



COMP4342

Mobile Computing

Group project (2022)

Cheung Sui Wing	21027547D
Kwong Chun Him	21028468D
Lau Man Chun	21027257D
Cheng Chi Kit	21028079D

Contents

Overview of the project	2
System components	5
Client-side application	5
Server	5
Database	5
Third party API	5
Functional Requirements	6
Mobile Application	6
RESTful Server	7
Data Model.....	8
Testing.....	9
Installation guide.....	14
Backend server setup	14
Application server setup	15
Run the Application with mobile	17
User Guide	18
Mobile Applications User guide	18
Peer Review	27

Overview of the project

As the public has become more aware of the concept of healthiness, calculating calories has become one of the most important things in diet control. Many people have their target calorie absorption per day calculated by themselves or provided by their doctor.

At the same time, there are many applications in the market that allows users to input food name to create a record. And calculate the calories for the user. However, the process is not user-friendly since the users need to input the food by name and then search one by one.

The main purpose of our mobile application is to help the user to improve the food recording experience. It allows people to take/choose a picture and send it to the server. Using an object detection skill to recognize the foods and find the food nutrition data from a public food database API. The user can choose the food by clicking the result response from the server instead of inputting and searching by the name to create a food calorie record every time.

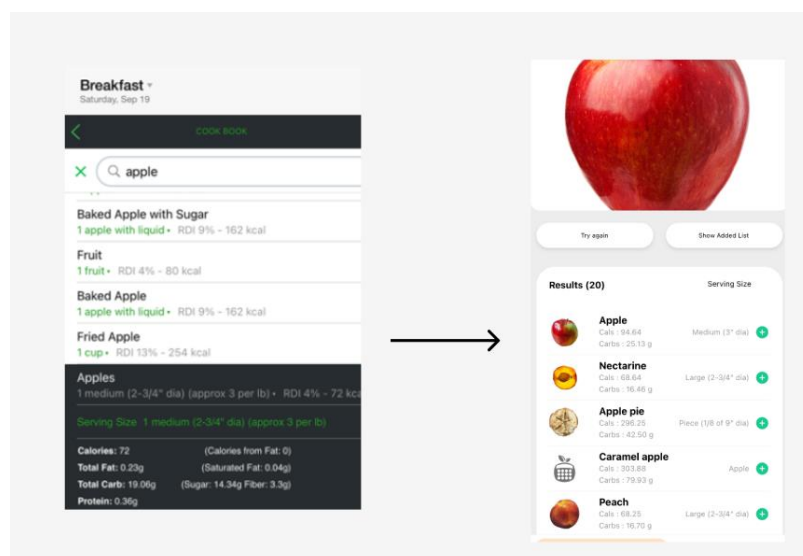


Figure 1 Left are using input, Right using AI (our app)

Left side of figure 1 show the application require user to type “Apple” to search the food one by one.

Right side of figure 1 shows our application, which allow user to take picture to recognized that is an apple.

If there is **more** food needed to record, our application is much more convenient.

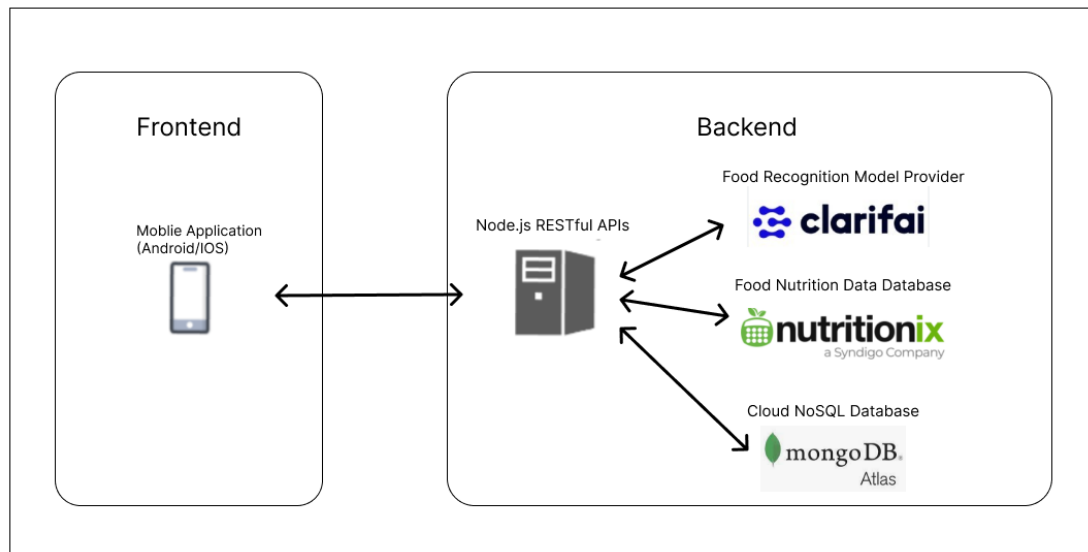


Figure 2 System architecture

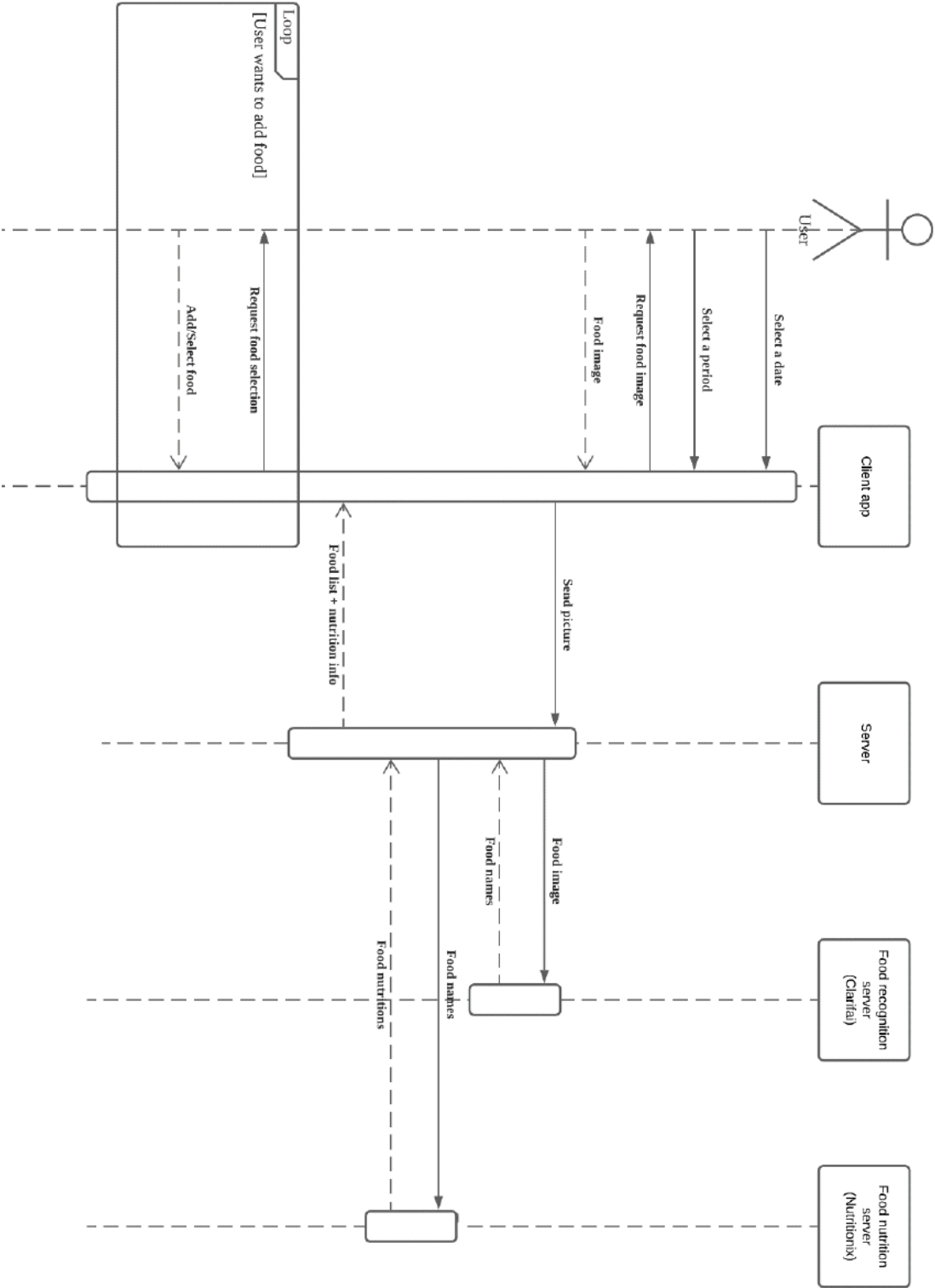
As Figure 2 shows, our system includes frontend and backend applications.

