Boiler Bite

Purdue Dining

Design Document

February 7th, 2020

Team 16

Ridwan Chowdhury
Jeremy Yang
Uday Chaudhary
Isha Mahadalkar

Table of Contents

Purpose

Students at Purdue often have a problem with keeping track of their calorie intake during a semester since there is no app to provide them with the knowledge of the food being served at the dining courts. Between rushing to classes and keeping up with homework, there simply isn't time for Purdue students to meticulously plan out their meals and ensure that they meet their own dietary goals. This often causes them undue stress, as they lose or gain more weight than they desire while attempting to keep up with their course load. The best way to combat this would be to develop a time-efficient way for students to track what they eat.

There are certain apps that inform students of the various foods at specific dining courts during different times of days like Purdue Meals and Purdue Menu. Both these apps only give information about the food items being served at each dining court on a specific day without tailoring it for a specific user. There are also a lot of meal planning apps like PlateJoy and FoodPrint which allow you to create a daily nutrition plan and keep track of your daily calorie intake. These meal planning apps are not specific to Purdue. The user also has to work with a complicated interface by having to enter detailed information for every meal that they eat. Another limitation with these apps is that they might suggest food items that are not available in the dining courts.

With BoilerBite, we want to give students the convenience of spending just a few minutes a day to keep track of all their macronutrients. We plan to make this app available for anyone who uses the dining courts on campus and wants to stick to a diet plan that provides a specific amount of calories per day.

Functional Requirements

1. User Accounts:
   <u>As a user,</u>
   a. I want to be able to create an account.
   b. I want to be able to login.
   c. I want to be able to delete my account when I want to.
   d. I want to be able to access my account anytime.

2. User Profiles:
   <u>As a user,</u>
   a. I want to be able to input my weight, height, and age into my profile.

- b. I want to be able to input my desired daily calorie intake into my profile.
- c. I want to be able to update my profile with a new weight, height, and age.
- d. (If time allows) I want to be able to choose the units used to display my information (lb. vs kg).
- e. (If time allows) I want to be able to set a desired body weight as a goal.
- f. (If time allows) I want to be able to compare my current weight to my past weight.

3. Dining court information:
   <u>As a user,</u>
   - a. I want to be able to check open hours of the dining courts.
   - b. I want to be able to view the menu of the dining courts.
   - c. I want to be able to filter out certain dining courts.
   - d. I want to be able to check for allergens in the dishes of the dining courts.
   - e. I want to be able to check the main ingredients of the items on the menus.
   - f. (If time allows) I want to be able to filter the foods recommended to me.

4. Recording meals:
   <u>As a user,</u>
   - a. I want to be able to record the dishes I eat each meal.
   - b. I want to be able to know how many calories I ate each meal, calculated based on the dishes that I recorded.
   - c. I want to be reminded when I consume a certain amount of calories over the limit that I desire.
   - d. I want to be able to see how many carbs, grams of protein, fat, etc. I consume each meal.
   - e. (If time allows) I want to be reminded when I consume a certain amount of calories under the limit that I desire.
   - f. (If time allows) I want to be able to turn off notifications about consuming too many calories.
   - g. (If time allows) I want to be able to turn off notifications about consuming too few calories.

5. Tracking Macronutrients:
   <u>As a user,</u>
   - a. I want to be able to calculate how much carbs, protein, and fat I've consumed each meal.
   - b. I want to be able to see how far away I am currently from my desired daily calorie intake.

   c. I want be presented a visualization of my daily/weekly macronutrient progress (i.e. chart showing calories eaten)
   d. (If time allows) I want to be recommended to add certain food to increase their calorie intake if I’m not eating enough.
   e. (If time allows) I want to be recommended to remove certain foods to decrease my calorie intake if I’m overeating.

6. Feedback and Maintenance
   <u>As a user,</u>
   a. I want to be able to provide feedback/report bugs to the developers.

   <u>As a developer,</u>
   a. I want to be able to send users emails with information regarding the app (server down, server maintenance downtime).
   b. I want to be able to delete a user’s account from the server.

Non-Functional Requirements

1. Architecture and Performance
   <u>As a developer,</u>
   a. I would like the application to run on iOS
   b. I would like the front-end to be developed in Swift 5.1
   c. I would like the back-end to be developed using SQL
   d. I would like the application to open within 5 seconds
   e. I would like API requests to have a response within 500ms
   f. I would like to server to support up to 1000 requests per minute
   g. I would like the application to support up to 10,000 users
   h. I would like the application to run smoothly without crashing

2. Security
   <u>As a developer,</u>
   a. I would like to utilize a permissions system for information access
   b. I would like to validate all API requests
   c. I would like to audit all logins into the database
   d. (If time allows) I would like to encrypt user information
   e. (If time allows) I would like to implement stored procedures to prevent malicious SQL injections

3. Usability

   As a developer,

   a. I would like to have a streamlined user experience
   b. I would like the GUI to be easily navigated and intuitive
   c. I would like the app to be usable 24/7

4. Hosting/Deployment

   As a developer,

   a. I would like to host the client on the iOS appstore for public download and usage
   b. I would like to deploy the backend to a Google Cloud Platform server to run 24/7
   c. I would like to easily update the front-end and back-end separately after deployment

5. Appearance

   As a developer,

   a. (If time allows) I would like to create a visually pleasing user interface
   b. (If time allows) I would like to support customized color schemes

Design Outline

Our project is a system that provides meal plans in dining courts and the total calories consumed per day to all users who eat in the dining courts. The implementation of this model will be through a client-server design pattern since multiple users will be accessing the shared resources i.e dining courts menus. A database will contain all the data which will be accessed by the server which will allow multiple IOS clients to access information using a web API. Apart from the main architecture i.e Client-Server, we will also use transaction-processing to determine the daily calorie intake.

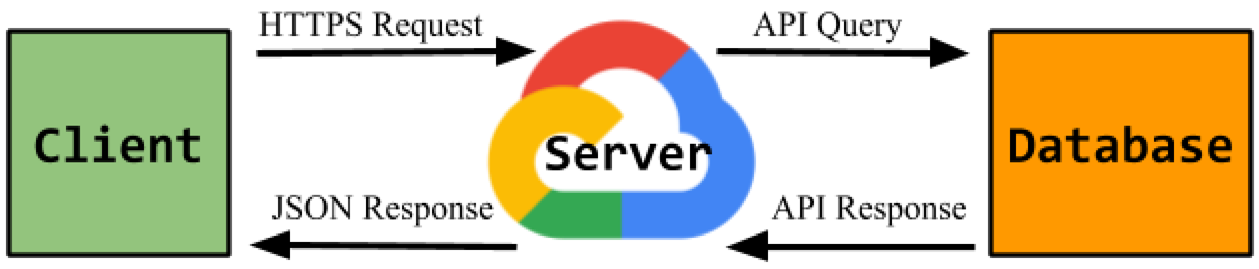
1. **Client**
    A. The client will be running on the user's phone and will be the interface of our system
    B. The client will display UI features where the user can login, look at the menu and create a meal plan according to user's calorie requirements
    C. The client will allow the data from the database to be seen by the user to show the calorie intake per day

2. **API Server**
    A. The server will respond to the user's request to transfer the data from the Purdue dining courts to the user.
    B. The server will transfer the data ,like calories and proteins consumed, entered by the user into the database.
    C. The server will also retrieve information from the database.
3. **Database**
    A. The database will store all the data used in BoilerBite
    B. The data stored in the database will be the meal served in the dining courts at a specific time, the user information and the calorie intake per day.
    C. The database will be responsible for responding to the server's request to access data and executing it.
4. **Statistics module**
    A. The statistics module will analyze the information from the database and tell the user about the daily calorie intake.
    B. The statistics module will later be used to display information about proteins and fats.

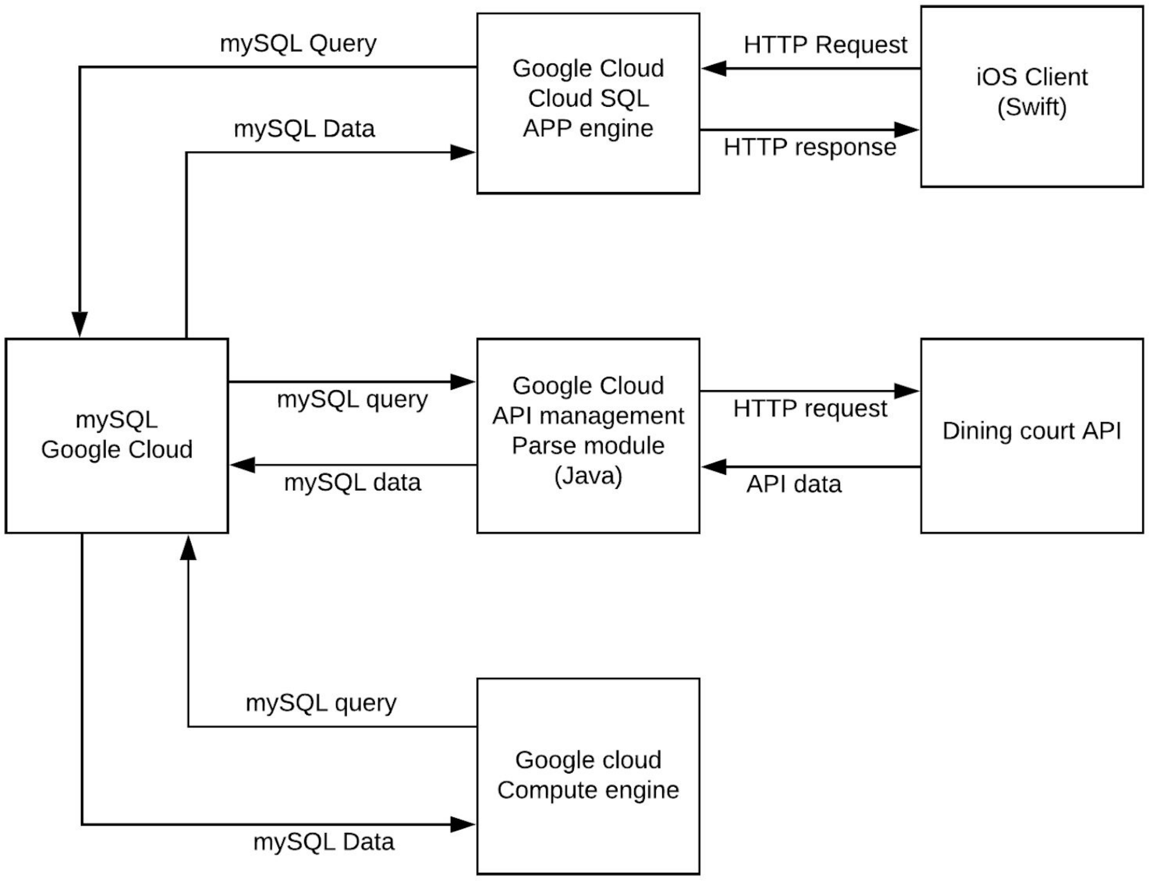
High Level Overview of the System

We will have a number of different clients using the application and attempting to access the database simultaneously. As such, we will be implementing a client-server model. The client (i.e. the mobile application) will communicate with the server by sending HTTPS requests. When the client seeks information from the database, such as information for a user profile, it will send the server a GET request. If the client is updating information in the database, it will instead send a POST request with the necessary info, such as if a user decides to change their calorie goals. The server that will process these requests will be running on a Google Cloud Platform Computer Engine to maintain a 24/7 running time. It will be utilizing the open-source Django framework to handle requests. When the server receives a request, it will make the necessary API query to the database, either to obtain or update information in the database. If the API query is to obtain info, it will return the necessary information. If the API query is to update information, it will return whether or not the update was successful. Once the server receives a response from the database, it will send a JSON object response to the client as confirmation.

