Universiti Malaysia PAHANG
Engineering • Technology • Creativity

BCI3283 MOBILE APPLICATION DEVELOPMENT

Mini Project

Semester 2018/2019

MyRecipe Application

Course Lecturer : Dr. Mohammed Falah Mohammed

Section : 01A, 01B

Group members :

Student Name	Student ID
Cheng Tzyy En	CB16030
Low Chai Wei	CB16084
Chan Pui Fen	CB16095
Kong Shin Yee	CB16110

Abstract

The prime objective of “My Recipe Application” is to create a full-fledged Android application which could save a list of recipes based on the category menu which are breakfast, lunch and dinner that entered by the user. The user can sign up for an account if they are visiting as a guest. In addition, if the user already had an account, they can log in via their email address and password to view the recipe. A home page with 3 selections is shown to the user which allows the user to choose for the function and access it. There is “Breakfast”, “Lunch” and “Dinner”. The user or guest not only can search and view the recipes in the menu but also can add and delete their own recipe so that their recipes can share with others. If the users do not know what to cook when they are trying to cook something, they can make a choice of the best recipe based on the feedback and rating from others’ users. Moreover, there are 4 options in the toolbar to enable the user to click the function in a shortcut way which is “My Account”, “Search”, “Feedback” and “About Us”. “My Account” is a shortcut way to allow the user to login to the account. “Search” is a function to allow the user to search the recipe based on their preference. “Rate Us” allows the user to give rating and feedback. “About Us” include some simple information such as phone number and email address so that the user can contact our team if needed.

Table of content

1.0 Introduction

Cooking is a process that uses human knowledge to combine suitable ingredient. Recipes also a piece of knowledge, through the recipe that human cook, we able to convey some information about their personality, culture and habits.

Nowadays, less number of people are less time and confident to cook. The cooking recipe application as a recommender system that provide way to help people search for food and cook from recipes in a more flexible and easy way. This application is offering the types of ingredients and step-by-step preparation. User able to buy the ingredients based on the recipe from recipe recommender. Making recommendations on which food to prepare based on recommending recipes is an interesting functionality in itself. [1]

2.0 Case Study

Nowadays, majority of people are busy on work that cause they have no idea to prepare and cook their meals. It is normal phenomena to think that couples or family members who work at company or a person who lives alone want to cook food for themselves as quickly as possible and no need to worry about what to cook when they are rush in. However, if they are having same types of food, they will get bored and therefore they need an easy platform to get more recipes. Thinking of what to cook is also tough problem. To attract children liking, parents are responsible to change the menu every single day. Parents are not only think to what recipe to changes, they also need to consider the nutrition that their children taken. In addition, some people are easily to forget but ingredients to restock their kitchen. This also as problem when they have lack of ingredients to prepare for cooking a perfect meal.

Regarding on the problem that we have been observed, we have decided to build a mobile application for community use. We had done a handful research on community. After a thorough discussion, a few functions have been finalized to be implemented in the mobile application with the objective of providing cooking social platform especially for hectic people experience. There are few of function that have been finalized as a list is shown below:

- Allow user to create their personal recipe.
- Allow user to delete their recipe if they are not satisfy the recipe.
- Allow user to search recipe based on what they want to seek.

- Allow user to view recipe based on recipe categories.
- Allow user to rate and comment on the performance of application.

Firstly, user able to create their personal recipe in their account by inserting the recipe name, photo, ingredients and preparation. Secondly, user able to delete the recipe that already been created if they are not satisfy with it by clicking the “delete” button. Thirdly, user also able to search the recipe by inserting the recipe name at the search bar, the system will bring user to the recipe page regarding on user is searching. Forth, user able to view recipe based on recipe categories which are “breakfast”, “lunch” and “dinner”. Fifth, user able to rate and comment on application by putting the number of stars and texting the comment. In addition, user also able to use voice sensor as texting action instead of using typing action to comment the performance of application. Besides that, if user have any queries or problems, they may contact the application official office to seek for help and solutions by clicking “contact us” button.

The external database that we used to store user data in application is “Firebase”.

