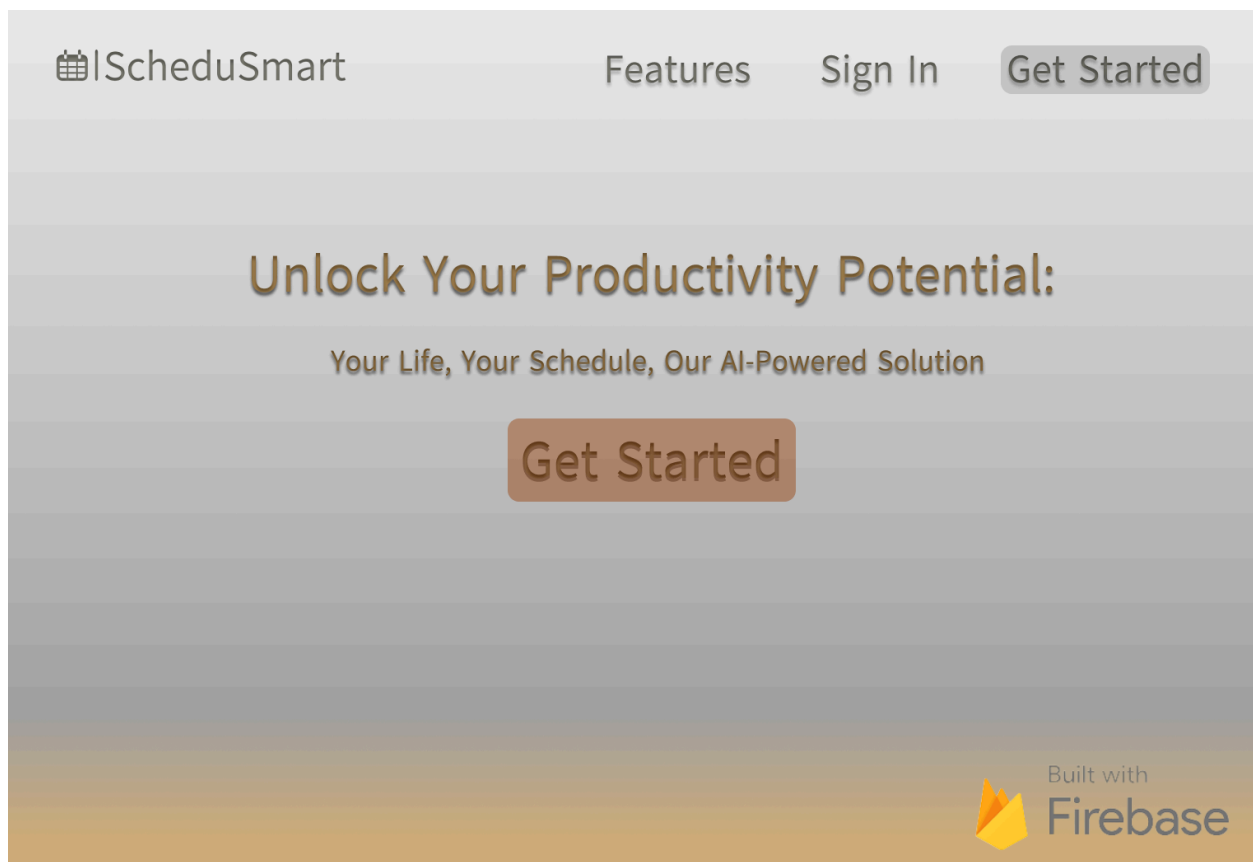
4.3.5 Generate Schedule with AI



4.4 UI Mockups

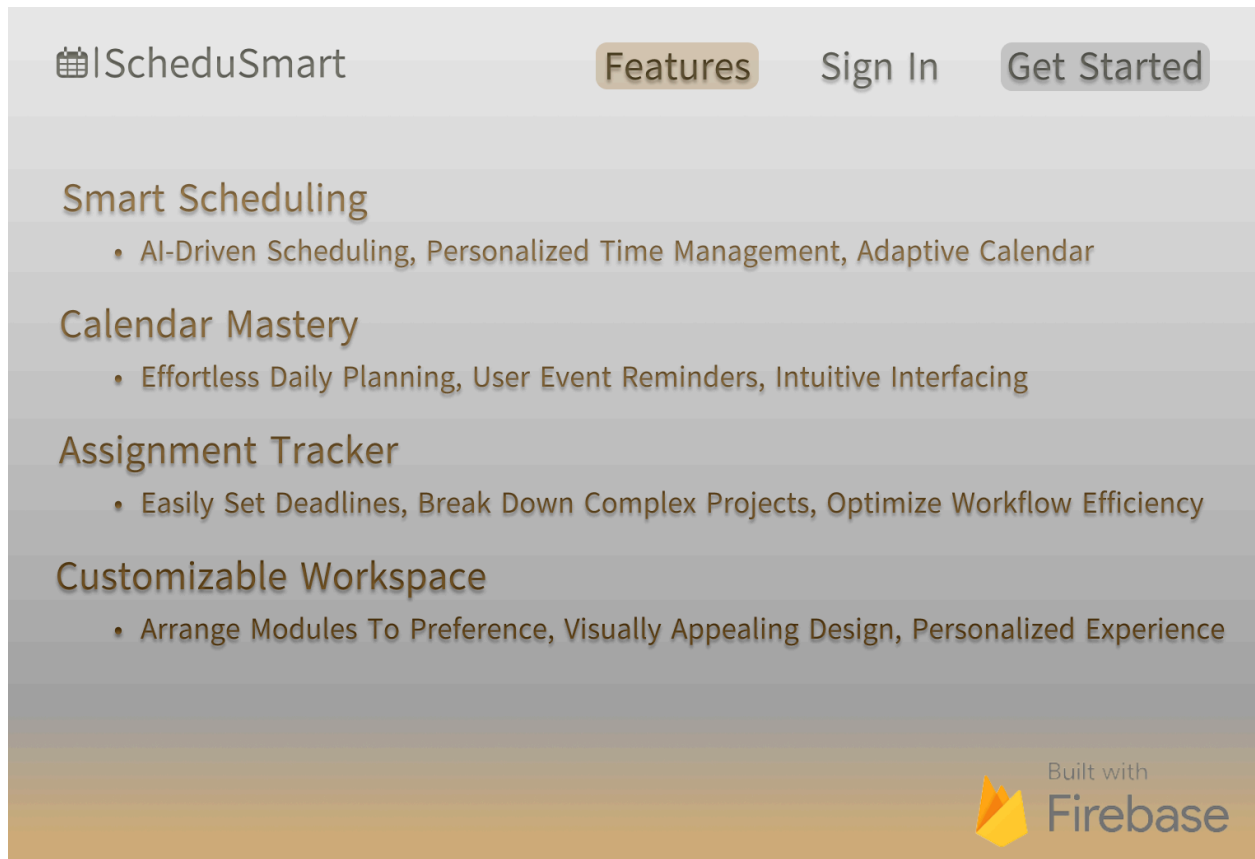
4.4.1 Landing Page

The first mock-up below depicts what the initial landing page could look like. This page will serve as the location for new users to learn about what ScheduSmart has to offer, as well as create a new account or sign in to an existing user account. Clicking the ScheduSmart icon (which is present on all pages) in the top left will take any user back to this page.