The server's main purpose is to process simultaneous requests from many clients, validate them, perform the necessary database operations, and return a properly formatted response to the clients. It will allow for many users to utilize the app and request from the database at the same time.



Detailed View of the Client, Server, and Database



The UML diagram shows the sequence of events when the application is started. The iOS client communicates with Google Cloud SQL App Engine using HTTP requests between the two. The SQL app engine uses mySQL query to communicate with mySQL Google Cloud while mySQL Google Cloud communicates with SQL app engine using mySQL data. The mySQL google cloud also contains the parsed data obtained from the Dining court API. The Google cloud Compute engine will be used to compute the calories consumed daily and will send the calculated data to the mySQL Google cloud to be stored and accessed.

Design Issues

Functional Issues

1. **Do we want a login system?**

   Issue statement: How we decide to implement a login system will have a major impact on the structuring of our data and our application's interface

   - **Option 1**: No login system
   - **Option 2**: Login system with username and password unique to our service
   - **Option 3**: Login system with integration of other accounts (e.g. Google, Facebook)

   Pros and Cons: Option 1's not having a login system has no overhead, however it would make progression of data across multiple devices very difficult. Option 2's unique login system would have a decent bit of overhead, but make progression across multiple devices and individual user profiles much easier to handle. Option 3's integrated login system would have a very large amount of overhead, however it would allow the user to connect their BoilerBite account with other major brand accounts for social purposes.
   Decision: We plan to implement option 2. We decided that a login system would be very useful, as it would allow users to keep track of their nutrition goals over a very long term and customize their goals to better fit their needs. In addition, it would allow them to maintain their progress should they get a new device. However, we felt that integration of accounts from other services was not needed, as it provides no real benefit to our application and incurs unnecessary overhead.

2. **What nutrients do we want to allow the user to track?**

   Issue Statement: Deciding which nutrients to track will impact our nutrition progression system, one of the main focuses of our application

   - **Option 1:** Allow the users to track only calories
   - **Option 2:** Allow the users to track a pre-selected set of nutrients
   - **Option 3:** Allow users to track any nutrient they specify

   Pros and Cons: Option 1 would make a user's nutrition very easy to track and set goals for, however it is not very expansive and encompassing of all nutrition. Option 2 would make tracking a user's nutrition more difficult, but would allow for tracking various different health measures. Option 3 would require a large amount of overhead, but allow for the user to fully customize all of their nutrition goals.
   Decision: We plan to implement option 2. Although calories is undoubtedly the most important macronutrient to track due to its major role in weight loss/gain, users may want to track other nutrients in order to meet other fitness goals. For example, someone who goes to the gym very often may want to ensure they are eating enough protein throughout their day. As such, we have pre-selected a set of nutrient that users may

choose to track: calories, protein, and fat. We did not allow the user to track any nutrient they specify because there are simply too many nutrients to account for when looking at things like vitamins and minerals, and the implementation would prove extremely time-consuming for our current project scope.

3. What information is necessary to create a user profile?

Issue Statement: How we implement user profiles will influence the structure of our data and how we store data in the database

- **Option 1:** No user profiles
- **Option 2:** Create a user profile with just their name
- **Option 3:** Create a user profile with name, weight, height, and age
- **Option 4:** Create a user profile with name, weight, height, age, and profile picture

Pros and Cons: Option 1 would be very simple to implement, but would leave a lot of the data unstructured when we store it. Option 2 would allow for data to be stored in a more structured format, however would not have a lot of information in the user's profile and would restrict the user's goal system. Option 3 would require us to store significant information about the user in the database, however it would make implementing a goal system easier. Option 4 would require the size of each user profile to increase significantly due to the picture, but would allow for more social functions on the application.

Decision: We plan to implement option 3. A user profile with just a name would not be useful in helping the user accomplish their fitness goals. With their weight, height, and age, we can provide much more tailored recommendations regarding the food they eat and their overall meal plan. However, because our app will not be a social app, it is unnecessary for the profiles to have a profile picture.

4. What is our recommendation system going to suggest for users?

Issue Statement: Our recommendation system is very important in helping users achieve their nutrition goals. What it recommends could influence how likely the user is to follow its suggestions.

- **Option 1:** No recommendation system
- **Option 2:** Recommend users specific items to supplement their plan
- **Option 3:** Recommend users combinations of items as meals
- **Option 4:** Recommend an entire daily meal plan for the user

Pros and Cons: Option 1 would be very simple to implement, however not having a recommendation system would prove detrimental for the user. Option 2 would be helpful to the user as a supplement to their own changes, however may not be able to help the user completely fulfill their goals. Option 3 would be more comprehensive in helping the user with their nutrition goals, however it is more likely we recommend items the user

does not like. With Option 4 it would be possible to completely fulfill the user's nutrition goals, however it is much more likely that they will not follow the plan in its entirety, thus rendering it less useful.
Decision: We plan to implement option 2. Having a recommendation system in place would make it much easier for users to see what they are lacking in their meals and make the appropriate changes. However, we don't want to recommend too much, as that would be stifling and make it more likely that the user won't follow the recommended plan because they don't enjoy those food items. Instead of recommending entire meals, we are opting to recommend individual items. This will help fill any sort of nutritional gaps while still giving the user more freedom regarding what they eat.

Non-Functional Issues

1. How do we want to implement our database?

Issue Statement: The way in which we implement the database will change what software we use and how we integrate the database with the server

- **Option 1:** A relational database - SQL
- **Option 2:** A non-relational database - NoSQL
- **Option 3:** An alternate non-relational database - MongoDB
- **Option 4:** No database

Pros and Cons: Option 1 would require that we structure our data rather strictly, however it would allow for complex queries and eliminate duplicate data. Option 2 means our data structure would not have to be as strict, however we would be limited to simpler queries and have the possibility of dealing with duplicate data if we have any mistakes in entries. Option 3 faces the same problems as Option 2. Option 4 is simply not a possibility, as we need a database in order to store the data for all of our application's users.
Decision: We plan to implement option 1. A database is certainly required for this project, as we need to store not only user account and login information, but also the user profile information that is tied to each account. We chose a relational database over a non-relational database for a few reasons. Firstly, because our data is already set and structured, there is less reason to use a non-relational database meant for dynamically structured data. Secondly, the scalability of a relational database would prove very useful as our userbase expands into the thousands of users and our database grows proportionally. Finally, the popularity and extensive documentation of SQL makes it easier for us to learn it, given that none of our team members have significant experience using any database.

2. What platform do we want our application to run on?

Issue Statement: The platform our application targets will affect the audience we are able to reach, as well as how we will develop the application

- **Option 1:** iOS mobile application

- **Option 2:** Android mobile application
- **Option 3:** Web application

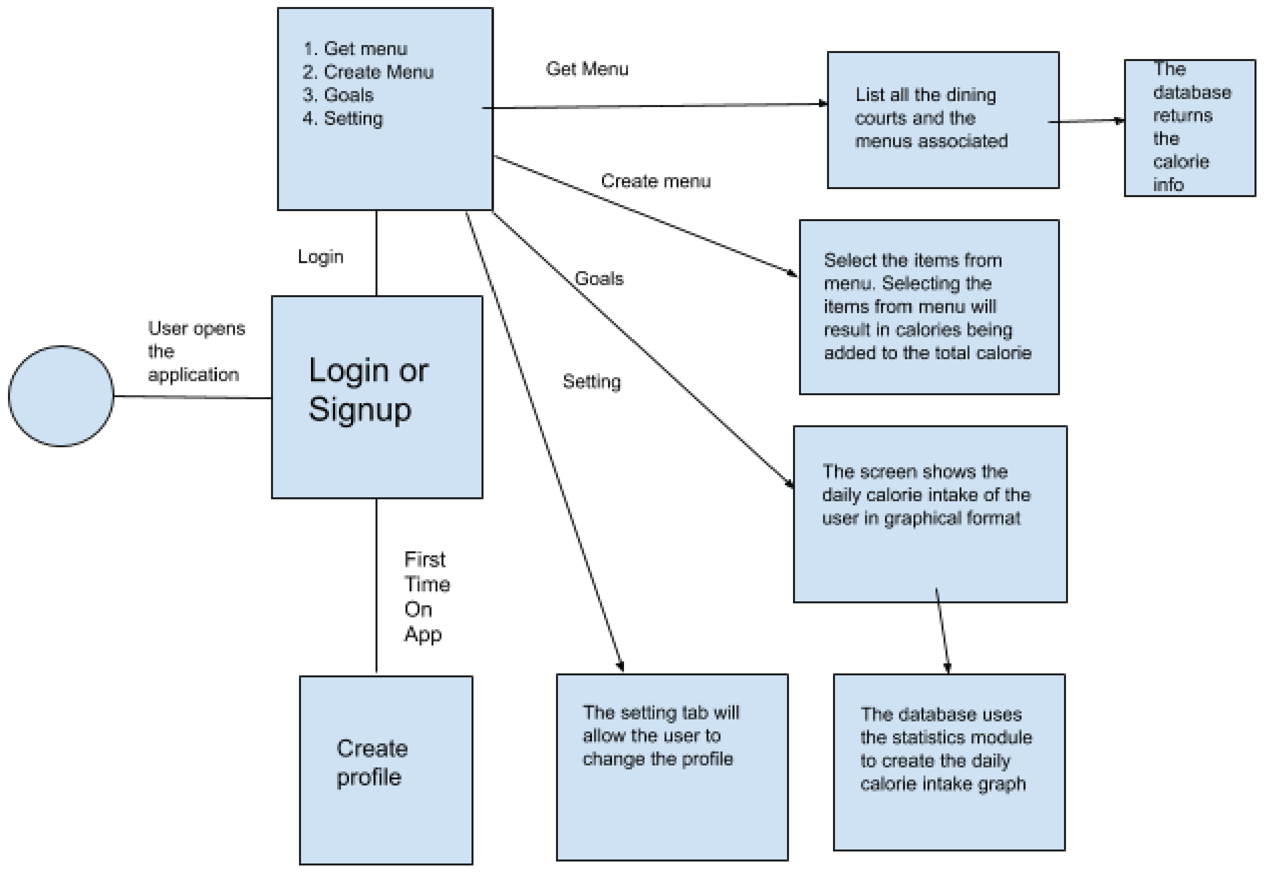
Pros and Cons: Option 1 is great for us as all members of our team have iPhones and our vision for the app had mobile in mind, however we would be unable to reach those who don't have iPhones. Option 2 would still allow us to develop for mobile, however none of our team members have an iPhone and we would be unable to reach those without an Android phone. Option 3 means we would not be developing specifically for mobile like we envisioned, however we would have access to the largest audience.
Decision: We plan to implement option 1. All of the members on our team have iPhones, and so developing for iOS would prove much more optimal. Although a web application was also an option, we ultimately decided on a mobile app because we believe our project would work best there and we would reach the largest audience that way.

