CramBrûlée

Technical Design Documentation
Orbital 2024

Members:

Denzel Loke
Timothy Loh

Table of Contents

Key Details

Team Name: Crambrûlée

Members:
Loke Mun Kong Denzel (A0281142N)
Loh Yan Xun, Timothy (A0272149B)

Project Name: Crambrûlée

Proposed Level of Achievement: Gemini

Github Link: https://github.com/timothyloh0523/crambrulee

Project Overview

Motivation

Students often have trouble with tracking deadlines for many different assignments, and find it a chore to plan and optimise their schedules to complete their tasks without affecting their sleep schedule or quality of work.

Being university students ourselves, we understand how frustrating this can be, especially given how little time we already have to even complete our work, let alone plan when to complete them. Hence we wanted to make an application that can help alleviate this, by helping the user to both plan and optimise their schedules.

There are many existing calendar apps, such as Google Calendar, which allow users to enter and keep track of their schedules. However, all of them require entirely manual inputs from their users, a process we felt that we could automate and hence streamline, improving the overall user experience by doing the planning for them. In other words, we wished to create a smart planner that could harness the power of technology to automate the planning process for users.

Aim

We hope to streamline the planning of one's daily & weekly tasks by creating a smart calendar web application, CramBrûlée, with an additional functionality to automatically plan users' schedules for them: using user inputs (due date of a task, existing schedule, preferences) to determine the most optimal time that the user should start cramming an assignment, and automatically add it into their CramBrûlée calendar.

Tech Stack

- Languages used: Javascript, CSS, HTML
- Frontend, user interface: Javascript
- Backend: Express, Node.js
- Database: MongoDB
- APIs: FullCalendar, OpenAI

User Stories

User Profiles

This web app was created with university students in mind, to help streamline and automate their task planning when juggling schoolwork and other commitments. However, it is definitely generalisable and possible for other demographics to use as well.

High Priority

- As a user, I want to tie my calendar to a user account, so that I can view it on multiple devices if necessary.
- As a user, I want to clearly view my schedule, so that I can easily keep track of my plans for the week.
- As a user, I want to optimise my schedule, so that I can make better use of my limited time.
- As a user, I want to automate my schedule planning, so that I can free up more time towards completing my tasks.
- As a user, I want to be able to edit the suggested schedule, so that I can cater for last-minute changes and flexibility of my schedule.
- As a user, I want to view my deadlines clearly, so that I can keep track of them and avoid missing them.
- As a user, I want to split my tasks by colour, so that I can clearly differentiate between different types or subjects of tasks.

Medium Priority

- As a user, I want to block out 12am to 8am time slots from my calendar and prevent tasks from being scheduled then, so that I can maintain a healthy sleep schedule.
- As a user, I want to be alerted of my deadlines a few days before, so that I ensure those tasks are completed on time.
- As a user, I want to toggle between weekly and monthly view of my calendar, so that I can keep track of both upcoming and future deadlines.

Low Priority

- As a user, I want to toggle between landscape and portrait calendar display, so that I can personalise my calendar.
- As a user, I want to use my calendar in dark mode, so that I can cater for different lighting conditions.

README

Documentation:
https://docs.google.com/document/d/1NJYrrDUmPWrZGAuYLO0DFdPRINOexmdeYmMg_YjxGRM/edit?usp=drive_link (this document)

Description:
This app is intended to be a calendar app that intuitively helps you plan your tasks based on your calendar availability and task urgency.

Proposed Level of Achievement: Project Gemini

Project Scope: To create a Web App that displays users' personalised calendars, and intuitively suggests timeslots for users to begin working on new tasks, based on factors like task urgency and calendar availability.

Tech Stack
We used Javascript for our frontend, Express and Node.js for our backend, and MongoDB for the database to store information required for user authentication.

Local Deployment

1. Prerequisites

- Before you begin, ensure that your machine has the following software and dependencies downloaded. If the terminal is unresponsive, use Ctrl + C and try again.
    - Node.js: Install Node.js from nodejs.org.
    - npm: npm is installed with Node.js.
    - MongoDB: Install MongoDB from mongodb.com. Ensure MongoDB is running on localhost:27017.
    - express: Type into terminal "npm i express"
    - mongoose: Type into terminal "npm i mongoose"
    - hbs: Type into terminal "npm i hbs"
    - openai: Type into terminal “npm i openai@^4.0.0”

2. Installation

- To set up the project locally, clone the CramBrulee repository into your machine using the gitHub desktop app. Alternatively, you can enter the following terminal command to clone the repository using Git "git clone https://github.com/timothyloh0523/crambrulee.git"

3. .env file

- Before running CramBrulee, a file called “.env” containing the OpenAI API key needs to be created first. Do so in the root directory (the path should look something like this, e.g. C:\Users\joe12345\...\crambrulee\.env). For privacy reasons, do message @dndnzel or @timo_lyx on Telegram for the necessary file.

