# National College of Ireland

Computing Project (BSHCSD4)

Software Development

Academic Year 2022/2023

Sutthita Phromchomcha

X19345621

X19345621@student.ncirl.ie

# URecipe

# Technical Report

## Contents

Executive Summary ... 3

1.0 Introduction ... 3

1.1. Background ... 3

1.2. Aims ... 3

1.3. Technology ... 3

1.4. Structure ... 3

2.0 System ... 3

2.1. Requirements ... 3

2.1.1. Functional Requirements ... 3

2.1.1.1. Use Case Diagram ... 4

2.1.1.2. Requirement 1: User Registration ... 4

2.1.1.3. Description & Priority ... 4

2.1.1.4. Use Case ... 5

2.1.1.5. Requirement 2: User Sign-In ... 6

2.1.1.6. Description & Priority ... 6

2.1.1.7. Use Case ... 6

2.1.1.8. Requirement 3: Recipe Creation ... 7

2.1.1.9. Description & Priority ... 7

2.1.1.10. Use Case ... 7

2.1.1.11. Requirement 4: Video Player ... 8

2.1.1.12. Description & Priority ... 8

2.1.1.13. Use Case ... 9

2.1.1.14. Requirement 5: Voice Speech ... 10

2.1.1.15. Description & Priority ... 10

2.1.1.16. Use Case ... 10

2.1.1.17. Requirement 6: Favourite Recipes ... 11

2.1.1.18. Description & Priority ... 11

2.1.1.19. Use Case ... 11

2.1.1.20. Requirement 7: Manage and Update ... 13

2.1.1.21. Description & Priority ... 13

2.1.1.22. Use Case ... 13

2.1.2. Data Requirements ... 14

2.1.3. User Requirements ... 14

2.1.4. Environmental Requirements ........ 14
2.1.5. Usability Requirements ........ 14
2.2. Design & Architecture ........ 15
2.3. Implementation ........ 16
2.4. Graphical User Interface (GUI) ........ 24
2.5. Testing ........ 27
2.6. Evaluation ........ 28
3.0 Conclusions ........ 28
4.0 Further Development or Research ........ 29
5.0 References ........ 29
6.0 Appendices ........ 29
6.1. Project Proposal ........ 30
6.1.1 Objectives ........ 30
6.1.2 Background ........ 30
6.1.3 State of Art ........ 30
6.1.4 Technical Approach ........ 31
6.1.5 Technical Details ........ 31
6.1.6 Special Resources Required ........ 32
6.1.7 Project Plan ........ 32
6.1.8 Testing ........ 34
6.2 Reflective Journals ........ 34

# Executive Summary

The purpose of this technical report is to describe the application, the steps that were undertaken, the utilization of technology and tools, and the requirements implemented within the application.

This report provides a background of the URecipe App how people can improve their culinary skills with this application and how it aims for any age and gender. Many technologies and platforms used to complete the application are also discussed.

The major section of this report is the System Requirements and its User Cases such as **user registration, user sign-in, recipe creation, video player, voice speech, favourite recipes**, and **manage and update**. Following by design and architecture, GUI, and evaluation from the application tests.

# 1.0 Introduction

## 1.1. Background

Cooking is a fulfilling activity that allows people to turn basic food into something appealing and captivating. There were a lot of food photos provided by people who loved to spend the lockdown improving their culinary abilities during Covid-19 and the number of apps downloaded each day is expanding drastically, this is why this project was chosen to be undertaken.

## 1.2. Aims

This project aims to create a virtual kitchen assistant with simple functions for anyone to use that allows users to create, save and watch videos at any time. Additionally, to distinguish this project from other recipe applications on the market such as Yummly and Tasty.

## 1.3. Technology

The platform used for creating and designing the application is **Android Studio**. Since I had an experience with this platform before, therefore it is easier and quicker to use for the project. User accounts are stored in **Firebase Realtime Database**. This was accomplished with the connection between Android Studio and Firebase. It will also be used to save user recipes as they are created.

**Spoonacular** API contains many recipes, therefore it will be used to display on the Explorer page in the application.

Android Studio is connected to **GitHub** where all the coding can be added and updated in the repository.

## 1.4. Structure

The structure of this document begins with a description of the application system requirements and use cases. The following discussions are about functional requirements and the authentication from the user. Other sections are designing, implementing, graphical interfaces, testing, and evaluating the application.

# 2.0 System

## 2.1. Requirements

### 2.1.1. Functional Requirements

1. **User Registration** - This function is required as a user must create an account before creating and saving recipes in the application.

2. **User Sign-in –** This function is required as a user must have it before they can make the recipe and other features on the Explore page.
3. **Recipe Creation –** This function is required as every new recipe added needs to be stored in the application under the user's account name.
4. **Video Player –** This function is required as users can play the recipe video and use voice speech on the same page.
5. **Voice Speech –** This function is required as users can have access to videos and use voice speech to open the video's recipe.
6. **Favourite Recipes –** This function is required as the system will use the user's favourite recipe to recommend another recipe in the application.
7. **Manage & Update -** This function is required since the program will require system upgrades as well as additional functions that may be implemented in the future. In addition to updates, the system administrator needs to maintain the application daily so that the application functions properly.

#### 2.1.1.1. Use Case Diagram

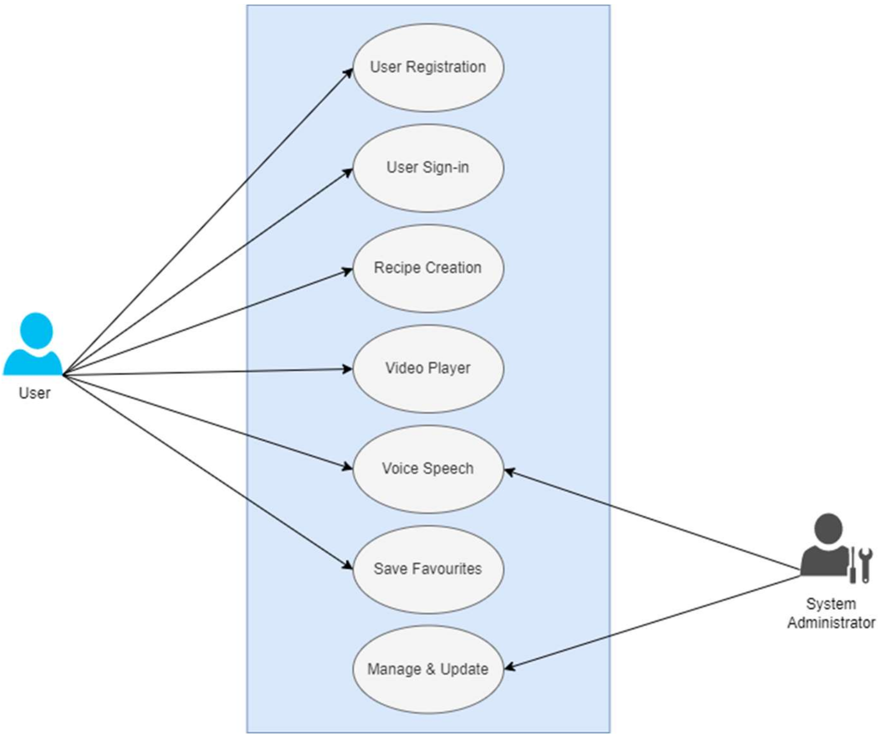


*Figure 1 Use Case Diagram*

#### 2.1.1.2. Requirement 1: User Registration

#### 2.1.1.3. Description & Priority

Requirement 1 allows users to create an account to access and organize their recipes in the application.

### 2.1.1.4. Use Case

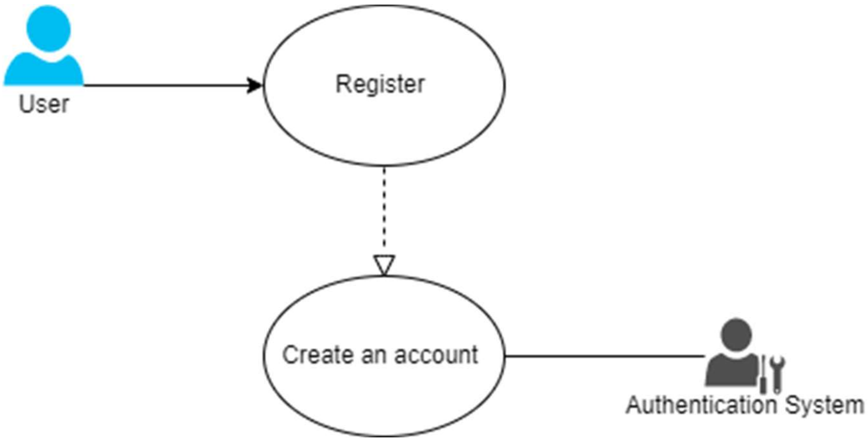


*Figure 2 Sign-Up Use Case*

**Scope**

The scope of this use case is to allow users to create an account.

**Description**

This use case describes how to register an account.

**Use Case Diagram**

This use case diagram is between a user and an authentication system.

**Flow Description**

**Precondition**

The application has been opened by the user.

**Activation**

This use case starts when a user selects the “Create an account” button on the opening page.

**Main flow**

1. The application displays the choice to log in or create an account.
2. The user clicks on the 'Create Account button.
3. The user enters a valid email address and a password.
4. The application continues to create a new account from the authentication system.
5. The account has been created, and the user can access it via the mobile application.

**Alternate flow**

A1 : Incorrect details

1. The authentication system alerts the user if the details they entered are invalid.
2. The user can retry with their correct details.
3. The use case continues at position 4 of the main flow.

**Exceptional flow**

E1 : Account Registration Error

1. An error occurs during the creation process with the authentication system.
2. The user comprehends an error has occurred.
3. The use case continues at position 1 of the main flow.

**Termination**

The system presents the home screen of the application.

**Post condition**

The authentication system creates a new account for the user.

### 2.1.1.5. Requirement 2: User Sign-In

### 2.1.1.6. Description & Priority

Requirement 2 allows users to log in to their accounts to access their information and all data associated with the application.

### 2.1.1.7. Use Case

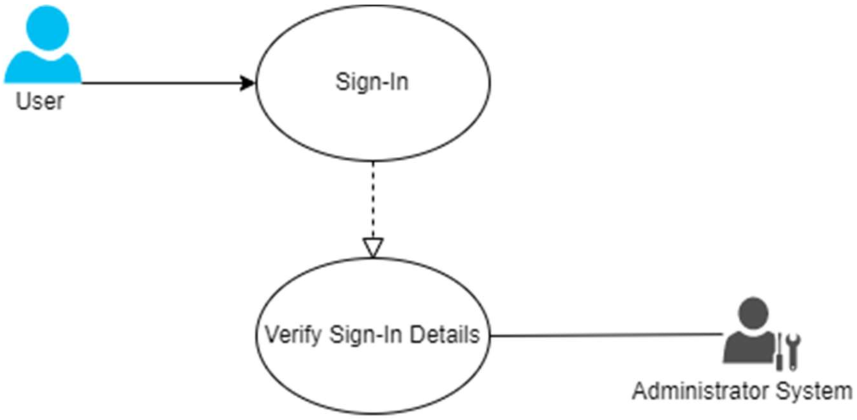


*Figure 3 Sign-in Use Case*

**Scope**

The scope of this use case is to allow users to log in to their accounts.

**Description**

This use case describes how to log in to their account.

**Use Case Diagram**

This use case diagram is between a user and an authentication system.

**Flow Description**

**Precondition**

The application has been opened by the user.

**Activation**

This use case starts when a user selects the “Log In” button on the opening page.

**Main flow**

1. The application displays the choice to log in or create an account.
2. The user clicks on the ‘Login’ button.
3. The user enters their email and password.
4. The authentication system scan and confirms the login details are correct.
5. The user is logged in successfully leading to a home page.

**Alternate flow**

A1 : Incorrect details

1. The authentication system alerts the user if the details they entered are incorrect.
2. The user can retry with their correct details.
3. The use case continues at position 4 of the main flow.

**Exceptional flow**

E1 : Login Error

1. An error occurs during the creation process with the authentication system.
2. The user comprehends an error has occurred.
3. The use case continues at position 1 of the main flow.

**Termination**

The system presents the home screen of the application.

**Post condition**

The application logs the user into their account.

### 2.1.1.8. Requirement 3: Recipe Creation

### 2.1.1.9. Description & Priority

Requirement 3 allows users to create and display a recipe in the application.

### 2.1.1.10. Use Case

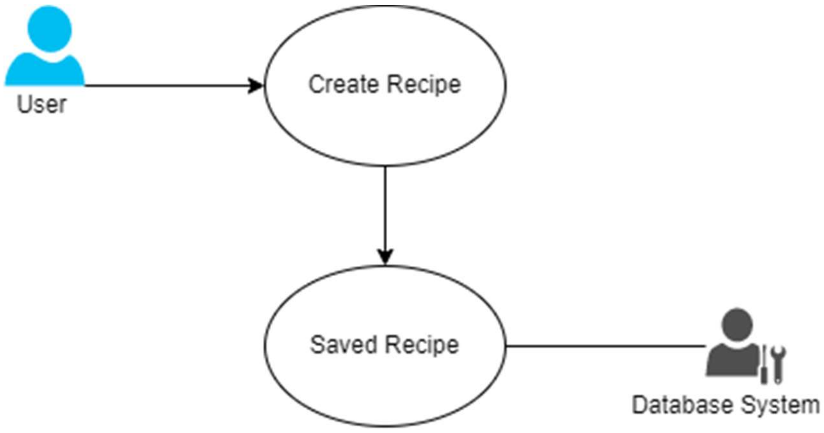


*Figure 4 Recipe Use Case*

**Scope**

The scope of this use case is to allow users to create a recipe.

**Description**

This use case describes how to create a recipe.

**Use Case Diagram**

This use case diagram is between a user and a database system.

**Flow Description**

**Precondition**

The application has been opened by the user.

**Activation**

This use case starts when a user selects the "My Recipe" button on the homepage screen.

**Main flow**

1. The application displays an add button to create a new recipe.
2. A dialog shows up as the add button is clicked.
3. The user enters the recipe name and adds it.
4. The application saves the recipe and continues to the main page.
5. The recipe has been created, and the user can view, edit, and delete them at any time.

**Alternate flow**

A1 : Replication

4. The authentication system alerts the user if the recipe name has been created.
5. The user can retry with the new recipe name.
6. The use case continues at position 2 of the main flow.

**Exceptional flow**

E1 : System Error

7. An error occurs during the creation process with the database system.
8. The user comprehends an error has occurred.
9. The use case continues at position 1 of the main flow.

**Termination**

The system presents the main recipe page of the application.

**Post condition**

The new recipe has been created and stored in the database.

### 2.1.1.11. Requirement 4: Video Player

### 2.1.1.12. Description & Priority

Requirement 5 allows users to play a recipe video in the application.

### 2.1.1.13. Use Case

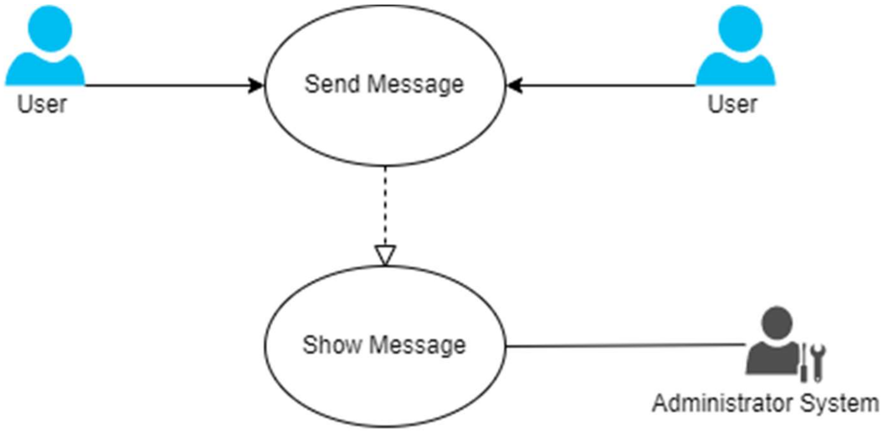


*Figure 5 Video Player Use Case*

**Scope**

The scope of this use case is to allow users to play the video.