3. Where do we want to host our server?

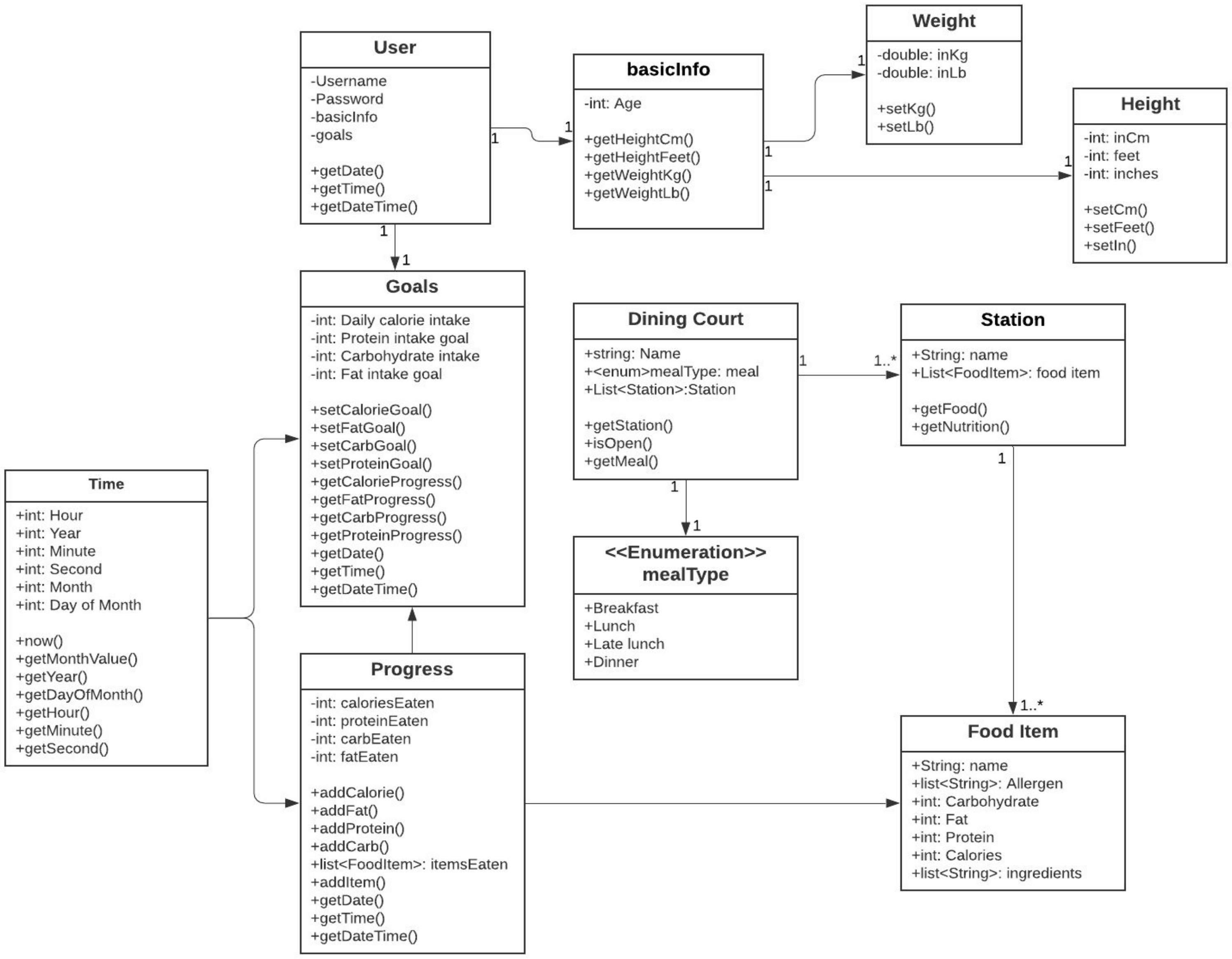
Issue Statement: The service we choose to host our server will impact how we integrate the database, as well as the overall runtime of the server
Option 1: Amazon Warehouse Services (AWS)
Option 2: Azure
Option 3: Google Cloud Platform
Pros and Cons: Option 1 might be a little bit easier to work with, as one of our members has a little bit of experience with AWS, however it would prevent us from running the server 24/7 on the free tier. Option 2 also has a free tier, however once again it is not enough for us to run the server continously. Option 3 would require more research because none of our team members have worked with it before, its free tier would allow us to run the server 24/7.
Decision: We plan to implement option 3. We require a server in order to properly implement our client-server-database model. In order to have the server running 24/7 like we originally outlined, the best solution is to host it on a cloud computing provider. Although one of our team members has some experience using AWS, the free tiers for AWS and Azure don't meet our needs. They have some rather restrictive limits on the number of hours a free server can use. Google's Cloud Platform, however, allows us to run one instance for free non-stop onto which we can deploy our server.

Design Details

Activity/State Diagram


1. Get menu
2. Create Menu
3. Goals
4. Setting
Get Menu
List all the dining courts and the menus associated
The database returns the calorie info
Create menu
Select the items from menu. Selecting the items from menu will result in calories being added to the total calorie
Login
Goals
User opens the application
Login or Signup
Setting
The screen shows the daily calorie intake of the user in graphical format
First Time On App
Create profile
The setting tab will allow the user to change the profile
The database uses the statistics module to create the daily calorie intake graph

Class Diagram



Description of Classes and their Interactions

Since the application will be object-oriented, the class diagram will also describe how the classes and “structs” work together.

User:

- Represents the user of the application
- Contains login information, user basic information, user goals, and user progress
- Created when a user creates an account

BasicInfo:

- Contains the basic information of the users(Age, weight, height)
- Created when a user creates an account

Weight:

- Stores the users’ weight information in kg and lb
- Whenever the user enters/changes his/her weight, depending on the unit that the user prefers, the other unit will be updated as well.(if the user input/update the weight in lbs., the app will update the value of the weight in kg as well)
- Created when the user first enters his/her weight

Height:

- Stores the users height in centimeters and feet
- Feet and inches will be entered separately and therefore will be stored with two variables
- Whenever the user updates the height in his/her preferred unit, the other unit will be updated based on the new input.
- Created when the user first enters his/her height

Goals:

- Stores the users’ daily intake goals
- Each time the user changes any of his/her goals, getDateTime() will record the date and time the goal is changed
- Created when the user creates an account(initialized to null before user input)

Time:

- Has the functions that provide time information to Progress and Goals.
- Created whenever the user changes his/her goals and whenever the user records a food item to his/her progress

Progress:

- Stores the users' consumed nutrition information based on the food items that the users' eat
- The progress will be stored into the database daily and will be cleared
- itemEaten will store the items that the users' eat on a given day into a list
- Each time the user records a food item, getDateTime() will record the date and time the item is recorded
- First created when the user input his/her daily consumption goals, and cleared daily

FoodItem:

- Stores the information of a food item that will be pulled from the Purdue dining court API (allergens, ingredients, and nutrition information)
- Created as the user records the food he/she has eaten at the dining court, and when the user selects a station to check the food items being served at the station

Dining court:

- Represents the Purdue dining courts
- Each dining court will have a name, a list of stations, enum representing the meal of the day, and a boolean to check if the dining court is open or not
- All the information will be pulled from the dining court API
- Created as the user selects a dining court to check the menu

mealType:

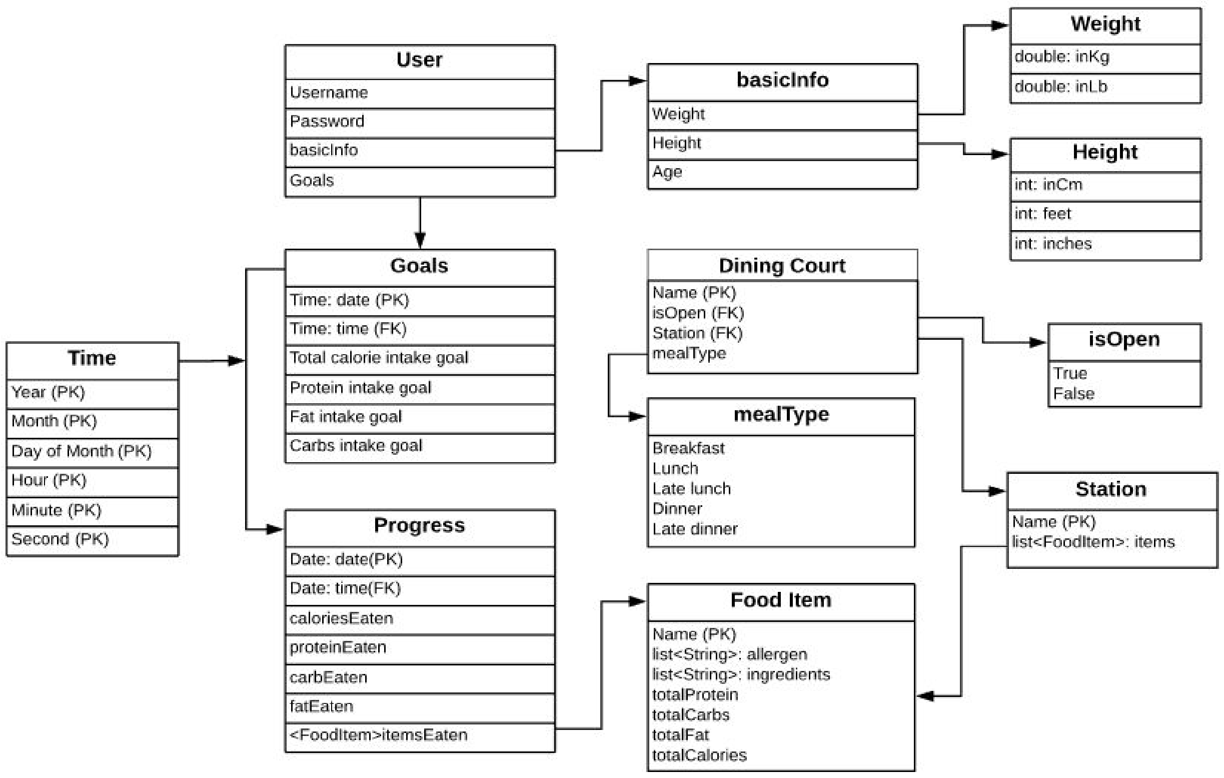
- An enumeration that informs the users of the meal of the day that the selected dining court is currently serving (breakfast, lunch, late lunch, dinner)

Station:

- Represents the stations of a given dining court.
- Each station will have a name and a list of food item being served at the station
- Created when the user selects a dining court to check the stations in the dining court