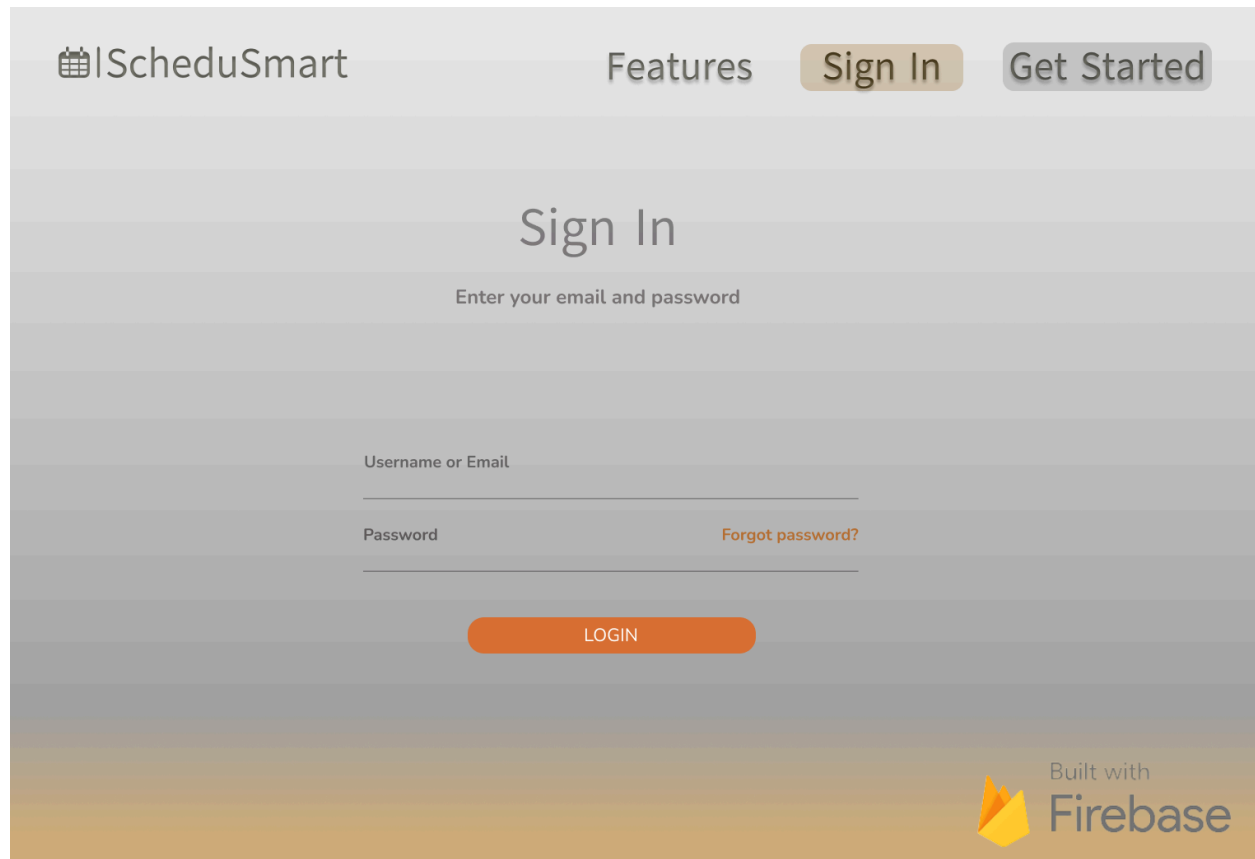
4.4.2 Features

This is the “Features” page. We plan to include information on this page which will explain how ScheduSmart differs from popular organization alternative tools.



