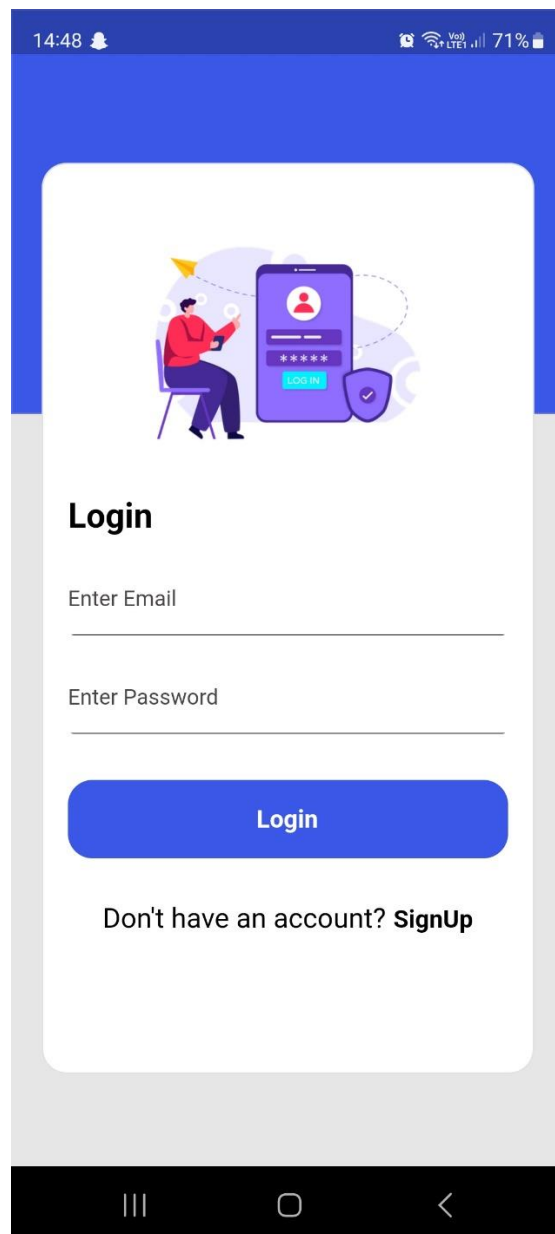


# **Barcode Scanner Documentation**

## Chapter 9: Implementation

### 9.1 Screenshots

#### 9.1.1 Screenshots for Login



*Figure 9.1.1.1 Screenshot for login*

This is the first page that every user goes through, whether its admin or normal user. This page is used for logging in to the system with the registered email and password. There is

also a button for logging in below the text field. User can also click on the bold 'Sign Up', text to register a new account.

### 9.1.2 Screenshot for Register

14:48

←

**Let's Get Started!**

Create an account and start creating.

Full Name

Male

Age

Username Address

Password

Confirm Password

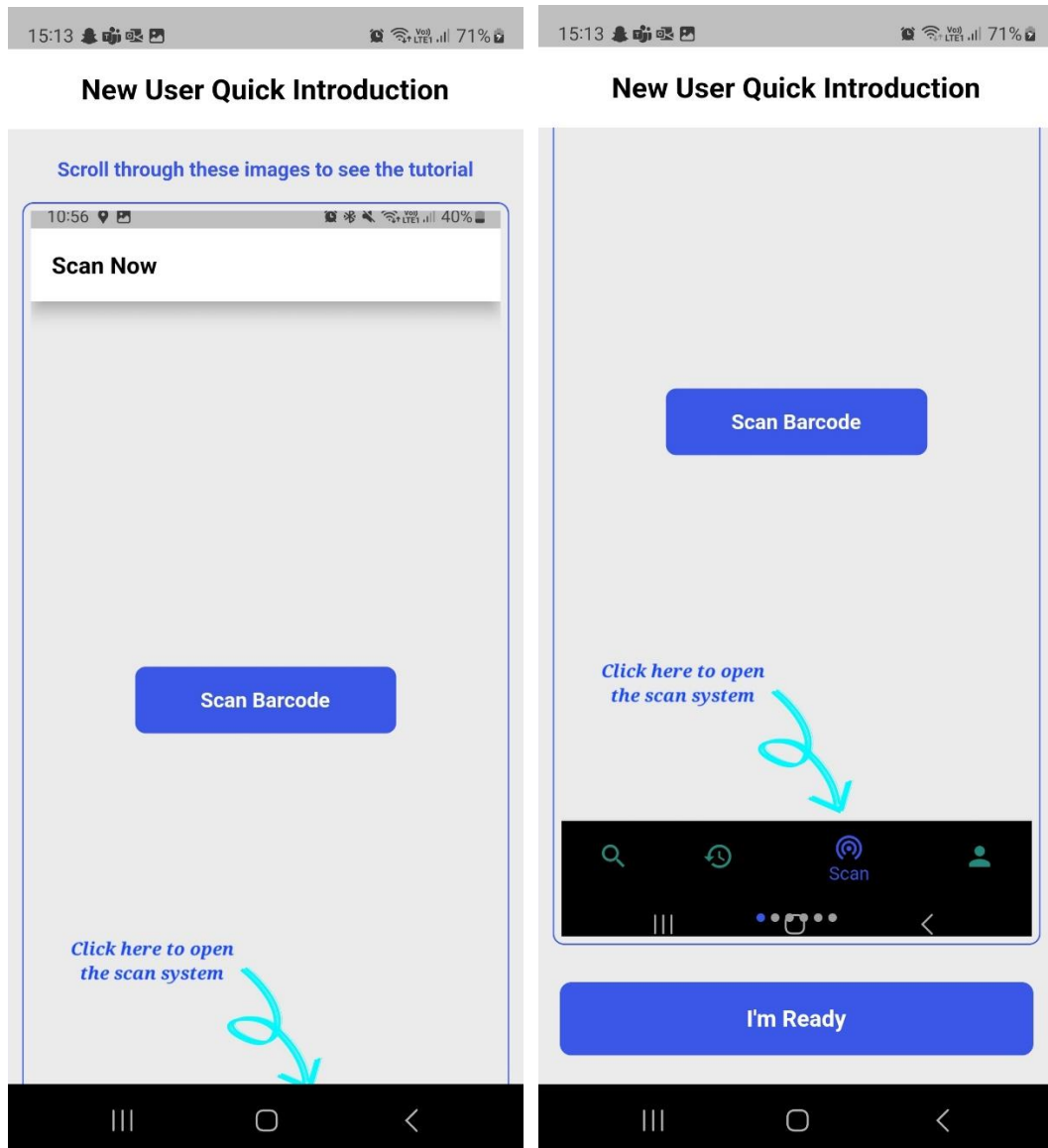
**Sign Up**

By creating an account or continuing to use My Scanner, you acknowledge and agree that you have accepted the Terms of Services and have reviewed the Privacy Policy.

*Figure 9.1.2.1 Screenshot for register*

This page is used for users who do not have any credentials to log in to the system. Users will need to key in some data, such as full name, gender, their current age, chosen username, and password. Then the user just needs to click the button, and if there is an empty field, the system will just show a warning popup message.