Database Design

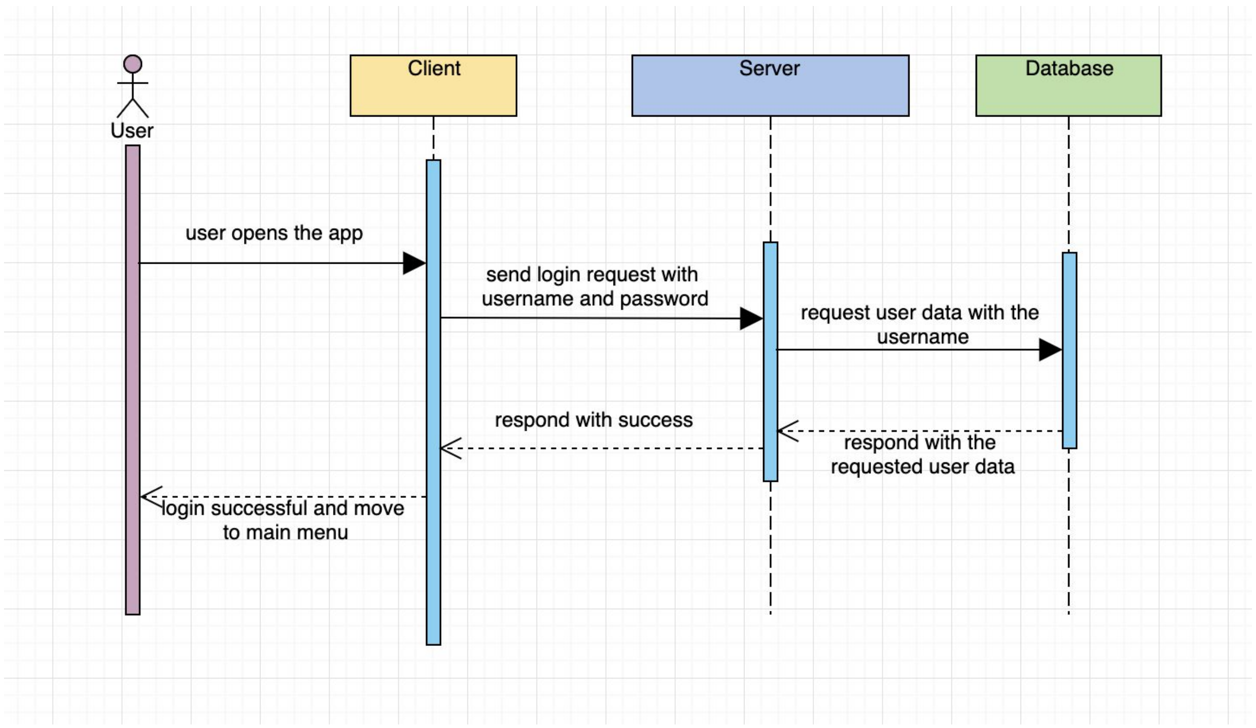
The database will be set up using mySQL and will be hosted on the Google cloud platform. Each class in the class diagram will have a table and be indexed by a Primary key. There will also be foreign keys to show relationships between keys (stations in dining courts). Functions that parse data from the dining court API will be implemented to make updating menus and checking nutrition information more efficient for the users.



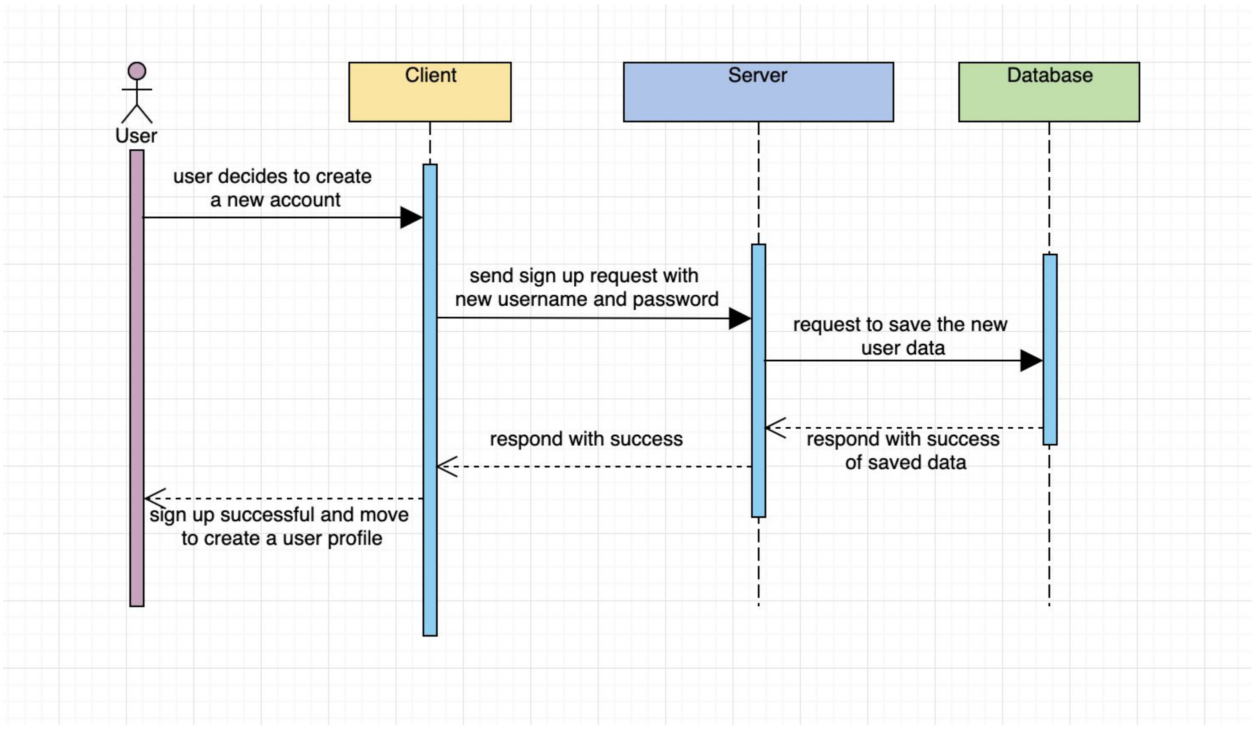
Sequence Diagrams

The diagrams show the sequence of major events in our application. The diagrams show that when a user performs an action the client will send a message to the server which in turn will either acquire data from the database or save data in the database. The server then sends the response back to the client and updates the screen or GUI to showcase the output.

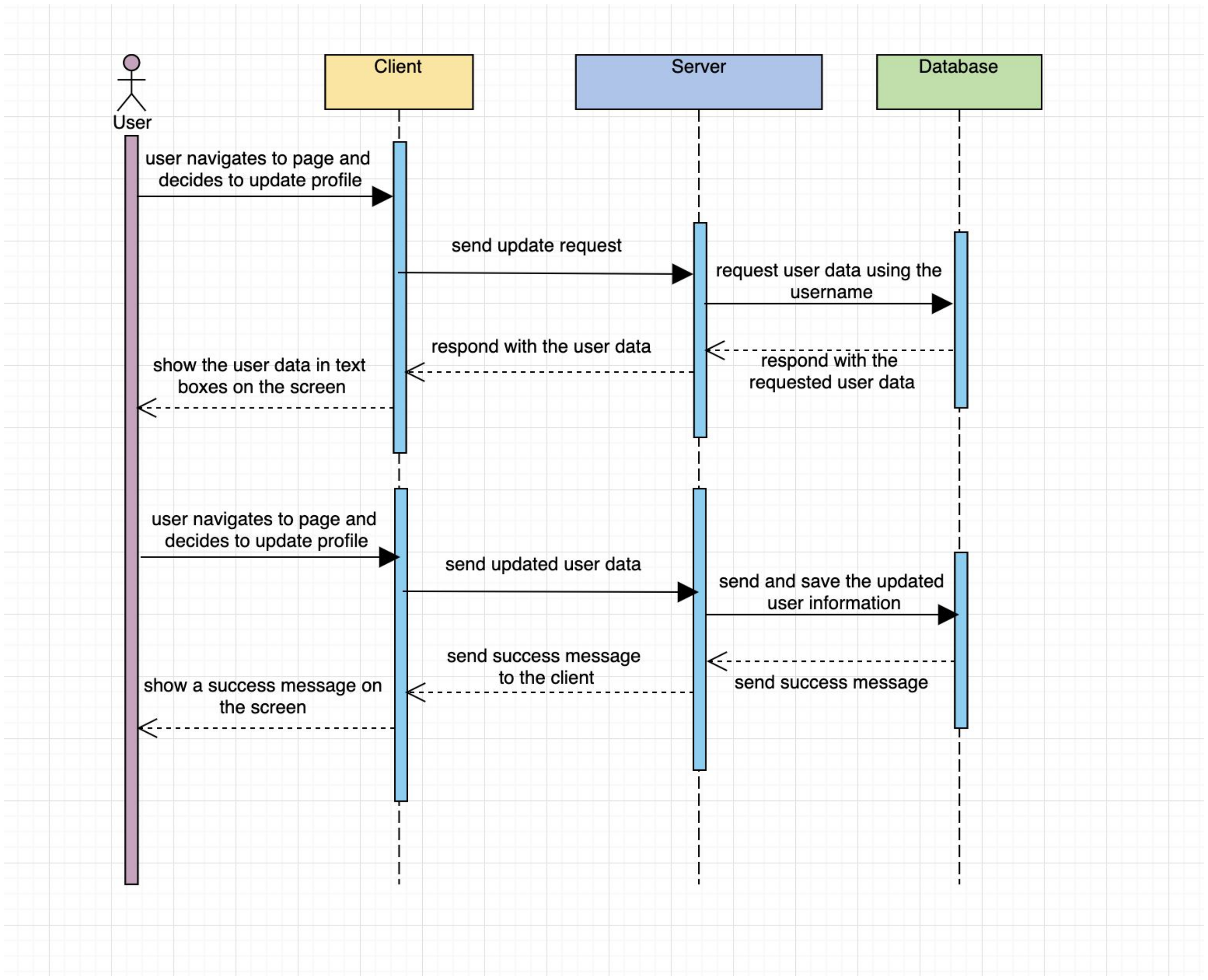
- Sequence of events when the user first opens the application and tries to log in


User
Client
Server
Database
user opens the app
send login request with username and password
request user data with the username
respond with the requested user data
respond with success
login successful and move to main menu