**Description**

This use case diagram showed that the user can control the video in the application.

**Use Case Diagram**

This use case diagram is between a user and an administrator system.

**Flow Description**

**Precondition**

The application has been opened by the user.

**Activation**

This use case starts when a user opens the video activity page.

**Main flow**

1. The application displays a recipe video on the screen.
2. The user clicks to play the video provided by the system.
3. The user has the option to control the video.
4. The system operates the user's request.

**Alternate flow**

A1 : Video Playing Error

1. The user checks the connection.
2. The use case continues at position 1 of the main flow.

**Exceptional flow**

E1 : System Error

1. An error occurs during playing the video.
2. The user comprehends an error has occurred.
3. The use case continues at position 1 of the main flow.

**Termination**

The system presents the video to the user.

**Post condition**

The application performs video controlled by the user.

### 2.1.1.14. Requirement 5: Voice Speech

### 2.1.1.15. Description & Priority

Requirement 4 allows users to activate a voice speech to open a recipe page from the video.

### 2.1.1.16. Use Case

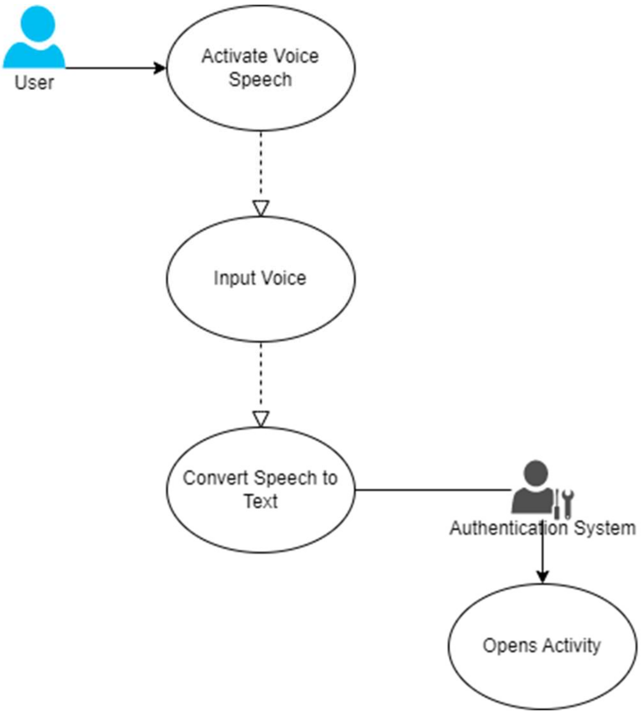


*Figure 6 Voice Speech Use Case*

**Scope**

The scope of this use case is to allow users to use voice speech.

**Description**

This use case describes how voice speech can be activated.

**Use Case Diagram**

This use case diagram is between a user and a voice recognition system.

**Flow Description**

**Precondition**

The application has been opened by the user.

**Activation**

This use case starts when a user turns on “Voice Speech” on the video page.

**Main flow**

1. The application displays the voice speech button.
2. The user presses the voice speech button.
3. The user input the keyword to activate the function.
4. The application converts the speech to text and operates the function.
5. The recipe instructions are opened.

**Alternate flow**

A1 : Voice Speech Inactive

1. The user can restart the application.
2. The use case continues at position 1 of the main flow.

**Exceptional flow**

E1 : System Error

1. An error occurs during the speech operation.
2. The user comprehends an error has occurred.
3. The use case continues at position 1 of the main flow.

**Termination**

The system presents the recipe page of the application.

**Post condition**

The application performs a command from the user.

### 2.1.1.17. Requirement 6: Favourite Recipes

### 2.1.1.18. Description & Priority

Requirement 6 allows users to save recipes from the main recipe page.

### 2.1.1.19. Use Case

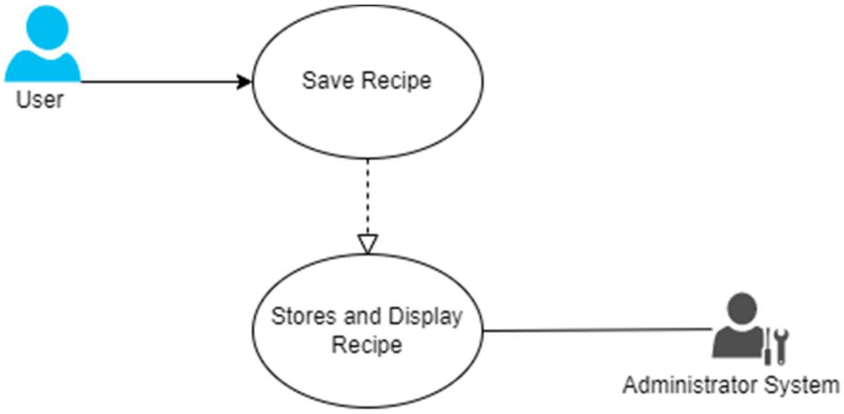


*Figure 7 Favourite Recipe Use Case*

**Scope**

The scope of this use case is to allow users to save recipes.

**Description**

This use case describes how recipes are saved in the application.

**Use Case Diagram**

This use case diagram is between a user and the system.

**Flow Description**

**Precondition**

The application has been opened by the user.

**Activation**

This use case starts when a user clicks a save button on each recipe.

**Main flow**

1. The application display recipes.
2. The user clicks a save-to-favourite button.
3. The saved recipe is sent to the user's favourite recipe page.
4. The application stores the saved recipe.
5. The saved recipes are displayed on the user's favourite recipe page.

**Alternate flow**

A1 : Saved Recipe Error

1. The user checks the connection.
2. The use case continues at position 1 of the main flow.

**Exceptional flow**

E1 : System Error

1. An error occurs during messaging or sending.
2. The user comprehends an error has occurred.
3. The use case continues at position 1 of the main flow.

**Termination**

The system presents the saved recipes on the user page.

**Post condition**

The application stores the data of the saved recipe and performs a recommendation.

### 2.1.1.20. Requirement 7: Manage and Update

### 2.1.1.21. Description & Priority

Requirement 7 allows the administrator system to manage and update the application.

### 2.1.1.22. Use Case

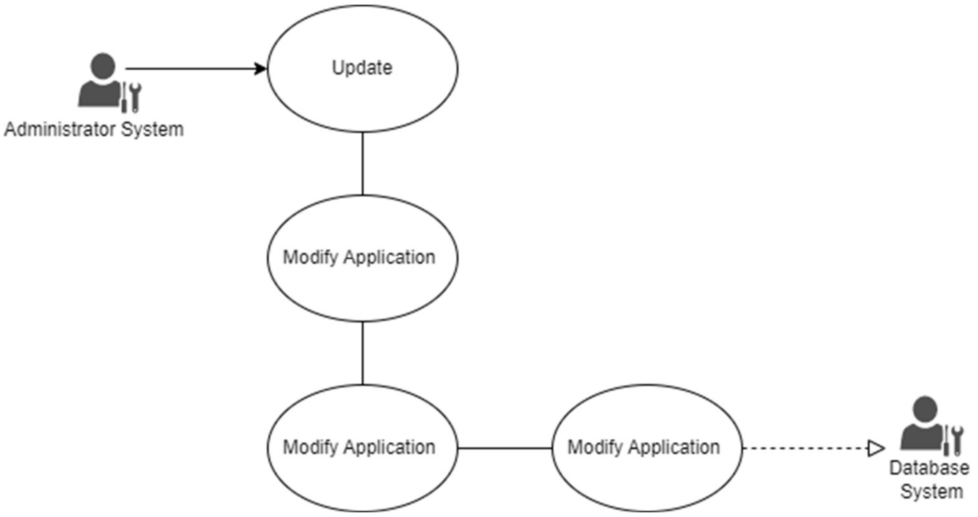


*Figure 8 Manage and Update Use Case*

**Scope**

The scope of this use case is to allow the system to maintain and update the application.

**Description**

This use case describes how the administrator system can manage and update the application.

**Use Case Diagram**

This use case diagram is between the administrator system and the database system.

**Flow Description**

**Precondition**

The application has been opened by the user.

**Activation**

This use case starts when the administrator opens the database system and modifies the application.

**Main flow**

1. The database system displays the system administrator with an administrator-only login.
2. The system enters their login information and hits 'Login'.
3. The database system allows access to the system's administrator.
4. The system administrator monitors, modifies, and/or applies any modifications to the database system they desire.
5. The system administrator saves the adjustments and pushes the information to the database system.

**Alternate flow**

A1 : Incorrect details

1. The application informs the system administrator of the wrong login information.
2. The system administrator may retry with the right information.
3. The use case continues at position 3 of the main flow.

**Exceptional flow**

E1 : System error

1. A database system error occurs while the system administrator is attempting to implement changes.
2. The system administrator comprehends that an error has occurred.
3. The use case continues at position 3 of the main flow.

**Termination**

The system administrator has altered and modified the database system.

**Post condition**

The application has been controlled and updated by the system administrator.

### 2.1.2. Data Requirements

**Performance** - The application should transmit information often while running in the background to guarantee dependability and ensure that users can complete the tasks without assistance. All graphical interfaces should be capable of retrieving and displaying data in less than 2 seconds.

### 2.1.3. User Requirements

**Security** – A user must have a functioning internet connection and the appropriate credentials to access their account. They must validate their email address upon registration, and they will be the only ones who can reset their passwords by email or a verification code delivered to their mobile devices. Each user's password will be encrypted with Firebase authentication, preventing unauthorized individuals from accessing other users' accounts.

**Reliability** - The program must not crash as a consequence of simulation problems. When an error occurs, it should be addressed at the system level to restart these processes. The system cannot work alone without the internet; however, it should not crash if the connection is lost; instead, it should halt and wait for the connection to be re-established.

### 2.1.4. Environmental Requirements

The application requires an Android system 4.0 (JellyBean) or higher.

### 2.1.5. Usability Requirements

**Recover** – If the user forgets their password, which prevents them from accessing the app, they can reset it by selecting 'Forget Password?'. The user will receive an email with a code that they must enter for security purposes. They can log in to their account when their password has been reset, and all of their information is saved in the application system.

**Reusability** - The code that was implemented in the program will be reused in many elements of the application. Other additional features that may be developed in the future will be added to pages and options.

## 2.2. Design & Architecture

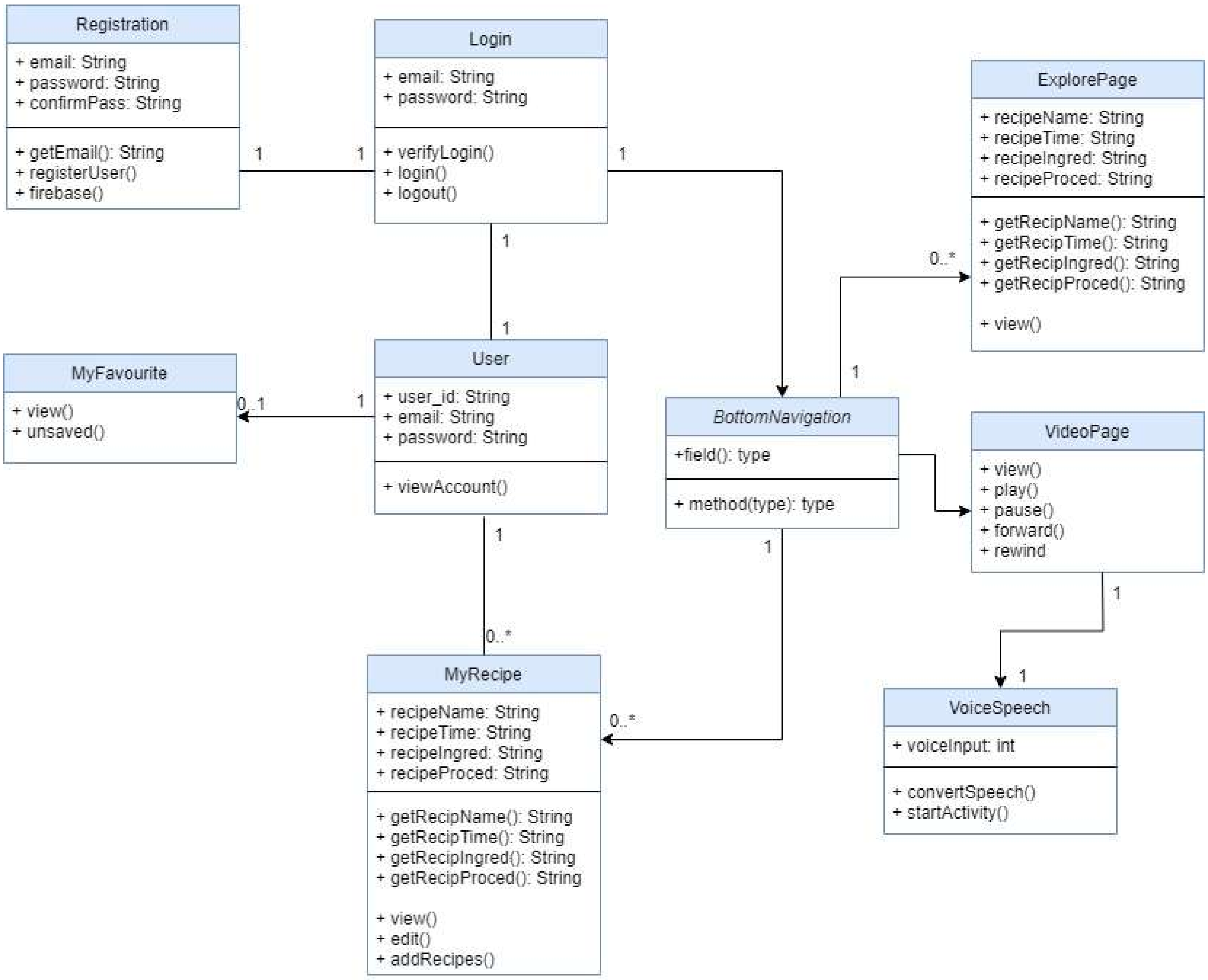
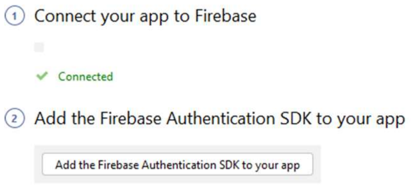


*Figure 9 Architecture Diagram*

## 2.3. Implementation

This code implementation indicates a user registration into the application. The user creates an account through the application and their details will be stored in the Firebase. Below is the snapshot of the application successfully connected to Firebase.



