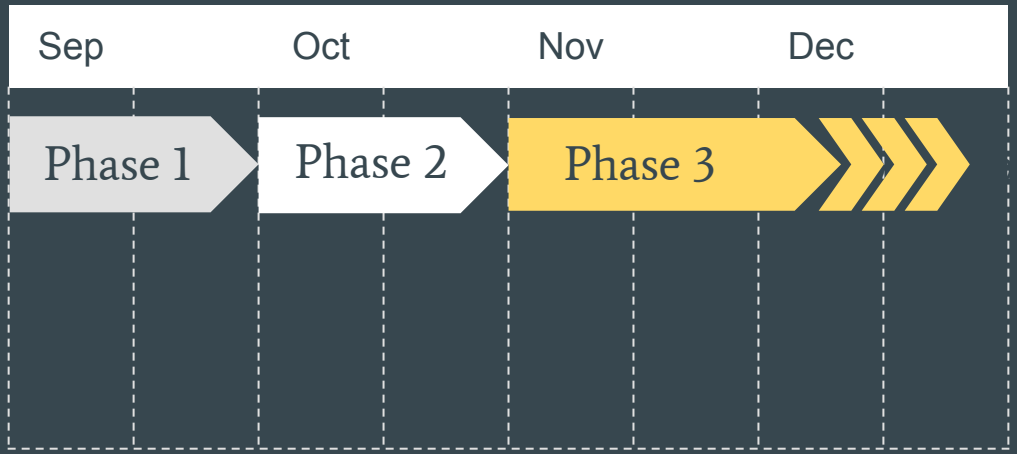
Evan Adis

Health made easy

CALORIE CONSCIOUS

Creating Timeline/Design Plans



- I began by sketching out a timeline broken into phases to follow.

Phase 1: Planning and Research (1 week)

1. Conceptualization and Research:
   - Define the purpose and target audience of your app.
   - Identify your unique selling points (USPs).
   - Research competitors and similar apps.
   - Create a preliminary feature list.
   - Survey potential users.
   - Analyze market trends and user needs.
   - Develop a revenue strategy. (e.g., freemium, subscription, ads).(if for profit)
   - Refine your feature list based on research findings.

Phase 2: Design and Prototyping (2-3 weeks)

1. Wireframing and Prototyping (1 week):
   - Create wireframes and prototypes for the app's user interface.
   - Gather feedback/User testing.

2. UI/UX Design (1 week):
   - Design the app's user interface.
   - Develop a user experience that's intuitive and user-friendly.

Phase 3: Development (6-8 weeks)

1. Backend Development (2 weeks):
   - Set up servers and databases.
   - Implement user authentication and data storage.
   - Find/Develop APIs for data retrieval.

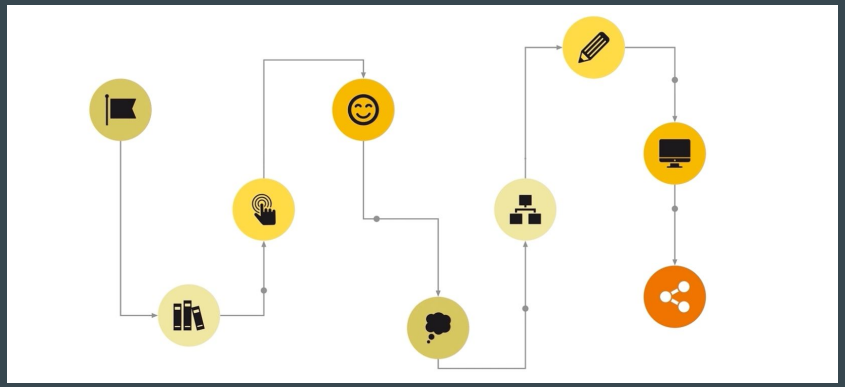
2. Frontend Development 2 weeks):
   - Build the app interface based on the designs.
   - Implement core features like calorie tracking and food logging.

3. Integration (2 weeks):

   - Integrate with third-party APIs for food databases, fitness tracking, etc.
   - Implement login options.

Phase 4: Testing and Demo(-)