Users Table:

Variables	**Data Type**
Email	String
Name	String

Table 1: Attributes of users

Breakfast/Lunch/Dinner Table:

Variables	**Data Type**
Duration	String
Image	String
Ingredient	String
Preparation	String
RecipeName	String
Serving	String

UserID	String
Username	String

Table 2: Attributes of Breakfast/ Lunch/ Dinner

Rate & Comment Table:

Variables	**Data Type**
Rate	Int
Comment	String
UserID	String
UserName	String

Table 3: Attributes of Rate & Comment for User

Variables	**Data Type**
Rate	Int
Comment	String
UserID	String
UserName	String

Table 4: Attributes of Rate & Comment for Guest

3.0 Storyboard

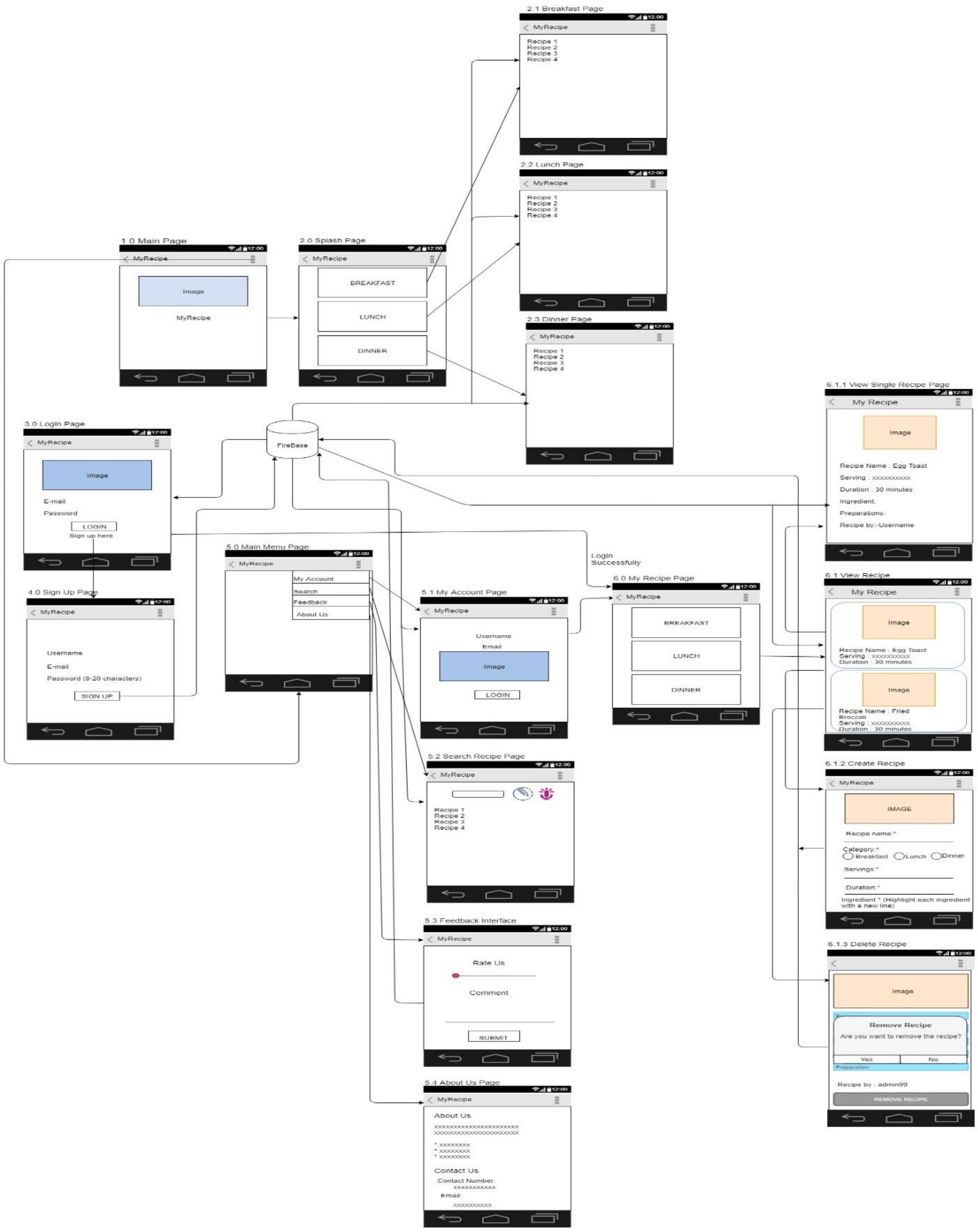


Figure 1: Storyboard of My Recipe Mobile Application