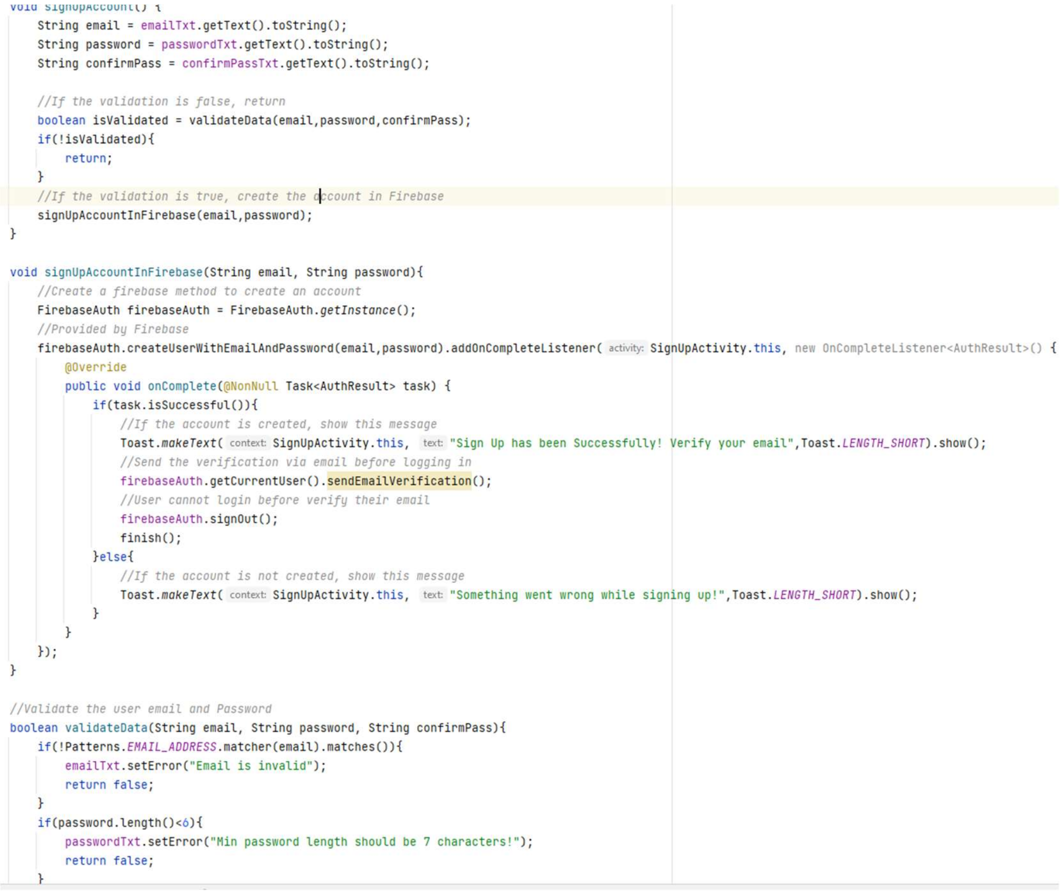
① Connect your app to Firebase

✓ Connected

② Add the Firebase Authentication SDK to your app

*Figure 10 Firebase*

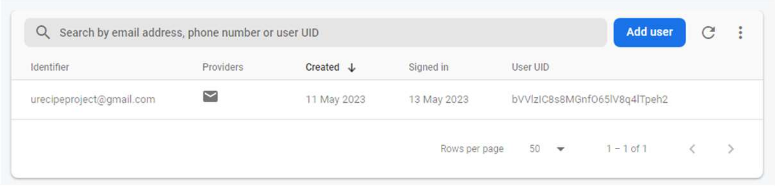
Firebase Authentication method was provided by Firebase to utilize the user's email and password to successfully create their account. I have also implemented the validation method to validate the user's email and password. If the user's email is valid, verification will be sent via the user's email to verify their email before logging into the application. If the user's account has been successfully created, a message will be displayed; otherwise, an error message will notify the user. Then they can retry the registration.



```java
void signUpAccount() {
    String email = emailTxt.getText().toString();
    String password = passwordTxt.getText().toString();
    String confirmPass = confirmPassTxt.getText().toString();

    //If the validation is false, return
    boolean isValidated = validateData(email,password,confirmPass);
    if(!isValidated){
        return;
    }
    //If the validation is true, create the account in Firebase
    signUpAccountInFirebase(email,password);
}

void signUpAccountInFirebase(String email, String password){
    //Create a firebase method to create an account
    FirebaseAuth firebaseAuth = FirebaseAuth.getInstance();
    //Provided by Firebase
    firebaseAuth.createUserWithEmailAndPassword(email,password).addOnCompleteListener( activity: SignUpActivity.this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if(task.isSuccessful()){
                //If the account is created, show this message
                Toast.makeText( context: SignUpActivity.this,  text: "Sign Up has been Successfully! Verify your email",Toast.LENGTH_SHORT).show();
                //Send the verification via email before logging in
                firebaseAuth.getCurrentUser().sendEmailVerification();
                //User cannot login before verify their email
                firebaseAuth.signOut();
                finish();
            }else{
                //If the account is not created, show this message
                Toast.makeText( context: SignUpActivity.this,  text: "Something went wrong while signing up!",Toast.LENGTH_SHORT).show();
            }
        }
    });
}

//Validate the user email and Password
boolean validateData(String email, String password, String confirmPass){
    if(!Patterns.EMAIL_ADDRESS.matcher(email).matches()){
        emailTxt.setError("Email is invalid");
        return false;
    }
    if(password.length()<6){
        passwordTxt.setError("Min password length should be 7 characters!");
        return false;
    }
```

*Figure 11 Sign-Up Code*

Once the user has created their account, their data will be stored in Firebase. An example has been provided with the creation of the application's email:

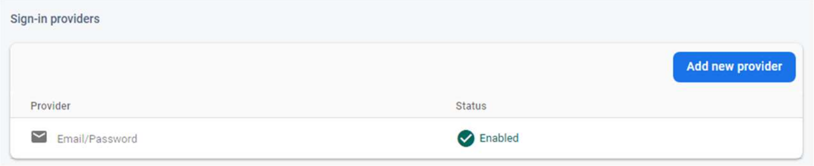


| Identifier | Providers | Created ↓ | Signed in | User UID |
|---|---|---|---|---|
| urecipeproject@gmail.com | ✉ | 11 May 2023 | 13 May 2023 | bVVlzIC8s8MGnfO65lV8q4lTpeh2 |

*Figure 12 Firebase User Data*

This implementation of code is for Login Authentication in Firebase. The application will require the user to log into the application with their email and password.

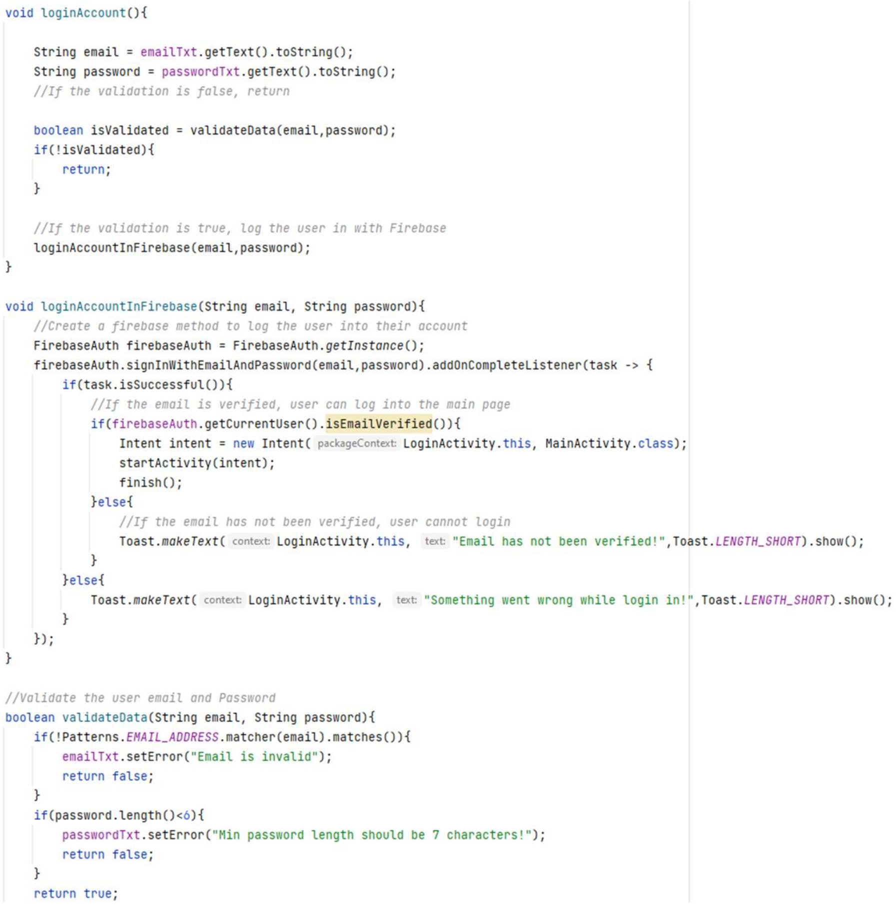
In Firebase, I have enabled the login method to perform this function.



| Provider | Status |
|---|---|
| ✉ Email/Password | ✓ Enabled |

*Figure 13 Firebase Sign-In Method*

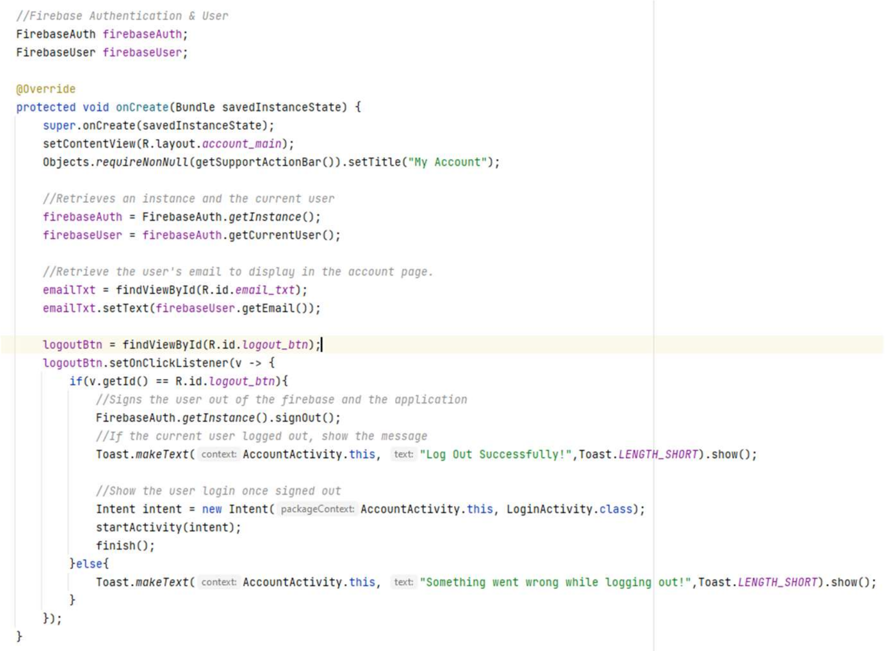
I have also implemented the validation of the user's email and password. If the user's email is valid or if the email has not been verified, the application will display a message to notify the user before continue logging in.



```
void loginAccount(){

    String email = emailTxt.getText().toString();
    String password = passwordTxt.getText().toString();
    //If the validation is false, return

    boolean isValidated = validateData(email,password);
    if(!isValidated){
        return;
    }

    //If the validation is true, log the user in with Firebase
    loginAccountInFirebase(email,password);
}

void loginAccountInFirebase(String email, String password){
    //Create a firebase method to log the user into their account
    FirebaseAuth firebaseAuth = FirebaseAuth.getInstance();
    firebaseAuth.signInWithEmailAndPassword(email,password).addOnCompleteListener(task -> {
        if(task.isSuccessful()){
            //If the email is verified, user can log into the main page
            if(firebaseAuth.getCurrentUser().isEmailVerified()){
                Intent intent = new Intent( packageContext: LoginActivity.this, MainActivity.class);
                startActivity(intent);
                finish();
            }else{
                //If the email has not been verified, user cannot login
                Toast.makeText( context: LoginActivity.this,  text: "Email has not been verified!",Toast.LENGTH_SHORT).show();
            }
        }else{
            Toast.makeText( context: LoginActivity.this,  text: "Something went wrong while login in!",Toast.LENGTH_SHORT).show();
        }
    });
}

//Validate the user email and Password
boolean validateData(String email, String password){
    if(!Patterns.EMAIL_ADDRESS.matcher(email).matches()){
        emailTxt.setError("Email is invalid");
        return false;
    }
    if(password.length()<6){
        passwordTxt.setError("Min password length should be 7 characters!");
        return false;
    }
    return true;
```

*Figure 14 Sign-in Code*

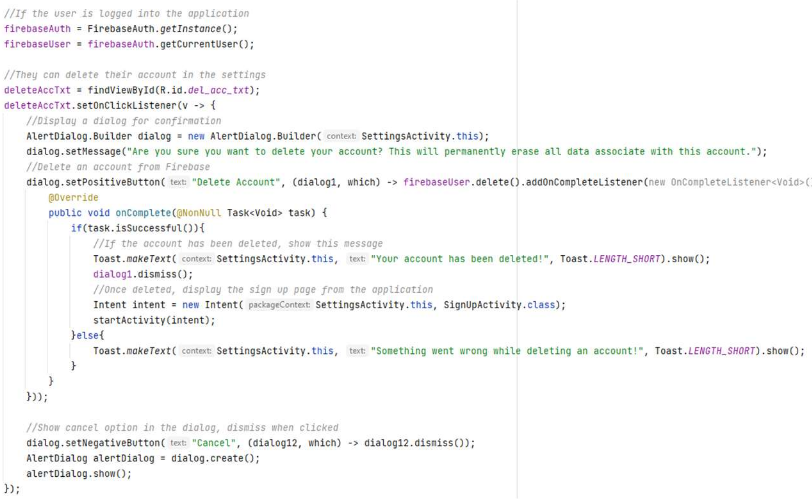
This implementation indicates a user account after they have registered in the application and the logout function. The application has to get the user email from Firebase to display on the Account Page. The code below also performs logging the user out from the application whenever the user selects the “Log Out” option on the account page. If the user logged out successfully, the application would display a message and the Login Activity (referring to Login GUI). This will prevent the user from accessing the main page of the application.



```
//Firebase Authentication & User
FirebaseAuth firebaseAuth;
FirebaseUser firebaseUser;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.account_main);
    Objects.requireNonNull(getSupportActionBar()).setTitle("My Account");

    //Retrieves an instance and the current user
    firebaseAuth = FirebaseAuth.getInstance();
    firebaseUser = firebaseAuth.getCurrentUser();

    //Retrieve the user's email to display in the account page.
    emailTxt = findViewById(R.id.email_txt);
    emailTxt.setText(firebaseUser.getEmail());

    logoutBtn = findViewById(R.id.logout_btn);
    logoutBtn.setOnClickListener(v -> {
        if(v.getId() == R.id.logout_btn){
            //Signs the user out of the firebase and the application
            FirebaseAuth.getInstance().signOut();
            //If the current user logged out, show the message
            Toast.makeText( context: AccountActivity.this,  text: "Log Out Successfully!",Toast.LENGTH_SHORT).show();

            //Show the user login once signed out
            Intent intent = new Intent( packageContext: AccountActivity.this, LoginActivity.class);
            startActivity(intent);
            finish();
        }else{
            Toast.makeText( context: AccountActivity.this,  text: "Something went wrong while logging out!",Toast.LENGTH_SHORT).show();
        }
    });
}
```