-User Testing
-Final Demo



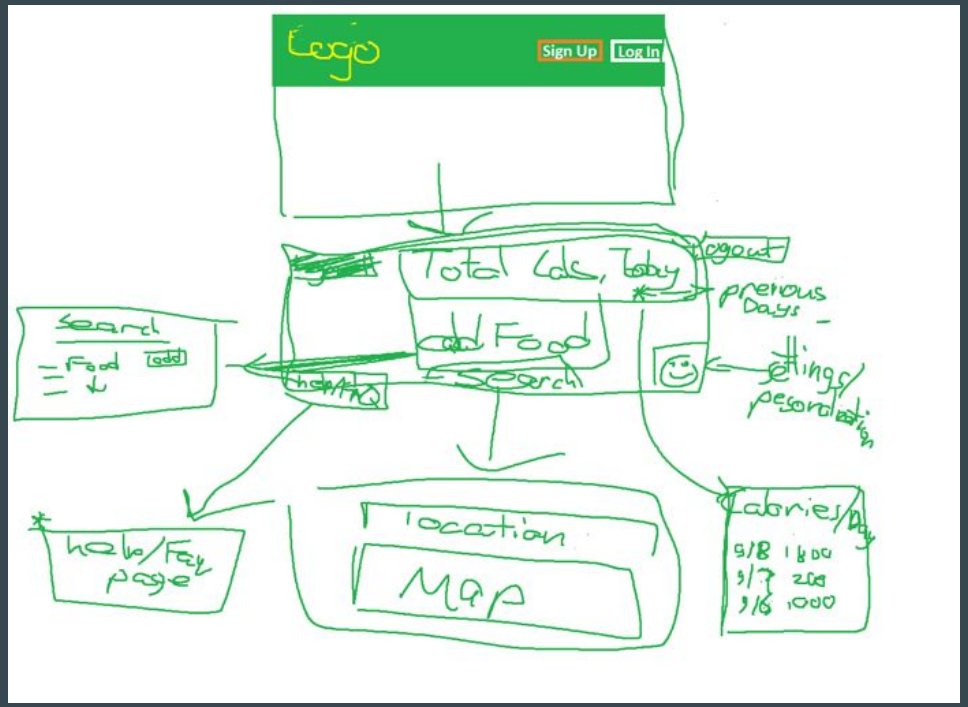
Planning & Research

- First step put simply, was to think of a problem, then a solution. In my case being the subject of making it easier to find and track food. Rich, poor, picky, young, or old. It's this project's aim to deliver an easier experience.
- Second step is to research similar ideas. Ex:'MyfitnessPal'
- Third step was to create a proposal and design and plan out user-flow, important, and critical features.

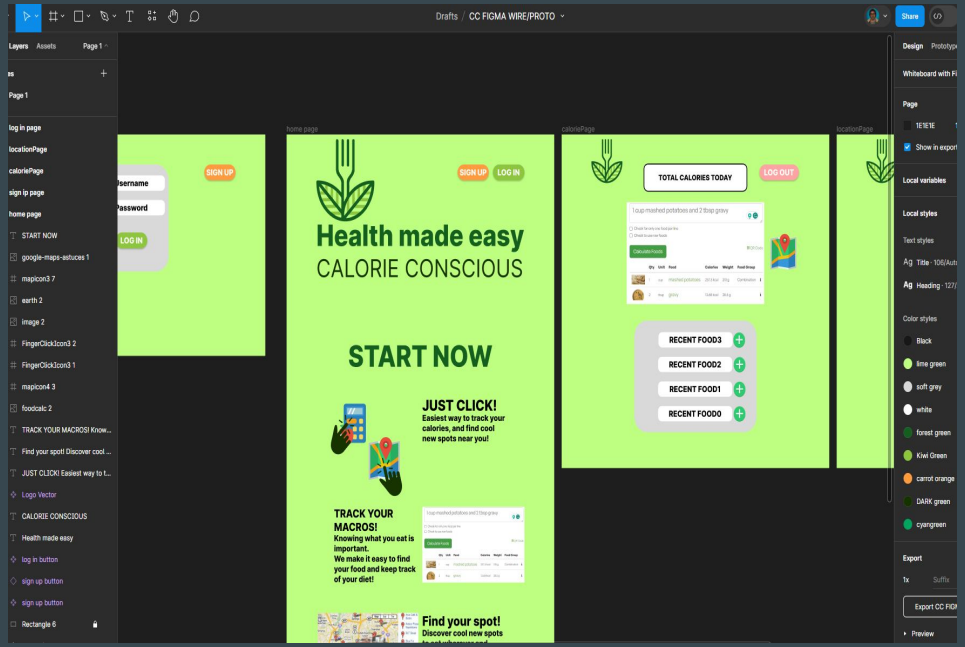
Designing the look. UI/UX

Sketches and wireframes

- The next step was to create basic sketches and wireframes of how the layout should incorporate basic critical features to the project, such as a calorie calculator, location services page, and even login/signup page.
- Based on my wireframe I can settle on a design and begin working on the next step.



Designing the look. UI/UX



Prototyping

- The next step was researching about prototyping.
- The program I used to create my prototype for Demo 1 was ‘Figma’.
- In Figma, I was able to create every detail down to the custom logo. This is the design my project will be based off.

Meeting expectations

Minimum Viable Product-

Meets very basic UI/UX Design.

Features include:

- Calorie Calculator,
- Google Places API,
- Basic site features like login/signup.

Expected/Planned

Meets basic UI/UX Design.

Features include:

- Calorie Calculator,
- Google Location API,
- Working database to store information and saved foods/locations using Nutritionix API/Google Places API
- Basic site features like login/signup.

Maximum Viable Product+

Exceeds UI/UX Design.

Features include:

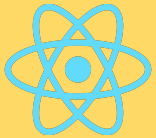
- **MVP(-)/Expected**
- Personalization page

Researching Full Stack

This project is my first time working full stack. I would need to research how to go about building the database, server, backend, and frontend of the project.

It has been especially helpful to research tutorials for similar projects. This helped me identify which tools I would need.