4. Usage

- Open Command Prompt or Terminal and navigate to the directory crambrulee/src
- Run the application using the following terminal command: nodemon index.mjs (if successful, terminal should display "port connected" and "mongoDB connected")
- Open a web browser and go to http://localhost:3000. You should see the login page.
- You will be able to create a new user using the signup form, once a new user has been created you will be directed to the login page. Users will only be permitted to log into the home page after logging in with the correct credentials. Different messages have been implemented to feedback different types of incorrect login credentials.
- At the home page, select different views to navigate between them, and click to add an event.

Milestone Progress:

- Milestone 1: Created Home Page, Login form, and Signup form.
- Milestone 2: Created 3 different calendar views (Month, Day, Week), and ability to add events by day
- Milestone 3:
    - Added “Suggest Event” button with OpenAI functionality to both suggest date and times for currently unscheduled events, and suggest new events for user consideration.
    - Added ability to add events at specific times of the day
    - Added “Task List” as a view option
    - Added quality-of-life changes, such as ability to colour-code events

Features

Do watch our milestone 3 video submission for a demonstration of these features in action!

User Account Login

Crambrûlée has a simple user account system, requiring each user to have a username and password to log into their account. The data will be used to allow users to log in to view and use Crambrûlée on multiple devices. Each user account will have a userID tagged to it, which is used to fetch events unique to each user.

Fig. 1 below shows the login page. Users enter their credentials in their respective fields, and clicking "Let Me In" will complete the login process (if the credentials are correct) and redirect the users to the calendar homepage. If the username and password combination entered is invalid, the message "incorrect credentials, please try again" will be displayed, and the user can press the back button on the toolbar to return to the login page.

If a user does not have an account, they can use the "Register" button to create a new account (Fig. 2), following which they will be redirected to the login page and can use their freshly created account to log into Crambrûlée.