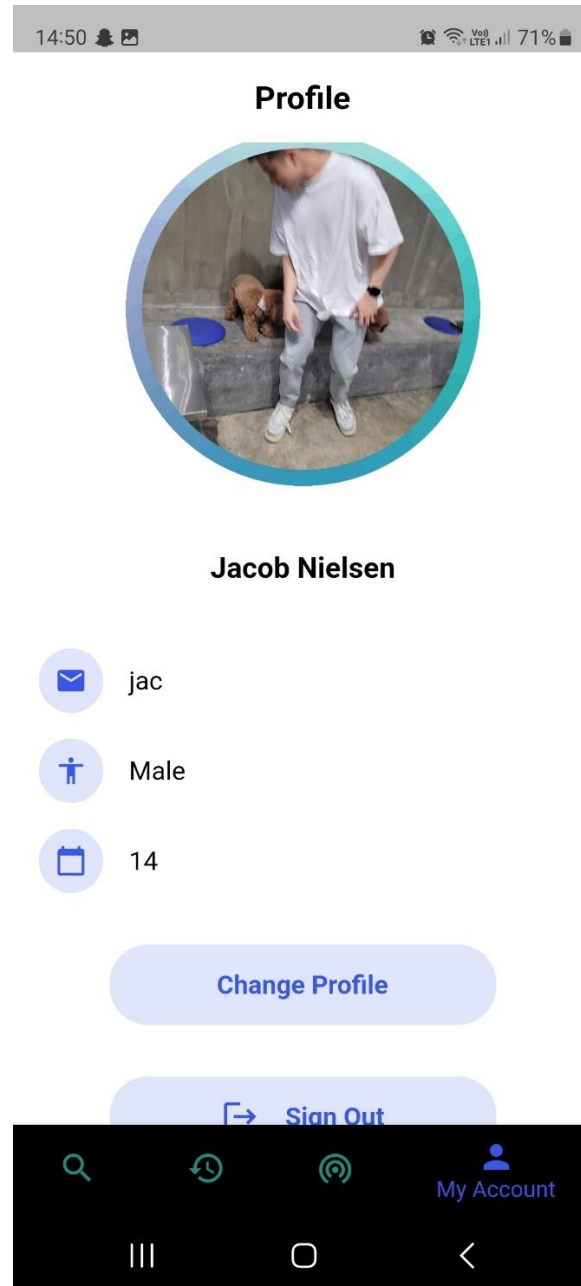
### 9.1.3 Screenshot for User Tutorial



*Figure 9.1.3.1 Screenshot for user tutorial*

This is a page that is rendered for first time users only. The tutorial page contains several pictures that user can scroll through left and right to see how the system works. Once the user clicks 'I am Ready' the page will be closed and direct user to the main user page.

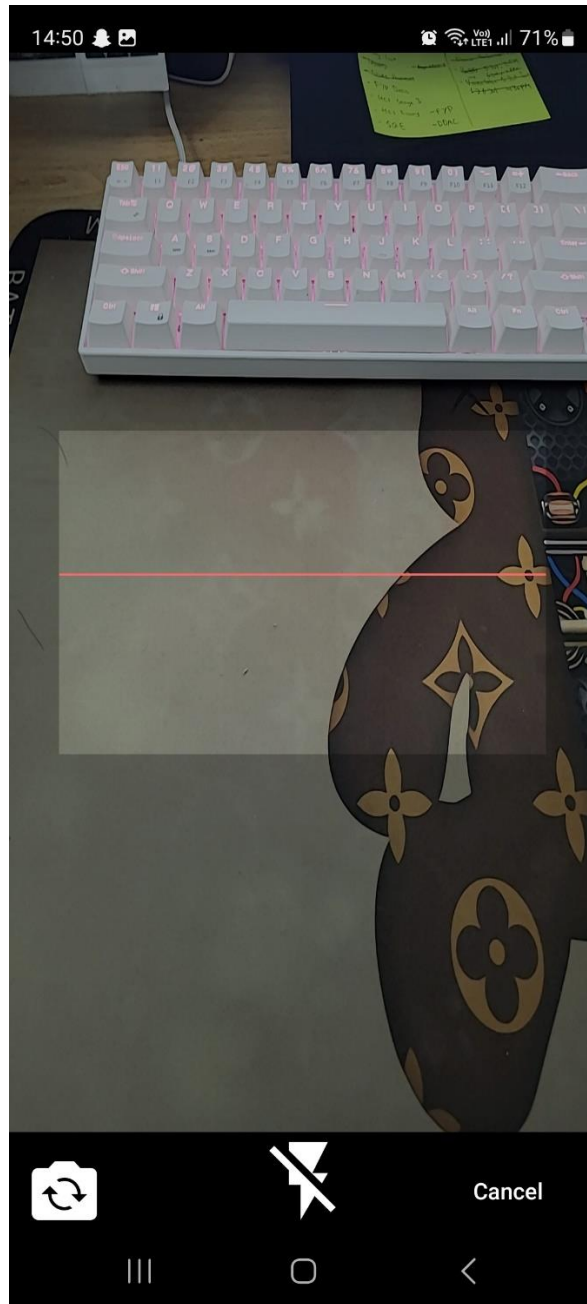
### 9.1.4 Screenshot for User Profile



*Figure 9.1.4.1 Screenshot for user profile*

This page is for user to see their profile data. Here, they can also change their profile picture if they want to, and there is also a button for signing out of the system, which will direct back to the login page.

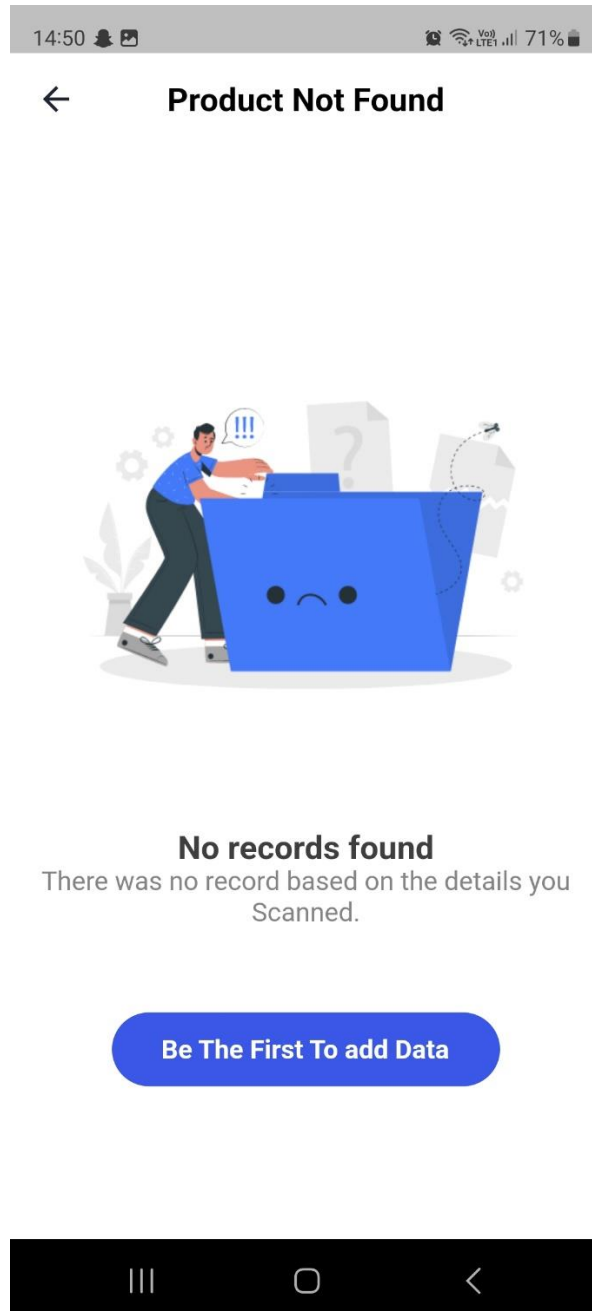
### 9.1.5 Screenshot for User Scan Barcode



*Figure 9.1.5.1 Screenshot for scan barcode*

This page is one of the main functions for user. It is the scan page, which will automatically open up the phone's camera to scan a barcode. There are some buttons below it, such as a button to open up a front camera, a button to turn on flashlight, and a button to cancel scanning.