Identifying Technologies

Database 	• Postgres, is a free and open-source relational database management system emphasizing extensibility and SQL compliance.
Backend/Server 	• Node.js is a cross-platform, open-source server environment that can run on Windows, Linux, Unix, macOS, and more.
Backend/framework 	• Express.js, or simply Express, is a back end web application framework for building RESTful APIs with Node.js
Frontend 	• React is a free and open-source front-end JavaScript library for building user interfaces based on components.
Frontend/framework 	• Bootstrap is a free and open-source CSS framework

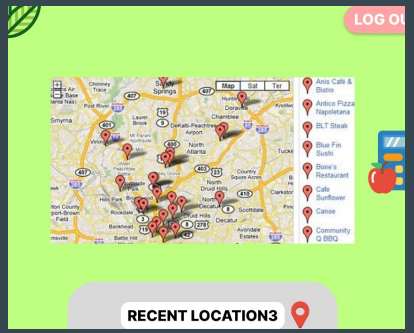
Identifying Technologies cont. - 3rd-Party API

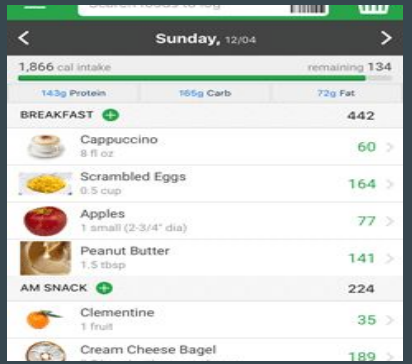
Google Places API

- Places API is a service that accepts HTTP requests for location data through a variety of methods. It returns formatted location data and imagery about establishments, geographic locations, or prominent points of interest.

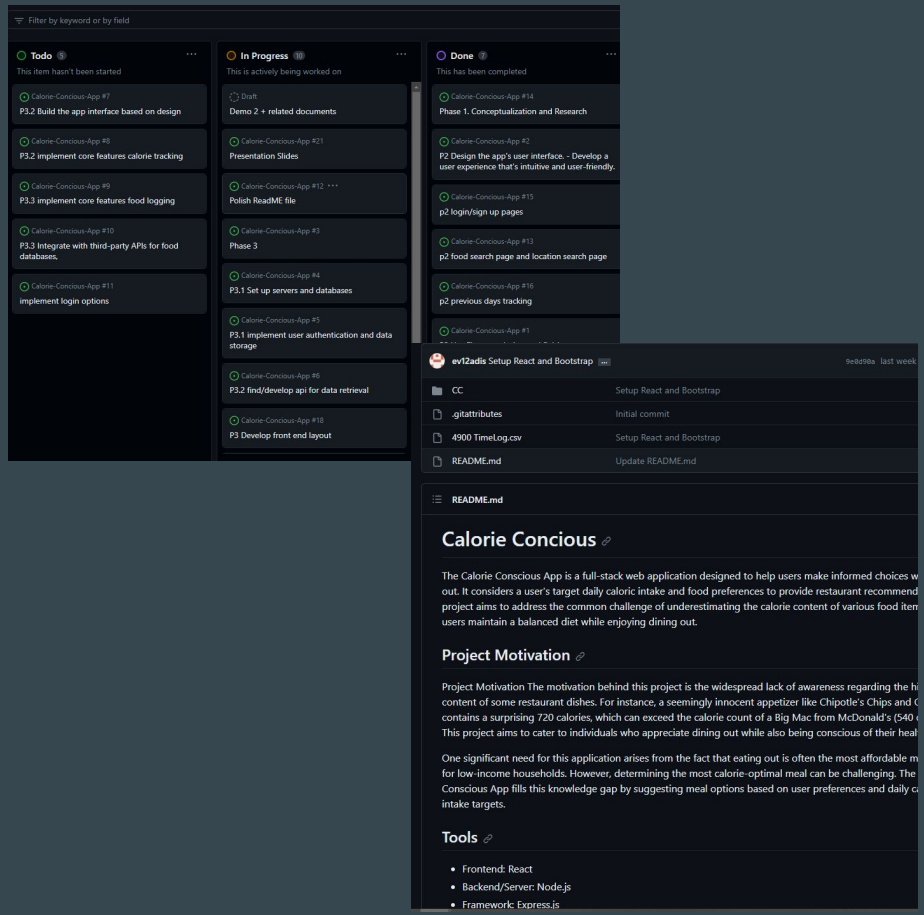
Nutritionix API

- Nutritionix API for sourcing food items and calorie data.
- Nutritionix maintains the world's largest verified nutrition database.





Setting up Repository and Project



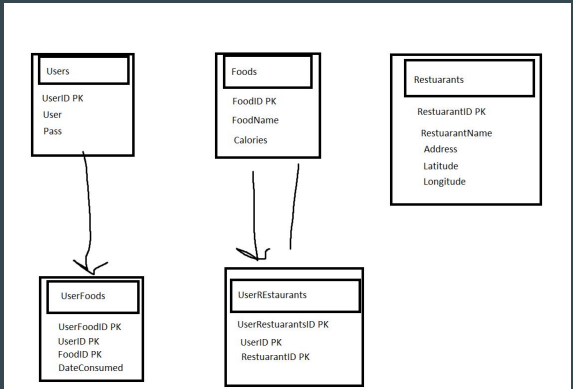
- The next step was to create a repository for the project, and a project management board to help me manage the progress and tasks pertaining to the project.