*Figure 15 User's Account & Logout Code*

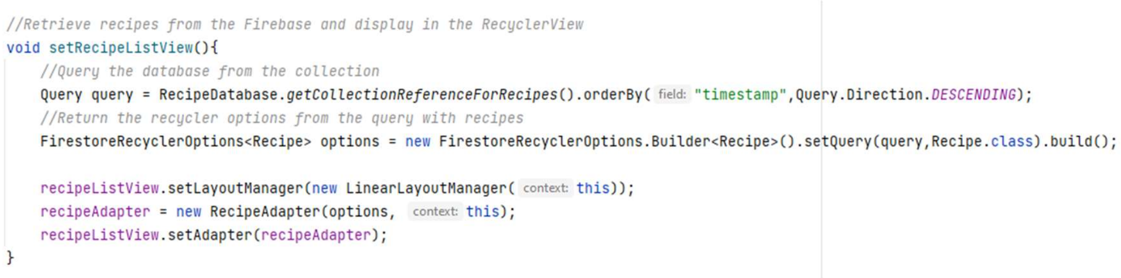
The implementation of this code is to delete an account from Firebase. Firstly, the application will get the current user if they are logged in to perform the delete function. When the user clicks “Delete Account” on the main settings page an alert dialog will appear for the user confirmation to completely delete their account. The code indicates that if the account has been successfully deleted from Firebase, a message will be displayed; otherwise, an error message will notify the user if the user account was not successfully deleted.



```
//If the user is logged into the application
firebaseAuth = FirebaseAuth.getInstance();
firebaseUser = firebaseAuth.getCurrentUser();

//They can delete their account in the settings
deleteAccTxt = findViewById(R.id.del_acc_txt);
deleteAccTxt.setOnClickListener(v -> {
    //Display a dialog for confirmation
    AlertDialog.Builder dialog = new AlertDialog.Builder( context: SettingsActivity.this);
    dialog.setMessage("Are you sure you want to delete your account? This will permanently erase all data associate with this account.");
    //Delete an account from Firebase
    dialog.setPositiveButton( text: "Delete Account", (dialog1, which) -> firebaseUser.delete().addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            if(task.isSuccessful()){
                //If the account has been deleted, show this message
                Toast.makeText( context: SettingsActivity.this,  text: "Your account has been deleted!", Toast.LENGTH_SHORT).show();
                dialog1.dismiss();
                //Once deleted, display the sign up page from the application
                Intent intent = new Intent( packageContext: SettingsActivity.this, SignUpActivity.class);
                startActivity(intent);
            }else{
                Toast.makeText( context: SettingsActivity.this,  text: "Something went wrong while deleting an account!", Toast.LENGTH_SHORT).show();
            }
        }
    }));

    //Show cancel option in the dialog, dismiss when clicked
    dialog.setNegativeButton( text: "Cancel", (dialog12, which) -> dialog12.dismiss());
    AlertDialog alertDialog = dialog.create();
    alertDialog.show();
});
```

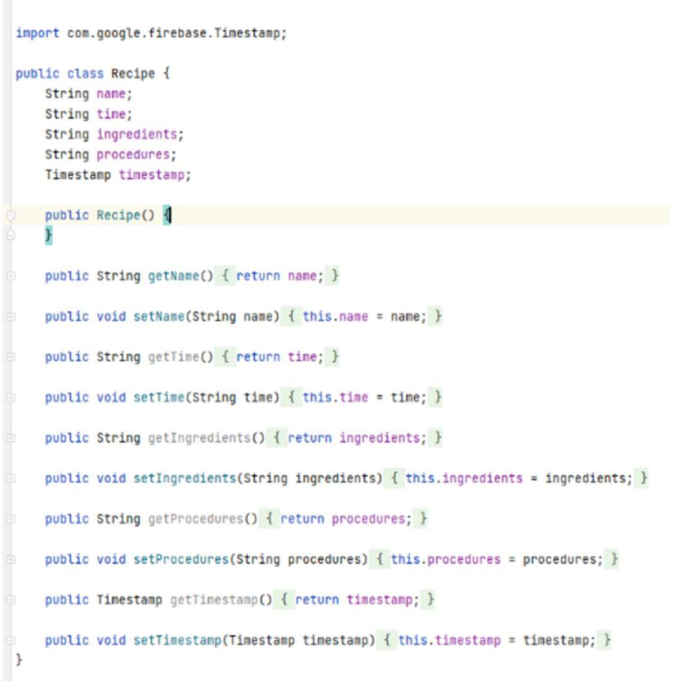
*Figure 16 Delete Account Code*

This implementation of code is for the “My Recipe” Page. User can create their recipe and store it in Firebase under their unique ID (see Figures 19 & 20). The recipes are stored as document types. When the user clicks the add button an activity will display containing text fields requesting the user’s input. For that function, I have created a class to store recipe names, time, ingredients, and procedures (see Figure 18).To display the user recipe, the application has to retrieve the data from the Firebase collection (see Figure 20). The recipes will be displayed in a recycler list view that reuses the cardholder (see Fig 17).



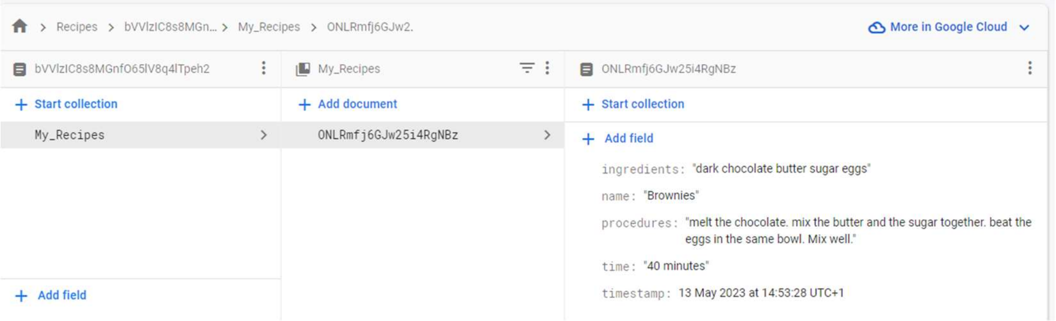
```java
//Retrieve recipes from the Firebase and display in the RecyclerView
void setRecipeListView(){
    //Query the database from the collection
    Query query = RecipeDatabase.getCollectionReferenceForRecipes().orderBy( field: "timestamp",Query.Direction.DESCENDING);
    //Return the recycler options from the query with recipes
    FirestoreRecyclerOptions<Recipe> options = new FirestoreRecyclerOptions.Builder<Recipe>().setQuery(query,Recipe.class).build();

    recipeListView.setLayoutManager(new LinearLayoutManager( context: this));
    recipeAdapter = new RecipeAdapter(options,  context: this);
    recipeListView.setAdapter(recipeAdapter);
}
```

*Figure 17 My Recipe Main Code*



```java
import com.google.firebase.Timestamp;

public class Recipe {
    String name;
    String time;
    String ingredients;
    String procedures;
    Timestamp timestamp;

    public Recipe() {
    }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String getTime() { return time; }

    public void setTime(String time) { this.time = time; }

    public String getIngredients() { return ingredients; }

    public void setIngredients(String ingredients) { this.ingredients = ingredients; }

    public String getProcedures() { return procedures; }

    public void setProcedures(String procedures) { this.procedures = procedures; }

    public Timestamp getTimestamp() { return timestamp; }

    public void setTimestamp(Timestamp timestamp) { this.timestamp = timestamp; }
}
```

*Figure 18 My Recipe Class Code*



ingredients: "dark chocolate butter sugar eggs"

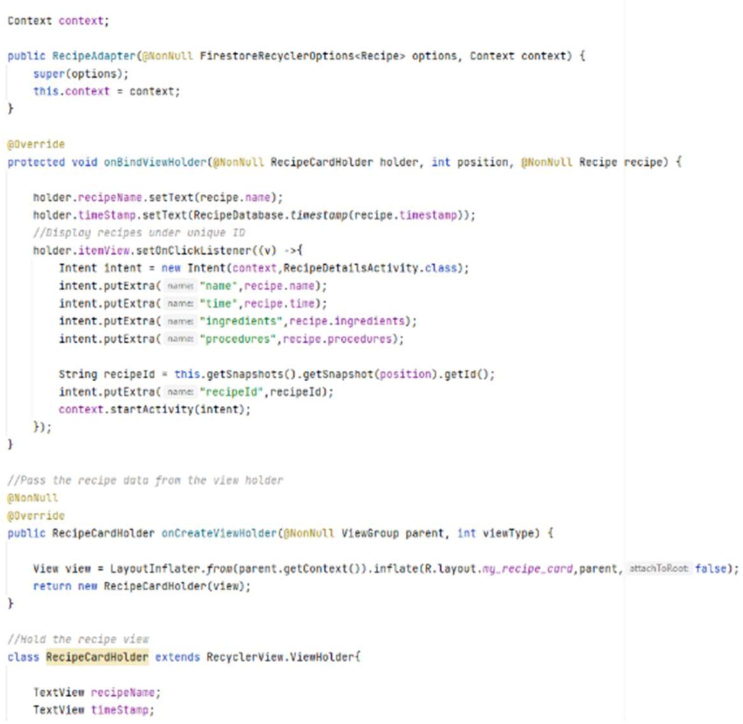
name: "Brownies"

procedures: "melt the chocolate. mix the butter and the sugar together. beat the eggs in the same bowl. Mix well."

time: "40 minutes"

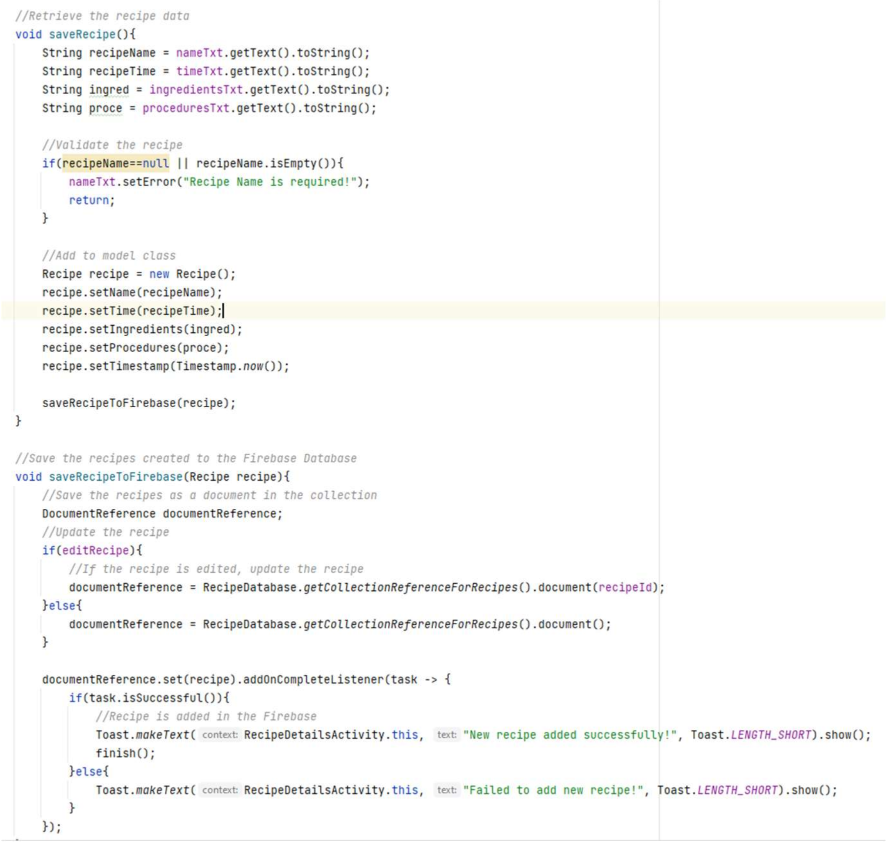
timestamp: 13 May 2023 at 14:53:28 UTC+1

*Figure 19 My Recipe Firebase Code*



```java
Context context;

public RecipeAdapter(@NonNull FirestoreRecyclerOptions<Recipe> options, Context context) {
    super(options);
    this.context = context;
}

@Override
protected void onBindViewHolder(@NonNull RecipeCardHolder holder, int position, @NonNull Recipe recipe) {

    holder.recipeName.setText(recipe.name);
    holder.timeStamp.setText(RecipeDatabase.timestamp(recipe.timestamp));
    //Display recipes under unique ID
    holder.itemView.setOnClickListener((v) ->{
        Intent intent = new Intent(context,RecipeDetailsActivity.class);
        intent.putExtra( name: "name",recipe.name);
        intent.putExtra( name: "time",recipe.time);
        intent.putExtra( name: "ingredients",recipe.ingredients);
        intent.putExtra( name: "procedures",recipe.procedures);

        String recipeId = this.getSnapshots().getSnapshot(position).getId();
        intent.putExtra( name: "recipeId",recipeId);
        context.startActivity(intent);
    });
}

//Pass the recipe data from the view holder
@NonNull
@Override
public RecipeCardHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.my_recipe_card,parent, attachToRoot: false);
    return new RecipeCardHolder(view);
}

//Hold the recipe view
class RecipeCardHolder extends RecyclerView.ViewHolder{

    TextView recipeName;
    TextView timeStamp;
```

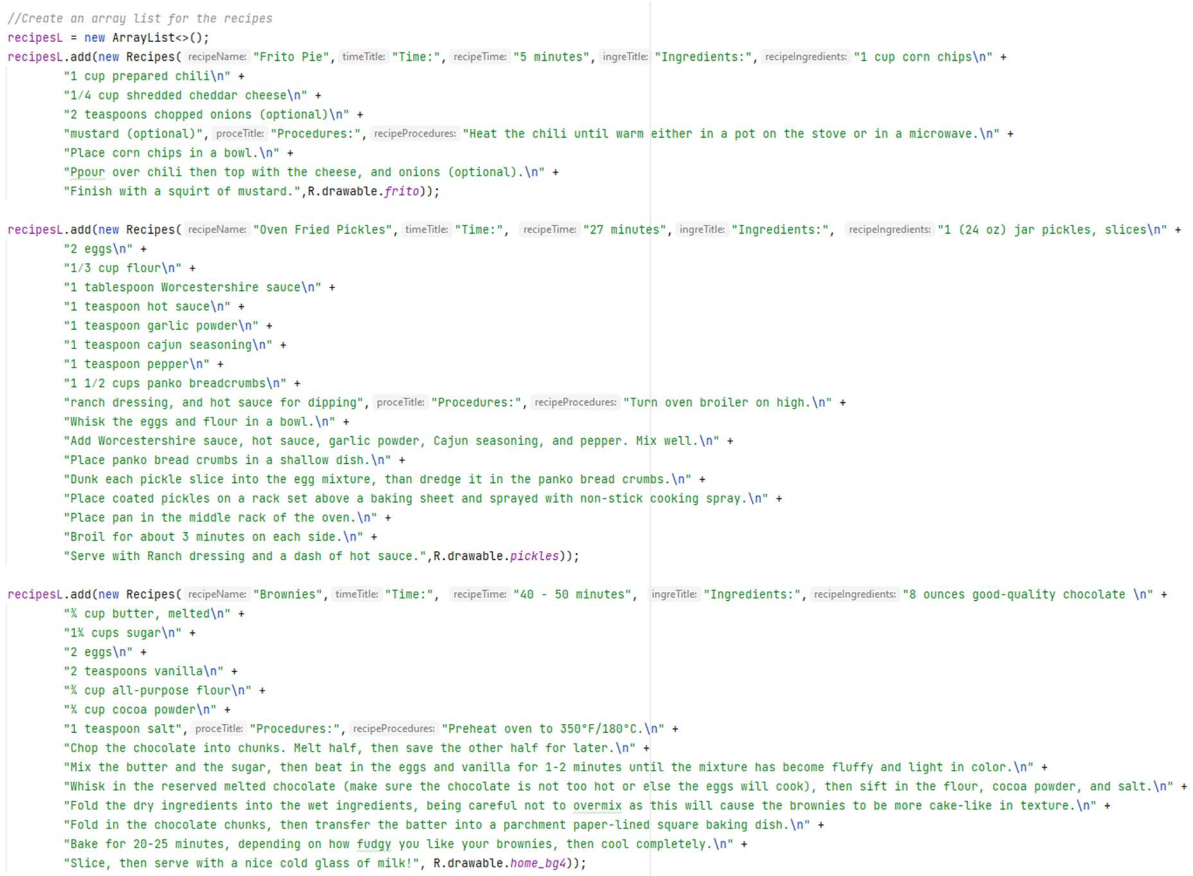
*Figure 20 My Recipe Adapter Code*



```java
//Retrieve the recipe data
void saveRecipe(){
    String recipeName = nameTxt.getText().toString();
    String recipeTime = timeTxt.getText().toString();
    String ingred = ingredientsTxt.getText().toString();
    String proce = proceduresTxt.getText().toString();

    //Validate the recipe
    if(recipeName==null || recipeName.isEmpty()){
        nameTxt.setError("Recipe Name is required!");
        return;
    }

    //Add to model class
    Recipe recipe = new Recipe();
    recipe.setName(recipeName);
    recipe.setTime(recipeTime);
    recipe.setIngredients(ingred);
    recipe.setProcedures(proce);
    recipe.setTimestamp(Timestamp.now());

    saveRecipeToFirebase(recipe);
}

//Save the recipes created to the Firebase Database
void saveRecipeToFirebase(Recipe recipe){
    //Save the recipes as a document in the collection
    DocumentReference documentReference;
    //Update the recipe
    if(editRecipe){
        //If the recipe is edited, update the recipe
        documentReference = RecipeDatabase.getCollectionReferenceForRecipes().document(recipeId);
    }else{
        documentReference = RecipeDatabase.getCollectionReferenceForRecipes().document();
    }

    documentReference.set(recipe).addOnCompleteListener(task -> {
        if(task.isSuccessful()){
            //Recipe is added in the Firebase
            Toast.makeText( context: RecipeDetailsActivity.this,  text: "New recipe added successfully!", Toast.LENGTH_SHORT).show();
            finish();
        }else{
            Toast.makeText( context: RecipeDetailsActivity.this,  text: "Failed to add new recipe!", Toast.LENGTH_SHORT).show();
        }
    });
```