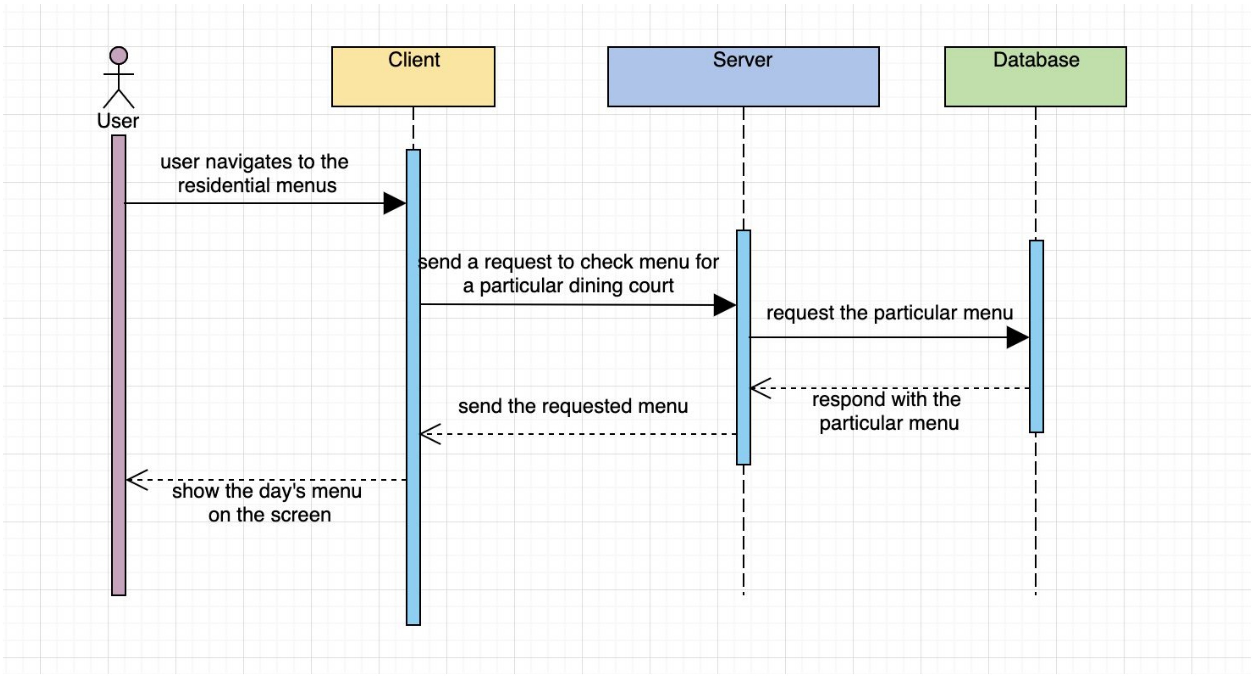
- Sequence of events when the user creates a new account


User
Client
Server
Database
user decides to create a new account
send sign up request with new username and password
request to save the new user data
respond with success of saved data
respond with success
sign up successful and move to create a user profile

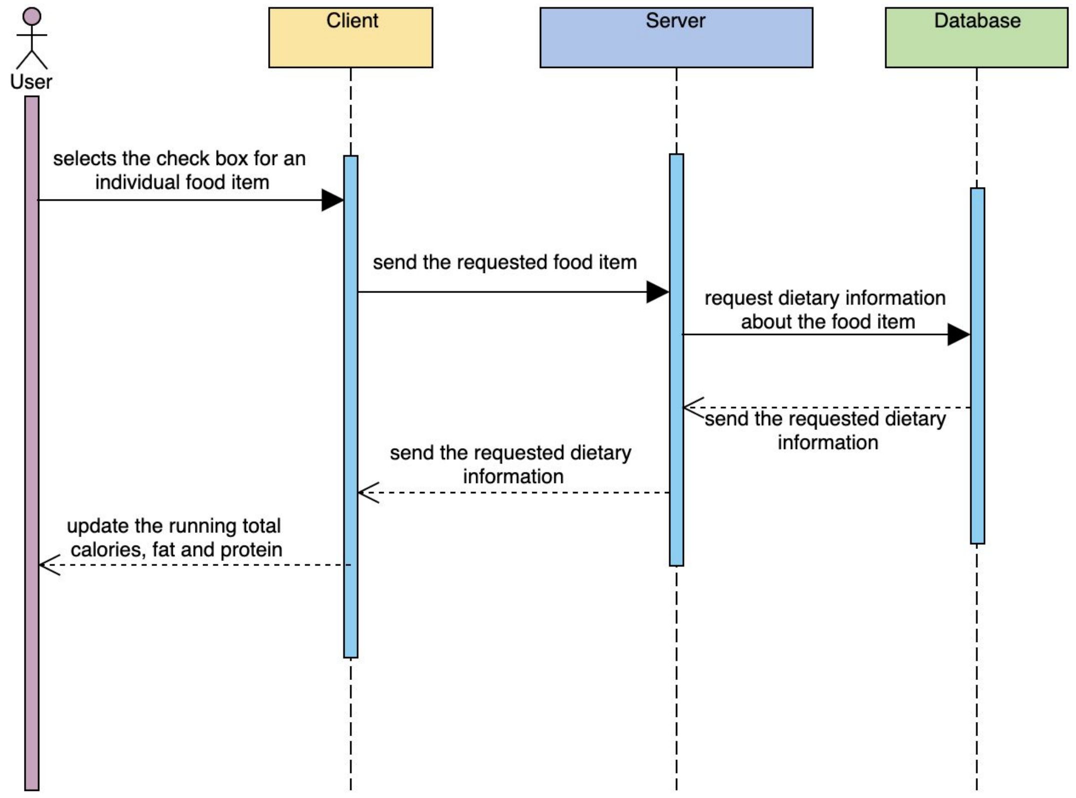
- Sequence of events when the user wants to update their profile


User
Client
Server
Database
user navigates to page and decides to update profile
send update request
request user data using the username
respond with the requested user data
respond with the user data
show the user data in text boxes on the screen
user navigates to page and decides to update profile
send updated user data
send and save the updated user information
send success message
send success message to the client
show a success message on the screen

- Sequence of events when the user requests to see the days menu

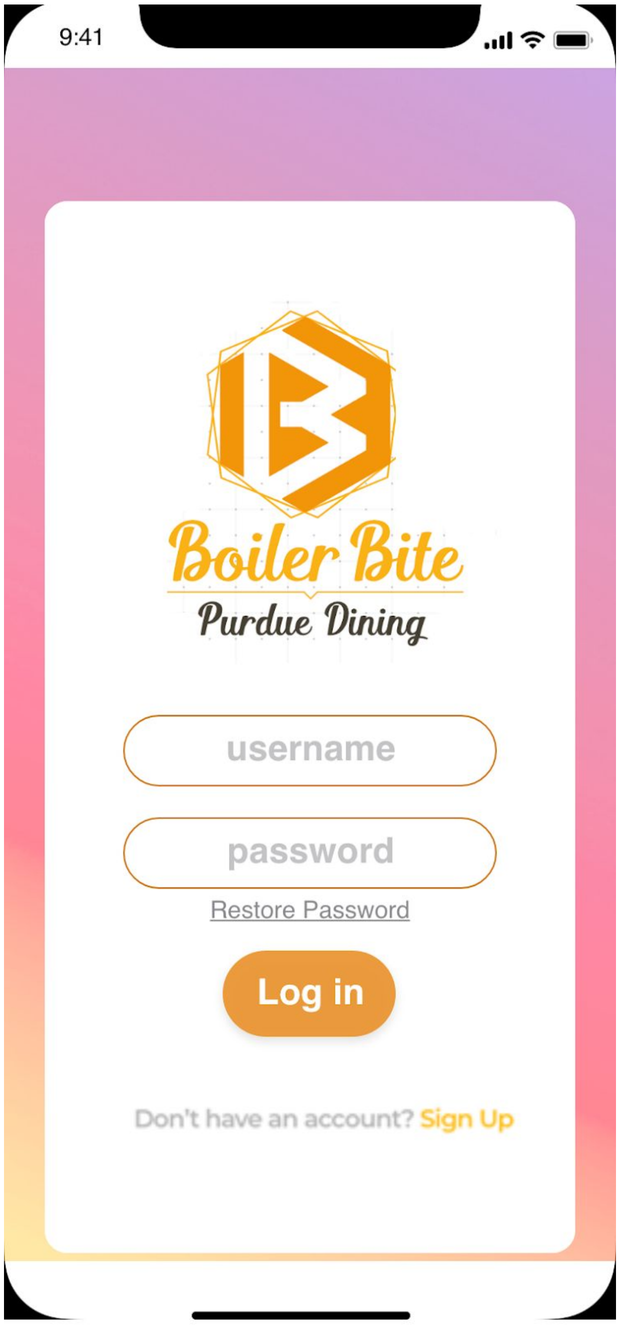

User
Client
Server
Database
user navigates to the residential menus
send a request to check menu for a particular dining court
request the particular menu
respond with the particular menu
send the requested menu
show the day's menu on the screen

- Sequence of events when the user adds a food item to their meal

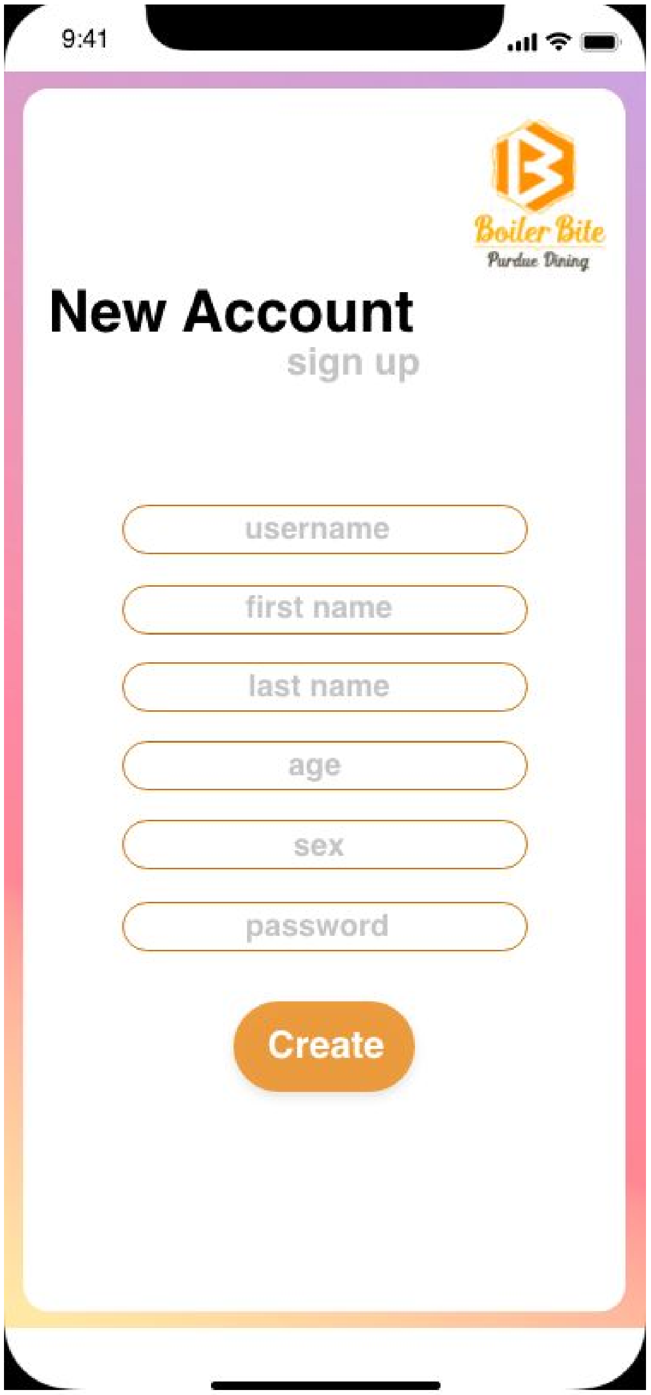

User
Client
Server
Database
selects the check box for an individual food item
send the requested food item
request dietary information about the food item
send the requested dietary information
send the requested dietary information
update the running total calories, fat and protein