Our main function is to allow the user to take/pick a picture of the food in the mobile app. This will then be sent to the server which will use the image to get the food recognition result and nutritional data by accessing some third party APIs, and finally return to the front-end phone so that the user can select and record the food directly.

Here is system logic of this function:

1. Users take/pick a picture by using the mobile application.
2. The application sends the picture to the server
3. The server sends the picture to Clarifai API and get the food recognition data.
4. After received the food recognition data, use the result food list to generate a query. Then send to Nutritionix API to get the nutrition data.
5. The server processes the data then send back to the user.

Sequence diagram of the main function (Record diet)



System components

Client-side application

- Development platform: React Native expo (SDK47, IOS13.0+/ Android 5.0+)
- Main Development environment: iPhone 11 (IOS 15.6.1)
- Supported platform: Android and IOS

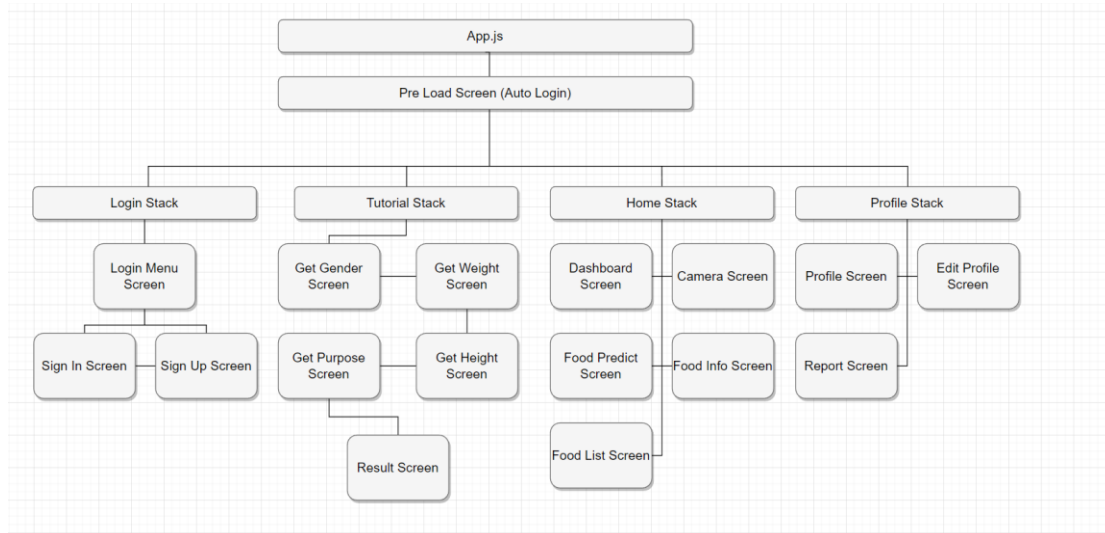


Figure 3 Mobile App architecture

- Figure 3 shows all the screen included in our application.
- App.js keep the auth context to control the which stack of screen to show
- The 4 “xxx Stack” in the third row means the group of screens, we keep it in a navigation stack to handle screen navigation.

Server

- Node.js
- Express.js (Create RESTful API)
- Mongoose.js (Manage access/operation with MongoDB Atlas)
- <https://jwt.io/> (Java web token, Route protection for security reason)
- 4 APIs for user (signup/login/edit/verify)
- 1 API for food recognition

Database

- MongoDB Atlas (Free 500MB NoSQL Cloud DB provider)

Third party API

- Clarifai (Food-Item-recognition API Provider)

<https://clarifai.com/clarifai/main/models/food-item-recognition?inputId=https%3A%2F%2Fs3.amazonaws.com%2Fsamples.clarifai.com%2Ffeatured-models%2Ffood-pepperoni-pizza.jpg>

- Nutritionix (Food nutrition data database)

<https://www.nutritionix.com/natural-demo?q=1%20cup%20mashed%20potatoes%20and%202%20tbsp%20gravy&s=1>

Functional Requirements

Mobile Application

Function	Description
Sign up	Create account
Sign in	Login account and get the “userToken” (i.e. Java web token)
Store the “userToken” in storage	Store token for accessing the server/auto login
Auto Login	If there are “userToken” in the storage, try to use this token for login
View personal profile	User can view their own profile
Edit personal profile	User can edit their own profile
Tutorial for new account	For new accounts, it will be a tutorial for them and lead them to input the basic information used to calculate the BMR.
Calculate BMR (Basal Metabolic Rate)	By using the Mifflin-St Jeor Equation (Calculator.net, n.d.) For men: $BMR = 10W + 6.25H - 5A + 5$ For women: $BMR = 10W + 6.25H - 5A - 161$ where: W is body weight in kg H is body height in cm A is age
Calculate BMI	Calculate the Body Mass Index
Take/pick picture to get food item	User can take/pick a picture and send to the server and get the food items with nutrition data
Record the food item	User can add the food item to record list according to “Date” and “Slot” “Date” means the record day, user can change to different day to record “Slot” means “Breakfast” / “lunch”, “Dinner”, “Other”
Warning if the absorbed calories is over the recommended BMR	/
Report	Simple bar chart report to show the record of calories absorbed in the last week.