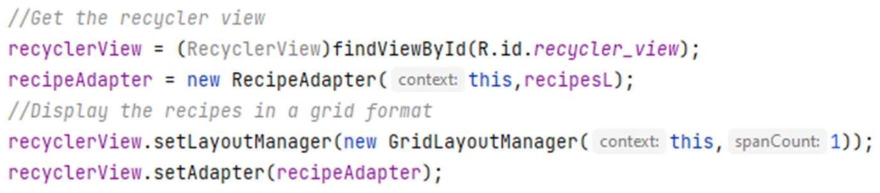
*Figure 21 My Recipe Details Code*

This implementation shows the array list of recipes to display in the main Explore Page (Explore GUI). One of the application objectives is to utilize the recipes from Spoonacular API. Unfortunately, the implementation did not meet the requirements. The reason for this purpose was due to the crashing of my computer device and the error I had encountered. Consequently, I have created my recipe list to display in the application. The method is similar to My Recipe containing the class java (see Figure 23) and the adapter (see Figure 24). The recipes are displayed in a recycler view that reuses the code implementation (see Figure 22).



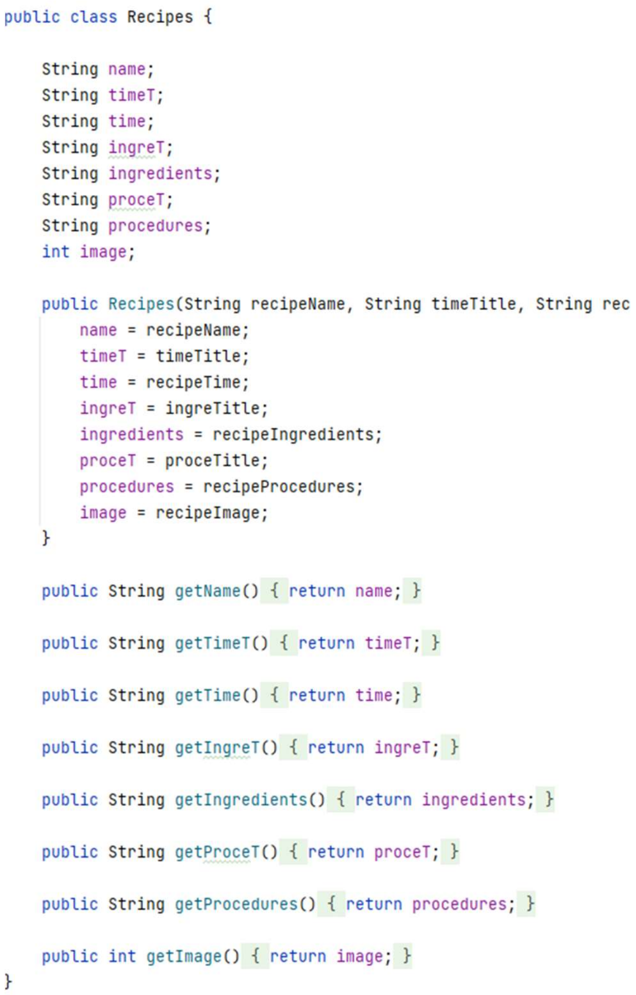
```
//Create an array list for the recipes
recipesL = new ArrayList<>();
recipesL.add(new Recipes( recipeName: "Frito Pie", timeTitle: "Time:", recipeTime: "5 minutes", ingreTitle: "Ingredients:", recipeIngredients: "1 cup corn chips\n" +
        "1 cup prepared chili\n" +
        "1/4 cup shredded cheddar cheese\n" +
        "2 teaspoons chopped onions (optional)\n" +
        "mustard (optional)", proceTitle: "Procedures:", recipeProcedures: "Heat the chili until warm either in a pot on the stove or in a microwave.\n" +
        "Place corn chips in a bowl.\n" +
        "Ppour over chili then top with the cheese, and onions (optional).\n" +
        "Finish with a squirt of mustard.",R.drawable.frito));

recipesL.add(new Recipes( recipeName: "Oven Fried Pickles", timeTitle: "Time:",  recipeTime: "27 minutes", ingreTitle: "Ingredients:",  recipeIngredients: "1 (24 oz) jar pickles, slices\n" +
        "2 eggs\n" +
        "1/3 cup flour\n" +
        "1 tablespoon Worcestershire sauce\n" +
        "1 teaspoon hot sauce\n" +
        "1 teaspoon garlic powder\n" +
        "1 teaspoon cajun seasoning\n" +
        "1 teaspoon pepper\n" +
        "1 1/2 cups panko breadcrumbs\n" +
        "ranch dressing, and hot sauce for dipping", proceTitle: "Procedures:", recipeProcedures: "Turn oven broiler on high.\n" +
        "Whisk the eggs and flour in a bowl.\n" +
        "Add Worcestershire sauce, hot sauce, garlic powder, Cajun seasoning, and pepper. Mix well.\n" +
        "Place panko bread crumbs in a shallow dish.\n" +
        "Dunk each pickle slice into the egg mixture, than dredge it in the panko bread crumbs.\n" +
        "Place coated pickles on a rack set above a baking sheet and sprayed with non-stick cooking spray.\n" +
        "Place pan in the middle rack of the oven.\n" +
        "Broil for about 3 minutes on each side.\n" +
        "Serve with Ranch dressing and a dash of hot sauce.",R.drawable.pickles));

recipesL.add(new Recipes( recipeName: "Brownies", timeTitle: "Time:",  recipeTime: "40 - 50 minutes",  ingreTitle: "Ingredients:", recipeIngredients: "8 ounces good-quality chocolate \n" +
        "¾ cup butter, melted\n" +
        "1¼ cups sugar\n" +
        "2 eggs\n" +
        "2 teaspoons vanilla\n" +
        "¾ cup all-purpose flour\n" +
        "¼ cup cocoa powder\n" +
        "1 teaspoon salt", proceTitle: "Procedures:", recipeProcedures: "Preheat oven to 350°F/180°C.\n" +
        "Chop the chocolate into chunks. Melt half, then save the other half for later.\n" +
        "Mix the butter and the sugar, then beat in the eggs and vanilla for 1-2 minutes until the mixture has become fluffy and light in color.\n" +
        "Whisk in the reserved melted chocolate (make sure the chocolate is not too hot or else the eggs will cook), then sift in the flour, cocoa powder, and salt.\n" +
        "Fold the dry ingredients into the wet ingredients, being careful not to overmix as this will cause the brownies to be more cake-like in texture.\n" +
        "Fold in the chocolate chunks, then transfer the batter into a parchment paper-lined square baking dish.\n" +
        "Bake for 20-25 minutes, depending on how fudgy you like your brownies, then cool completely.\n" +
        "Slice, then serve with a nice cold glass of milk!", R.drawable.home_bg4));
```

*Figure 22 Explore Main Code*



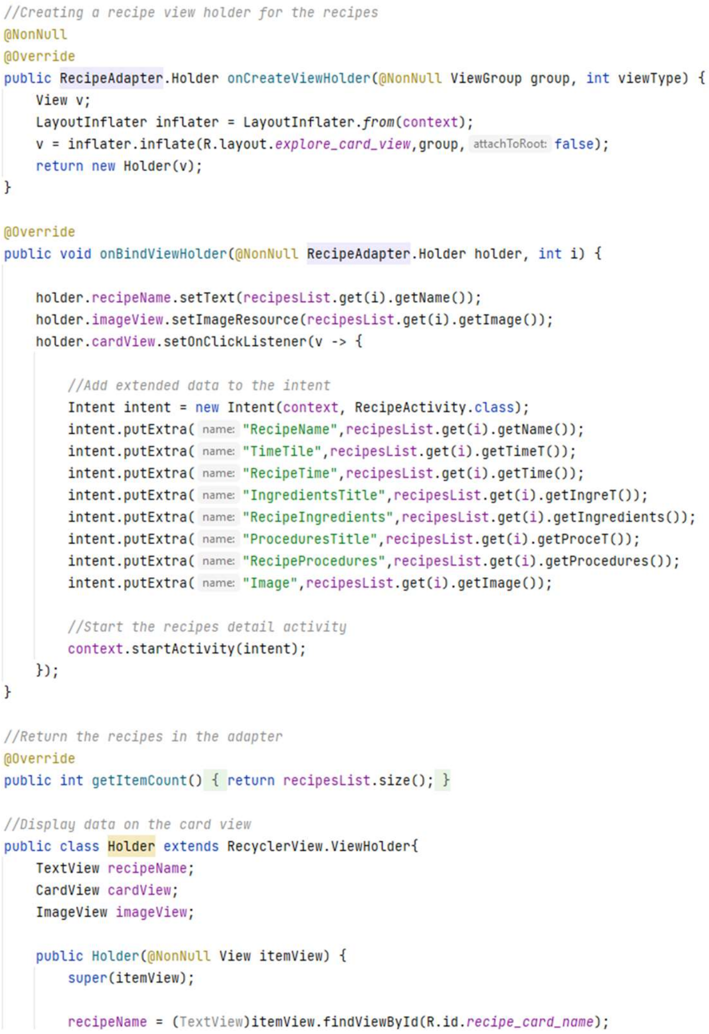
```
//Get the recycler view
recyclerView = (RecyclerView)findViewById(R.id.recycler_view);
recipeAdapter = new RecipeAdapter( context: this,recipesL);
//Display the recipes in a grid format
recyclerView.setLayoutManager(new GridLayoutManager( context: this, spanCount: 1));
recyclerView.setAdapter(recipeAdapter);
```

*Figure 23 Recipe RecyclerView Code*



```
public class Recipes {

    String name;
    String timeT;
    String time;
    String ingreT;
    String ingredients;
    String proceT;
    String procedures;
    int image;

    public Recipes(String recipeName, String timeTitle, String rec
        name = recipeName;
        timeT = timeTitle;
        time = recipeTime;
        ingreT = ingreTitle;
        ingredients = recipeIngredients;
        proceT = proceTitle;
        procedures = recipeProcedures;
        image = recipeImage;
    }

    public String getName() { return name; }

    public String getTimeT() { return timeT; }

    public String getTime() { return time; }

    public String getIngreT() { return ingreT; }

    public String getIngredients() { return ingredients; }

    public String getProceT() { return proceT; }

    public String getProcedures() { return procedures; }

    public int getImage() { return image; }
}
```

*Figure 24 Recipes Class Code*

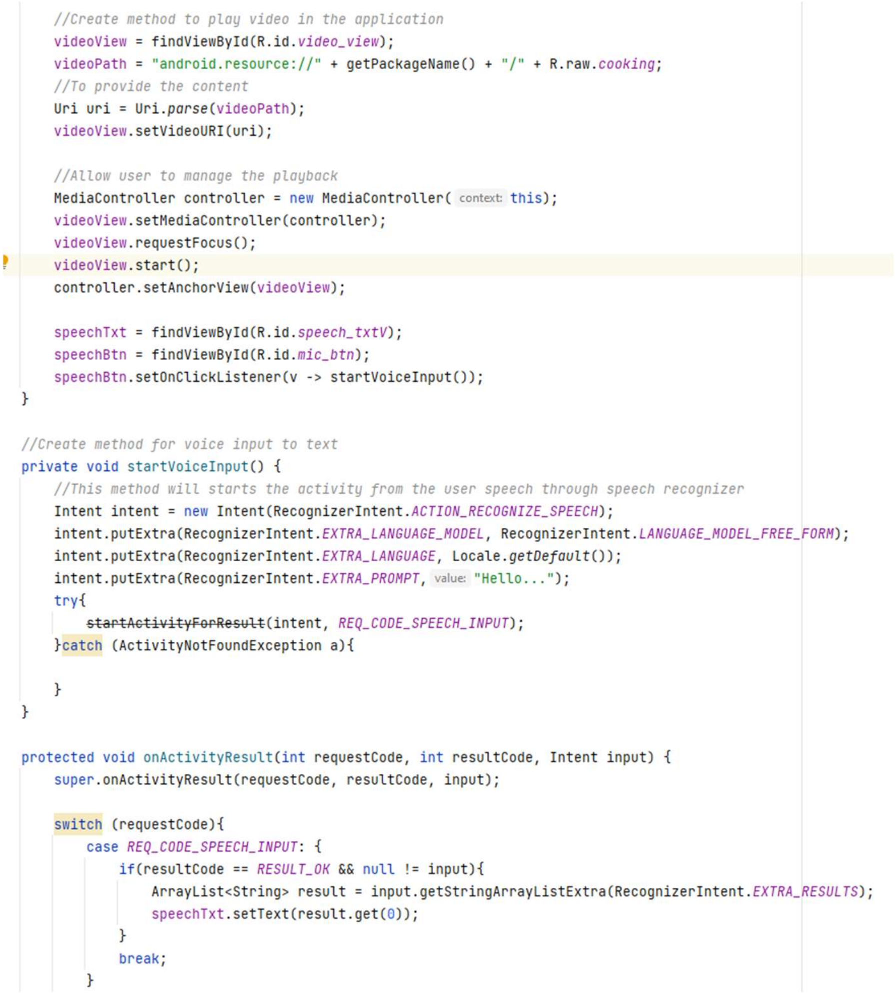


```
//Creating a recipe view holder for the recipes
@NonNull
@Override
public RecipeAdapter.Holder onCreateViewHolder(@NonNull ViewGroup group, int viewType) {
    View v;
    LayoutInflater inflater = LayoutInflater.from(context);
    v = inflater.inflate(R.layout.explore_card_view,group, attachToRoot: false);
    return new Holder(v);
}

@Override
public void onBindViewHolder(@NonNull RecipeAdapter.Holder holder, int i) {

    holder.recipeName.setText(recipesList.get(i).getName());
    holder.imageView.setImageResource(recipesList.get(i).getImage());
    holder.cardView.setOnClickListener(v -> {

        //Add extended data to the intent
        Intent intent = new Intent(context, RecipeActivity.class);
        intent.putExtra( name: "RecipeName",recipesList.get(i).getName());
        intent.putExtra( name: "TimeTile",recipesList.get(i).getTimeT());
        intent.putExtra( name: "RecipeTime",recipesList.get(i).getTime());
        intent.putExtra( name: "IngredientsTitle",recipesList.get(i).getIngreT());
        intent.putExtra( name: "RecipeIngredients",recipesList.get(i).getIngredients());
        intent.putExtra( name: "ProceduresTitle",recipesList.get(i).getProceT());
        intent.putExtra( name: "RecipeProcedures",recipesList.get(i).getProcedures());
        intent.putExtra( name: "Image",recipesList.get(i).getImage());

        //Start the recipes detail activity
        context.startActivity(intent);
    });
}

//Return the recipes in the adapter
@Override
public int getItemCount() { return recipesList.size(); }

//Display data on the card view
public class Holder extends RecyclerView.ViewHolder{
    TextView recipeName;
    CardView cardView;
    ImageView imageView;

    public Holder(@NonNull View itemView) {
        super(itemView);

        recipeName = (TextView)itemView.findViewById(R.id.recipe_card_name);
```

*Figure 25 Recipe Adapter*

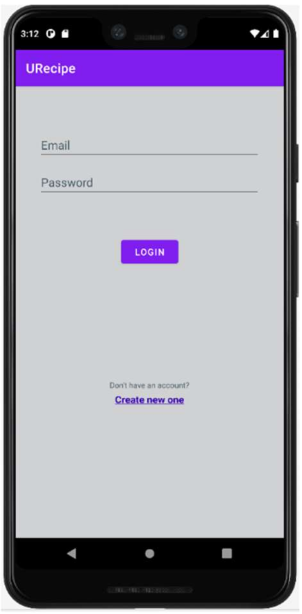
This Implementation of code is for Video Players and Voice Speech. The video of cooking is displayed on the page using a video view to hold the video and media controller to give user access such as play, pause, fast-forward, and rewind.