### 9.1.6 Screenshot for Scan Result: Not Found



*Figure 9.1.6.1 Screenshot for scan result 1*

In case that the barcode scanned is not found on the database, this page will be rendered on the user's screen. There is a button that let user key in the data they know about the food they just scanned. If the user does not want to do anything, then there is a back button on top left of the screen to go back to the scan page.

### 9.1.7 Screenshot for User Add Database

**Let's Add Your Food!**

Keep in mind that all data should cover the whole pack not serving size or your data might be rejected.

**Choose Product Photo**

Chosen File Name: url

Product Name

Details

Net Weight

Calorie

Fat

Protein

Carbohydrate

Sugar

Sodium

Halal

☒ true

Vegan

☒ true

Gluten Free

☒ true

Ingredients

**Add For Review**

*Figure 9.1.7.1 Screenshot for user add database*

This page will be shown to user if they choose to add a data for the product they scanned. User will need to enter all of the details that is shown above, and also choose the product photo.



### 9.1.8 Screenshot for Result Found

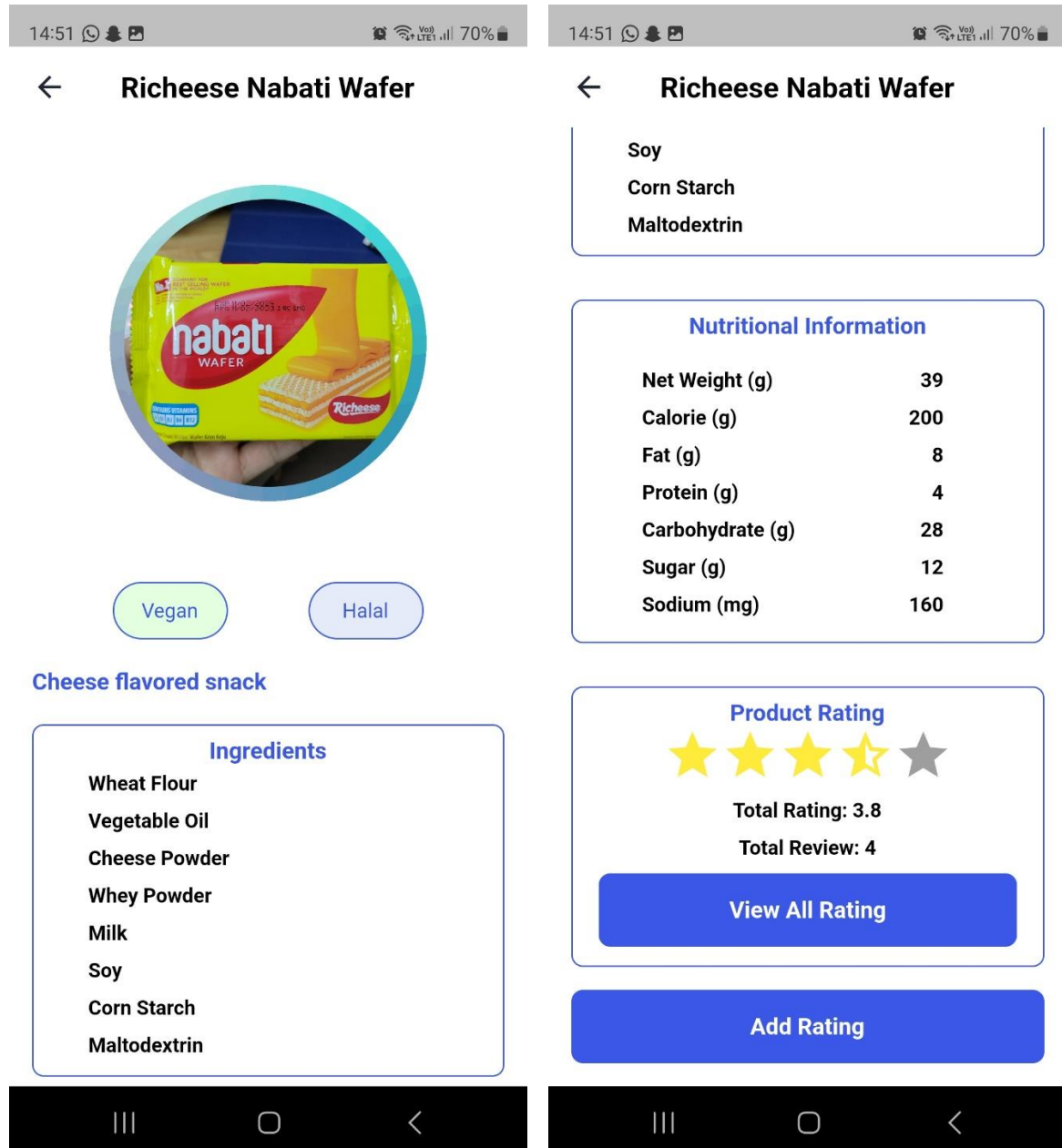
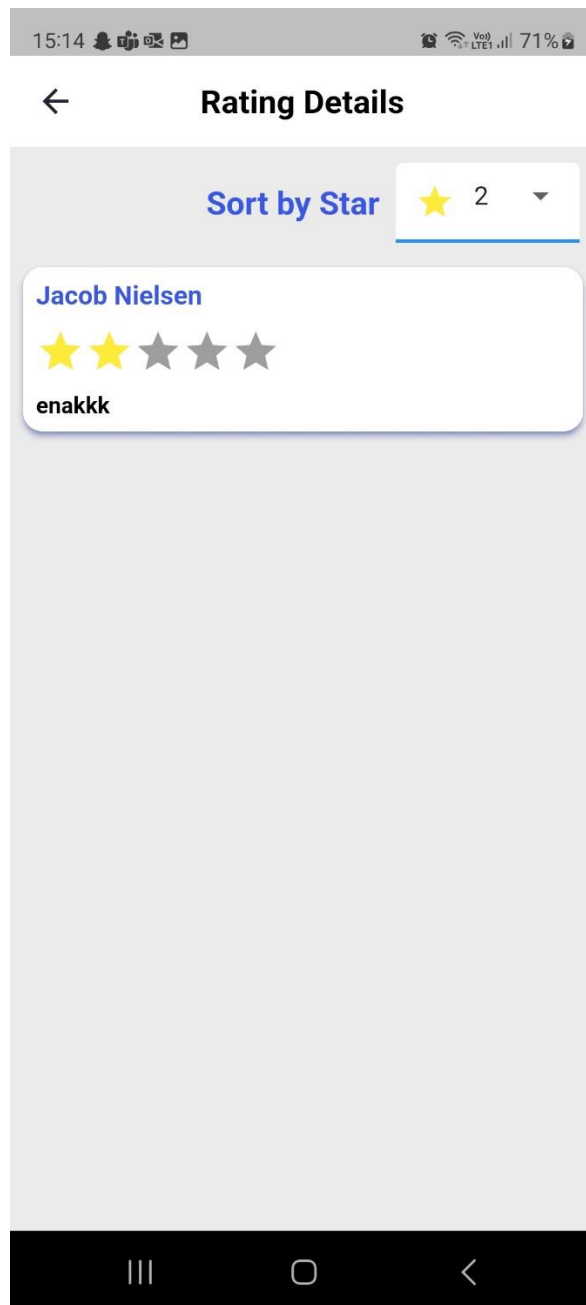


Figure 9.1.8.1 Screenshot for scan result 2

In case that the barcode scanned is found in the database, the system will show all data, including tags, ingredients, nutritional information, and rating.