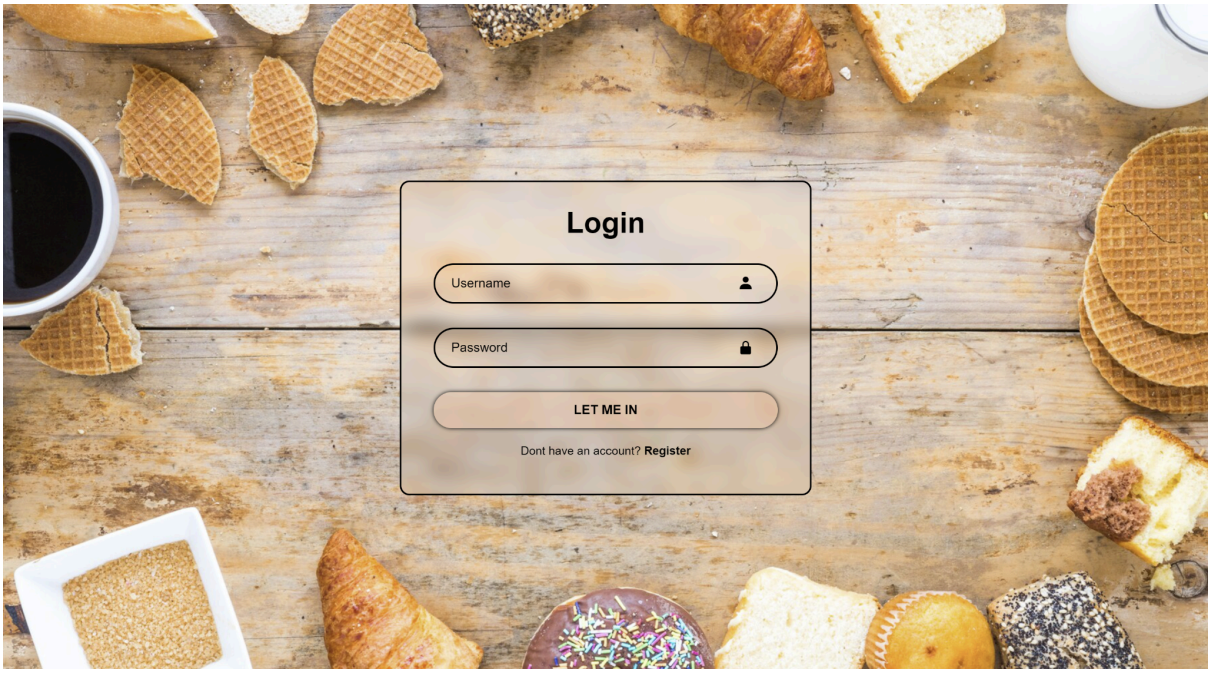

Fig. 1: Login Page

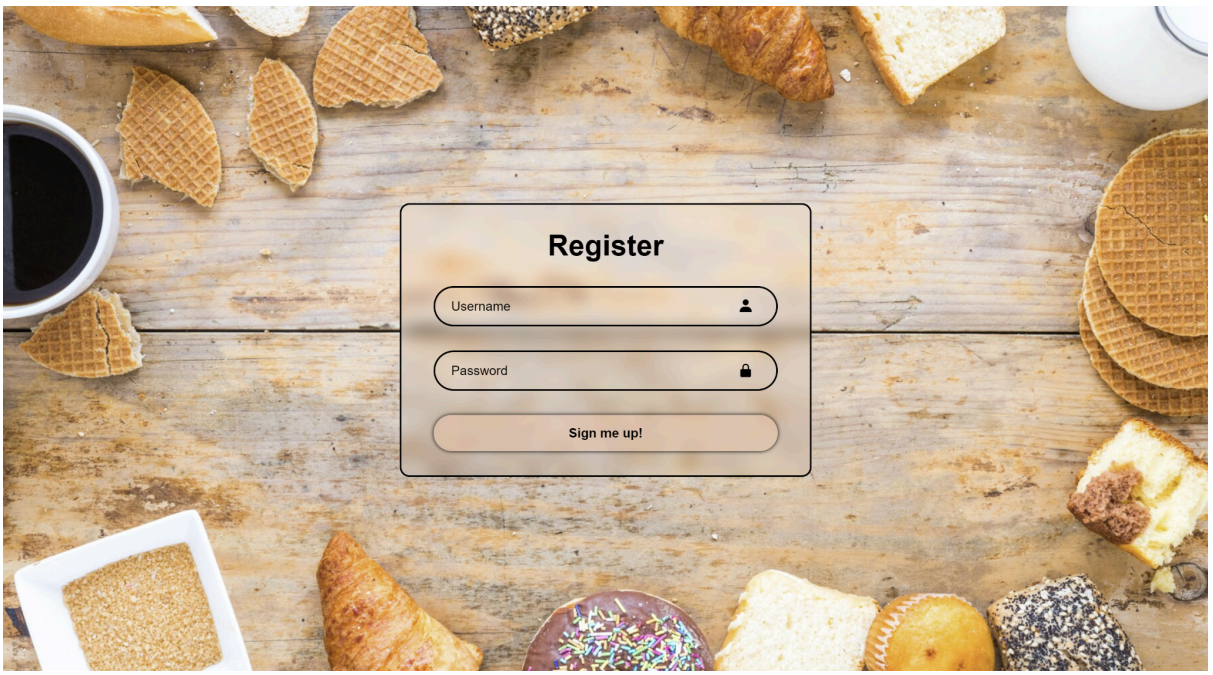

Fig. 2: Registration Page

Upon logging in successfully, the user will be redirected to the "month view" homepage, displaying the current month and any scheduled tasks (Fig. 3).