UI Mockup

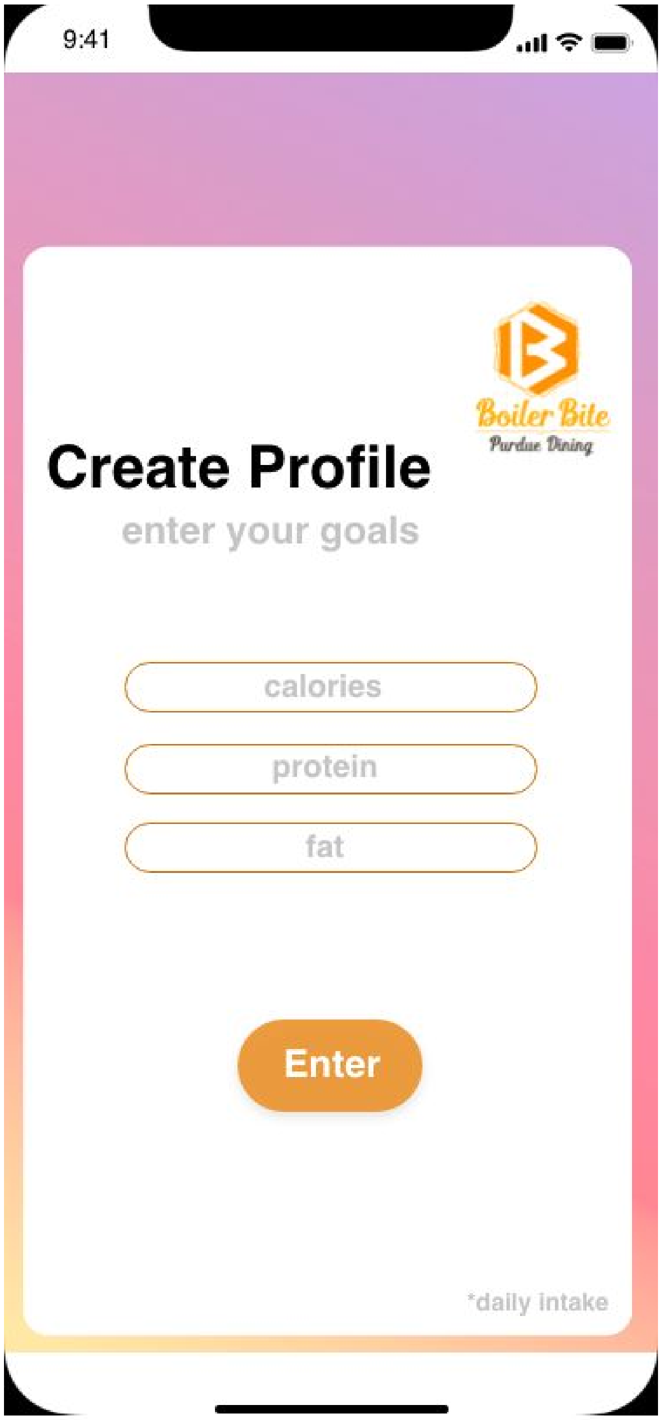
- Login Page



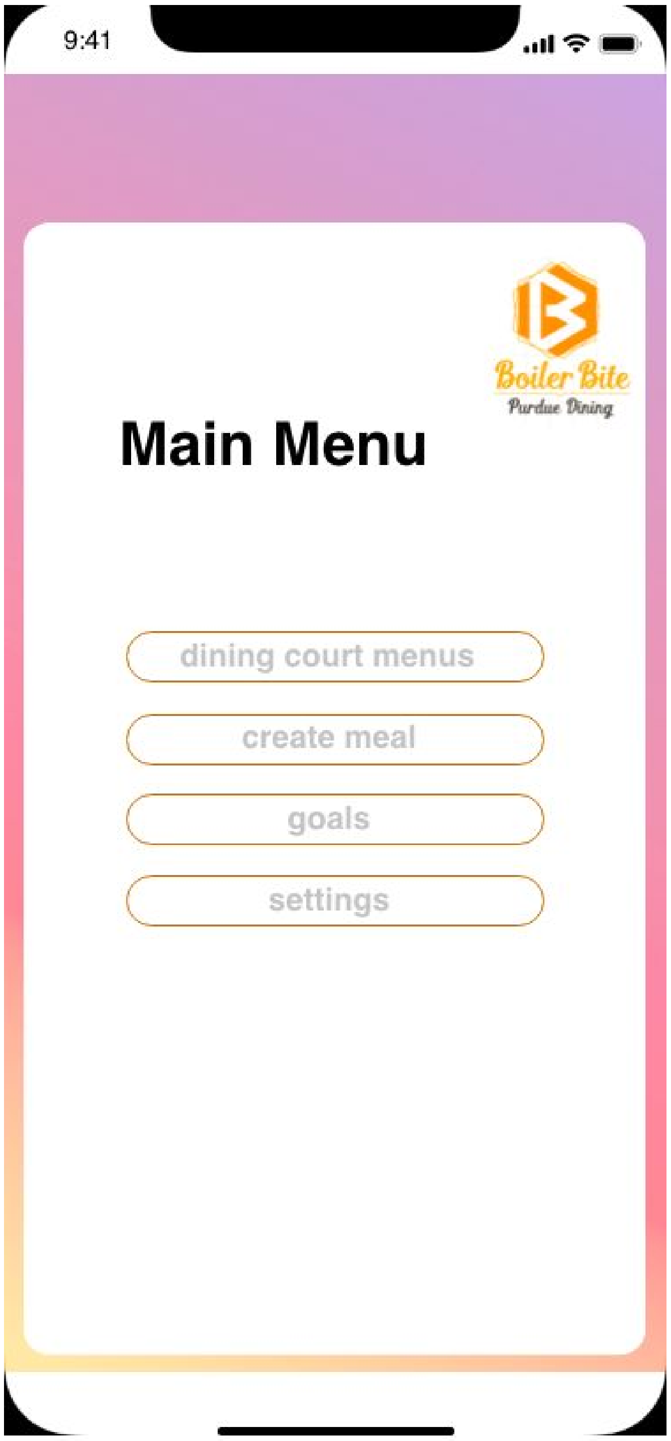
- Sign up Page


9:41
Boiler Bite
Purdue Dining
New Account
sign up
username
first name
last name
age
sex
password
Create

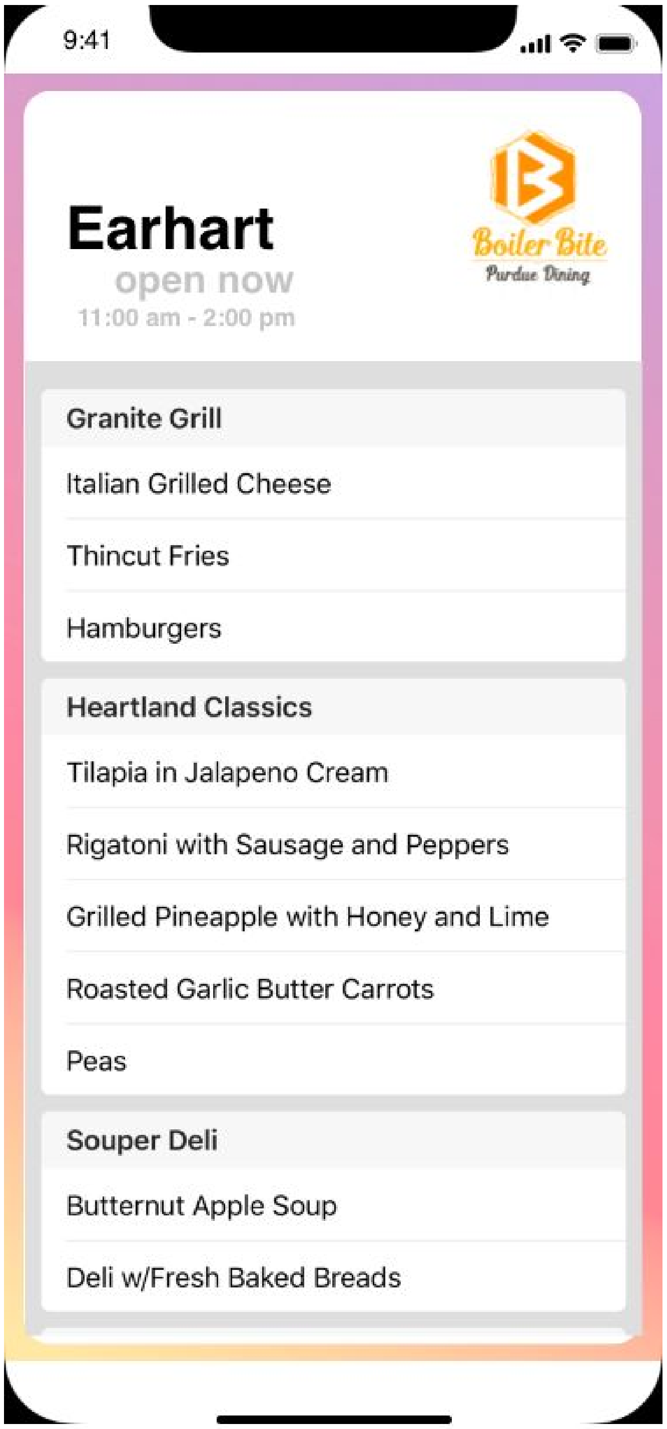
- Create Profile



- Main Menu


9:41
Boiler Bite
Purdue Dining
Main Menu
dining court menus
create meal
goals
settings