4.0 UML Use Case Diagram

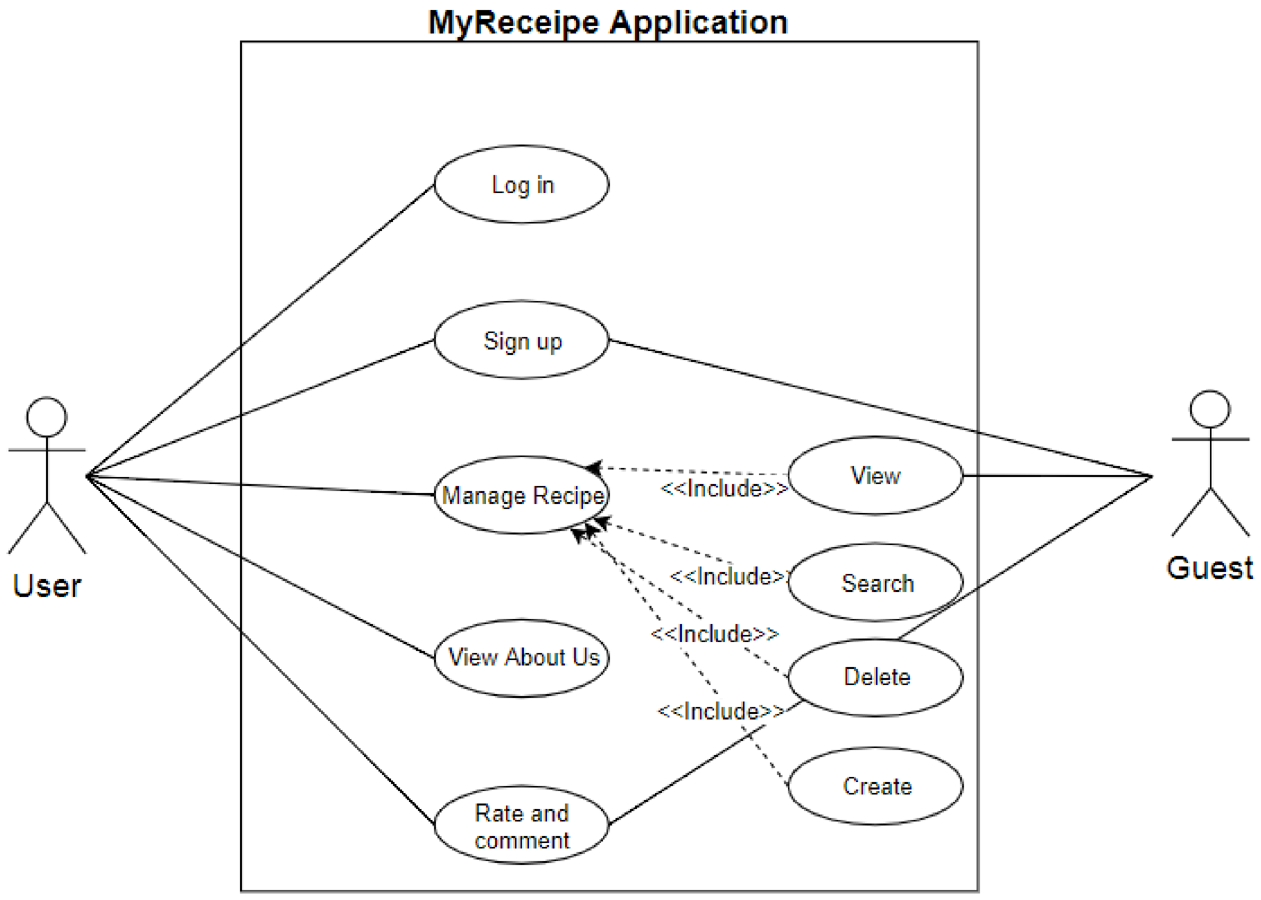


Figure 2: Use Case Diagram for My Recipe Application

5.0 FlowChart

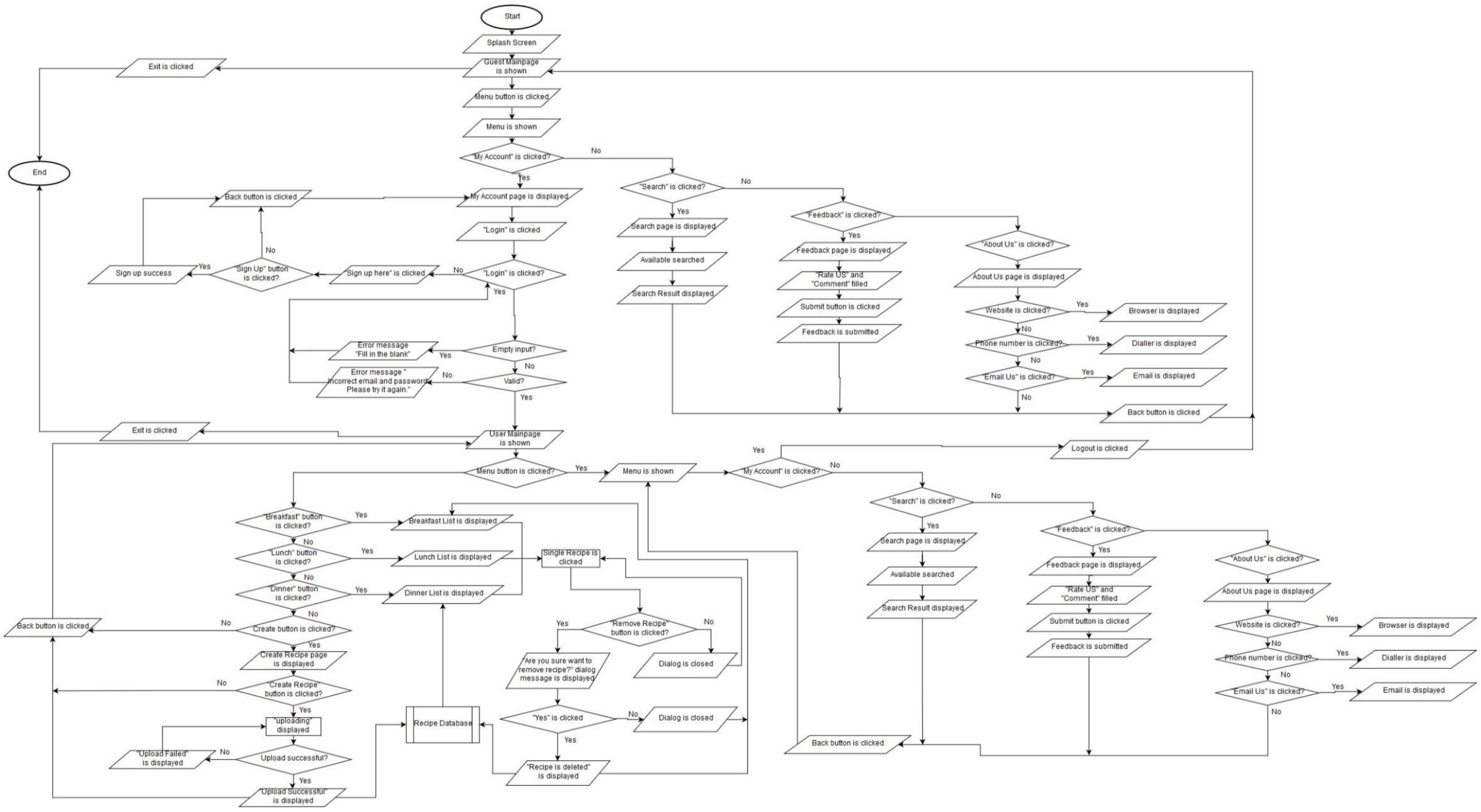


Figure 3: Flowchart of My Recipe Application

6.0 GUI Screenshots

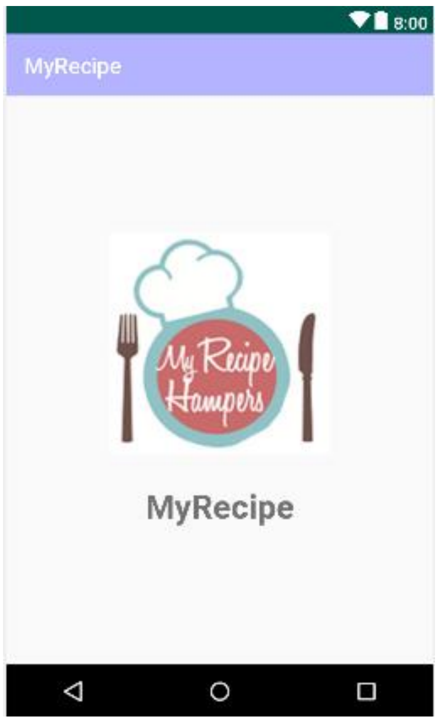


Figure 4: Main Interface

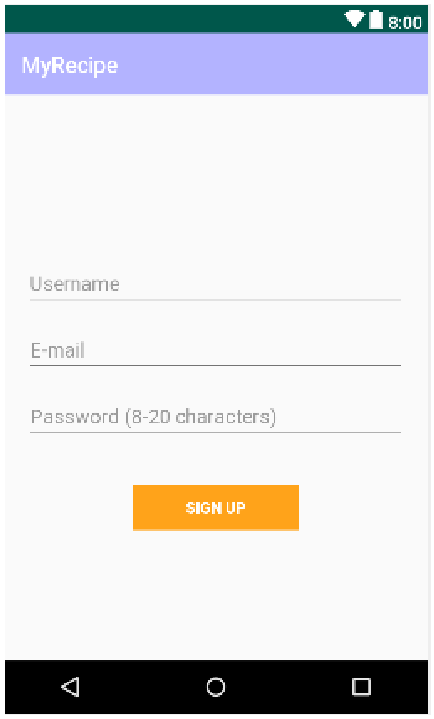