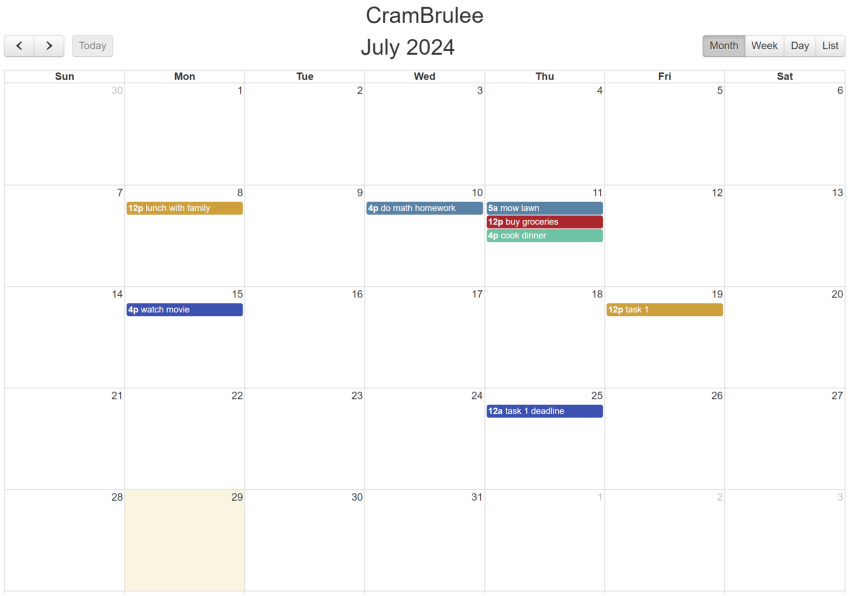

Fig. 3: Main calendar interface (month view)

Main Calendar Interface with View Options (fullCalendar API)

The calendar's interface was created using the fullCalendar API. The basic layout is like a normal planner. The following are some of the features we included.

Ability to switch between month view, week view, day view and task list

The different view options can be toggled via the buttons on the top right corner. The default is month view; the 3 other view options are week view (Fig. 4), day view (Fig. 5), and task list (Fig. 6).