Phase 3

Backend Development

- The first step for creating the backend was researching SQL and databases. To do this I spent countless hours watching tutorials and guides.

- I began designing sketches, tables, ER diagrams, and finally started settling with potential database designs.
- I then translated them to PostgreSQL.

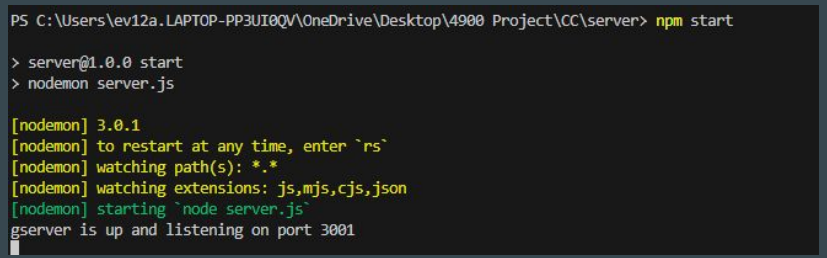


TempDatabase1.sql

```sql
CREATE TABLE Users (
    UserID INT PRIMARY KEY AUTO_INCREMENT,
    Username VARCHAR(255) NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Email VARCHAR(255) NOT NULL
);

CREATE TABLE CalorieTracking (
    TrackingID INT PRIMARY KEY AUTO_INCREMENT,
    UserID INT,
    FoodName VARCHAR(255) NOT NULL,
    Calories INT NOT NULL,
    Timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);

CREATE TABLE LastSelectedFoods (
    SelectionID INT PRIMARY KEY AUTO_INCREMENT,
    UserID INT,
    FoodName VARCHAR(255) NOT NULL,
    Timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);

CREATE TABLE Restaurants (
    RestaurantID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255) NOT NULL,
    Address VARCHAR(255) NOT NULL,
    Phone VARCHAR(20),
    Website VARCHAR(255),
    Rating FLOAT,
    UserID INT, -- to associate restaurants with users if needed
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
```

Setting up the Backend Server/Framework

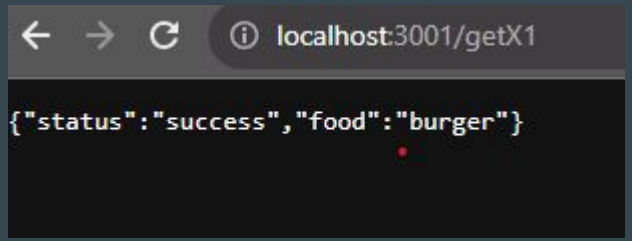
- Setting up Node.js and Express.js was a relatively simple task.
- I've also learned about and installed a couple of extra libraries/plugins to help run the project such as 'morgan', 'dotenv', and 'nodemon'.



Seeing the server start for the first time!

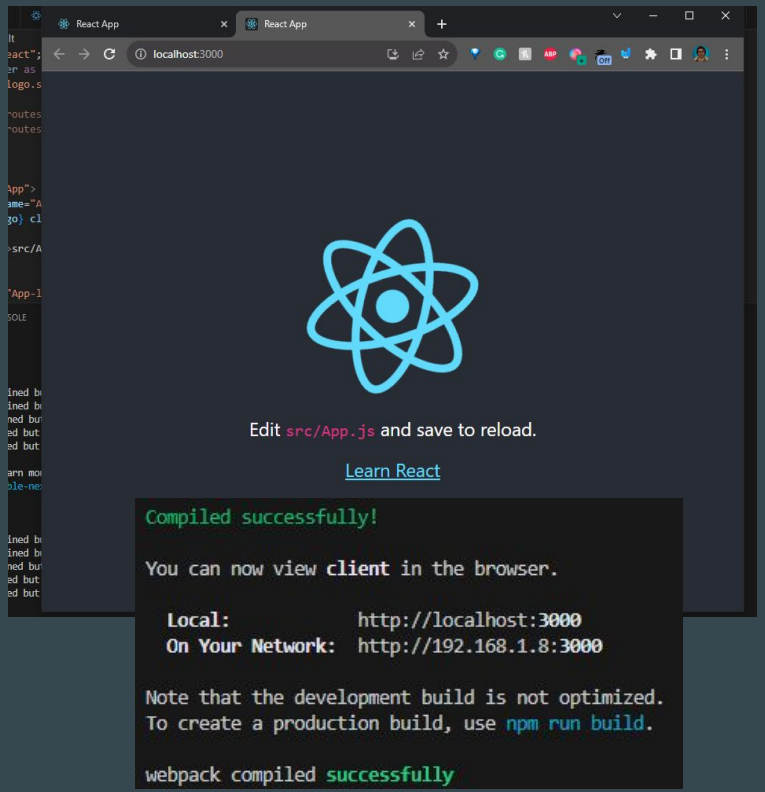
Setting up the Backend Server/Framework cont.

- I then learned to use HTTP request like ‘get’ an ‘post’ which are used to create REST API. With these I can manage my PostgreSQL database.



My first successful HTTP ‘get’ request.