The implementation below is the voice speech. The code indicates that the user's input will be converted to text with a recognizer method that will open the recipe activity if the user presses the speech button and follows the direction.



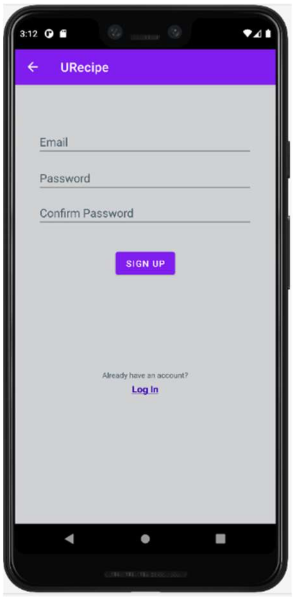
```
    //Create method to play video in the application
    videoView = findViewById(R.id.video_view);
    videoPath = "android.resource://" + getPackageName() + "/" + R.raw.cooking;
    //To provide the content
    Uri uri = Uri.parse(videoPath);
    videoView.setVideoURI(uri);

    //Allow user to manage the playback
    MediaController controller = new MediaController( context: this);
    videoView.setMediaController(controller);
    videoView.requestFocus();
    videoView.start();
    controller.setAnchorView(videoView);

    speechTxt = findViewById(R.id.speech_txtV);
    speechBtn = findViewById(R.id.mic_btn);
    speechBtn.setOnClickListener(v -> startVoiceInput());
}

//Create method for voice input to text
private void startVoiceInput() {
    //This method will starts the activity from the user speech through speech recognizer
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
    intent.putExtra(RecognizerIntent.EXTRA_PROMPT, value: "Hello...");
    try{
        ~~startActivityForResult~~(intent, REQ_CODE_SPEECH_INPUT);
    }catch (ActivityNotFoundException a){

    }
}

protected void onActivityResult(int requestCode, int resultCode, Intent input) {
    super.onActivityResult(requestCode, resultCode, input);

    switch (requestCode){
        case REQ_CODE_SPEECH_INPUT: {
            if(resultCode == RESULT_OK && null != input){
                ArrayList<String> result = input.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
                speechTxt.setText(result.get(0));
            }
            break;
        }
```

## 2.4. Graphical User Interface (GUI)



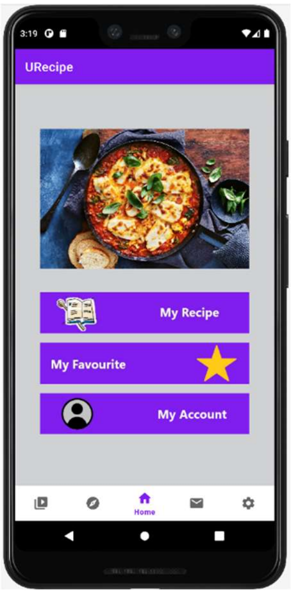
**Login GUI**

User can enter their email and password to log in once they created an account. If the user does not have an account, they create one below login credentials.



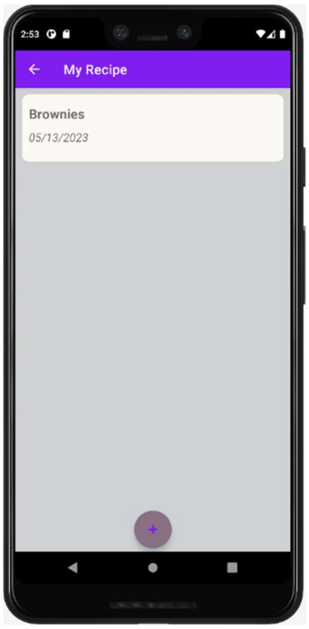
**Registration GUI**

Users must enter their email, password and confirm their password to complete the account registration. If the user already has an account, they can log in below the registration credentials.



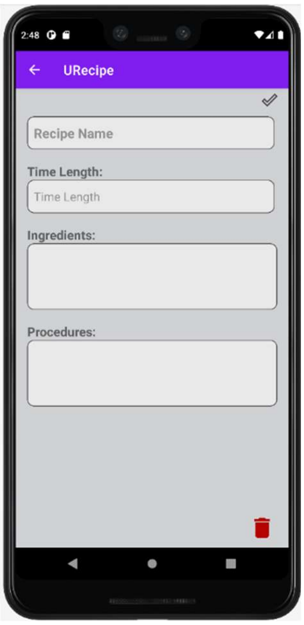
**Homepage GUI**

This is the main screen of the application. It displays three main options for users to select and bottom navigation with various selections and a slideshow of various dishes.



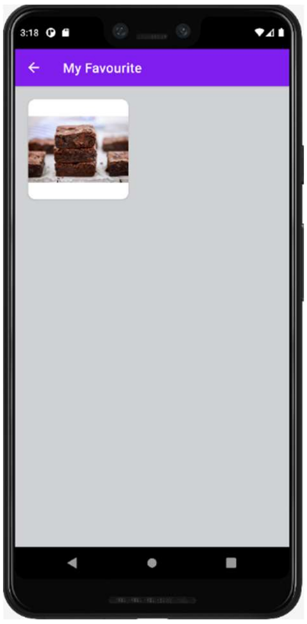
**My Recipe GUI**

User can create their recipe by clicking the add button on the screen. The card view above is



**Create Recipe GUI**

The user must input the recipe name to save the recipe. Users can

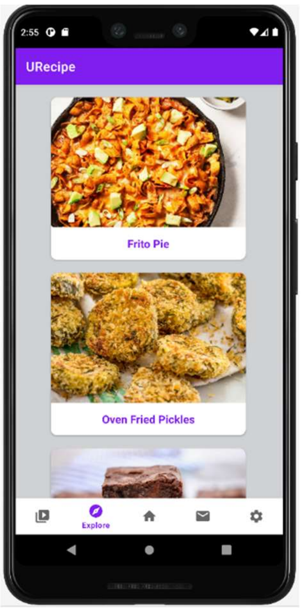


**My Favourite GUI**

When user saved their favourite recipe, they will be stored on this page. They how the new recipe will show once they are added.

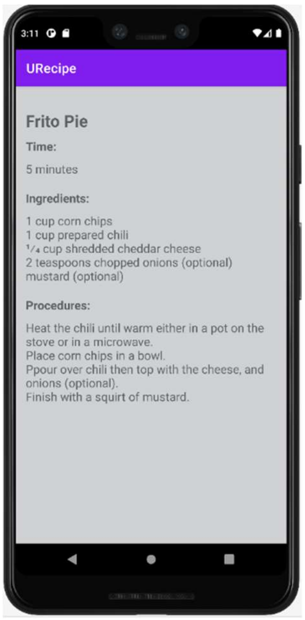
fill in the time, ingredients, and procedures for their recipe.

can also remove them with a remove button.



**Explore GUI**

Users can explore the recipes on this page. They can search, select, save, and rate each recipe.



Frito Pie

**Time:**

5 minutes

**Ingredients:**

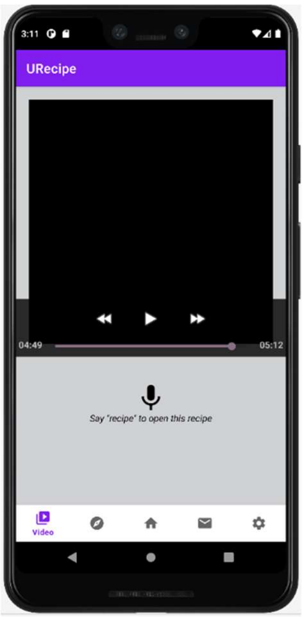
1 cup corn chips
1 cup prepared chili
¹/₄ cup shredded cheddar cheese
2 teaspoons chopped onions (optional)
mustard (optional)

**Procedures:**

Heat the chili until warm either in a pot on the stove or in a microwave.
Place corn chips in a bowl.
Ppour over chili then top with the cheese, and onions (optional).
Finish with a squirt of mustard.

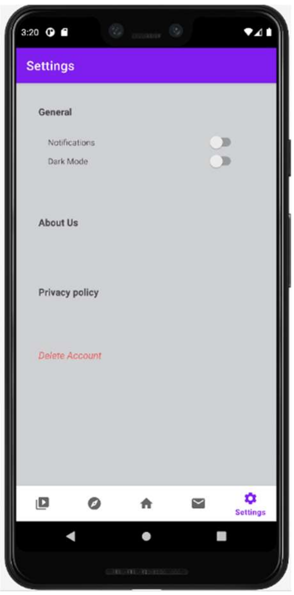
**Recipe GUI**

This activity displays the recipe when the user selects a dish from the Explore Page.



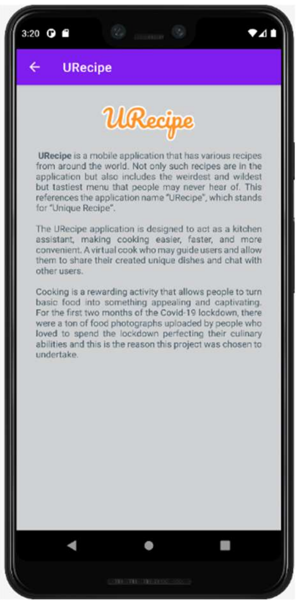
**Video GUI**

Users can control the video and use the speech button to open the recipe with their voice.



**Settings GUI**

This is the main settings page of the application. Users can select any options and delete their accounts in the settings.



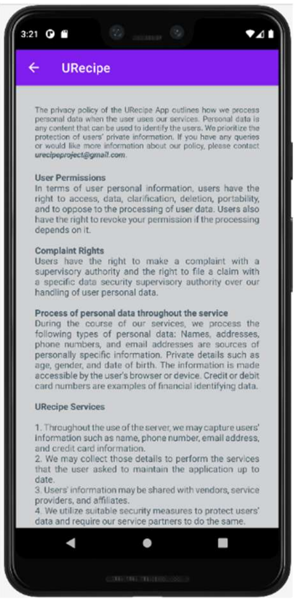
**URecipe** is a mobile application that has various recipes from around the world. Not only such recipes are in the application but also includes the weirdest and wildest but tastiest menu that people may never hear of. This references the application name "URecipe", which stands for "Unique Recipe".

The URecipe application is designed to act as a kitchen assistant, making cooking easier, faster, and more convenient. A virtual cook who may guide users and allow them to share their created unique dishes and chat with other users.

Cooking is a rewarding activity that allows people to turn basic food into something appealing and captivating. For the first two months of the Covid-19 lockdown, there were a ton of food photographs uploaded by people who loved to spend the lockdown perfecting their culinary abilities and this is the reason this project was chosen to undertake.

**About Us GUI**

When the user clicks the "About Us" button in the settings it will display the description and background of the URecipe App.



The privacy policy of the URecipe App outlines how we process personal data when the user uses our services. Personal data is any content that can be used to identify the users. We prioritize the protection of users' private information. If you have any queries or would like more information about our policy, please contact *urecipeproject@gmail.com*.

**User Permissions**
In terms of user personal information, users have the right to access, data, clarification, deletion, portability, and to oppose to the processing of user data. Users also have the right to revoke your permission if the processing depends on it.

**Complaint Rights**
Users have the right to make a complaint with a supervisory authority and the right to file a claim with a specific data security supervisory authority over our handling of user personal data.

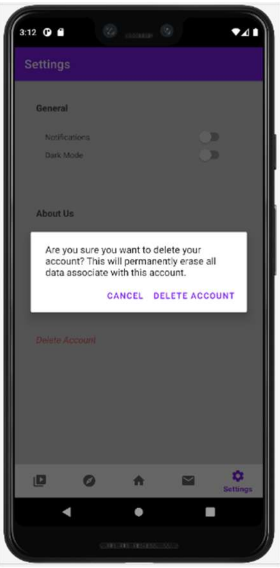
**Process of personal data throughout the service**
During the course of our services, we process the following types of personal data: Names, addresses, phone numbers, and email addresses are sources of personally specific information. Private details such as age, gender, and date of birth. The information is made accessible by the user's browser or device. Credit or debit card numbers are examples of financial identifying data.

**URecipe Services**

1. Throughout the use of the server, we may capture users' Information such as name, phone number, email address, and credit card information.
2. We may collect those details to perform the services that the user asked to maintain the application up to date.
3. Users' information may be shared with vendors, service providers, and affiliates.
4. We utilize suitable security measures to protect users' data and require our service partners to do the same.

**Privacy Policy GUI**

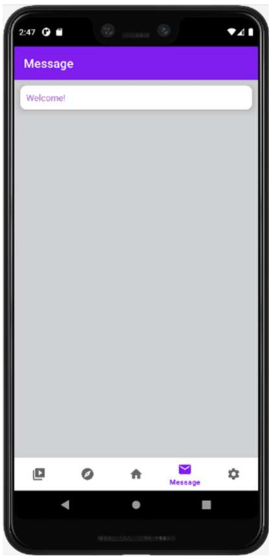
When the clicks the "Privacy Policy" button in the settings it will display the policy that the user wanted to know within the application.



Are you sure you want to delete your account? This will permanently erase all data associate with this account.

**Delete Account GUI**

This dialog is shown when the user clicked the “Delete Account” option in the settings.

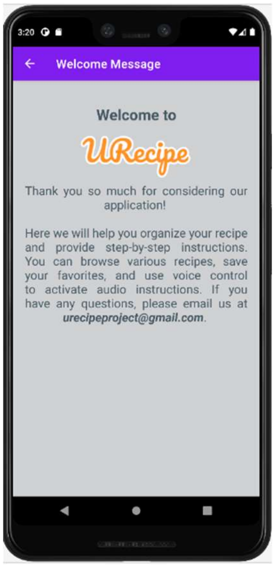


Welcome!

**Message GUI**

This page displays messages from the application.

Once the user first logs in this message will appear.



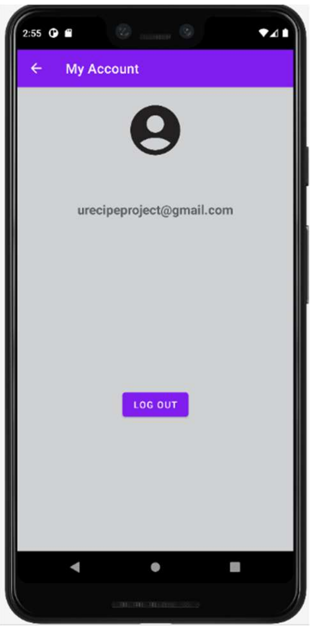
Welcome to

Thank you so much for considering our application!