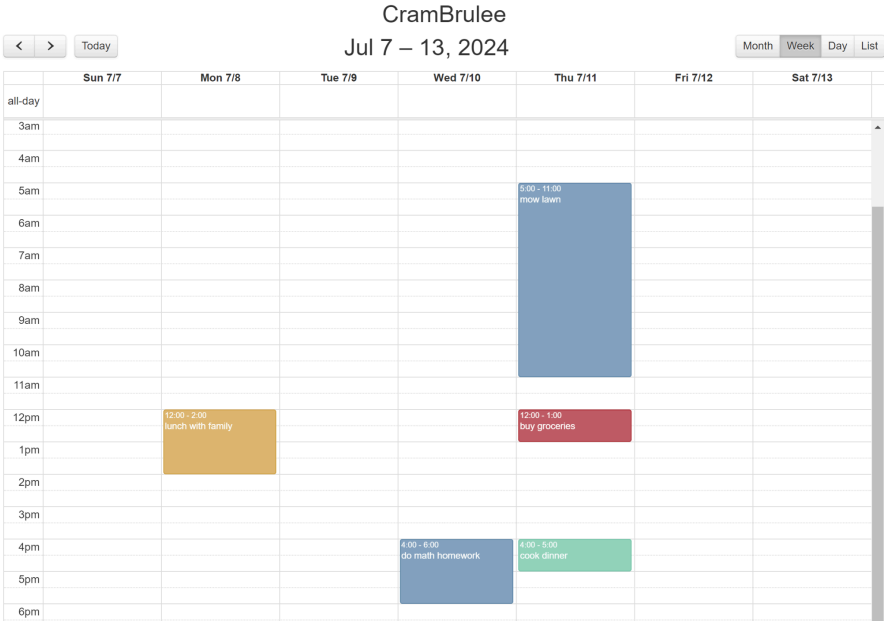

Fig. 4: Week View

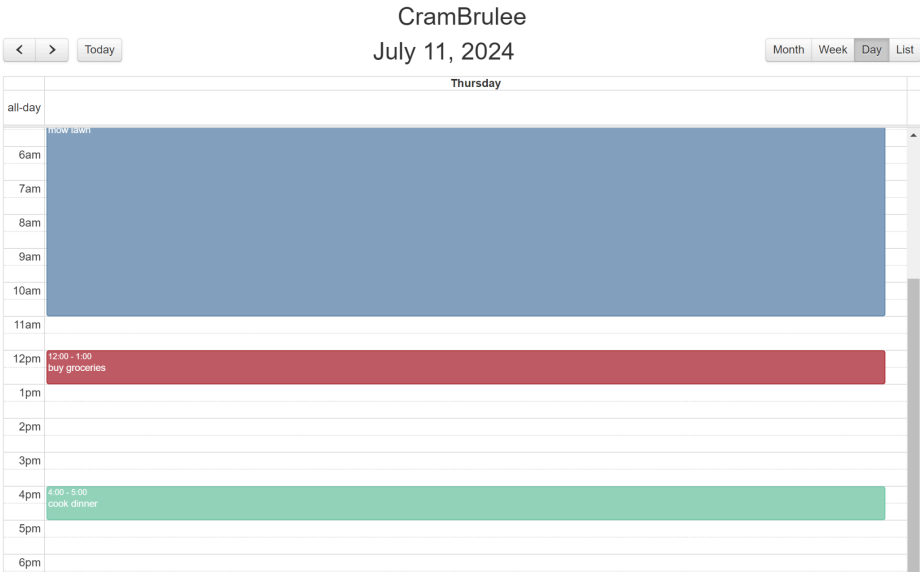

Fig. 5: Day View

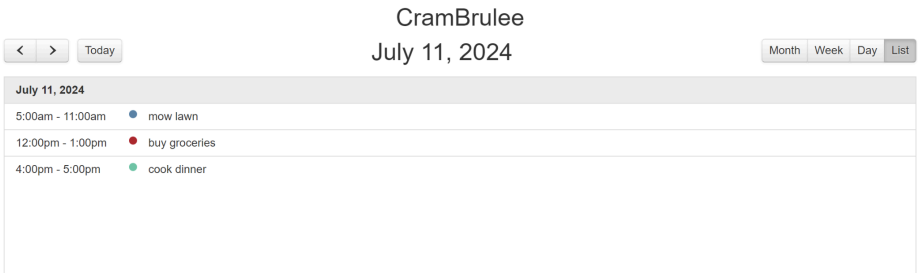

Fig. 6: Task List

Previous, next, and "Today" navigation for each view option listed above

The "<" and ">" buttons on the top left corner correspond to "previous" and "next" respectively, for the day/week/month depending on the view. For example, under "day view" for July 11, the "<" button will navigate to the day view for July 10. Additionally, the "Today" button will redirect the user to the present day/week/month.

Adding an event

An event can be added by clicking on the respective timeslot in each view (e.g. clicking on a day in the month view, or clicking on an hour in the day view).

Option to choose event colour

When creating or editing an event, the "event colour" option allows users to customise the event colour (Fig. 7).

Option to create full-day event

Also in Fig. 7, ticking the checkbox for "All Day" will make the event a full-day event without needing the user to manually type in the dates.

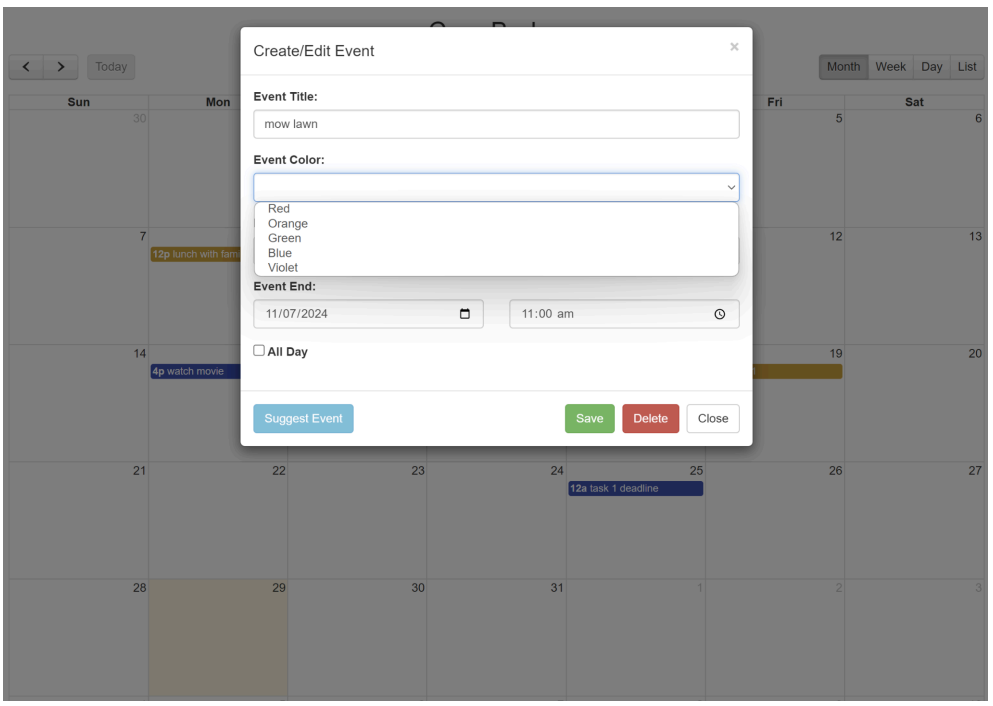

Fig. 7: Create/Edit Event Screen

Autoplanner (OpenAI API)

The autoplanner feature harnesses the power of OpenAI technology to help users automatically schedule their tasks based on existing commitments and deadlines. When the user presses the "Suggest Event" button (blue button as seen in Fig. 7), this function will be called.