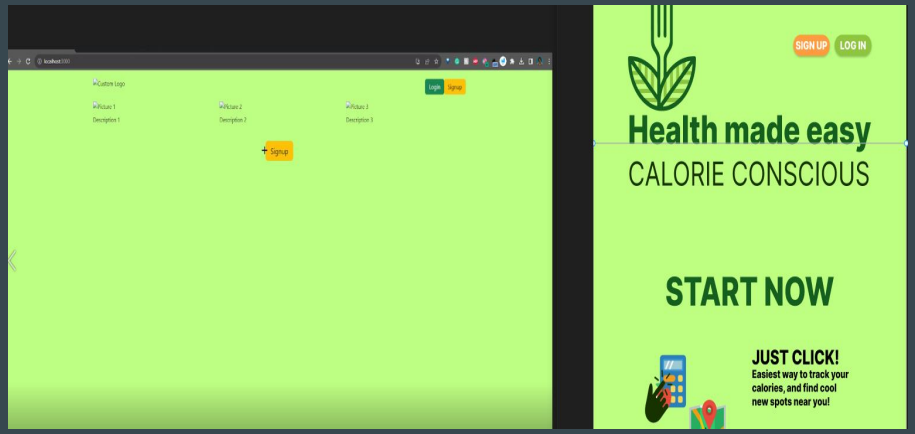
Phase 3 - Frontend Development



- I then setup React and Bootstrap for the first time.

Here is React and Bootstrap installed successfully and running for the first time!

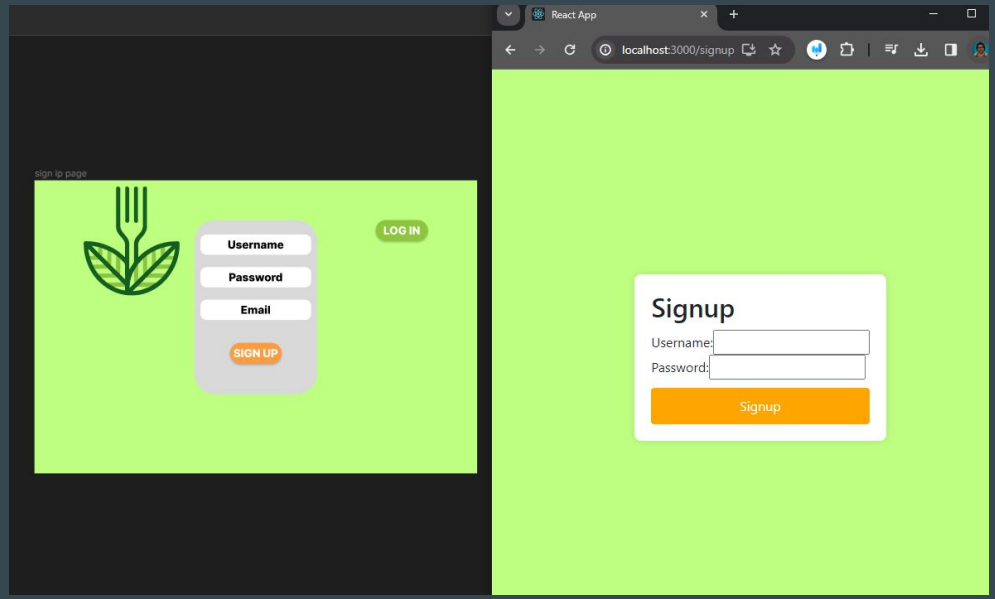
Frontend Development



- The idea was to design the front-end based on the previous sketches and figma designs.

Here is my first attempt at recreating the front page.

Frontend Development



I then moved on to other basic pages like the login and sign-up page.

Estimating Steps.

- Knowing the order of what functions i would need to add next would benefit me greatly in the construction of my program.

Dependant Properties

Users

- Users are create dependency on the foods but users are also dependant on logging in or signing up.

  Therefore Login function is of most importance.

API

- Foods information such as location, calories and more are dependant on API information. Therefore after login function, but before all else, API must be implemented.

Food or calorie info

Food and calorie information is completely dependant on the api information and dependant on users. Therefore, users must have functionality to register, and API must be implemented first.

Creating Users

The next step was to implement functionality to track users.