Figure 5: Signup Interface

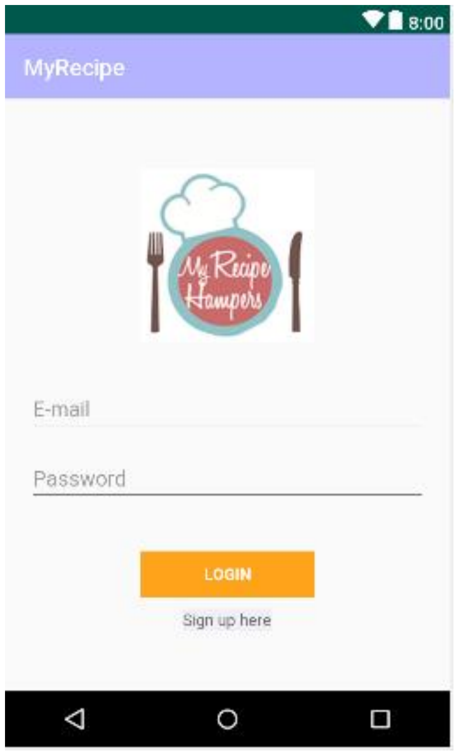


Figure 6: Login Interface

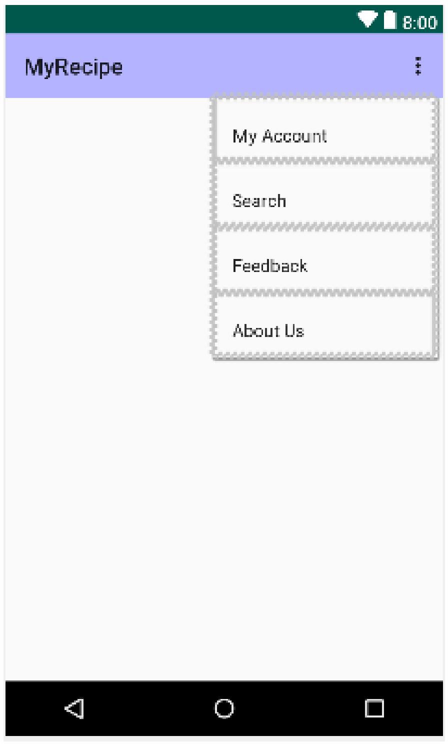


Figure 7: Main Menu Interface

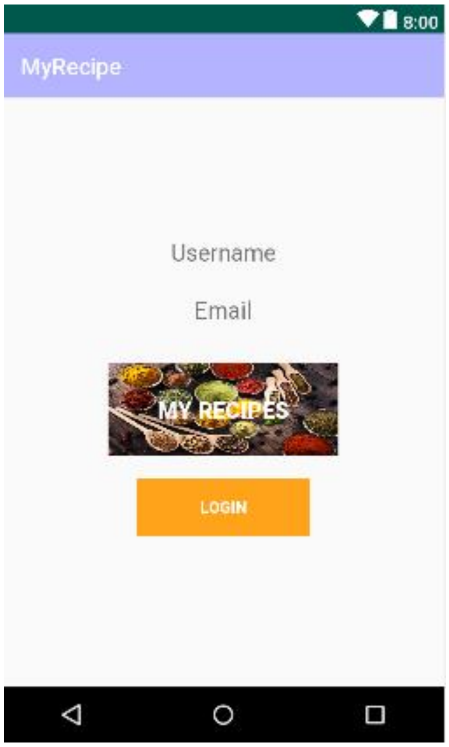


Figure 8: My Account Interface

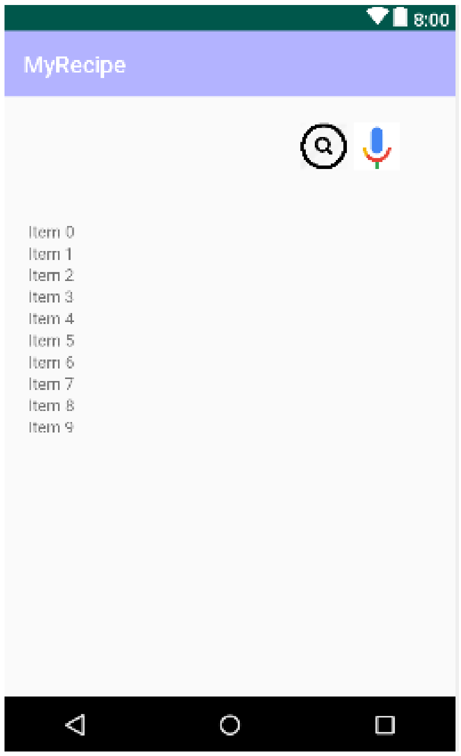


Figure 9: Search Interface

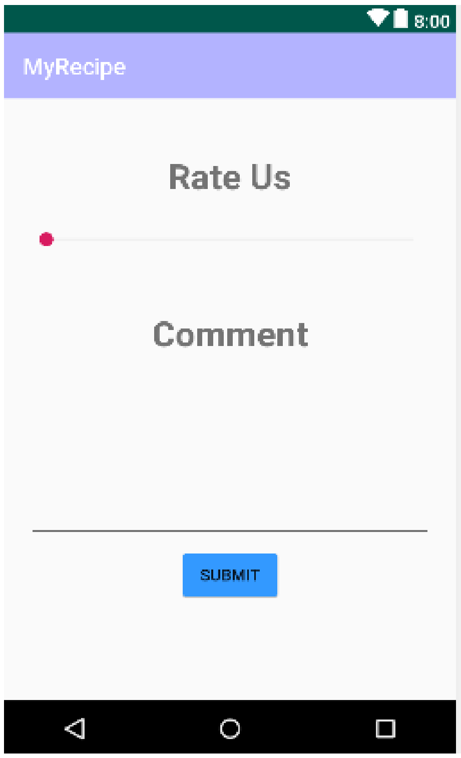