- Individual Residential Menus



Earhart

open now

11:00 am - 2:00 pm

Granite Grill

Italian Grilled Cheese

Thincut Fries

Hamburgers

Heartland Classics

Tilapia in Jalapeno Cream

Rigatoni with Sausage and Peppers

Grilled Pineapple with Honey and Lime

Roasted Garlic Butter Carrots

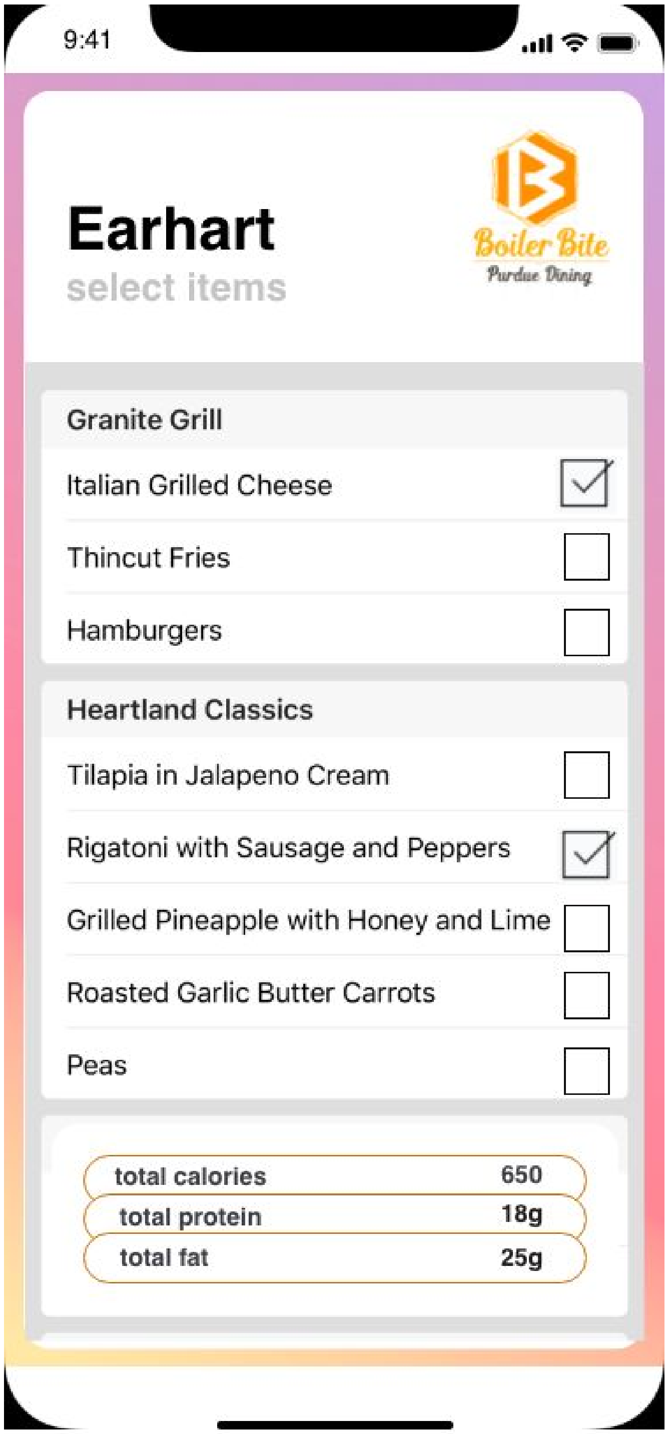
Peas

Souper Deli

Butternut Apple Soup

Deli w/Fresh Baked Breads

- Create Meal



Earhart
select items

Granite Grill

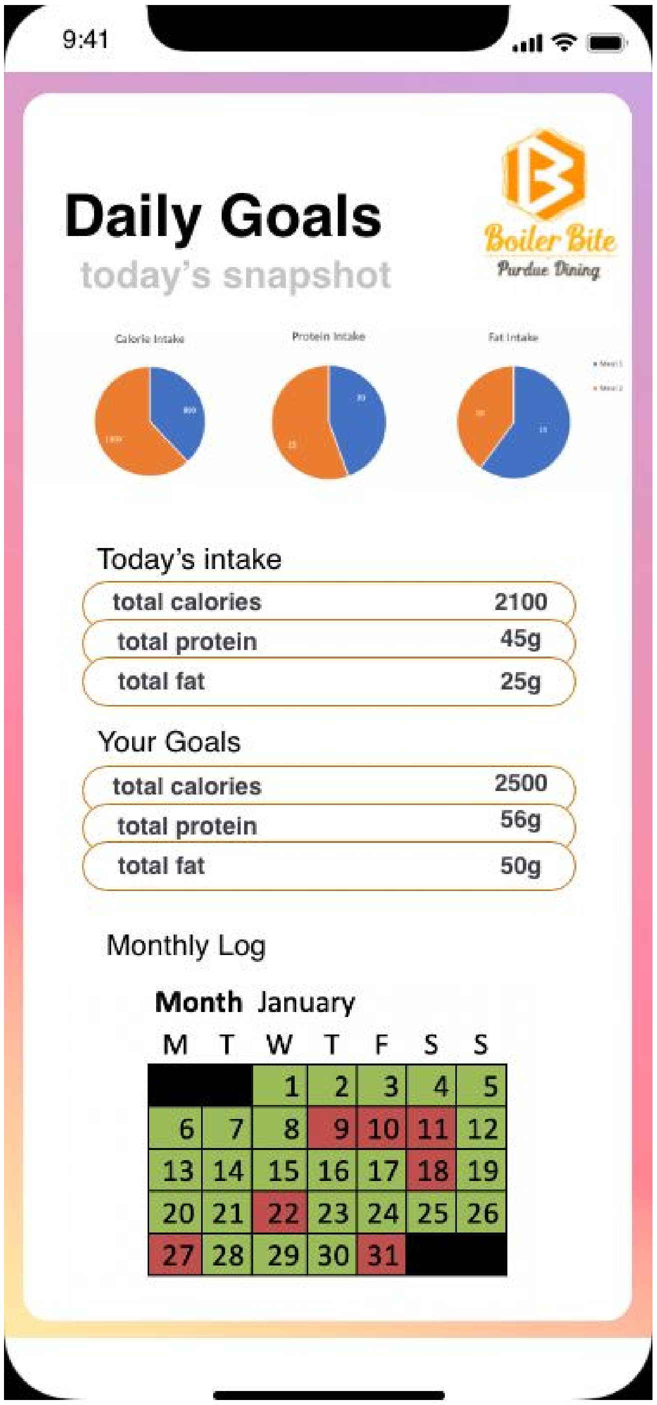
Italian Grilled Cheese	✓
Thincut Fries	☐
Hamburgers	☐

Heartland Classics

Tilapia in Jalapeno Cream	☐
Rigatoni with Sausage and Peppers	✓
Grilled Pineapple with Honey and Lime	☐
Roasted Garlic Butter Carrots	☐
Peas	☐

total calories	650
total protein	18g
total fat	25g

- Goals Summary



Daily Goals

today's snapshot

Today's intake

total calories	2100
total protein	45g
total fat	25g

Your Goals

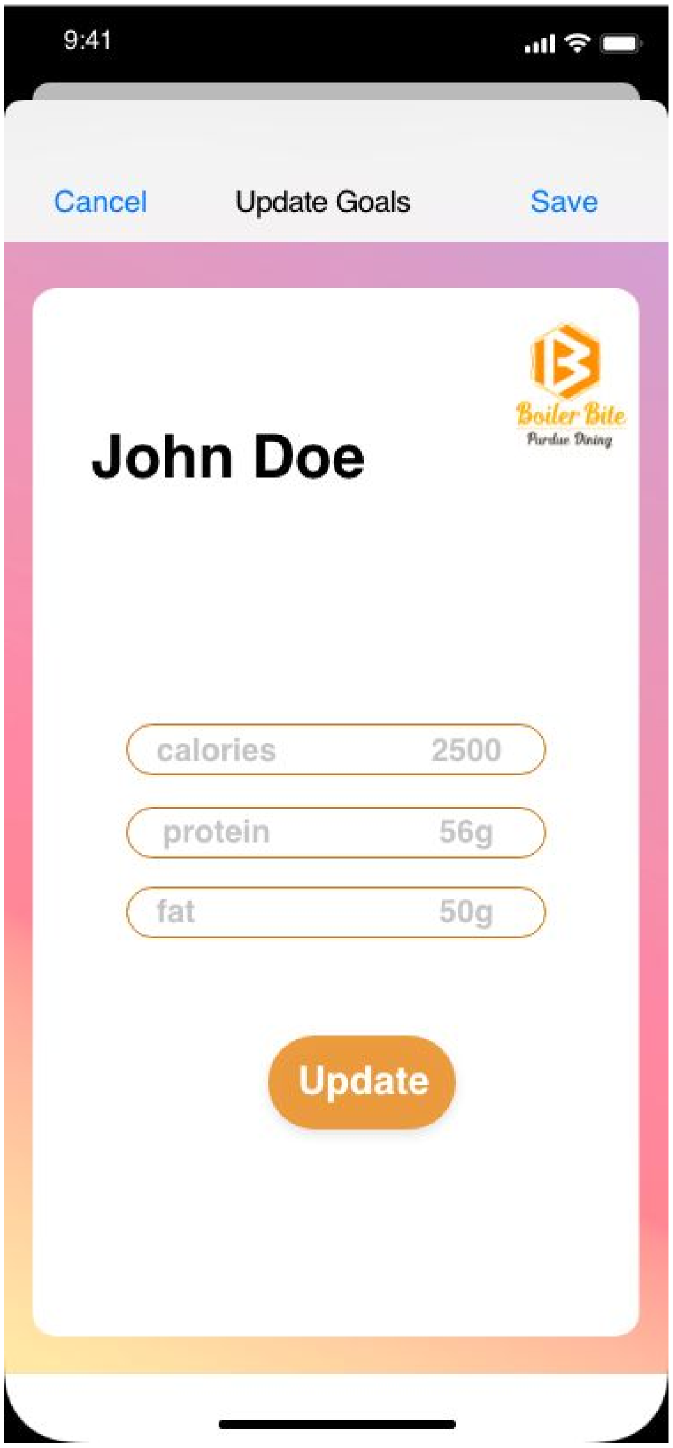
total calories	2500
total protein	56g
total fat	50g

Monthly Log

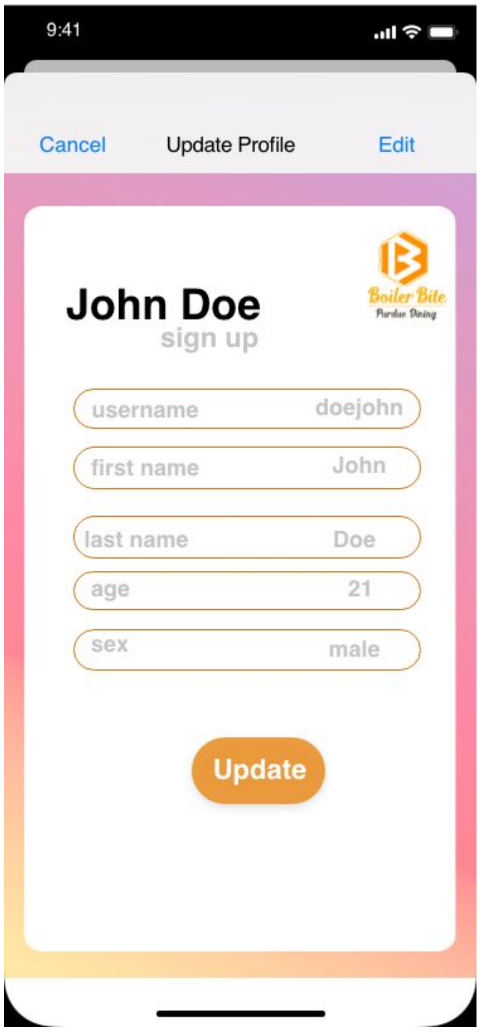
Month January

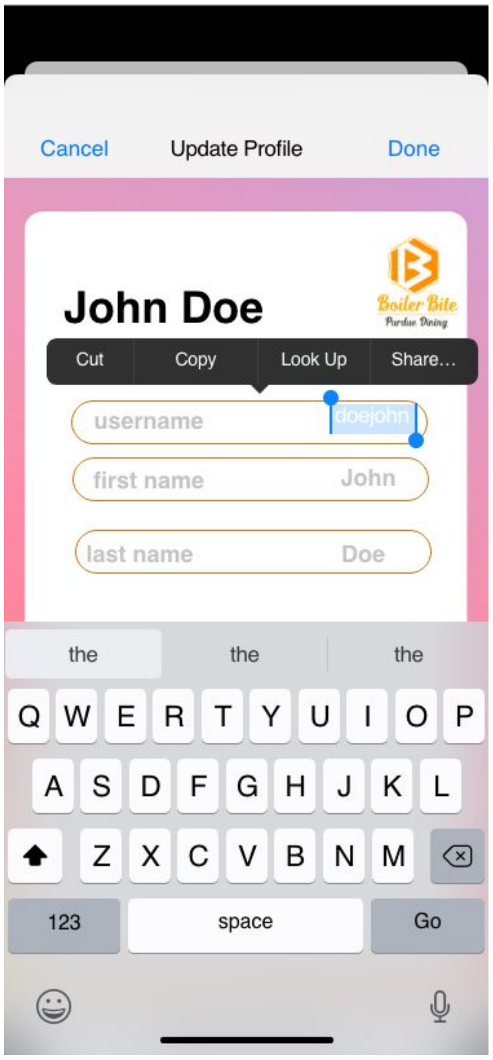
M	T	W	T	F	S	S
		🟩 1	🟩 2	🟩 3	🟩 4	🟩 5
🟩 6	🟩 7	🟩 8	🟥 9	🟥 10	🟥 11	🟩 12
🟩 13	🟩 14	🟩 15	🟩 16	🟩 17	🟥 18	🟩 19
🟩 20	🟩 21	🟥 22	🟩 23	🟩 24	🟩 25	🟩 26
🟥 27	🟩 28	🟩 29	🟩 30	🟥 31		