RESTful Server

Route	Description
/user/signup	<pre>/* create user URL:localhost:3000/user/signup Method: POST body: { "username": "test", "password": "12345" } */</pre>
/user/login	<pre>/* login URL:localhost:3000/user/login Method: POST body: { "username": "test", "password": "12345" } */</pre>
/user/verify	<pre>/* verify user by token URL:localhost:3000/user/verify Method: POST body: { "token": "xxx" } */</pre>
/user/edit	<pre>/* edit user URL:localhost:3000/user/edit Method: POST body: { "user": "{ _id : xxxxxxxxxxxxxx username: xxx xxx: xxx ... }, "token": xxxxxxxxx } */</pre>
/foodRec	<pre>/* receive form data URL:localhost:3000/foodRec Method: POST Form-Data: KEY: "FoodName" VALUE: image.png (image file) */</pre>

Data Model

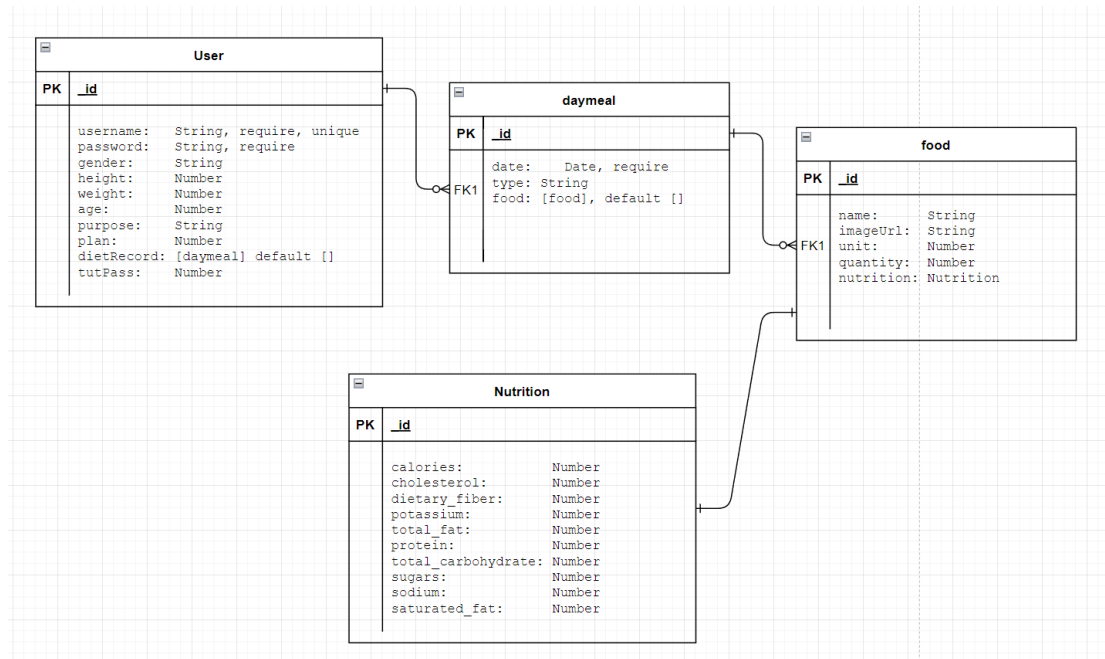


Figure 4 Data model

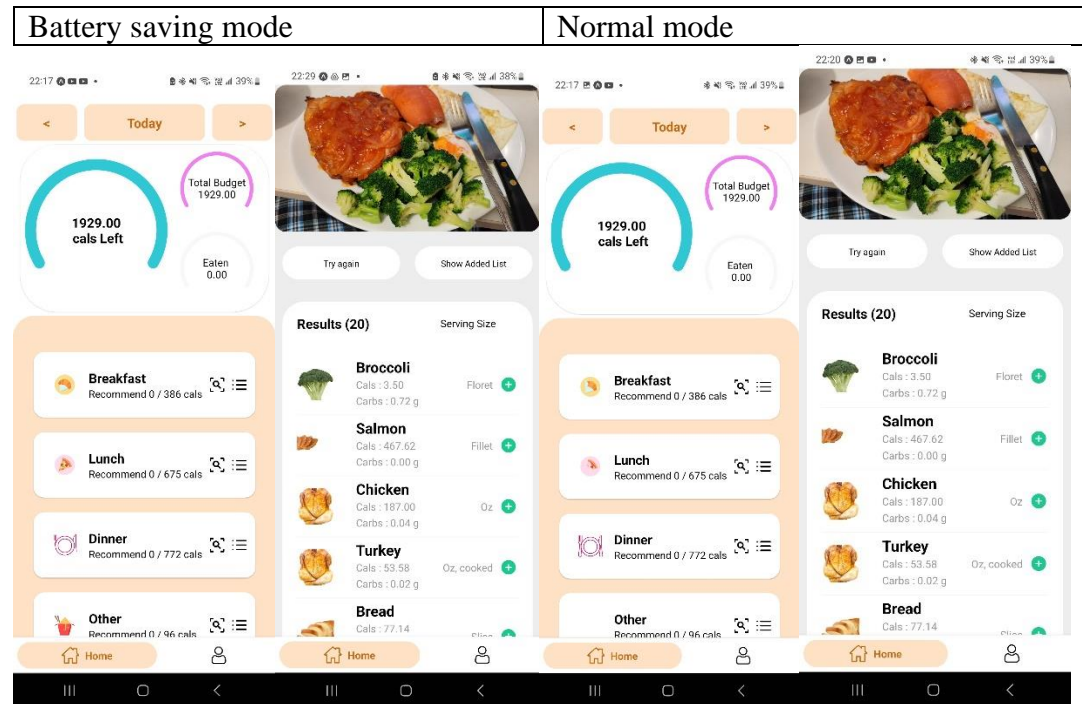
Figure 4 shows the data model we are using to store the user information in mongoDB.

Testing

We've conducted testing on two different devices, Xiaomi K20 pro and Galaxy Z Flip4.

Performance testing

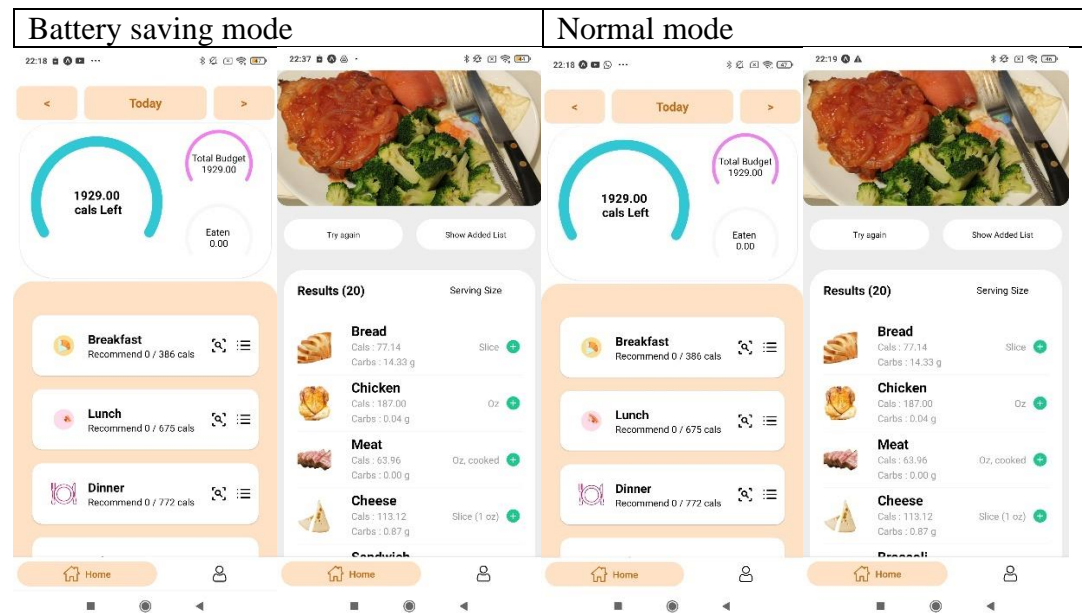
Galaxy Z Flip4:



To simulate low performance or low battery, we activated battery saving mode on the left.

The app works perfectly on both circumstances, with an expected frames dropped because of the battery saving mode.