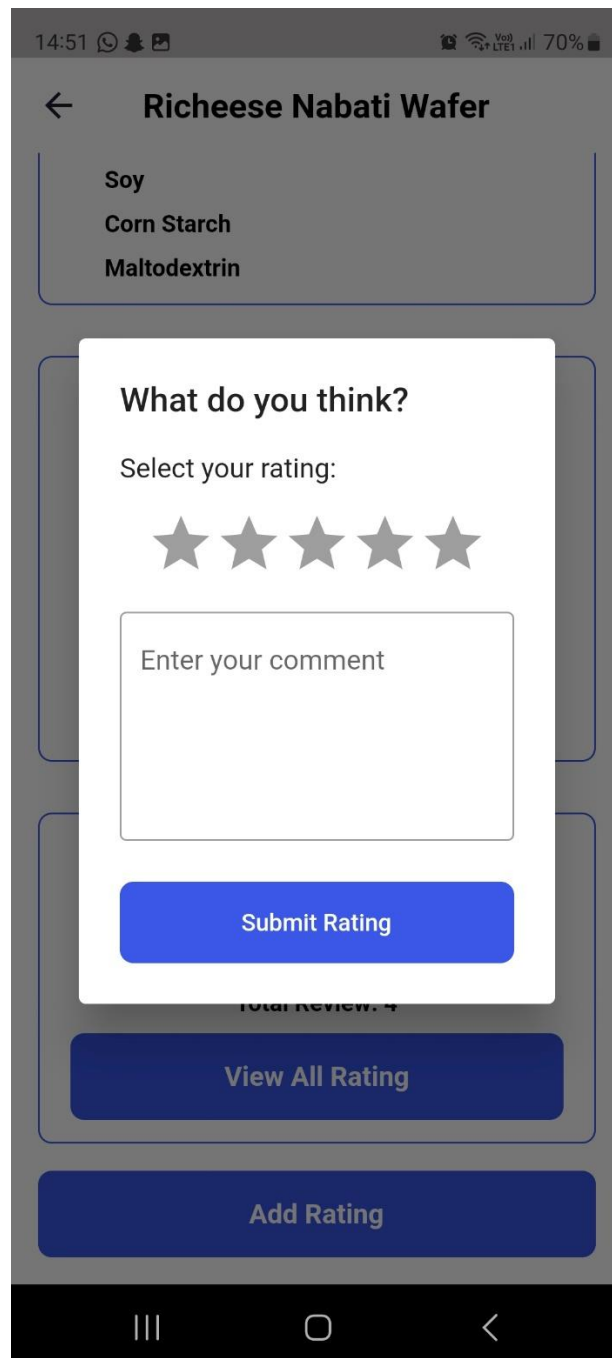
### 9.1.9 Screenshot for User View Rating



*Figure 9.1.9.1 Screenshot for view rating*

If user clicked the view all rating on the previous screen, it would show this page, which user can see all rating sorted by the star.

### 9.1.10 Screenshot for User Add Rating



*Figure 9.1.10.1 Screenshot for add rating*

If user clicked the view all rating on the previous 2 screen, it would show this popup, where user can key in their own rating towards the food they just scanned.

### 9.1.11 Screenshot for User Scan History

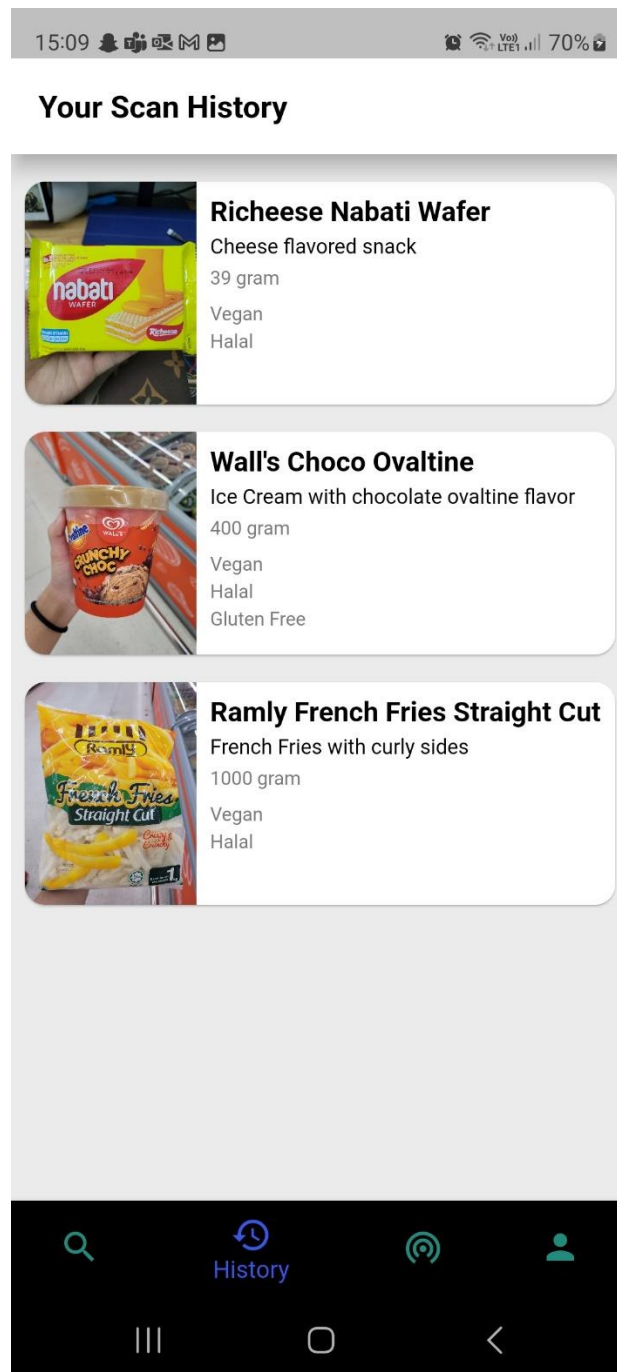


Figure 9.1.11.1 Screenshot for scan history

This page shows everything that user has scanned previously. The history page will save the data, so that user can go back to this page to see the data of a food without scanning it twice.

### 9.1.12 Screenshot for User Search Database

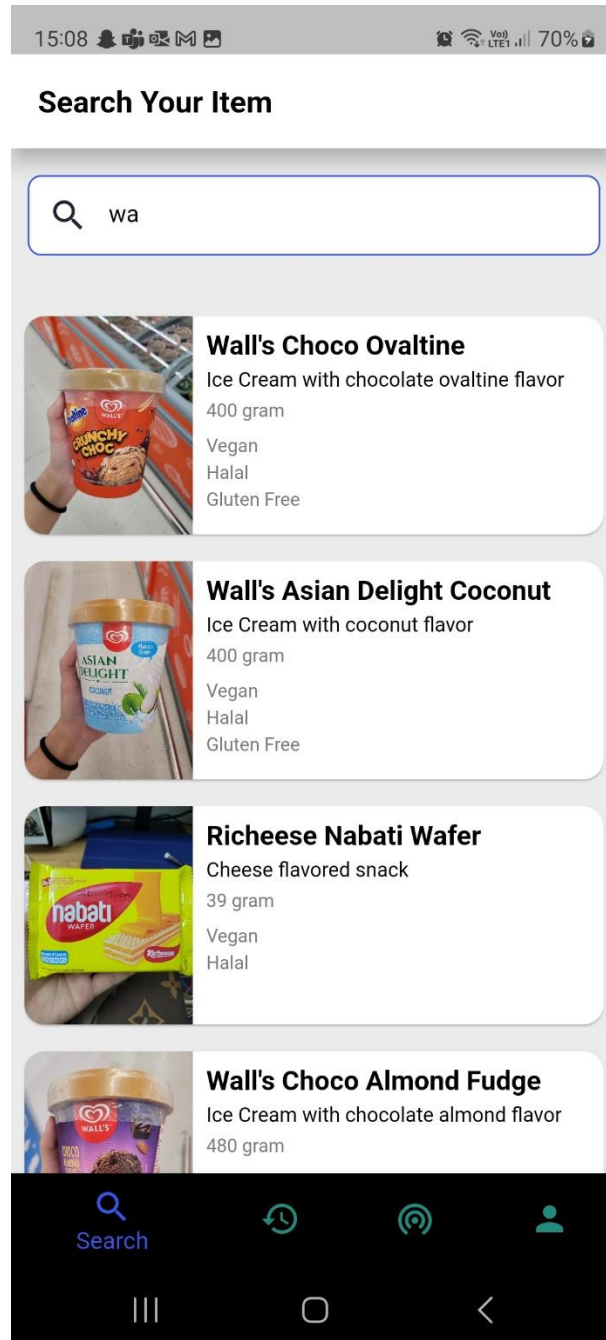


Figure 9.1.12.1 Screenshot for search database

This page is for user to search a data rather than scanning the barcode. This page is useful if user wants to find a data about a certain food, but currently does not possess the barcode of the food