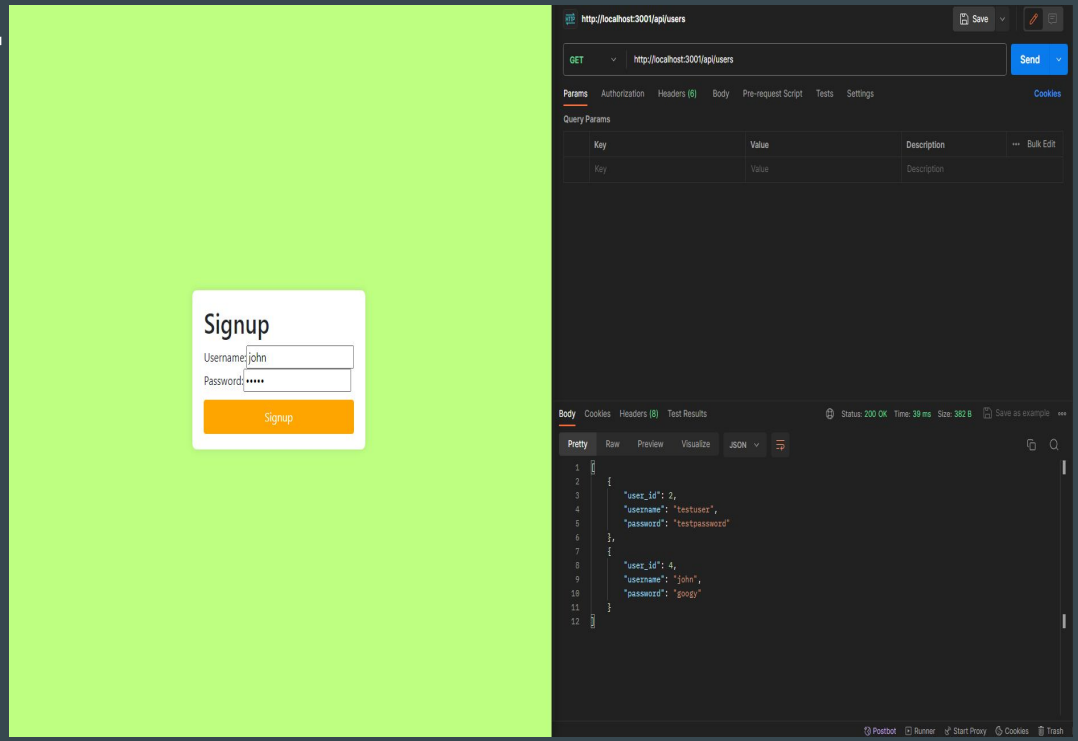
I chose to simply store them in a postgreSQL database early to be able to implement other web functionality.

Creating Users cont

Since the database was already setup, all that was left was to route the front end including the button.

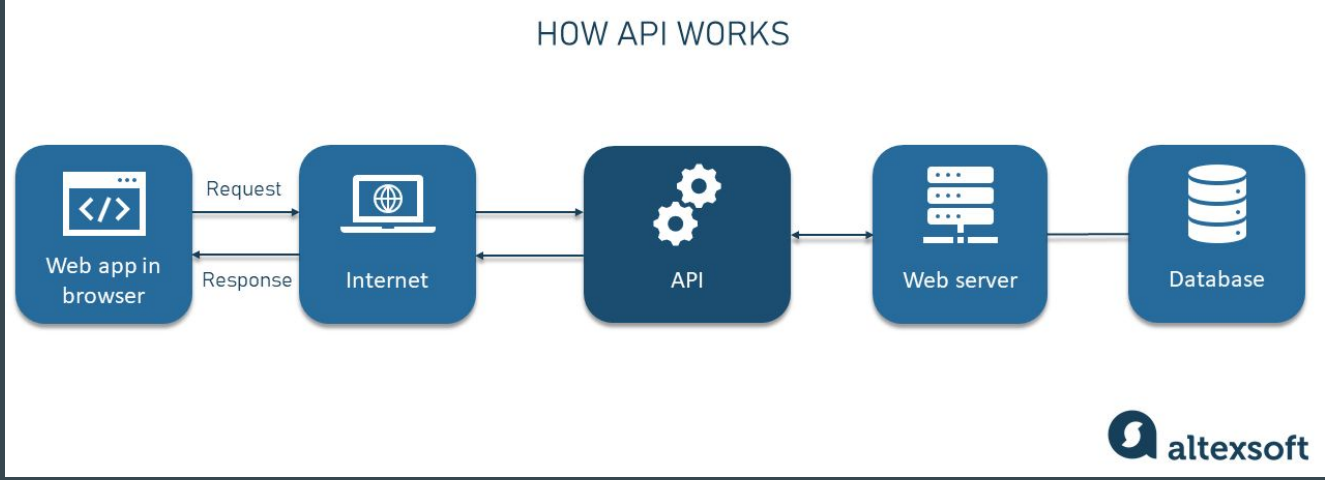
I used postman to verify the database entries.

Here is an example of the functioning signup button.



- Finally it was time to implement API. Prior research was very helpful in choosing API that i would need to use for this project.

API



Part of the journey was watching many tutorials on how API works and what it is exactly. The application will make an API call for a set of data to display for the end user to consume.

That means i would need to figure out what API I could use to find the data i need. Then I need to learn how to access that data.

Choosing API

The API i chose to use was the Nutrionix API.

Nutritionix API is for sourcing food items and calorie data.

From research i determined it maintains the world's largest verified nutrition database.

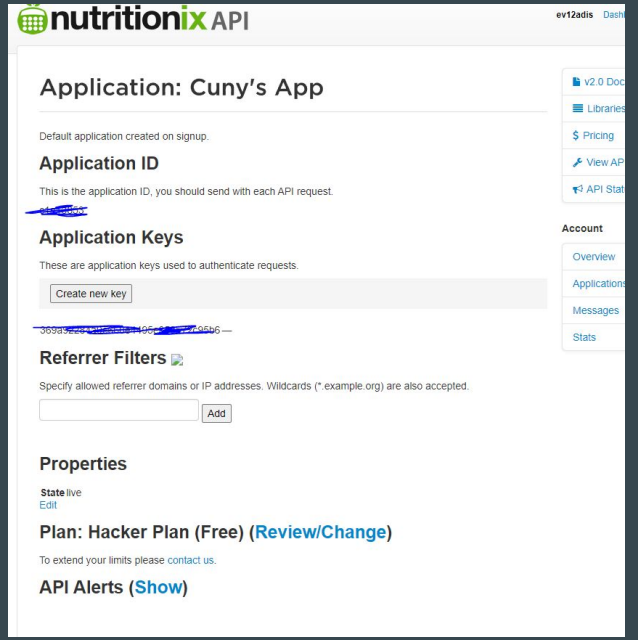
It also tracks all necessary location data so I wouldn’t need Google Places API.