Xiaomi K20 pro:

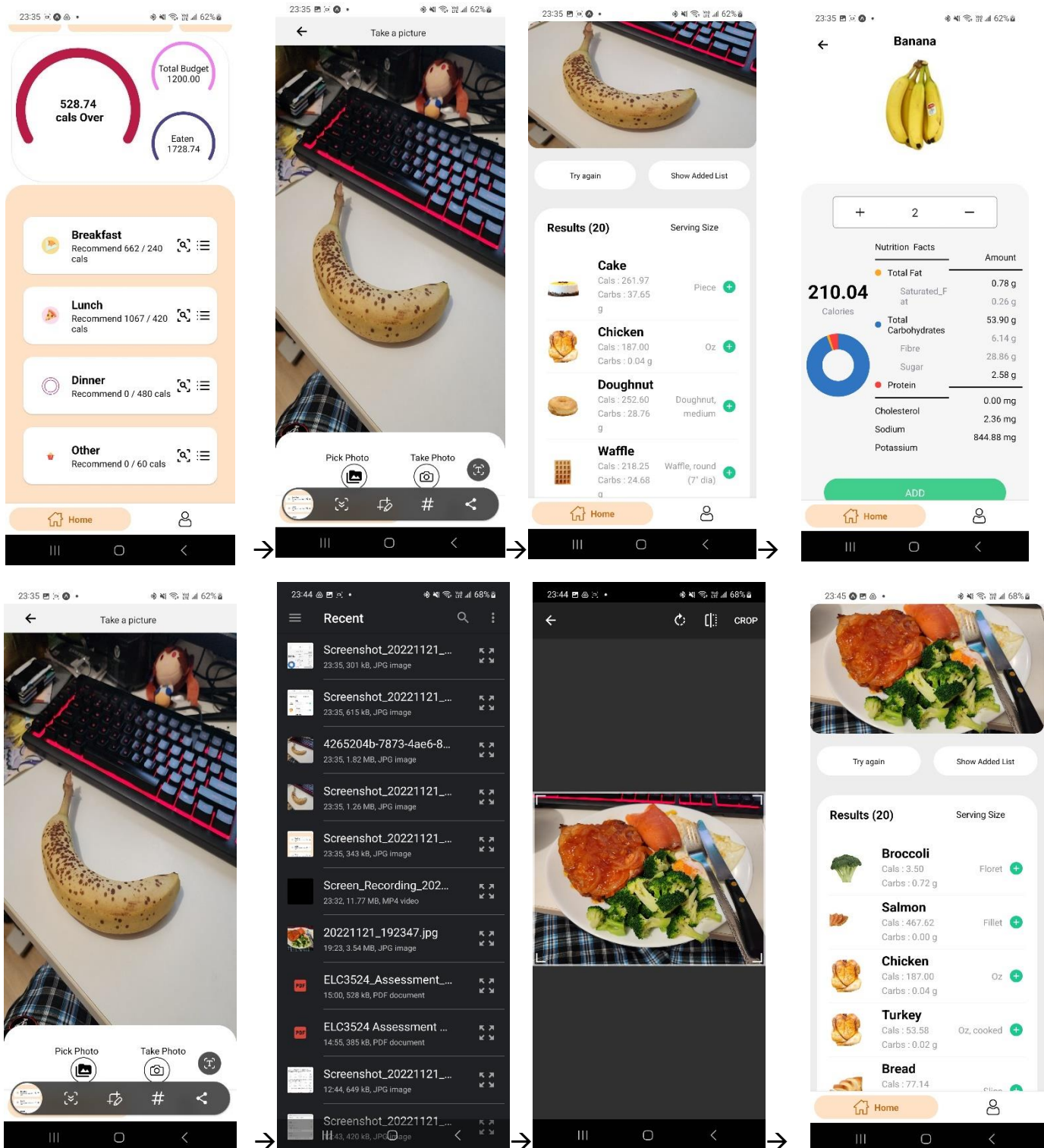


We did the same performance test as the Flip4, the results are the same, app worked perfectly fine with a few frames dropped.

Functional testing:

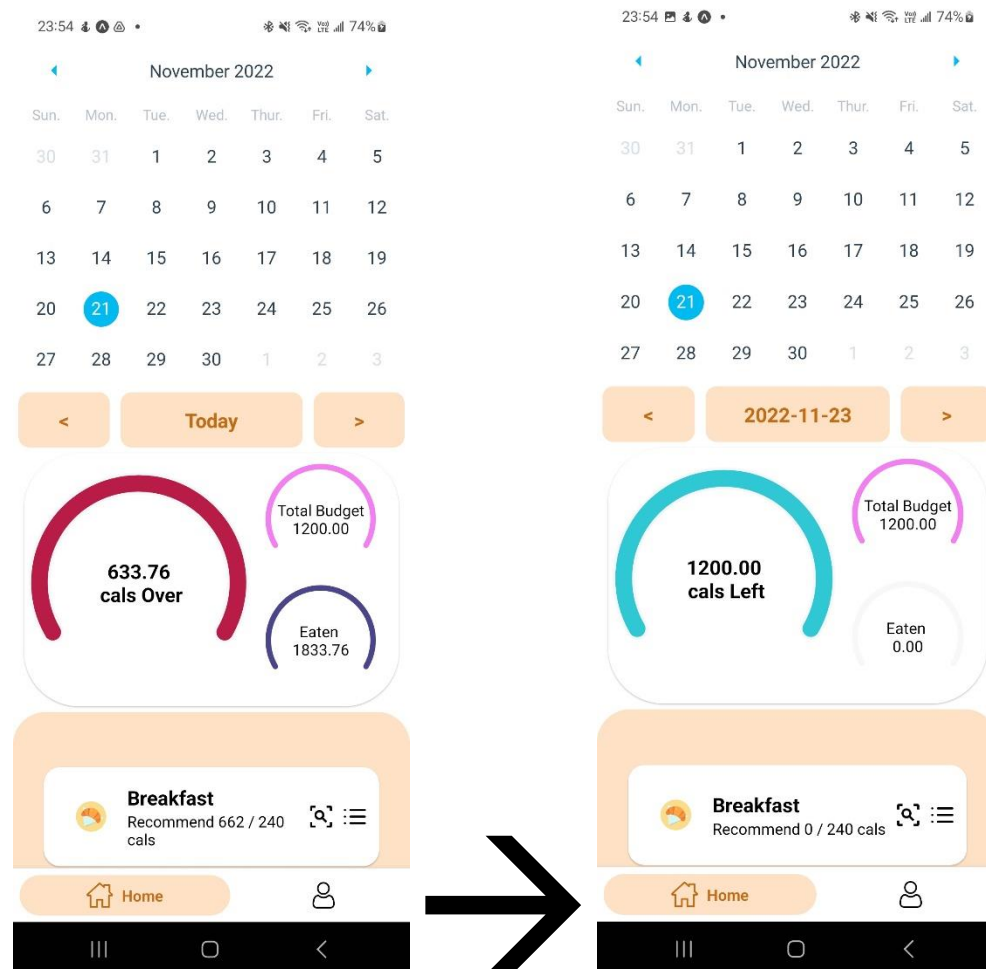
Our main functions are to capture a picture of a food and send it to the server side to perform the process of recognizing the food.

The below images show that we can perform our main function with no bug.