- Update Goals

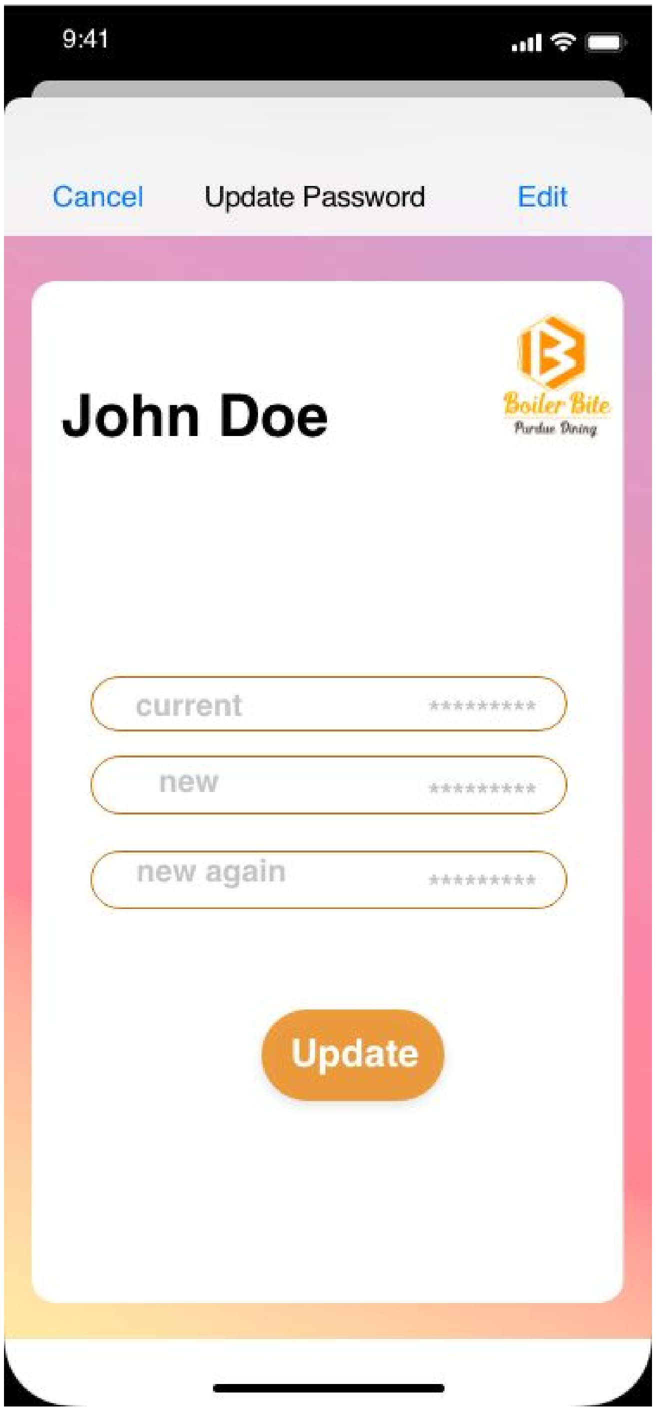


- Update Profile

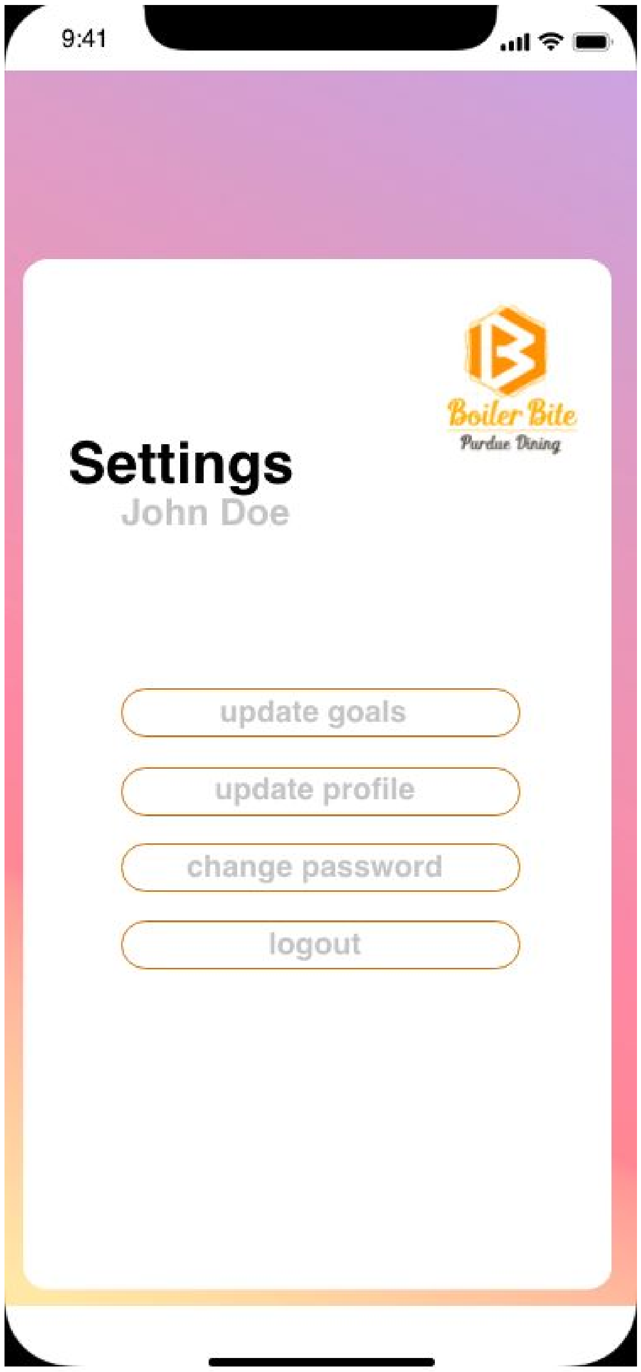




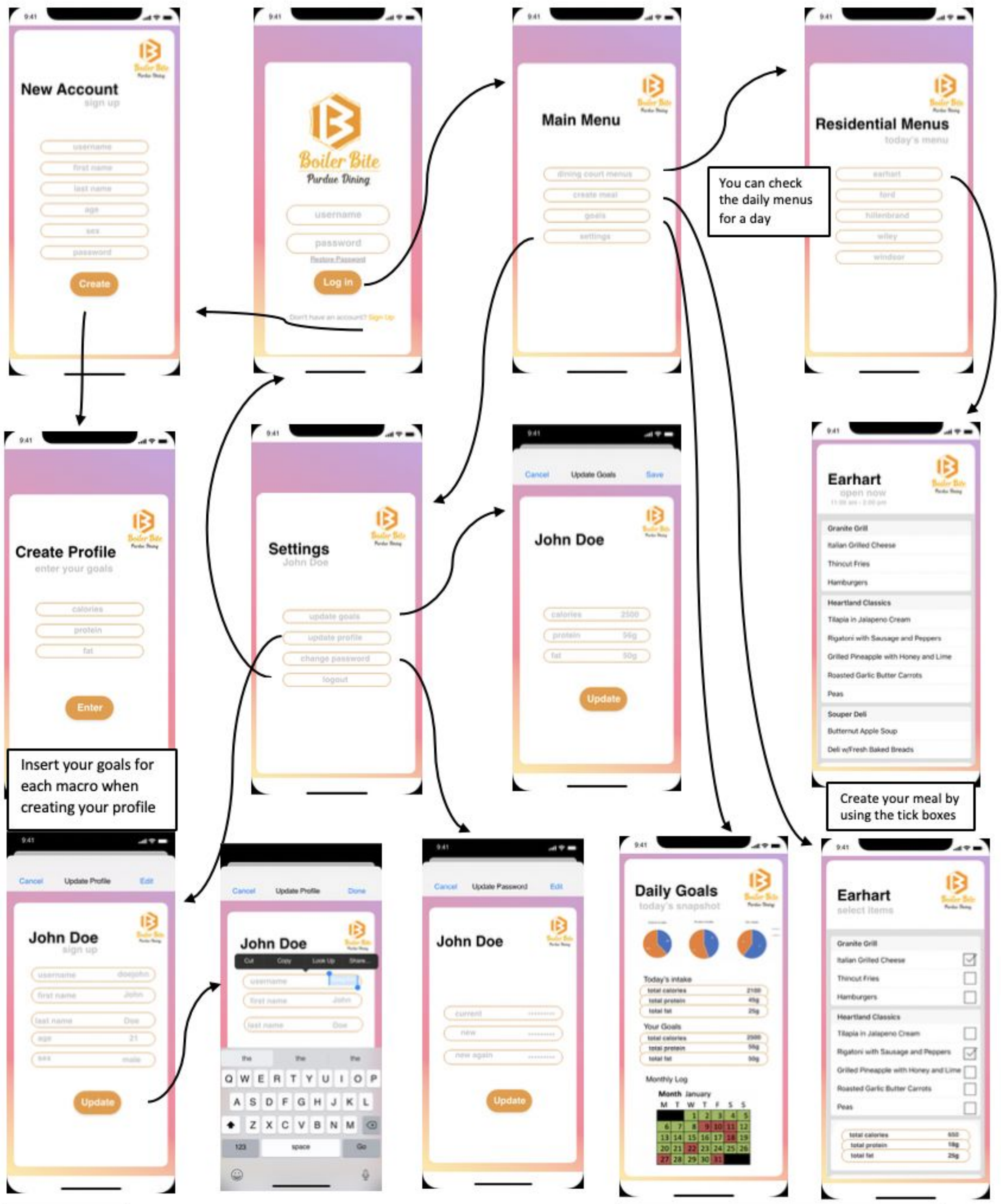
- Update Password



- Settings


9:41
Boiler Bite
Purdue Dining
Settings
John Doe
update goals
update profile
change password
logout

Program Flow



You can check the daily menus for a day

Insert your goals for each macro when creating your profile

Create your meal by using the tick boxes

Edit your information

Shows your daily goals for each macro as pie charts and a monthly board where green signifies a "goal met" and red signifies "goal not met"