### 9.1.13 Screenshot for Admin Database

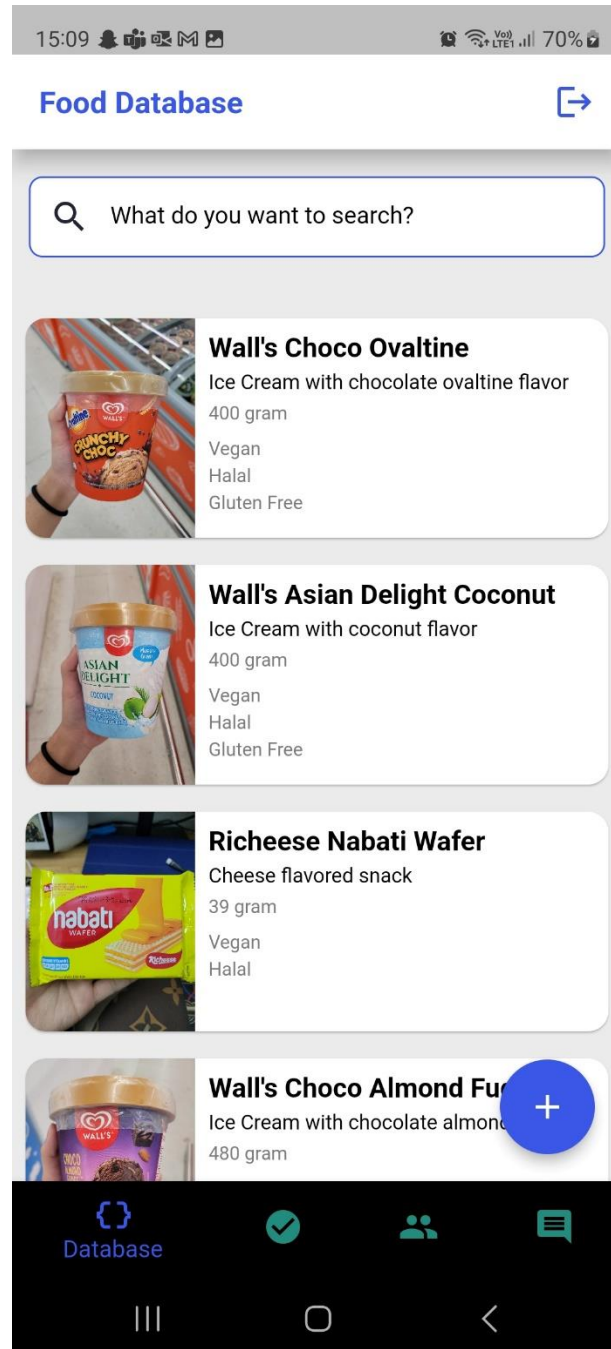


Figure 9.1.13.1 Screenshot for admin database

This page is the first page shown when an administrator credential is used. This page is similar to user search data page, but there is a floating button which is used for admin to add a new data to the database.

### 9.1.14 Screenshot for Admin Add Database

**Let's Add Your Food!**

Keep in mind that all data should cover the whole pack not serving size or your data might be rejected.

**Choose Product Photo**

Chosen File Name: url

**Product Name**

**Details**

**Barcode**

**Net Weight**

**Calorie**

**Carbohydrate**

**Sugar**

**Sodium**

**Halal**

**Vegan**

**Gluten Free**

**Ingredients**

**Add Directly to Database**

Figure 9.1.14.1 Screenshot for admin add database

When the admin clicks the floating button on the previous page, it will show this page. The difference with user is that here admin need to manually add the barcode number, and the data inserted by admin will actually be inserted to the main database.

### 9.1.15 Screenshot for Admin Database Details

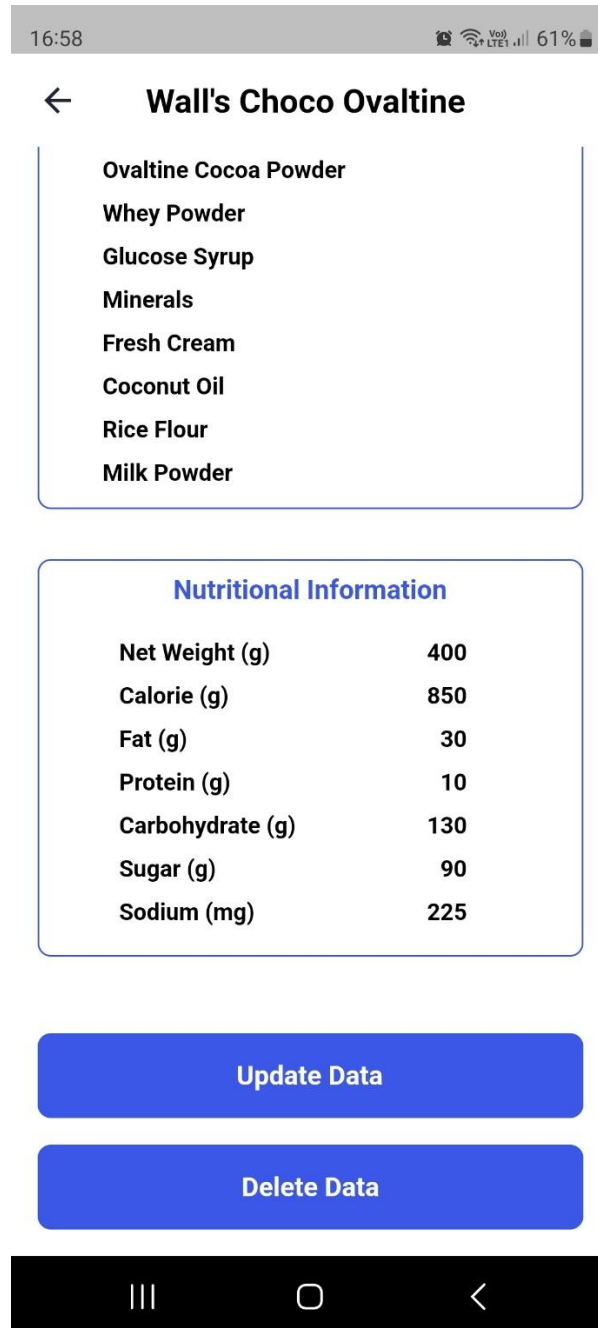
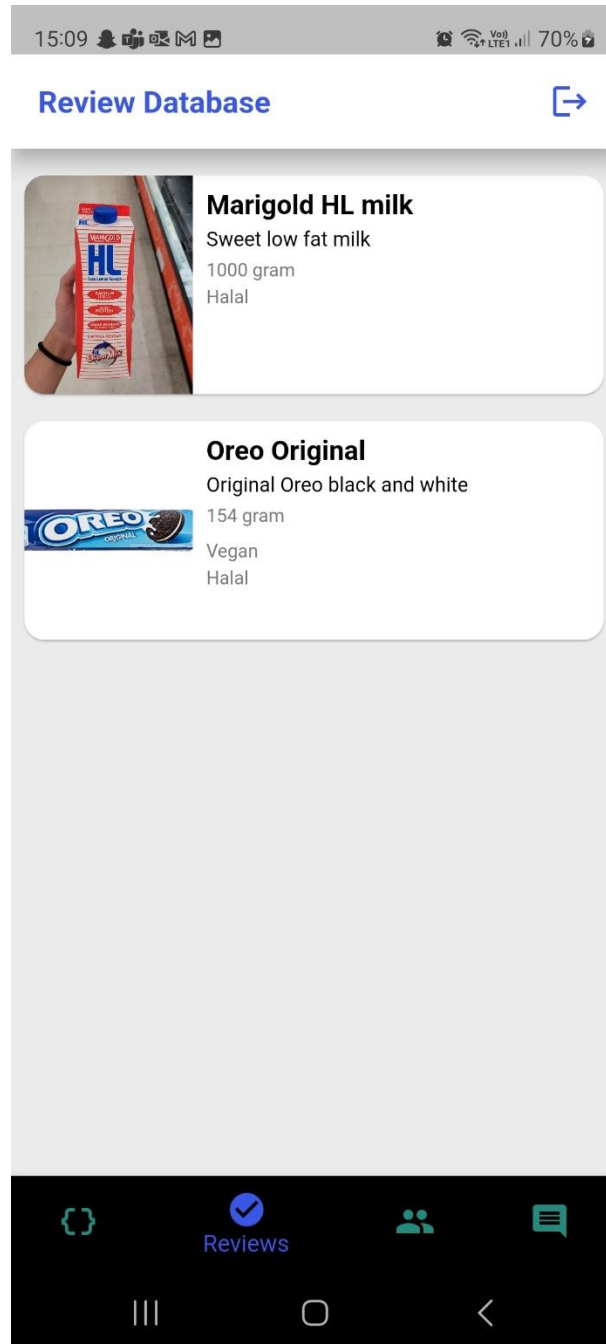