Here we will help you organize your recipe and provide step-by-step instructions. You can browse various recipes, save your favorites, and use voice control to activate audio instructions. If you have any questions, please email us at ***urecipeproject@gmail.com***.

**Welcome Message GUI**

Every new user will get a welcome message from the application.



urecipeproject@gmail.com

**My Account**

Once the user creates an account their details are displayed on this page.

## 2.5. Testing

Android Studio Monkey Testing was used for testing the application on the user's end. Monkey Testing provides code to be implemented once the project is completed. With a USB wire connection between the Window device and the mobile phone. The application can go thorough and active on mobile devices. Firstly, the testing began with registration and examining whether the details are saved in Firebase through the user's device. After succeeding with the connection between Android Studio and a test device, the following functions such as Login, Recipe Creation, Favourite Recipes, and Voice Speech were operated.

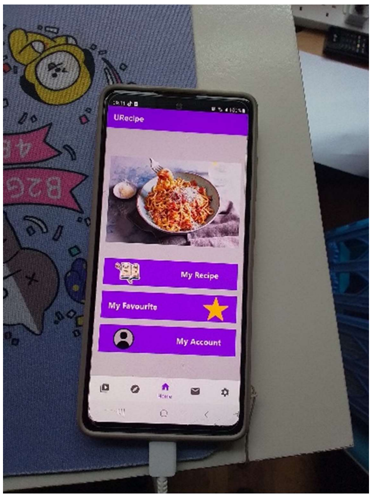
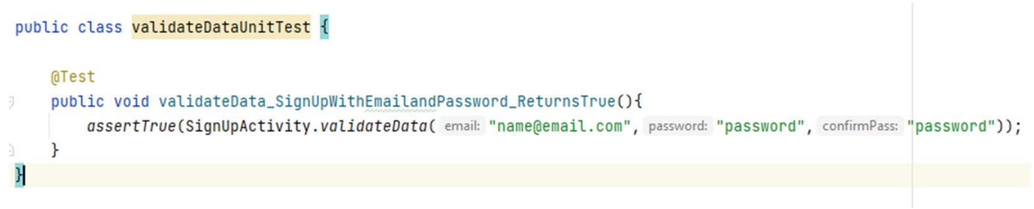


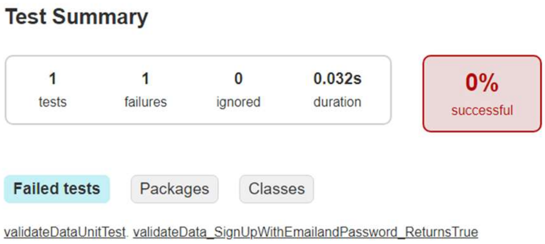
*Figure 26 USB Testing on Device*

The Unit Test was difficult to test the application as it relied extensively on external service (Firebase). I have tried to do a Unit Test with the registration data validation and unfortunately, it failed.



```
public class validateDataUnitTest {

    @Test
    public void validateData_SignUpWithEmailandPassword_ReturnsTrue(){
        assertTrue(SignUpActivity.validateData( email: "name@email.com", password: "password", confirmPass: "password"));
    }
}
```

*Figure 27 Unit Test Validation*



Test Summary

```
1        1          0         0.032s       0%
tests    failures   ignored   duration     successful
```

Failed tests | Packages | Classes

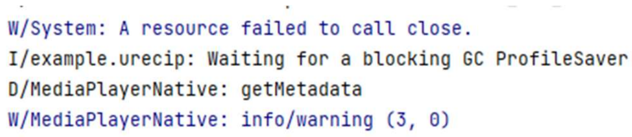
validateDataUnitTest. validateData_SignUpWithEmailandPassword_ReturnsTrue

*Figure 28 Unit Test Failure*

As for the emulator, all the graphical interfaces were well displayed including the action bar, images, texts, and buttons. Users can only access the main page of the application after they registered their account and if the user logged out from the application, they will be shown with the opening screen. Without login details, they cannot access the application. And once they delete their account, it will be removed from Firebase as well.

At first, I encountered the rendering problem after I made changes with ‘AndroidManifest’. I solved the issue by changing the theme of the application.

The video player and voice speech were the only tests that failed. The video played the first time it was tested, however, there was an error on the last day of testing. It was intractable.



```
W/System: A resource failed to call close.
I/example.urecip: Waiting for a blocking GC ProfileSaver
D/MediaPlayerNative: getMetadata
W/MediaPlayerNative: info/warning (3, 0)
```

*Figure 29 Video Player Failure*

The voiced speech did not go according to plan because there was no pop-up mic to allow the user to input their voice. The button was clickable, yet there were no actions in the emulator. Despite what the emulator indicates, the functionalities worked when performed on the real device including the voice speech.

## 2.6. Evaluation

The application was evaluated through debugging. While running the application on the emulator I noticed in the logcat that the enabling was late, and some files cannot be accessed such as raw video. In terms of performance evaluation, I noticed that when I opened the application from the device, it flashed, indicating a glitch problem, which was quite unexpected. When the buttons were clicked, they were slightly delayed.  The slideshow on the main page performs perfectly with smoothness and time. I also performed the connection between the application and the Firebase API. The data was sent rapidly without any issues. The performance between Android Studio and the test device was similarly excellent, and this included all the application's features. The only aspects that were unsettling were the video player and the explore page. The video was playing perfectly on the device; however, the emulator displayed a black screen. Unlike on the emulator, the explore page margins were off-screen. Overall, the performance was satisfactory, and it was gratifying to see the project you created running on your device.

I tried running the application with several emulator sizes for scalability testing, and it worked completely well for each device, even if the render while creating was different. Because of all the resources, the application performance was gradually delayed as more code was implemented.

Checking over the Proposal requirements, I believe I exceeded some of them, particularly the primary section of the application. However, given the proper utilization and resources, there could be additional gains based on performance.

# 3.0 Conclusions

Throughout the project's development, I came across several amazing sources of recipes that I am unable to access owing to the payment plans that some of the websites included. My objective was to use Spoonacular API as a source for my recipe. According to my research, Spoonacular uses standardized recipes, which makes it simpler and easier to search using food-related keywords. I failed to create it even though it was the sole basic source. If only I had anticipated the outcome of my computing device's crash, I would have continued to experiment with Spoonacular. Instead, I created an array list of recipes from the website (Food, n.d.).

The benefit I gained from this project was the ability to understand more in-depth the tools I have always used. I discovered many methods for implementing and utilizing Android Studio features.

Useful project templates which were ready for use for developers to save time. Android Studio speeds up the implementation of code and testing. Android Studio also has Firebase compatibility as well as integrated clouds. This is a huge benefit for project developers to add features from Firebase to the application and to store data in their database.

The disadvantages of this project were the high hardware and installation. Since Android Studio requires a lot of RAM and CPU to execute, the development process was slowed considerably due to the old version of my computing hardware. The installation process took longer than coding, which was inconvenient during the process.

Despite the ups and downs of the project, I have gained knowledge of how to debug the code more effectively and conquer the errors I have encountered. Being unable to accomplish all the requirements is not considered a failure. I will always analyze all the aspects of this project and apply the understanding to any future initiatives.

# 4.0 Further Development or Research

In terms of future advancements, if I had more time and resources, I would try to recreate the requirements that I neglected to complete, particularly the Voice Speech activity. The purpose of this study was to create a voice interaction that might help users cook. It is a fantastic opportunity to create a high-quality AI voice for recipe reading. It would be far more convenient if the user could listen to AI read the recipe ingredients and instructions while cooking. If there is an opportunity, I would like to learn more about the subject and complete my project.

# 5.0 References

API, s., n.d. s.l.: spoonacular API.

Bondarenko, A., n.d. *How to Build a Cooking or Recipe app?,* s.l.: Stormotion.

Demos, C., 2018. *Android Firebase - How to Delete a User Account Programmatically From Firebase (Explained),* s.l.: YouTube.

Firebase, 2023. *Manage Users in Firebase,* s.l.: Firebase.

Flow, C. i., 2017. *VideoView - Android Studio Tutorial,* s.l.: YouTube.

Food, n.d. *16 WEIRD & WACKY REGIONAL RECIPES,* s.l.: Food.

How To Develop A Cooking, F. R. M. A. C. &. F., 2021. *How To Develop A Cooking, Food Recipe Mobile App: Cost & Features,* s.l.: emizentech.

Navneet, 2023. *Convert Speech to Text in Android Application,* s.l.: StackTips.

Nilkanth, D., 2019. *Android Studio - Build A Simple Android Recipe Application,* s.l.: YouTube.

Sushko, D., 2018. *How to Add Voice to Your Mobile App,* s.l.: Clutch.

Tuto, E., 2023. *Notes App With Firebase | Android | 2023,* s.l.: YouTube.

# 6.0 Appendices

## 6.1.Project Proposal

### 6.1.1 Objectives

“**URecipe**” is shortened from “Unique Recipe”, it is a mobile application that features many recipes from around the globe and contains the weirdest and wildest but tastiest menu that people may never hear of. It is a one-of-a-kind recipe that people might want to attempt to improve and go beyond their culinary skills.

Searching can take a long time to find the perfect recipe on the internet, and it is unreliable to rewrite and bookmark the recipe we found. The URecipe application here is set out to play as an assistant in the kitchen that will make cooking simpler, faster, and more convenient. A virtual cook that can provide users with guidance, and the opportunity to share their homemade unique recipes and communicate with other users.

Cooking/Recipe application development has the potential to benefit many people. Everyone who cooks, whether they are men, women, adults, elderly, families, or children, can benefit from having specialized recipe software on their mobile phones. This application can help user to arrange their recipe along with step-by-step cooking instructions. Users can browse various recipes, save their favourite, and watch instructional videos available in the app. There will be a wide range of selections, including vegan, vegetarian, dessert, and traditional food.

### 6.1.2 Background

Cooking is a rewarding hobby that enables people to transform an ordinary dish into something enticing and captivating. For the first two months of the Covid-19 lockdown, there were a ton of food images uploaded by people who liked to spend the lockdown honing their culinary abilities and this is the reason why this project was chosen to undertake. The number of apps downloaded each day is increasing dramatically and these apps are becoming more widely used and are technology friendly.

As mentioned above, the goals for making their cooking easier are as follows: A video recipe will be available to users, which may help them understand the menu better than the written recipe. The application will have a camera feature that allows user to upload their videos or photos along with their recipes. Users can save their favourite menus or recipes for future reference. They can search and view all the recipes within the app and comment and rate the recipe owner. If the user has any questions about the recipe, they can leave them in the query box. URecipe App will provide simple step-by-step instructions in a distinguishable way.

### 6.1.3 State of Art

This idea was developed from many existing apps such as Yummly, Tasty, and SuperFood. Yummly is a clever culinary software made for cooking lovers. It provides fascinating video tutorials and practical tools for grocery shopping and gathering recipes. The Tasty application has 4000 recipes available with innovative step-by-step instructions like Yummly. Both applications have almost the same features. SuperFood mainly focused on healthy food and offers a variety of nutritious and quick-to-prepare meals.

This information inspired me to create a mobile application using unique recipes that many people might not be familiar with. The craziest but most delicious recipes are fascinating to be implanted in an application. It is fascinating to include the most bizarre but delectable recipes in an application. Although my project and other comparable applications may share some fundamental characteristics, I believe a Voice Control feature will make my application stand out from them. People might find it irritating to manually pause or rewind the video with their hands during their cooking. I intended to develop a voice control system for videos so that users could give voice speeches to the video. There will also be a recipe voice instruction for people who struggle to see or read. These features will make people more comfortable and enjoy their time spent cooking.

### 6.1.4 Technical Approach

Since I will be creating a recipe application, these precious details on how to combine different ingredients into something delicious play an important role in me. The first approach I will be taking is integrating already existing recipe APIs. The quickest approach to constructing a recipe application is to use pre-made databases, which contain hundreds of recipes and ingredients. This, in my opinion, will serve as a solid foundation from the outset. The database for the application can also be populated with uploaded user recipes. With both approaches, the database can expand to include new users' recipes and their adaptations of existing ones.

The second approach I will be taking is a Voice Controller. Voice-activated technology is popular among consumers in a variety of settings. Two common ways of interacting with mobile phones were put to the test by Stanford University researchers: speech recognition and touchscreen typing. In both instances, speech recognition algorithms processed voice input more quickly than individuals typing. This functionality can be added to recipe apps so that users can directly ask questions of voice assistance when they are cooking without using their hands.

### 6.1.5 Technical Details

The implementation language I will be using is mostly Java. Other languages that I might come across are C++, JSON, AJAX, and Node.js. I will also be using Firebase SDK to store user details and push notifications. Firebase is an application development that offers integrated technologies to support the creation, expansion, and monetization of your apps. The Firebase SDK provides idiomatic and simple accessibility to the Firebase services across several platforms.

The most important features and technical requirements are:

- User registration and account management
- Advanced Recipes Search and view recipe videos
- Customized recipes
- Upload photos and videos, and rate & review
- Push notifications
- Security and Usability

People often only have two alternatives when using a culinary app while preparing meals: wash their hands whenever they need to contact the screen or smear the devices with their hands. However, the consumers may relax thanks to a voice controller mode. They may browse the recipes they're following and focus on the procedure itself by simply repeating "next step" or "rewind". The tasks to be taken to implement this approach are:

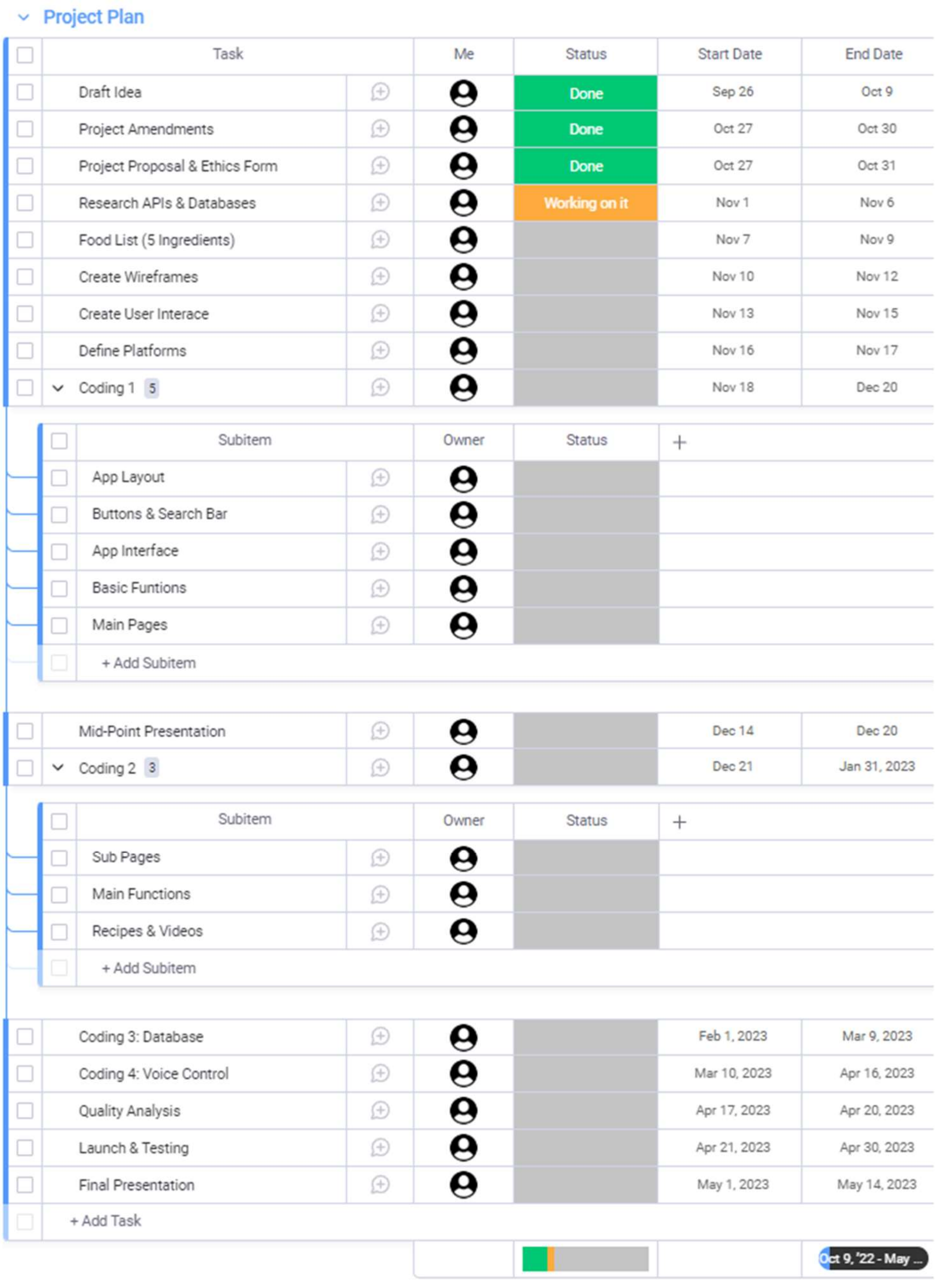
- Configure audio access rights.
- Design a user interface component that serves as a microphone trigger.
- Build up the Speech Recognizer instances from scratch.
- Develop a list of phrases that are supported to use as the app's voice-activated domain.
- Conduct app tasks by the user's voice speech.