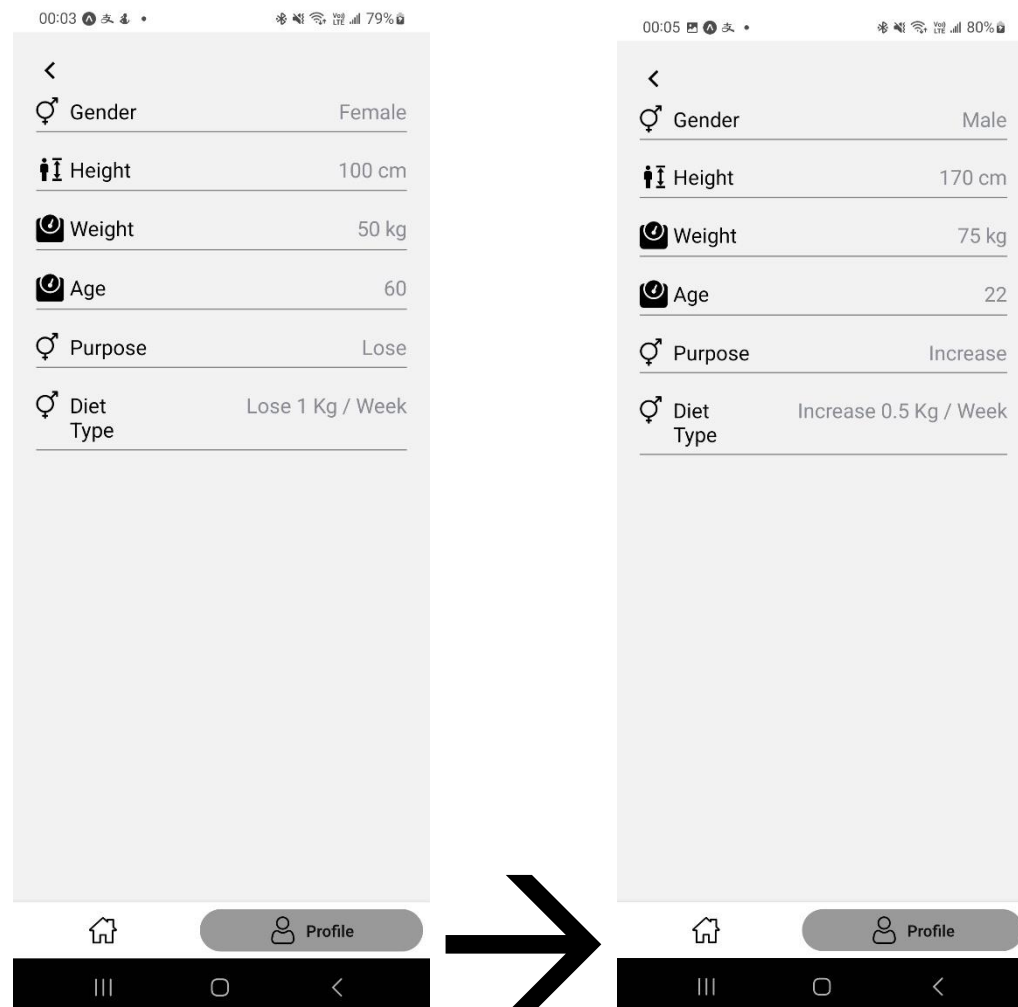
Date choosing function:

Choose the date by using the calendar, press the left arrow to go to yesterday, or press the right arrow to go to the next day.



Profile editing:

Edit the profile by typing the new value or choosing the new option.



Installation guide

Backend server setup

For the environment set up assume you installed node.js and npm. If not, you can follow the website to install them.

<https://phoenixnap.com/kb/install-node-js-npm-on-windows>

Step 0 use Terminal Command or Visual Studio Code to open “comp-4342-project” folder

Step 1 Run Terminal script to visit “server” folder path:

cd Server

```
PS C:\Users\kwong\Documents\GitHub\comp-4342-project> cd Server
PS C:\Users\kwong\Documents\GitHub\comp-4342-project\Server> |
```

Step 2

Install the package:

npm install

```
PS C:\Users\kwong\Documents\GitHub\comp-4342-project\Server> npm install

npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.
changed 1 package, and audited 314 packages in 682ms

22 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Step 3

Run Terminal script to run the server:

npm start

```
PS C:\Users\kwong\Documents\GitHub\comp-4342-project\Server> npm start
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.

> comp4342_project@1.0.0 start
> nodemon server.js

[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Listen: localhost:3000
Server is connected
```

Application server setup

Step 0: Install React Native Expo CLI by the following command:

```
npm install --global expo-cli
```

Step 1 Run Terminal script to visit “Application” folder path:

```
cd Application
```

```
PS C:\Users\kwong\Documents\GitHub\comp-4342-project> cd Application
PS C:\Users\kwong\Documents\GitHub\comp-4342-project\Application> |
```

Step 2: Install yarn

```
npm install --global yarn
```

Step 3 Install the package:

```
yarn install
```

Step 3 Find your IPv4 address in Terminal Command or Visual Studio Code:

Ipconfig

```
PS C:\Users\kwong\Documents\GitHub\comp-4342-project\Application> ipconfig

Windows IP 設定

乙太網路卡 乙太網路:

    連線特定 DNS 尾碼 . . . . . : mynet
    連結-本機 IPv6 位址 . . . . . : fe80::e77c:5c8f:e7ea:7570%11
    IPv4 位址 . . . . . : 192.168.128.55
    子網路遮罩 . . . . . : 255.255.255.0
    預設閘道 . . . . . : 192.168.128.1

乙太網路卡 VirtualBox Host-Only Network:

    連線特定 DNS 尾碼 . . . . . :
    連結-本機 IPv6 位址 . . . . . : fe80::71ed:dd9f:3b8e:d4bc%8
    IPv4 位址 . . . . . : 192.168.56.1
    子網路遮罩 . . . . . : 255.255.255.0
    預設閘道 . . . . . :

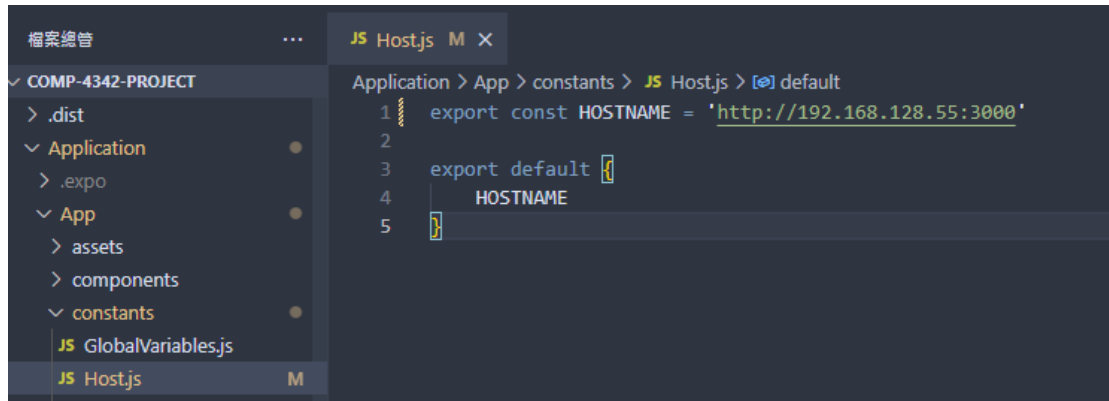
乙太網路卡 區域連線* 9:

    媒體狀態 . . . . . : 媒體已中斷連線
    連線特定 DNS 尾碼 . . . . . :
```


Step 4 set you host address:

Open the “Host.js” to change the “HOSTNAME” to you IPv4 address, like 'http://xxx.xxx.xxx.xx:3000', you should type your IPv4 address found in Step 3, and save the file.

Application/App/constants/Host.js



```
Application > App > constants > JS Host.js > [?] default
1  export const HOSTNAME = 'http://192.168.128.55:3000'
2
3  export default HOSTNAME
4
5
```

Step 5

Run Terminal script to run the server:

npm start



```
PS C:\Users\kwong\Documents\GitHub\comp-4342-project\Application> npm start
npm WARN config global '--global', '--local' are deprecated. Use '--location=global' instead.