Figure 9.1.15.1 Screenshot for admin see data details

Just like user, admin can also view the food details, but admin also has the power to delete the data or update the data by clicking the buttons shown in the picture.



### 9.1.16 Screenshot for Admin Review Database



*Figure 9.1.16.1 Screenshot for admin review database*

This page is for admin to see all the data that has been submitted by the user. By clicking the data, admin will see the details that has been submitted by user.

### 9.1.17 Screenshot for Admin Review Details

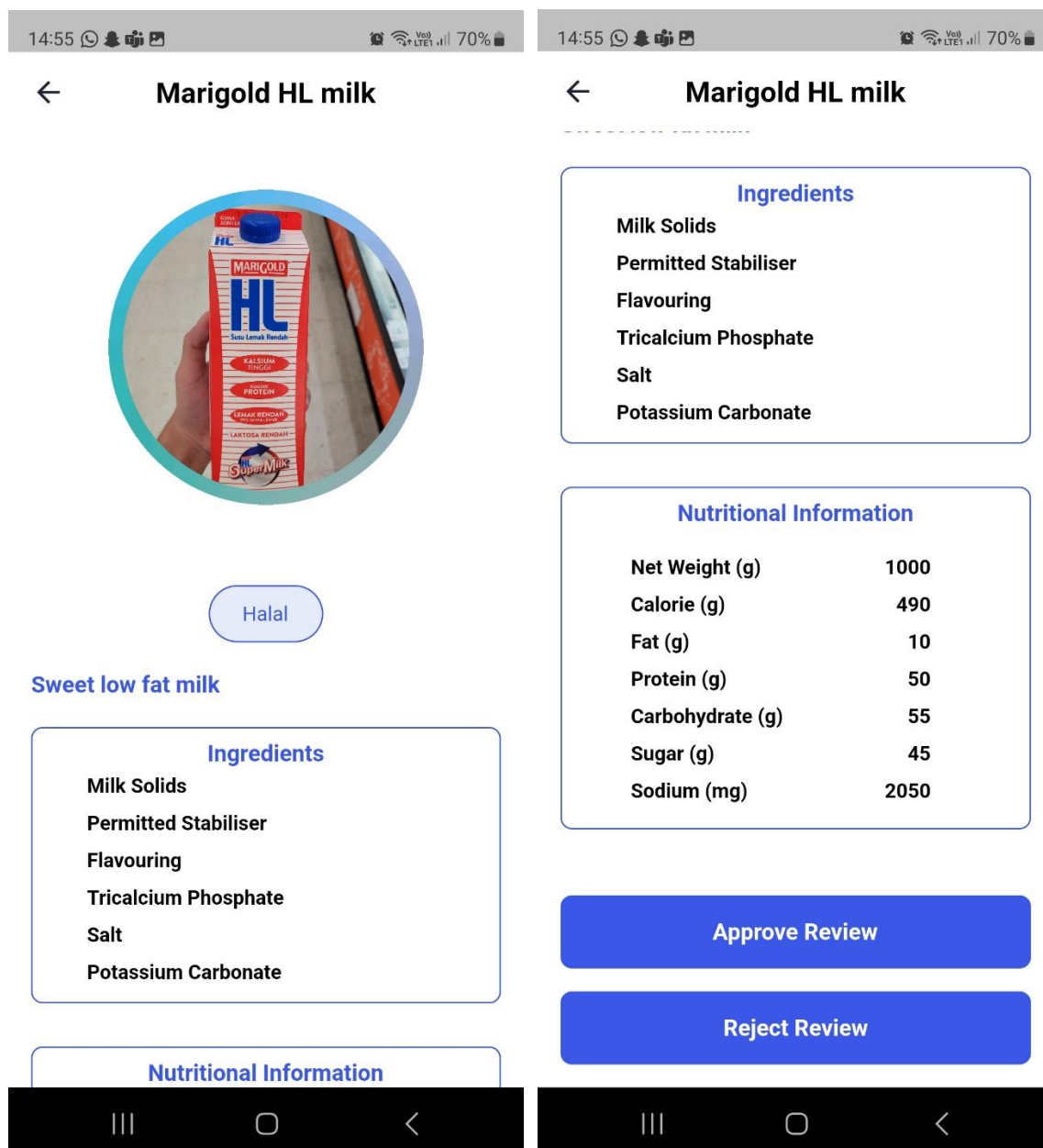
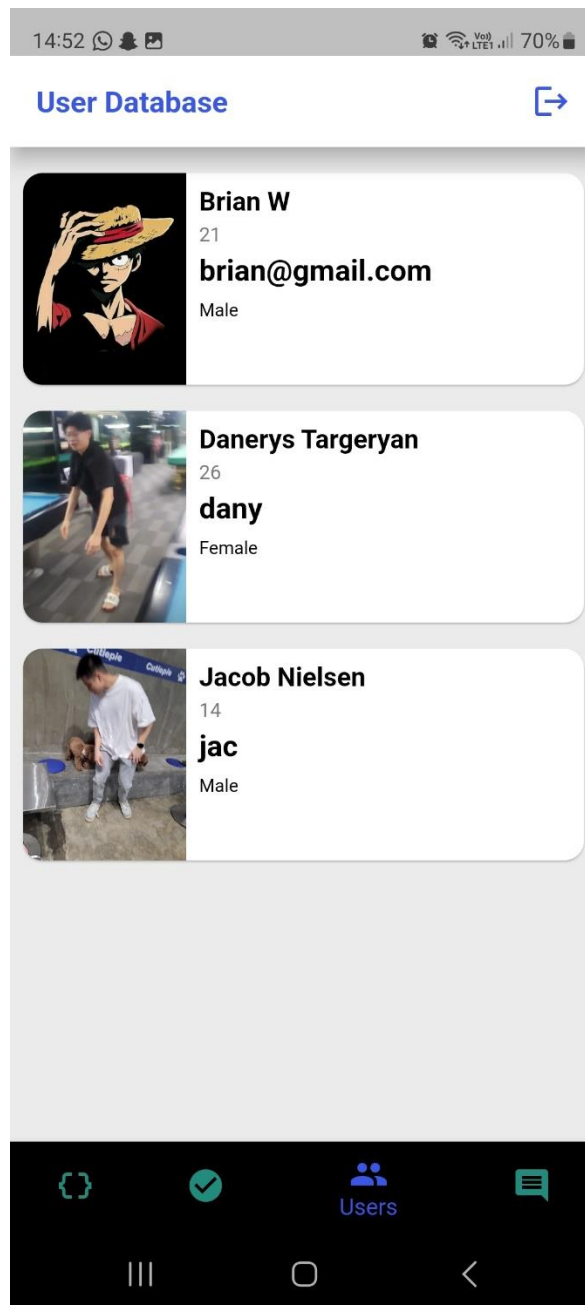