- If the user provides an event title, the autoplanner function will suggest a start and end timing for that event, following which the user can add it to their calendar.
- If the user does not provide an event title, the autoplanner function will suggest an event and start and end timings for it, following which the user can add it to the calendar. The event suggested will be based on existing events already inputted in the calendar. For example, if a user inputs a deadline on 30 July and tries to suggest an event for 28 July, the autoplanner will detect the aforementioned deadline and take that into consideration, allocating for that task to be done on 28 July if there are no other more pressing tasks to be allocated.

SWE Principles

Code Reuse

We opted to use existing APIs - the fullCalendar and OpenAI APIs - as the foundation for our project, instead of fully creating it from scratch, as they were viable frameworks that already existed and using them allowed us to focus our time and effort on implementing features and tying the entire system together instead.

BDUF Principle

Following the "Big Design Up Front" Principle, we clearly segregated our files - including our scripts, templates, and entry file - into necessary folders from the start, making our final code much easier to follow and debug.

YAGNI Principle

Following the "You Aren't Gonna Need It" Principle, we took our project step by step and only added in new functionalities after we were done implementing the previous ones, instead of pre-emptively adding in functionalities before being sure that we would definitely need them.

Interface Segregation Principle

The different view scripts (dayViewScript, monthViewScript and weekViewScript) have varying layout requirements, hence some require functions and calculations that others do not. For example, monthView's layout requires calculation of padding days (circled in Fig. 8) that other views do not, hence the function that

calculates these padding days is only added to monthViewScript, instead of having it as an overarching function used in all 3 scripts, as it is unnecessary for dayViewScript and weekViewScript.