> application@1.0.0 start
> expo start

Starting project at C:\Users\kwong\Documents\GitHub\comp-4342-project\Application
Some dependencies are incompatible with the installed expo version:
  expo-image-picker@14.0.1 - expected version: ~14.0.2
Your project may not work correctly until you install the correct versions of the packages.
Install individual packages by running npx expo install expo-image-picker@~14.0.2
Starting Metro Bundler



> Metro waiting on exp://192.168.128.55:19000
> Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

> Press a | open Android
> Press w | open web

> Press j | open debugger
> Press r | reload app
> Press m | toggle menu

> Press ? | show all commands

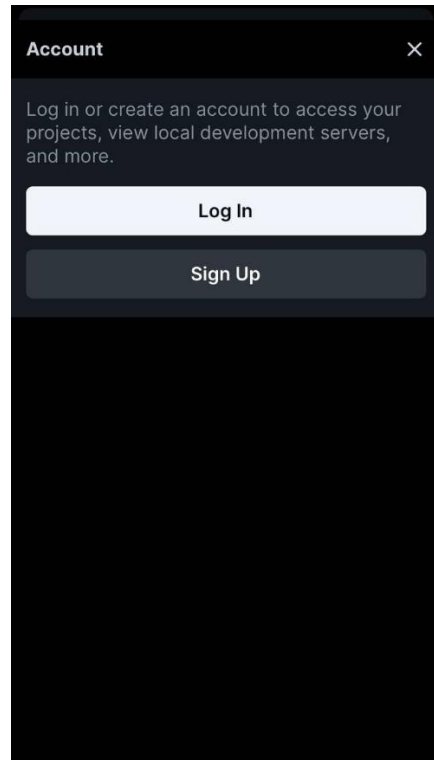
Logs for your project will appear below. Press Ctrl+C to exit.
```

Run the Application with mobile

Assume you run the application server successfully in the previous part

Step 0 Install the Expo Go from the App store <https://expo.io/tools>

Step 1 After you install “Expo Go”, you should Sign up a account and Log In



Step 2 Then use your mobile phone to scan the QR code generated when you run the application server to open the application on the real device

(Make sure both device and computer are under the same network)

(Some time the host in Host.js is:

192.168.x.x or

10.0.0.2, or

Your public IP or

Domain

Etc.

It really depends on your networking setup)

User Guide

Mobile Applications User guide

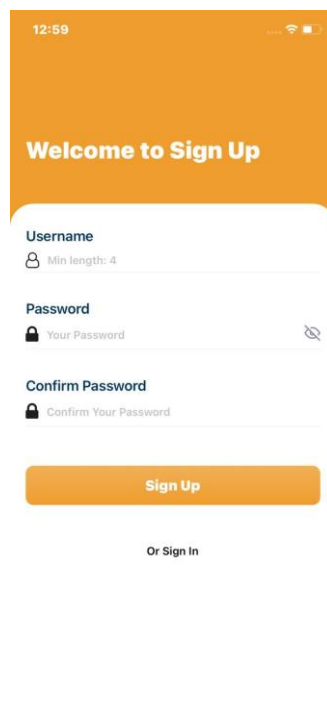
1 Create user account or login account



Press "I am New" button to create new account, go to Step 1.1


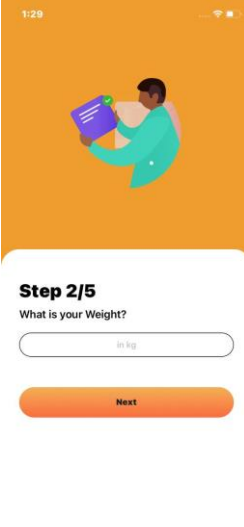
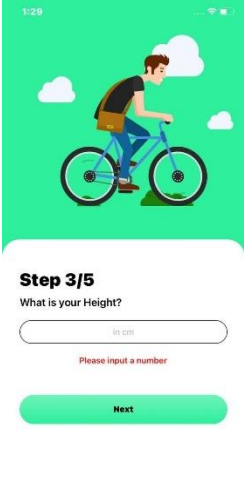
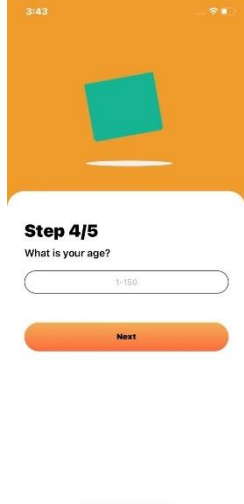
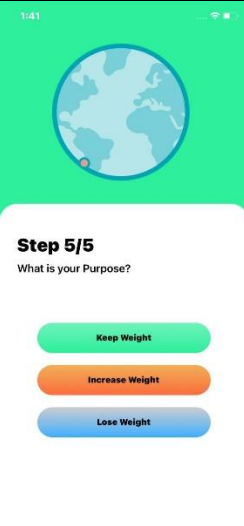

Press "I have an account already" button to login Account, go to Step 1.2