Figure 9.1.17.1 Screenshot for admin review details

Once admin click on a review data, admin will see this page that looks like the pages shown before. Here, admin can either approve or reject data submitted to the database by user. If admin approve, then the data will be added to the main database which can be seen by all users.

### 9.1.18 Screenshot for Admin User Database



*Figure 9.1.18.1 Screenshot for admin see user database*

This page is for admin to view all current users. Admin can see their full name, age, username, and also gender in the card preview.

### 9.1.19 Screenshot for Admin View User Detail

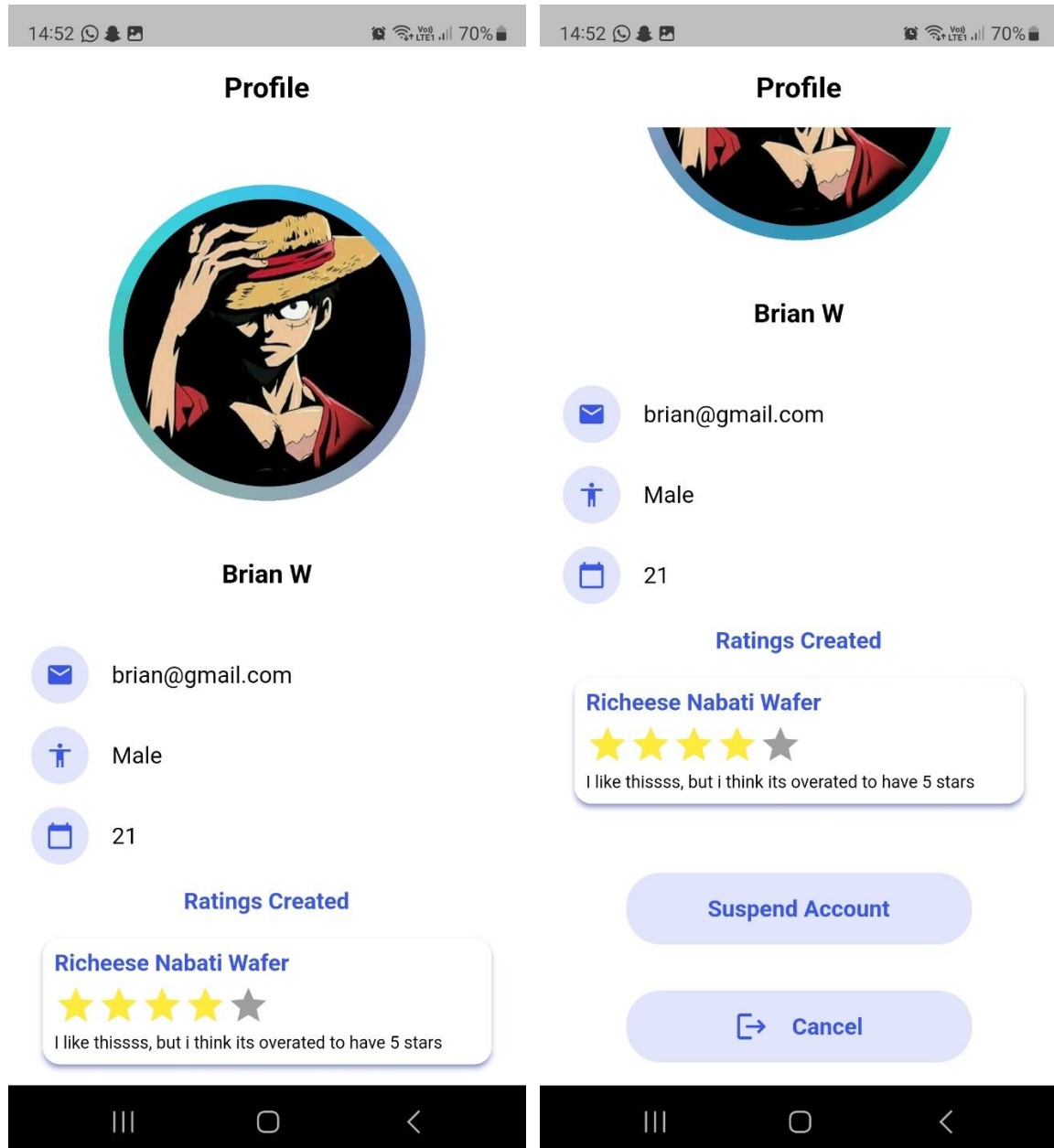


Figure 9.1.19.1 Screenshot for admin see user details

When a user is clicked, then admin can see all the details about the user, including the ratings that has been created under that account. Admin can decide to suspend a user account if admin judge that it might be appropriate.

### 9.1.20 Screenshot for Admin Rating Control

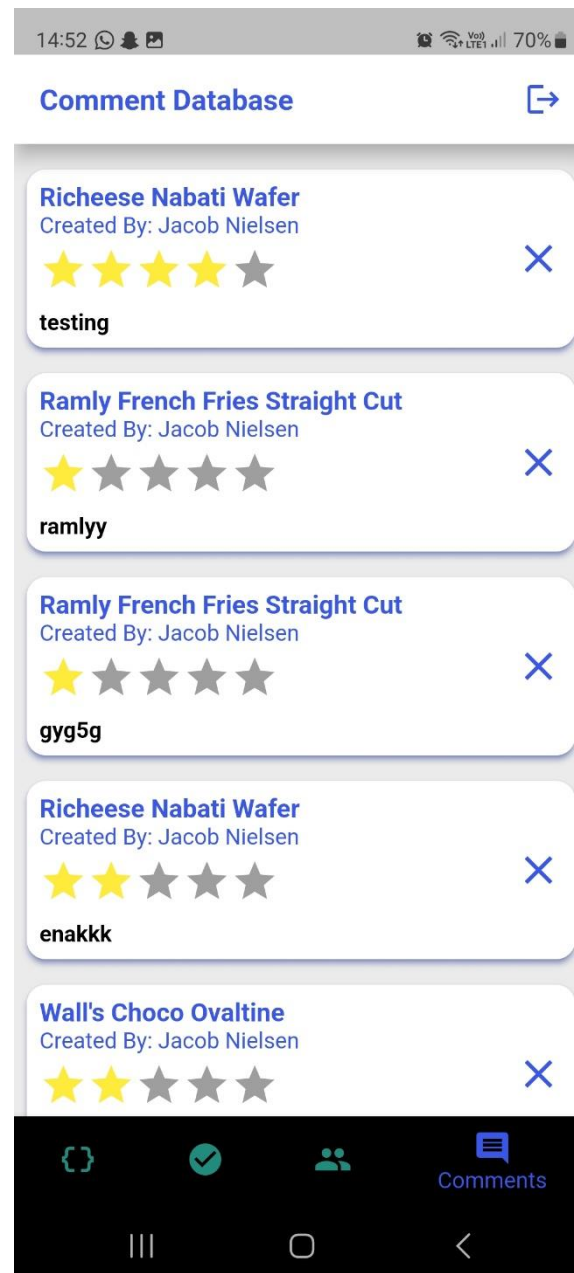


Figure 9.1.20.1 Screenshot for admin rating control

This last page is for admin to view all ratings made on all food. If admin thinks that the ratings are inappropriate, then admin can just delete the ratings by clicking the 'X' button on the right side of the ratings.