4.4.3 Sign In

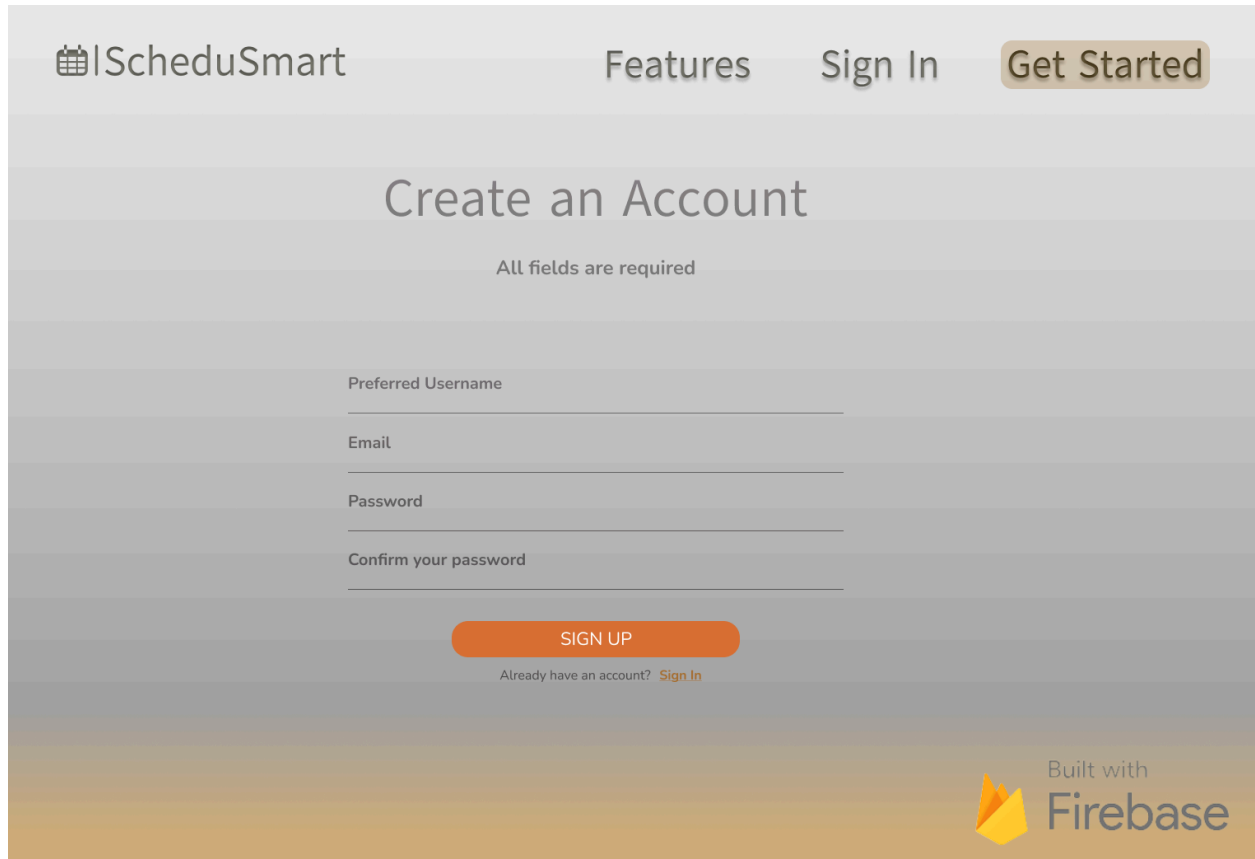
This is the Sign In page, where existing users will sign into their accounts using their chosen username or email and password.



The image shows a web page design for the 'Sign In' functionality of 'ScheduSmart'. The page has a light gray background with a subtle gradient. At the top, there is a navigation bar with the 'ScheduSmart' logo on the left, followed by the word 'Features', and two buttons: 'Sign In' (highlighted in orange) and 'Get Started' (gray). Below the navigation bar, the main heading 'Sign In' is centered in a large, dark gray font. Underneath the heading, the text 'Enter your email and password' is centered in a smaller, dark gray font. The form consists of two input fields: 'Username or Email' and 'Password', both with light gray borders and placeholder text. To the right of the 'Password' field, there is a link labeled 'Forgot password?' in orange. Below the input fields, there is a large, rounded orange button with the text 'LOGIN' in white. In the bottom right corner, there is a logo for 'Built with Firebase', featuring the Firebase logo (a stylized flame) and the text 'Built with Firebase'.

4.4.5 Sign Up

This is the Sign Up page, where new users come to create their accounts. Required information to open an account includes a preferred username, working email address, and a password.



The image shows a web page for signing up to ScheduSmart. The page has a light gray background with a subtle gradient. At the top, there is a navigation bar with the ScheduSmart logo (a calendar icon) on the left, and links for "Features", "Sign In", and "Get Started" on the right. The "Get Started" button is highlighted with a brown background. Below the navigation bar, the main heading "Create an Account" is centered in a large, dark gray font. Underneath the heading, a smaller line of text states "All fields are required". The sign-up form consists of four input fields, each with a label above it: "Preferred Username", "Email", "Password", and "Confirm your password". Below the form fields, there is a prominent orange "SIGN UP" button. Under the button, a link says "Already have an account? Sign In". In the bottom right corner, there is a logo for "Built with Firebase" featuring the Firebase flame icon.

ScheduSmart

Features Sign In Get Started

Create an Account

All fields are required

Preferred Username

Email

Password

Confirm your password

SIGN UP

Already have an account? [Sign In](#)

Built with
Firebase

4.4.5 Home Window

Finally, this is the main home window in which users are able to operate the multitudes of available functions. Shown in this mock-up are calendar events, recurring events (classes), assignment list items, Day/week/month/year view options, settings options, sign out option, and event completion functionality.