Step 1.1 Account Register

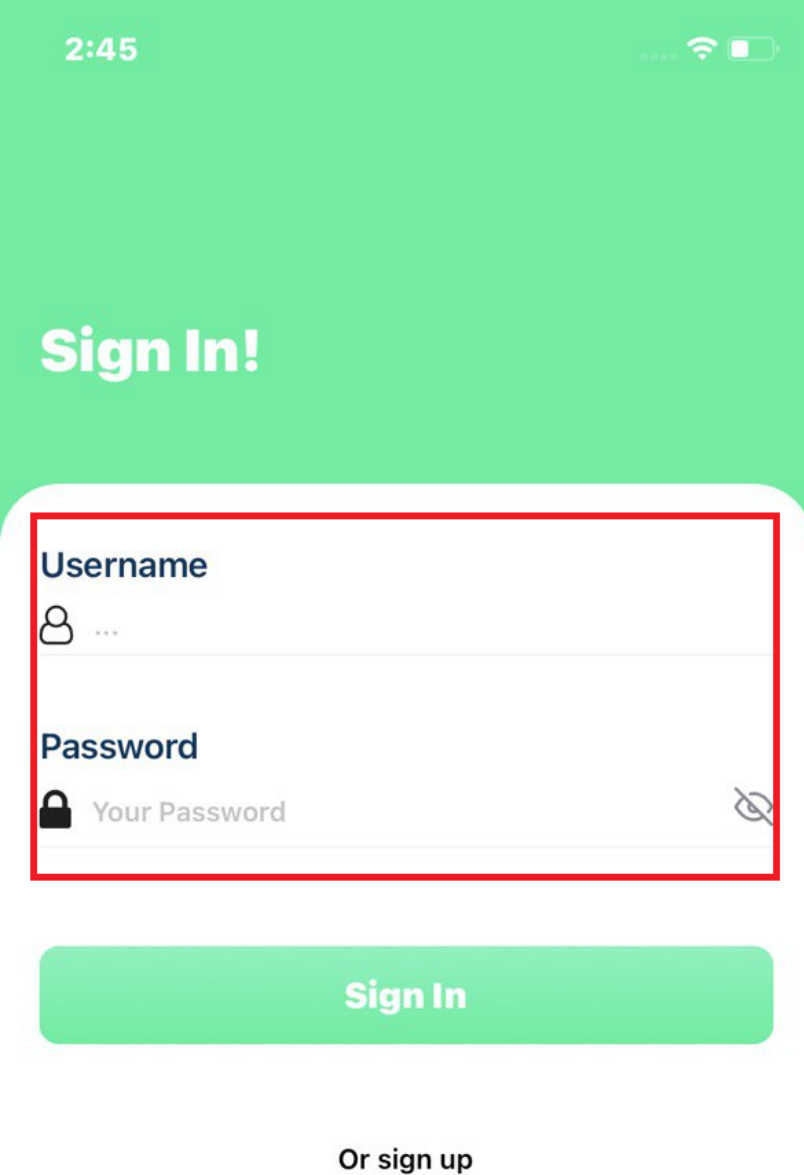


Fill in username, password, then press "Sign Up" button to sign up account.

1.1.1 Account Register

<p>1. Select The Gender (Male or Female):</p> 	<p>2. Enter user Weight in kg:</p> 
<p>3. Enter the user height in cm:</p> 	<p>4. Select user's birthday:</p> 
<p>6. Select user purpose:</p> 	<p>7. Click "Finish" to create account go to step 2</p> 

1.2 Login Account




A mobile application login screen with a green header and a white login card. The header displays the time 2:45 and status icons. The card contains a 'Sign In!' title, a 'Username' field with a person icon, a 'Password' field with a lock icon and a toggle for visibility, a green 'Sign In' button, and a link to 'Or sign up'.



2:45

Sign In!

Username

 ...

Password

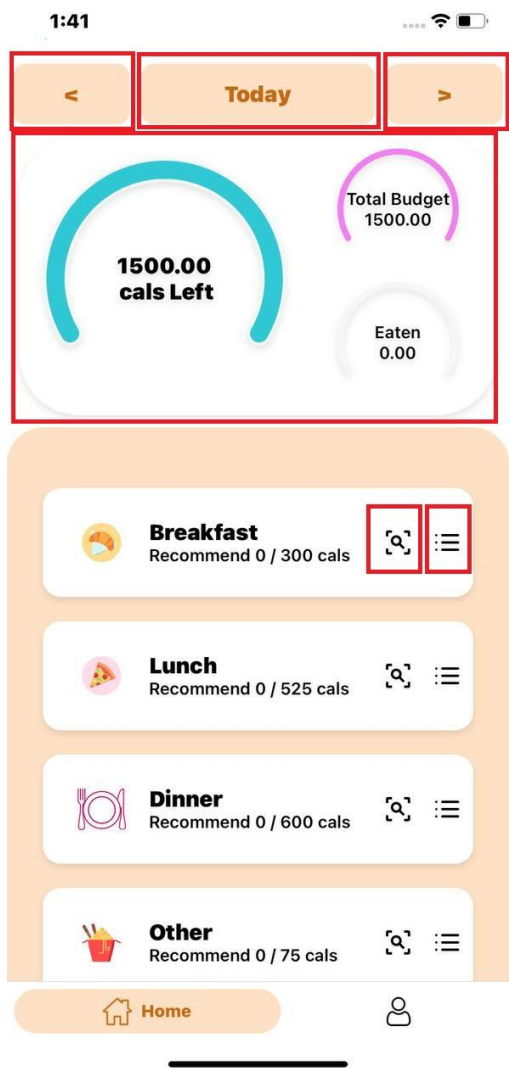
 Your Password 

Sign In

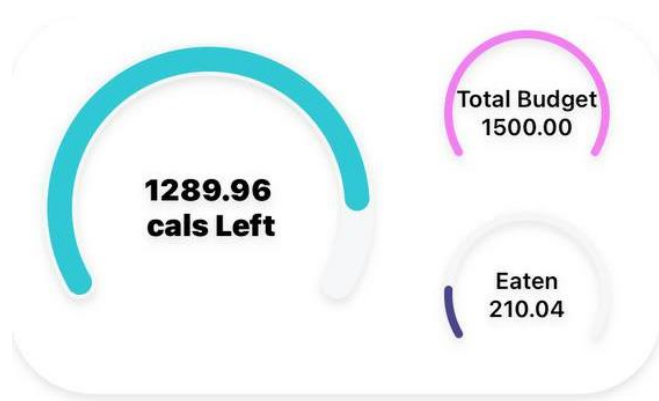
[Or sign up](#)

User need to fill in “Username”, “Password” and press “Sign In” button to sign in if the username and password is correct. Go to step 2

2 Main Page

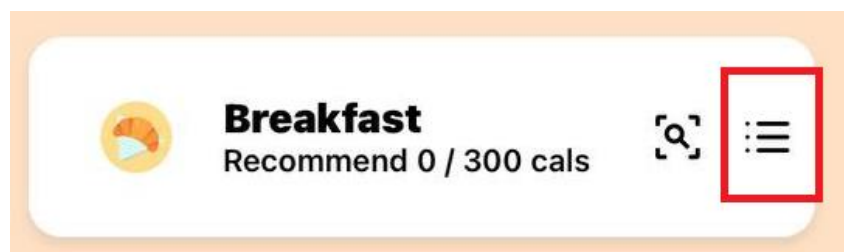



Previous day button (user click pervious button the selected day will display last day)	Selected day button (user click the selected day button will show calendar view for user select a day)	Next day button (user click next day button the selected day will display next day)

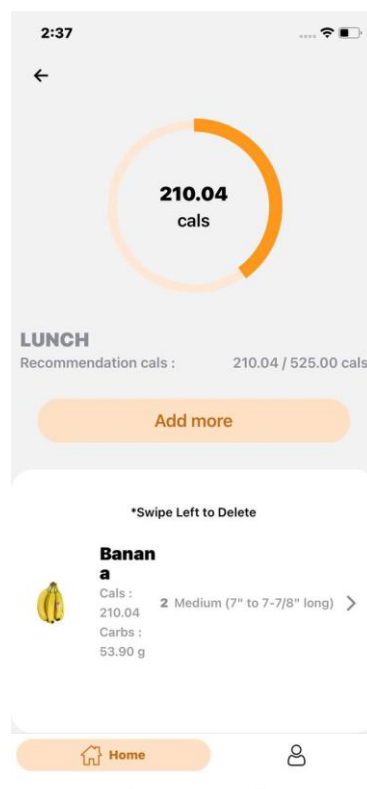


Show how many calories can user eat today, Total Budget of calories of a day and how many calories user eaten.

2.1 Food Check List

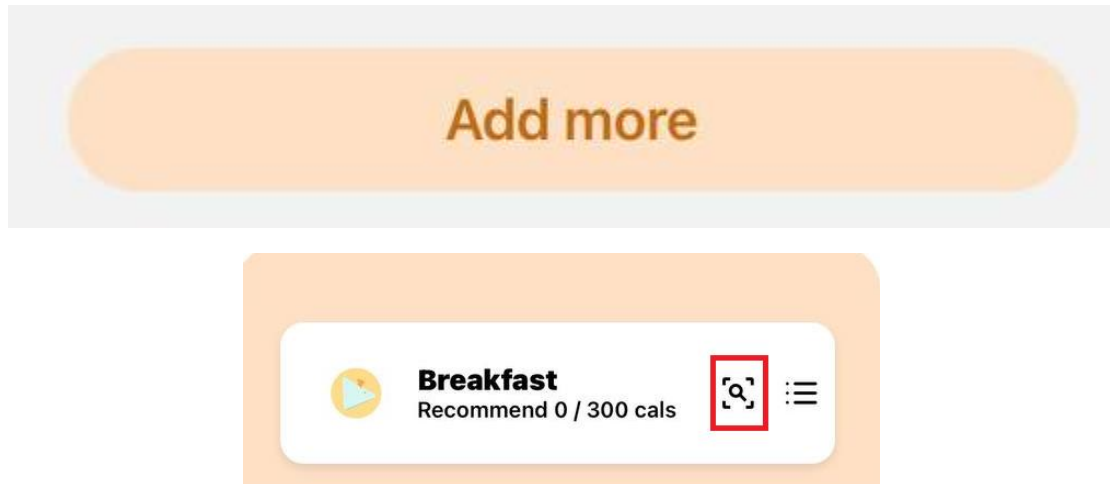


If the click  this button in “Breakfast”, Users can see the calculated calories and add more food button for this meal on this page

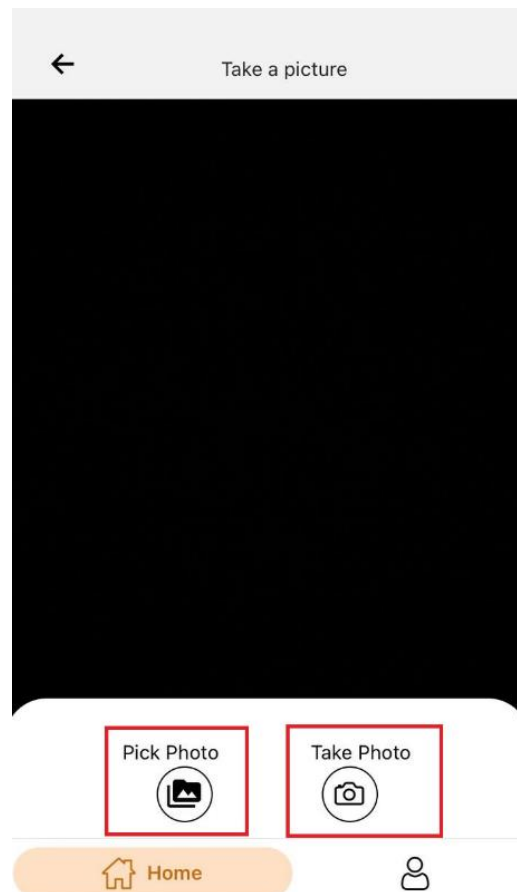