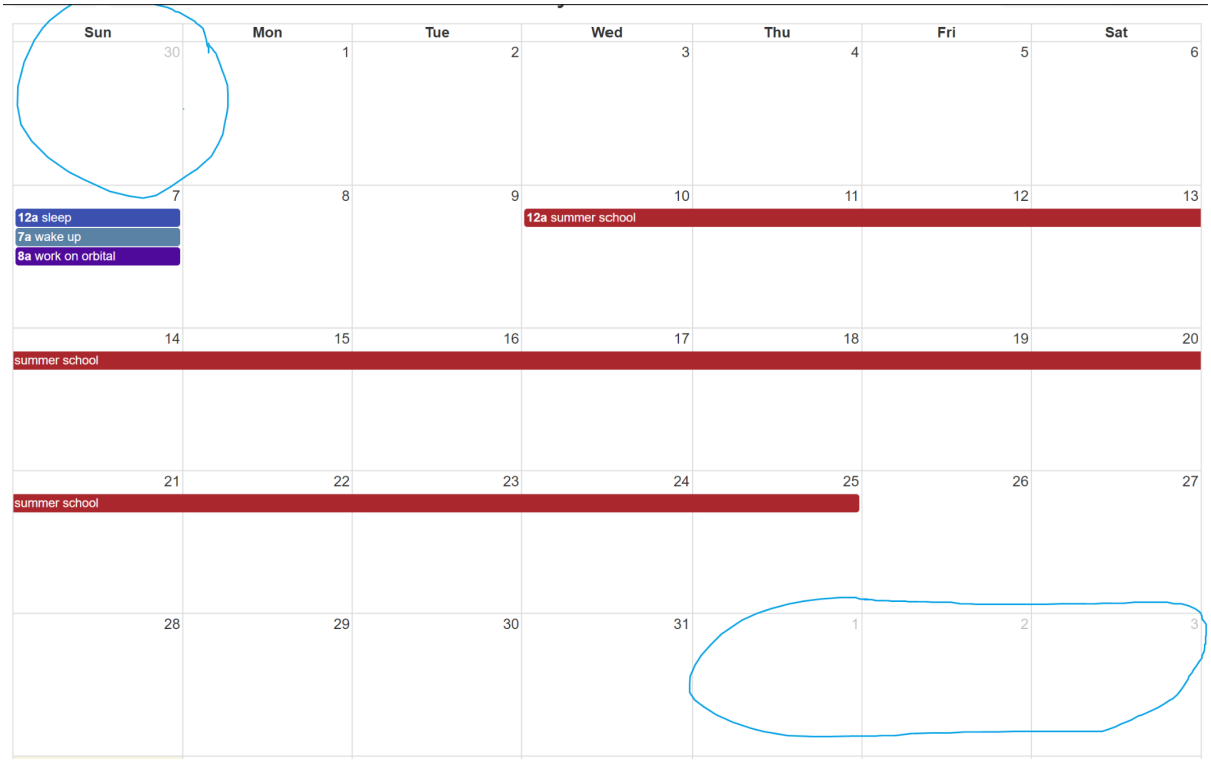

Fig. 8: monthView

Testing

Testing the app was done manually, by first setting up the app on localhost:3000 using the command nodemon index.mjs, then trying out all the different functionalities in different scenarios as specified below, and using the console.log and console.error functions to track system outputs, such as templates requested and errors encountered.

User Account Login

Login Page

Test Scenario	Test Case	Test Data	Expected Result	Actual Result	Pass/ Fail
Logging in	Correct username and password entered	Username: 1 Password: 1 Click "LET ME IN" Button or press "Enter" on keyboard	Displays calendar homepage on "month view"	Displays calendar homepage on "month view"	Pass
	Non- existent username	Username: 0 Password: 0	Displays message:	Displays message:	Pass

	entered (with any password)	Click "LET ME IN" Button or press "Enter" on keyboard	"incorrect credentials, please try again"	"incorrect credentials, please try again"	
	Username entered with wrong password	Username: 1 Password: a Click "LET ME IN" Button or press "Enter" on keyboard			Pass
Logging into calendar on a different account	Different username and password entered from first test case (done after populating account of username "1" with events)	Username: 2 Password: 2 Click "LET ME IN" Button or press "Enter" on keyboard	Displays calendar homepage on "month view", with no events (opposed to username "1" with existing events)	Displays calendar homepage on "month view", with no events	Pass
Clicking Register Button	Register button clicked	Button click	Displays signup page	Displays signup page	Pass

Signup Page

Test Scenario	**Test Case**	**Test Data**	**Expected Result**	**Actual Result**	**Pass/ Fail**
Signing up	Previously registered username entered (with correct password)	Username: 1 Password: 1 Click "Sign me up!" Button or press "Enter" on keyboard	Displays message: "This username is taken, please refresh and try a different username"	Displays message: "This username is taken, please refresh and try a different username"	Pass
	Previously registered username entered (with wrong password)	Username: 1 Password: 2 Click "Sign me up!" Button or press "Enter" on keyboard			Pass

Main Calendar Interface with View Options

Calendar view buttons

Test Scenario	Test Case	Test Data	Expected Result	Actual Result	Pass/ Fail
Navigating between different view options	Clicking on each of the 4 view options on the top right corner (“Month”, “Week”, “Day”, and “List”)	Respective button click	Displays corresponding view option correctly	Displays corresponding view option correctly	Pass
Navigating to previous/ next day/ month/ week	Clicking “<” button in each of the 4 views	Respective button click	Displays corresponding day/ month/ week correctly	Displays corresponding day/ month/ week correctly	Pass
	Clicking “>” button in each of the 4 views				Pass
Navigating to present day	Clicking “Today” button in each of the 4 views	Button click	Displays present day in the current view	Displays present day in the current view. Exception: button cannot be pressed if user is already in present day	Pass
Create/ edit event popup	Clicking on a timeslot on the calendar	Button click	Create/edit event popup appears	Create/edit event popup appears	Pass

Event creation/editing

Test Scenario	Test Case	Test Data	Expected Result	Actual Result	Pass/ Fail
Adding event	Create event in create/ edit	Type in: - event title - event colour - event	Register event according to given paramete	Register event according to given parameters Exception: If end time is during or before start	Pass

	event popup screen	start and end date and time, then save	rs, unless it is invalid	time, event is still displayed on calendar, but the end time is registered as “NA”	
Editing event	Edit event in create/ edit event popup screen	Change each of the parameters listed in the box above, then save	Update event according to given parameters, unless it is invalid	Update event according to given parameters Exception: If end time is during or before start time, event is still displayed on calendar, but the end time is registered as “NA”	Pass