## 9.2 Sample Code (3)

### 9.2.1 Sample Code Admin Approve Review Data

```
void approveReview() async {
  await Supabase.instance.client.from('detailsTable').insert({
    'barcode': widget.productData.barcode,
    'name': widget.productData.name,
    'vegan': widget.productData.vegan,
    'gluten_free': widget.productData.glutenfree,
    'halal': widget.productData.halal,
    'netto': widget.productData.netto,
    'calorie_kcal': widget.productData.calorie,
    'fat_g': widget.productData.fat,
    'protein_g': widget.productData.protein,
    'carbo_g': widget.productData.carbo,
    'sodium_mg': widget.productData.sodium,
    'sugar_g': widget.productData.sugar,
    'details': widget.productData.details,
    'image_url': widget.productData.imgurl,
    'ingredients': widget.productData.ingredients
  });

  rejectReview();
}

void rejectReview() async {
  await Supabase.instance.client
    .from('reviewTable')
    .delete()
    .eq('barcode', widget.productData.barcode);
}
```

*Figure 9.2.2.1 Sample code for approve data*

The code snippet above shows the process when admin decides to approve a review data that has been submitted from user. When admin clicks the button, the `approveReview()` async function is called. It create a connection with Supabase, and inserts the data into the main database table, which is 'detailsTable' and after finish inserting, it also calls another function, which is `rejectReview()`. Reject review is a function for deleting data from the 2<sup>nd</sup> table 'reviewTable'. This function is needed because logically after approving, admin should no longer be able to see the data on the review page, which is rendered from the table 'reviewTable'.

## 9.2.2 Sample Code Login

```

void login() async {
  Navigator.push(
    context,
    MaterialPageRoute(
      builder: (context) => LoadingScreen(),
    ), // MaterialPageRoute
  );

  final data = await Supabase.instance.client
    .from('userTable')
    .select()
    .eq('username', _email)
    .eq('password', _password);

  if (_email == 'admin' && _password == 'myscanneradmin') {
    Navigator.of(context).pushNamedAndRemoveUntil("/admin", (route) => false);
  } else if (data.isEmpty) {
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: Text('Oh No!'),
          content: Text('We cannot find your credentials'),
          actions: <Widget>[
            TextButton(
              child: Text('OK'),
              onPressed: () {
                Navigator.of(context).pop();
                Navigator.of(context).pop();
              },
            ), // TextButton
          ], // <Widget>[]
        ); // AlertDialog
      },
    );
  } else if (data[0]['suspended'] == true) {
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: Text('Oh No!'),
          content: Text('This account has been suspended!'),
          actions: <Widget>[
            TextButton(
              child: Text('OK'),
              onPressed: () {
                Navigator.of(context).pop();
                Navigator.of(context).pop();
              },
            ), // TextButton
          ], // <Widget>[]
        ); // AlertDialog
      },
    );
  } else {
    updateAllGlobal(
      _email,
      _password,
      data[0]['full_name'],
      data[0]['gender'],
      data[0]['age'].toString(),
      data[0]['profile_url'],
      data[0]['id'].toString());

    if (data[0]['intro'] == false) {
      Navigator.of(context)
        .pushNamedAndRemoveUntil("/user", (route) => false);
    } else {
      Navigator.of(context)
        .pushNamedAndRemoveUntil("/intro", (route) => false);
    }
  }
}

```

Figure 9.2.3.1 Sample code for login

The code snippet above is used for logging in. The first thing that happens when a user clicks the login button is to show a loading screen. After that it will make a connection with Supabase to get data which matches the username and password used. If the email and password are equal to the admin credentials, then it will just direct to admin page, otherwise it checks if the data returns something. If it's empty, it will render a popup message saying, "We cannot find your credentials" and if the account is found but suspended, then the message will be "This account has been suspended". If it passes the checking conditions, then the global variable will be updated. If it is the first-time user use the system, then it will direct to tutorial page, otherwise it will direct to user main page.

### 9.2.3 Sample Code User Search Database

```
void searchData(String search) async {
  final response = await Supabase.instance.client
    .from('detailsTable')
    .select()
    .ilike('name', '%$search%');

  final rows = response as List<dynamic>;

  productList = rows.map((row) {
    return ProductData(
      barcode: row['barcode'] as int,
      name: row['name'] as String,
      vegan: row['vegan'] as bool,
      glutenfree: row['gluten_free'] as bool,
      halal: row['halal'] as bool,
      netto: row['netto'] as String,
      calorie: row['calorie_kcal'] as String,
      fat: row['fat_g'] as String,
      protein: row['protein_g'] as String,
      carbo: row['carbo_g'] as String,
      sodium: row['sodium_mg'] as String,
      sugar: row['sugar_g'] as String,
      details: row['details'] as String,
      imgurl: row['image_url'] as String,
      ingredients: row['ingredients'] as String);
  }).toList();

  setState(() {});
}
```


*Figure 9.2.1.1 Sample code for search data*

The code snippet above is used by both admin and user for searching the database based on keywords. The function accepts one string parameter, which is the keyword used to search the database. This function is similar to SQL 'like', in which they search for everything that might be similar to the user input in any string position. If it finds a response, then the data will be put into a list, which will be mapped into a custom ProductData class so that it can be shown into the user interface.




## Appendices

### 13.1 Poster



**Food Barcode  
Scanner**


**A.P.U**  
ASIA PACIFIC UNIVERSITY  
OF TECHNOLOGY & INNOVATION

Ignatius Brian Widjaja - TP058847  
B.Sc. Software Engineering – APD3F2211SE

## Introduction

My Scanner mobile application is a cutting-edge mobile application using Flutter framework and Dart programming language. My Scanner will simplify shopping experience and make informed choices with just a scan. Unlock the wealth of product information instantly, from nutritional facts to allergens, ensuring healthier decisions. Say goodbye to confusion and hello to convenience.





### Problem Statement

Nutritional information on food packaging is often overlooked by a majority of consumers, increasing the risk of unhealthy eating habits and excessive calorie intake.

Misleading marketing claims on packaging can also lead to confusion and potential health risks. Serving size information may be misinterpreted.

Addressing these challenges is crucial to promote informed and healthier food choices.

### Purpose

To give information about the food easily which is trusted and reliable to anyone who use the mobile application.



To give additional info on the food such as average price, taste, and other possible comments provided from other people.

Improve overall shopping experience for users.

### Implementation

My Scanner implement some programming tools such as Dart, Flutter, and Supabase, enabling it to run on both IOS and Android platform.

The frontend uses flutter which specialize in mobile application interface, and the backend is handled by Supabase, providing a scalable SQL database and bucket for images


## How it Works

User just need to grant access to the phone's camera to use it. After that user need to scan the food's barcode, and the system will display necessary data about the food. In case user does not have the barcode, they can also just search the food based on keyword.

## Conclusion

The developer successfully completed the development of "My Scanner," the barcode scanner, within three months. The system functions exceptionally well, meeting all expectations.

My Scanner empowers user to make smarter choices, promotes healthier living, and eliminates confusion during your shopping trips.

Figure 13.1.1 Poster