2.2 Scan food

User can click these two buttons to add more food for this meal



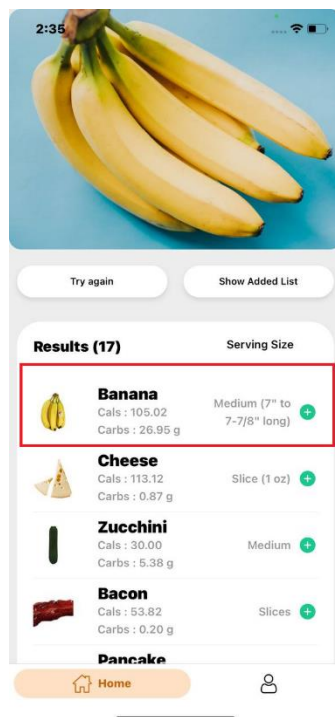
Users can choose two ways to add food pictures



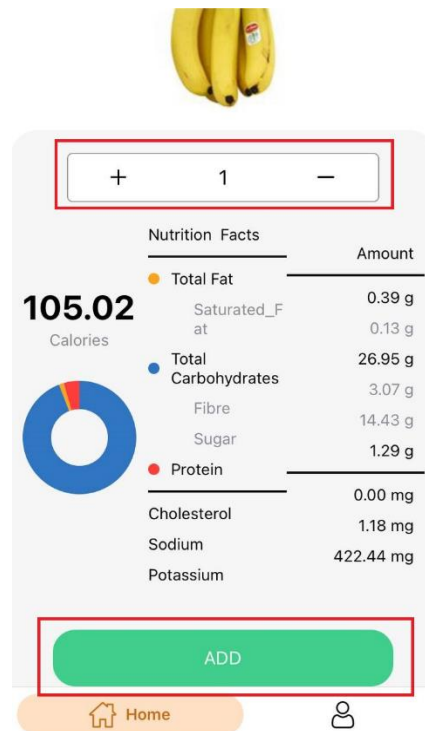
First, users can select pictures of related foods in the gallery for artificial intelligence recognition

The second type of user can use the built-in camera function of the mobile phone to take pictures of real food and manually identify food photos

2.3 Identify the food



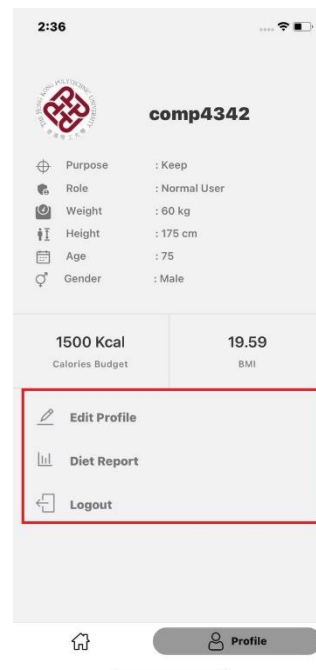
User can select the food that is in the picture



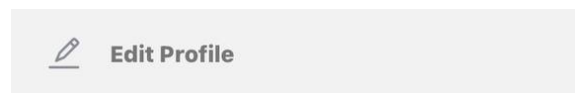
User can click “+” or “—” to increase or decrease the quantity of food. Then the Nutrition Facts will be calculated. Lastly click the "ADD" button to add it to the food list of the menu

3 Profile Page

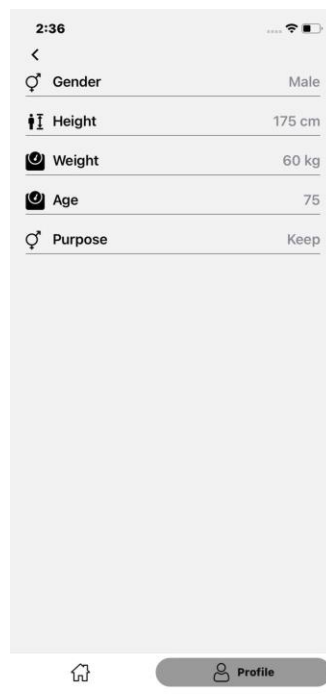
In the user profile, the user's basic information, user avatar, name, own calorie budget and body BMI will be displayed



List of functions available in the user profile:

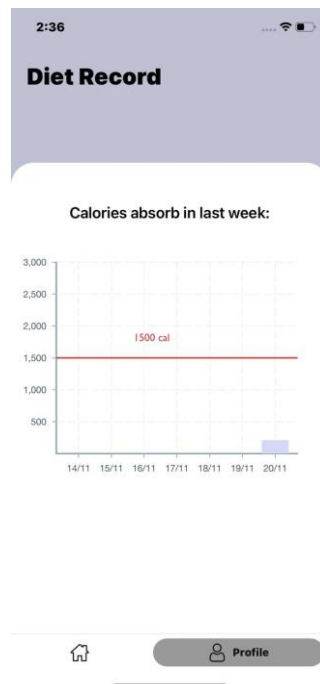



When user click this button, the edit user page will be reached. User can upload user icon and edit all information



Diet Report

If user click “diet plan” button, it will show the report of the diet plan.



 Logout

Where user “Logout” button, this user account will be log out

Peer Review

Cheung Sui Wing

- React Native App development (UI Screen/Logic in IOS)

Lau Man Chun

- Server (MongoDB, user route, signin/signup/verify/edit/Java web token)

Kwong Chun Him

- Food AI + nutritionix API function (combined them and process the return data)

Cheng Chi Kit

- Testing
- Bug fix in Android

Group Members (Name)	Contribution to the project (Total 100%)
Lau Man Chun	25%
Cheung Sui Wing	25%
Cheng Chi Kit	25%
Kwong Chun Him	25%