### 6.1.6 Special Resources Required

Creating a recipe application is a difficult, time-consuming, and expensive task. API is a collection of specific steps, functions, and constants for interacting with another website or application. My recipe application will retrieve a source of different types of recipes utilizing the Spoonacular API. Spoonacular offers more than 2,600 recipes with numerous data, 5,000 recipes with shoppable content, 600K products along with descriptions, and 115k menu items from various restaurants. It also performs an automated analysis of recipes to look for common allergies and calculate the nutritious data for recipes using a unique technology.

Some recipes that are not stored in the API may later be implemented in the application. However, building and maintaining my database will take significant time and resources.

### 6.1.7 Project Plan



Project Plan

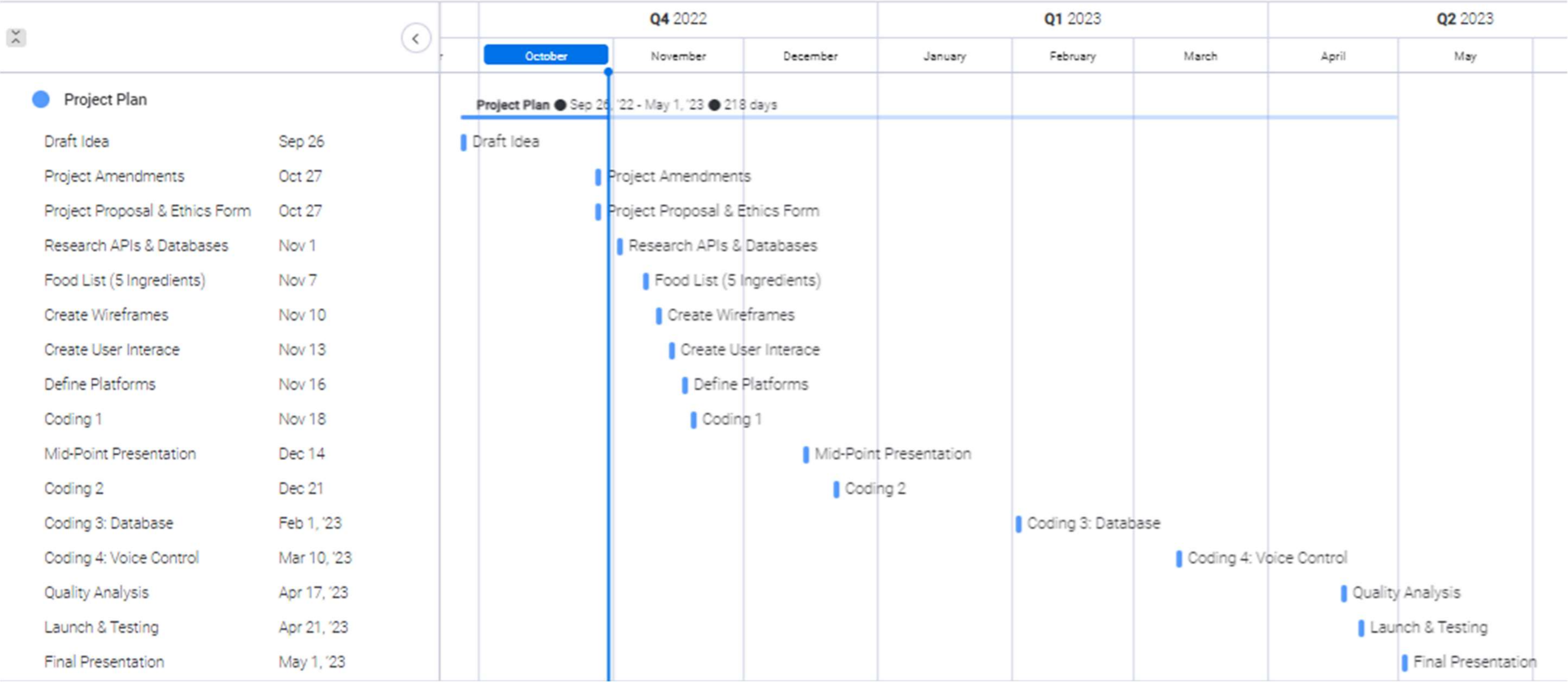
| □ | Task | Me | Status | Start Date | End Date |
|---|---|---|---|---|---|
| □ | Draft Idea | ● | Done | Sep 26 | Oct 9 |
| □ | Project Amendments | ● | Done | Oct 27 | Oct 30 |
| □ | Project Proposal & Ethics Form | ● | Done | Oct 27 | Oct 31 |
| □ | Research APIs & Databases | ● | Working on it | Nov 1 | Nov 6 |
| □ | Food List (5 Ingredients) | ● | | Nov 7 | Nov 9 |
| □ | Create Wireframes | ● | | Nov 10 | Nov 12 |
| □ | Create User Interace | ● | | Nov 13 | Nov 15 |
| □ | Define Platforms | ● | | Nov 16 | Nov 17 |
| □ | Coding 1 5 | ● | | Nov 18 | Dec 20 |

| □ | Subitem | Owner | Status | + |
|---|---|---|---|---|
| □ | App Layout | ● | | |
| □ | Buttons & Search Bar | ● | | |
| □ | App Interface | ● | | |
| □ | Basic Funtions | ● | | |
| □ | Main Pages | ● | | |
| | + Add Subitem | | | |

| □ | Task | Me | Status | Start Date | End Date |
|---|---|---|---|---|---|
| □ | Mid-Point Presentation | ● | | Dec 14 | Dec 20 |
| □ | Coding 2 3 | ● | | Dec 21 | Jan 31, 2023 |

| □ | Subitem | Owner | Status | + |
|---|---|---|---|---|
| □ | Sub Pages | ● | | |
| □ | Main Functions | ● | | |
| □ | Recipes & Videos | ● | | |
| | + Add Subitem | | | |

| □ | Task | Me | Status | Start Date | End Date |
|---|---|---|---|---|---|
| □ | Coding 3: Database | ● | | Feb 1, 2023 | Mar 9, 2023 |
| □ | Coding 4: Voice Control | ● | | Mar 10, 2023 | Apr 16, 2023 |
| □ | Quality Analysis | ● | | Apr 17, 2023 | Apr 20, 2023 |
| □ | Launch & Testing | ● | | Apr 21, 2023 | Apr 30, 2023 |
| □ | Final Presentation | ● | | May 1, 2023 | May 14, 2023 |
| | + Add Task | | | | |
| | | | 🟩🟧▨ | | Oct 9, '22 - May ... |



Project Plan

| | |
|---|---|
| Draft Idea | Sep 26 |
| Project Amendments | Oct 27 |
| Project Proposal & Ethics Form | Oct 27 |
| Research APIs & Databases | Nov 1 |
| Food List (5 Ingredients) | Nov 7 |
| Create Wireframes | Nov 10 |
| Create User Interace | Nov 13 |
| Define Platforms | Nov 16 |
| Coding 1 | Nov 18 |
| Mid-Point Presentation | Dec 14 |
| Coding 2 | Dec 21 |
| Coding 3: Database | Feb 1, '23 |
| Coding 4: Voice Control | Mar 10, '23 |
| Quality Analysis | Apr 17, '23 |
| Launch & Testing | Apr 21, '23 |
| Final Presentation | May 1, '23 |

### 6.1.8 Testing

**Testing 1:** Test the APIs created for the application. Send the request along with the required data and the API should run perfectly in the application.

**Testing 2:** I will test the application by creating a user account with a name, email address, and password. to check that the login information is correct, the Firebase platform will display a list of the user information in JSON format.

**Testing 3:** Select keywords and browse for a specific recipe. If the installation is successful, the menu should appear.

**Testing 4:** Post a comment and give a rating of the recipe. The application should add up user votes, save them, increase the rating bar, and send out user comments.

**Testing 5:** Video tutorials should be playing without any disruptions and sounds should be working if necessary.

**Testing 6:** Upload a recipe to test the functionality of this feature. Both the voice instructions and the recipes are legible.

**Testing 7:** To determine whether the strategy is effective or unsuccessful for the end user, I will work as the tester and give a voice speech to the application. Verify that all the recognition and connection are functional. If any more testing from the end user is required, I will submit an Ethic Application at a later stage.

**Testing 8:** Check the system to see if it is complying with the requirements. Then conduct a complete and integrated system.

**Testing 9:** Ensure the password security is effective. If the user's password follows the rule, it will allow it.

## 6.2 Reflective Journals

**Supervision & Reflection Template**

| | |
|---|---|
| **Student Name** | Sutthita Phromchomcha |
| **Student Number** | X19345621 |
| **Course** | BSH in Computing – Software Development |
| **Supervisor** | Lisa Murphy |

**Month:** November

**What**?
This month, I concentrated solely on wireframes and user interfaces. I implemented the application's basic functionalities, such as page activities and buttons. I was having difficulty with the bottom navigation bar because it is my first developing this function. It took me a long time to figure out how to implement this feature as I came across so many errors while debugging.

**So What?**
This function was very challenging for me as I have only been using the drawer navigation bar. The coding and binding of the bottom navigation were not working as I tried to run my application. The activities were not displaying as each page was clicked.

**Now What?**

| | |
|---|---|
| I searched for tutorials and decided to implement the function from scratch rather than using the default format. However, the coding might be longer than the default template. Another reason why the activities were not displaying was due to the names of each activity were incorrect in the 'AndroidManifest'. After implementing each activity, I must double-check the name of each activity before launching it. | |
| **Student Signature** | Sutthita Phromchomcha |

**Supervision & Reflection Template**

| | |
|---|---|
| **Student Name** | Sutthita Phromchomcha |
| **Student Number** | X19345621 |
| **Course** | BSH in Computing – Software Development |
| **Supervisor** | Lisa Murphy |

**Month:** October

| | |
|---|---|
| **What**?<br>This month, I brainstormed for my idea, a recipe mobile application, and filed a Project Pitch for it. I discussed with my supervisor after the project idea was "approved with revisions." The most important modification I had to make was to distinguish my app from other recipe applications. I devised the concept of culinary video reels, a suggestion box, and a recipe instructional voice. My supervisor advised that the third option was innovative and suggested using voice control for videos. | |
| **So What?**<br>Following our discussion, I followed my supervisor's advice and conducted additional research. I realized how these changes could make my application stands out from another recipe app. I began to search for more information, particularly about voice control for videos and discovered several recommendations on how to use them. The challenges are how to implement them and the recipe APIs I have to use in my application. | |
| **Now What?**<br>To overcome these challenges, I must conduct additional research on voice control, especially the coding aspect, and I need to keep in mind that implementing this function and recipe APIs will take a long time. | |
| **Student Signature** | Sutthita Phromchomcha |

**Supervision & Reflection Template**

| | |
|---|---|
| **Student Name** | Sutthita Phromchomcha |
| **Student Number** | X19345621 |
| **Course** | BSH in Computing – Software Development |
| **Supervisor** | Lisa Murphy |

**Month:** December

| |
|---|
| **What**?<br>This month I have been preparing for my mid-point presentation and demonstration video. I completed all the user interfaces to be displayed in the presentation and the basic functions for the application. |
| **So What?**<br>I was having difficulty with the screen recording of the demonstration and the editing app. So, I used a different screen recording but with a standard quality. |
| **Now What?**<br>I regretted doing the video presentation on the due date and I rushed to the end with a standard quality. After this month, I will be focusing on the users' accounts and the Explore page. The main function of this application will be a voice speech |

| | |
|---|---|
| **Student Signature** | Sutthita Phromchomcha |

**Supervision & Reflection Template**

| | |
|---|---|
| **Student Name** | Sutthita Phromchomcha |
| **Student Number** | X19345621 |
| **Course** | BSH in Computing – Software Development |
| **Supervisor** | Lisa Murphy |

**Month:** January

| |
|---|
| **What**?<br>This month, I am finishing up with the user account data and the user recipes. As the user signed up for their account their data should be saved within the application along with their saved recipes. |
| **So What?**<br>This function was very challenging as I could not get the data to display on the user account page along with their recipes. The errors were shown preventing the emulator to work. |
| **Now What?**<br>I followed the tutorial again from step 1 without missing any procedure. Next, I will be working on the Explorer Page to store the recipes from Spoonacular. |

| | |
|---|---|
| **Student Signature** | Sutthita Phromchomcha |

**Supervision & Reflection Template**

| | |
|---|---|
| **Student Name** | Sutthita Phromchomcha |
| **Student Number** | X19345621 |
| **Course** | BSH in Computing – Software Development |
| **Supervisor** | Lisa Murphy |

**Month:** February

| |
|---|
| **What**?<br>This month, I have been working on the Explorer Page and the template to hold the recipes for viewing. I have some small changes to the application details and colours. However, I could not extract the recipes from Spoonacular. Another problem occurred during my project: the computer I worked on stopped working, and I could not restart it. |

| | |
|---|---|
| **So What?**<br>The solution I found for my computer issue was to get a new motherboard as soon as possible. So that I can continue with my work. | |
| **Now What?**<br>While my computer is being repaired, I must work on my project on another device and attempt to reconnect to GitHub again. Once I have my computer, I must transfer all the data back. | |
| **Student Signature** | Sutthita Phromchomcha |

**Supervision & Reflection Template**

| | |
|---|---|
| **Student Name** | Sutthita Phromchomcha |
| **Student Number** | X19345621 |
| **Course** | BSH in Computing – Software Development |
| **Supervisor** | Lisa Murphy |

**Month:** April

| | |
|---|---|
| **What**?<br>During this month, I worked on the application's main functionalities and fixed the emulator's errors in preparation for the final submission. I was having trouble with the recipe video and the voice speech that was specified in the requirements. The video was not displaying or performing as I had hoped. | |
| **So What?**<br>My PC was not cooperating with the emulator. To launch the application, I had to enable the virtual machine in the BIOS settings every time I powered on my device. | |
| **Now What?**<br>The voiced speech is still unable to fix along with the recipe video. Nonetheless, I will do my best to get it working and ready for the final presentation. | |
| **Student Signature** | Sutthita Phromchomcha |