Autoplanner

Test Scenario	**Test Case**	**Test Data**	**Expected Result**	**Actual Result**	**Pass/ Fail**
Suggest event timing	In create/ edit event popup screen, suggest event given event title	Type in: - event title - event colour Leave blank: - date and time Then click “Suggest Event” button	Calendar will output a date and time based on user schedule	Calendar outputs a date and time based on user schedule Exception: When the OpenAI response limit is reached, no response will be received	Pass for most cases (minus exception)
Suggest event to carry out	In create/ edit event popup screen, suggest event without giving event title	Type in: - event colour Leave blank: - event title - date and time Then click “Suggest Event” button	Calendar will output an event with date and time based on user schedule	Calendar outputs an event, date and time based on user schedule Exception: When the OpenAI response limit is reached, no response will be received	Pass for most cases (minus exception)

Problems Encountered

Online Deployment

Our app was not deployed online - we initially attempted to use Vercel to deploy our application online with a website link. However, we ran into a database-related issue which caused the website to constantly timeout once a user submitted their login details. When viewing the same website locally on localhost:3000 (using the command "vercel dev" which replicates the Vercel environment locally), the app works perfectly well even though it should behave the same way as the online version (which was timing out constantly). We suspected the issue to be database-related, as the timeout would always occur right after logging in or registering, both acts which required accessing and fetching data from the MongoDB database. Additionally, we determined the pathing to be fine, since the code worked in the replicated local environment. However, even after adding the needed environmental variable MONGODB_URI and allowing the necessary permissions, the error still persisted. Other popular alternatives to Vercel that we knew of each had their own downsides (most notably, Heroku was now paywalled and hence was not an option). As we could not fix this issue even after days of debugging, we decided to focus our limited time on implementing other features instead.

"Suggest Event" Button

Initially, when implementing the OpenAI technology, the "Suggest Event" button would call the "getEventSuggestions" function multiple times on a single click, sending multiple prompts at once, even though we wanted just a singular prompt to be sent. This would result in the user's computer lagging out. Hence, we added a boolean variable called "isSuggesting" near the start of the "suggestEvent" function (Fig. 9), returning true if an event was already being suggested. The "suggestEvent" function would thus be terminated if "isSuggesting" is true, preventing the function from being called multiple times unnecessarily.

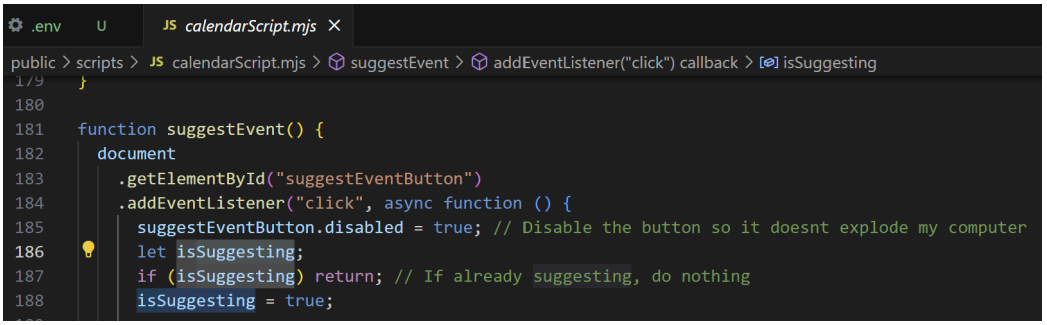

Fig. 9: "isSuggesting" boolean variable

Additionally, we also added some code to disable the "Suggest Event" button when it is loading a suggestion (also seen in Fig. 8, line 185), to prevent unnecessary multiple prompting and subsequent lagging.

Adding Events

During Milestone 2, we faced difficulties adding detailed features into our UI, such as adding events by hour instead of just by day, as well as other options like colour-coding events. Hence we had to overhaul our UI, eventually landing on the fullCalendar API that our current iteration uses.

Autoplanner Feature

The autoplanner feature successfully harnesses the power of OpenAI technology to help users automate their planning. However, the success of this feature also relies heavily both on the accuracy of users' inputs since it is based on them. While testing, we found more success when suggesting events within a packed schedule, since there was more data for the OpenAI technology to harness compared to a nearly blank schedule.

Additionally, we acknowledge that this feature could come with privacy concerns as the user's schedule is taken into consideration when the OpenAI technology is generating its outputs. The usage of third-party AI was a limitation we had to accept, as our scope of knowledge was insufficient to build an Artificial Intelligence programme from scratch for this project.

Finally, the API key we used is paywalled, so this might lead to some long-term maintenance costs. Yet, given the quick rise of ChatGPT and related technologies over the past few years, it is not unforeseeable that similar or better technologies could become more accessible in the near future and be used for implementation instead.