Researching API

The next Step is to research how to use and implement the API.

Through much trial and error I learned about API keys, and ID's. Using my own personalised key and id, i can have access to my database for free.

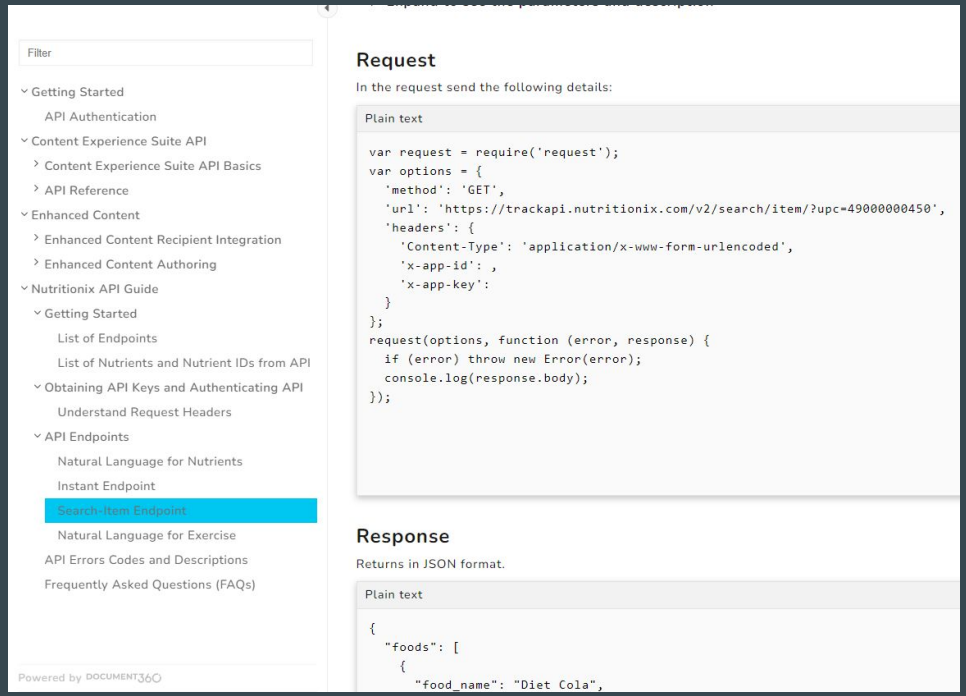
The next step would be to implement the functionality to access this database.



Researching API cont.

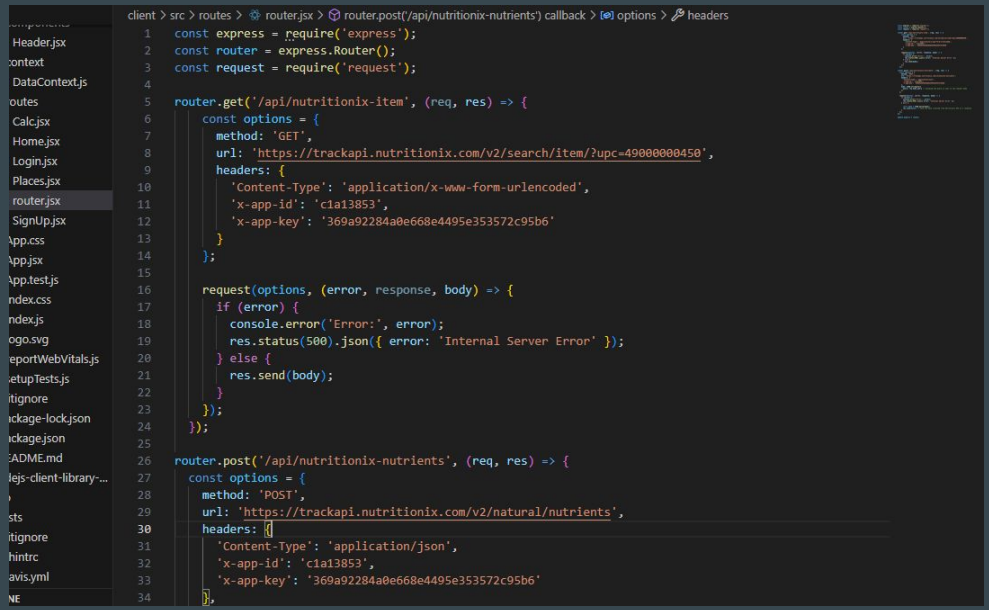
Researching APIs can be incredibly confusing, but i used the guide made publicly available to help me implement the functionality.

Most APIs come with guides on how to implement and use them.

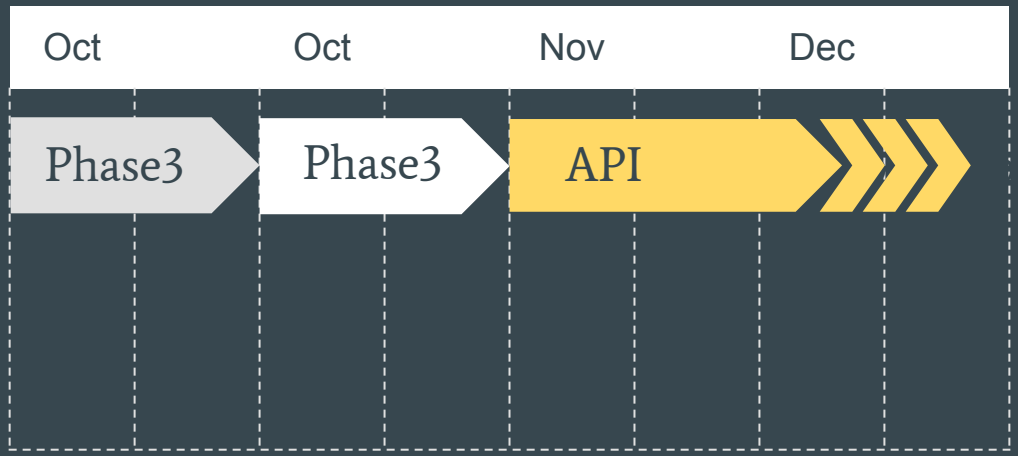


Implementing API

Implementing the API was very difficult. It took hours upon hours to get only parts working. Eventually i was able to implement authorization of the key and ID. Then i begin working on creating routes.



Timeline Restructuring

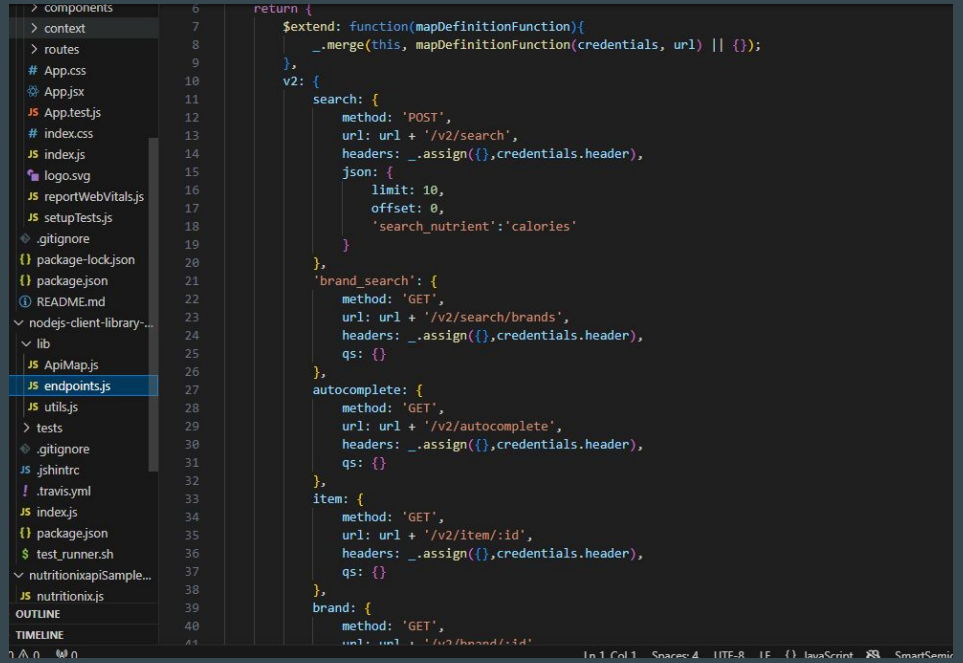


I began to realise that my timeline was quite working. Although creating a timeline at the start did help plan out my project. I would need to rethink how to plan better going forward. I did not expect the difficulty i would face in implement external API and how much time was needed in researching each portion of the project.

Implementing API cont.

After creating sample pages to test out NutrionixAPI functionality My next decision was to create a library in being able to easily access API information. This was a shortcut to creating external routes to be able to reuse API calls for each user.

Here i have creating a separate library with routes for each data call for the API.



FUTURE STEPS

1. API implementation

With the completion of the Nutrionix API library i can efficiently use calls with all endpoints being external. The next step would be to connect it to the database

2. User personalization

Features such as tracking height, weight, and other body metrics would be very effective in a calorie tracking, weight or other app relating to food.

3. Food Recommendation

Given the NutrionixAPI has access to location data, a feature such as ‘favorite place to eat’ can be added. Menus relating to the location may also be displayed.

Reflection

Having reached the endpoint of the semester, I can now reflect on the things I can improve on to help future students.

Reflection

When planning my time, if unprepared and lacking knowledge on a project or a particular field. ALWAYS PLAN FOR RESEARCH.

I believe I may have set my goals too high for this project in regards to my solo team size and prior knowledge. But I am very happy with how much I learned.

Repository & TimeLog
https://github.com/ev12adis/Calorie-Concious-App

Project Management Board
https://github.com/users/ev12adis/projects/1