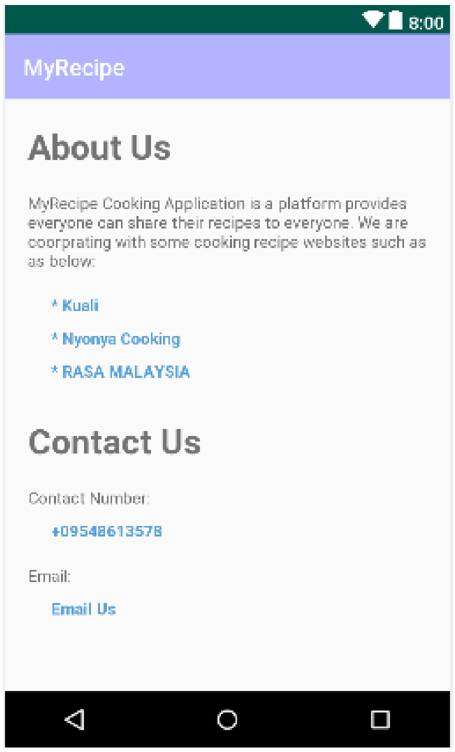


Figure 10: Feedback Interface



About Us

MyRecipe Cooking Application is a platform provides everyone can share their recipes to everyone. We are coorprating with some cooking recipe websites such as as below:

* Kuali
* Nyonya Cooking
* RASA MALAYSIA

Contact Us

Contact Number:

+09548613578

Email:

Email Us

Figure 11: About Us Interface

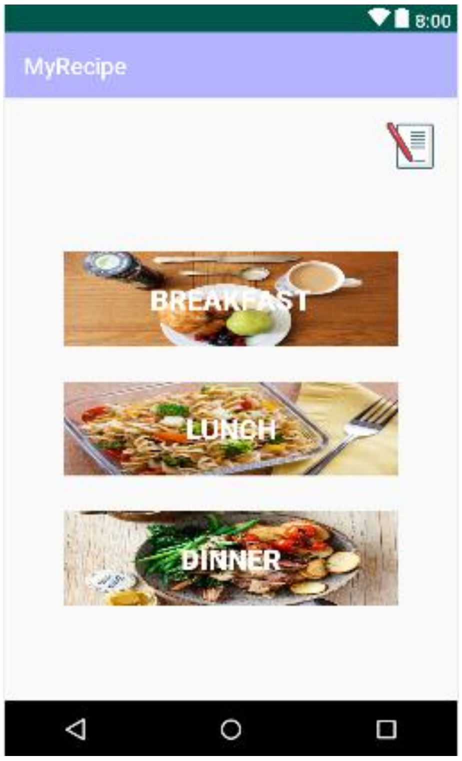


Figure 12: My Recipe Interface

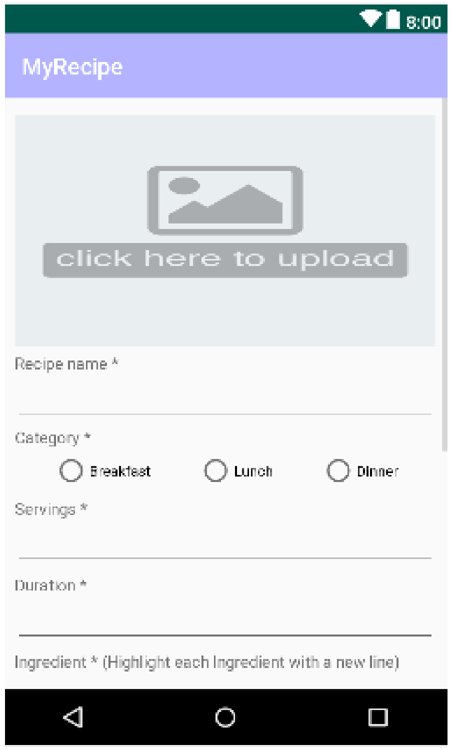


Figure 13: Create Recipe Interface

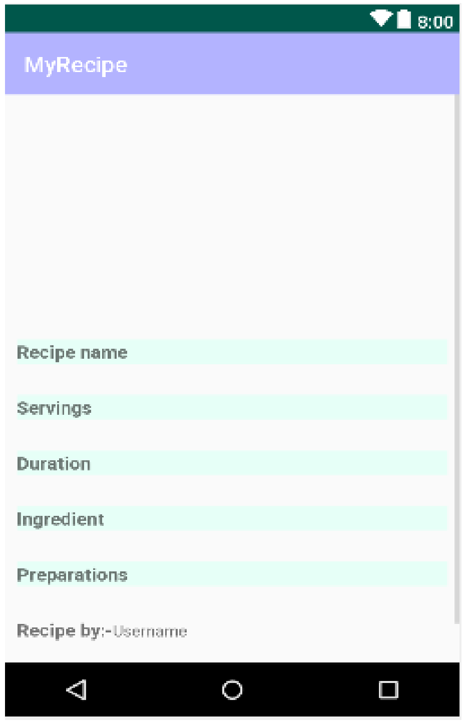


Figure 14: Single Recipe Interface

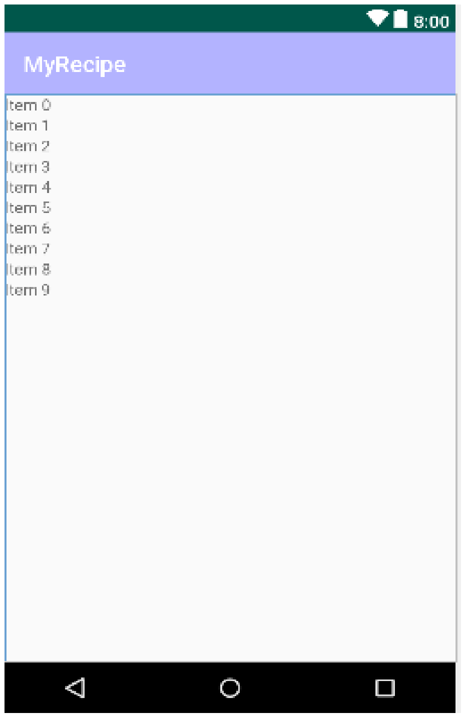


Figure 15: Breakfast, Lunch and Dinner Interface

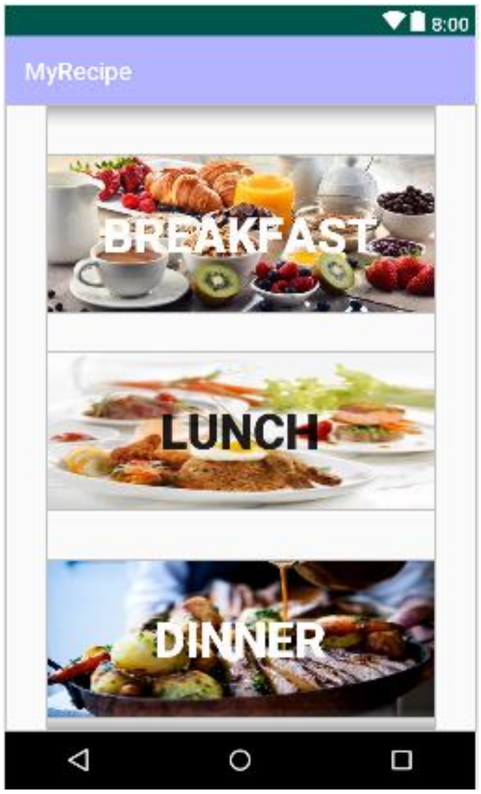


Figure 16: Splash Interface

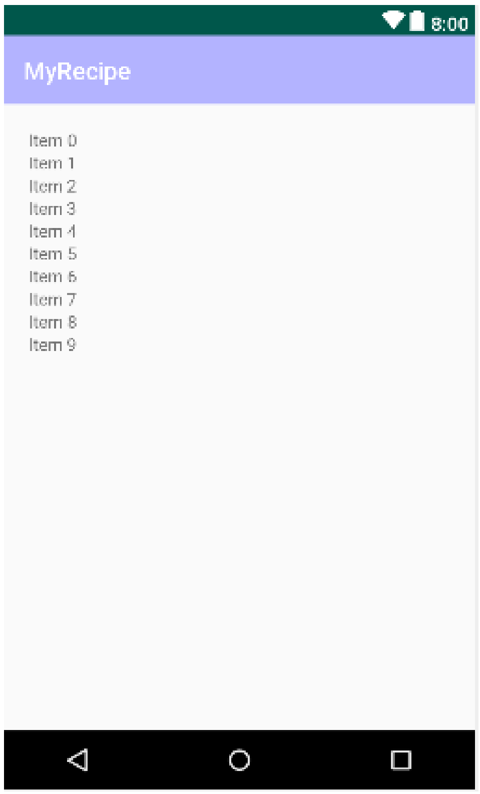


Figure 17: User Breakfast, Lunch and Dinner Interface

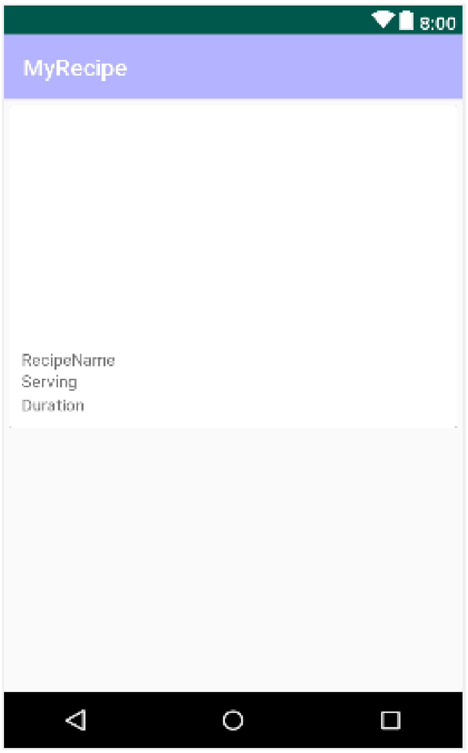


Figure 18: List of Recipe Interface

7.0 Problems and Obstacles Faced

There are some problems and obstacles faced when developing the mobile application. The first problem we faced is hard to distribute and define the functions of the mobile application. This is because the cooking recipe application has two actors which are user and guest. Both actors' functionality is similar but some of the functions only can access by user who logged in into the application such as manage recipe. Therefore, we study and do some research on cooking recipe application published on the market and find out some functionality able to implement into ours application.

Other than that, android studio's bug is the most critical problem we faced. While we are developing MyRecipe cooking application suddenly the software detected error about link resources failed. Those bugs drag our progress down and some of the bugs are hard to solve without expert knowledge especially the file corruption bug. The android studio design xml blue print gone and auto-keyword provide function also disable. Hence our progress delayed horribly and we keep trying to solve the bug by clean the project, restart software or laptop, the worst solution but solved the problem is re-install the software.

Another headache obstacle is determines which sensor we can apply in our application. Unlike the others application, cooking recipe application mostly applied one sensor which is speech to text. There are a lot of sensors but find a suitable sensor to implement is a challenge to us. A suitable sensor should useful and related to our topic. At last, we decide to implement speech to text on search recipe and accelerometer on main page.

Moreover, recycler view and card view also as a problem we faced while developing the application. Recycler view is list view enhanced view; the method to implement is different with list view. The main problem is we don't have knowledge and experience on implement the recycler view. It really takes a lot of time to study and get some example from the internet tutorials to overcome this problem. Hence we search for tutorials from YouTube and Google, and keep tried implement it until success.

To implement an external database also as an obstacle we faced during the development process. We tried to implement our application with mql but it is too complicated and lack of knowledge on how to connect android studio with mql and also mql required the mobile device and server network must be the same. We develop the project in laptop which connected campus internet server so the network is completely different from our mobile device. Even we changed our IP address but if we change location, the IP address also will be changed according to location. It will need t every time to modify the localhost address. This problem able to solve by upload to a server but we don’t have the resources and knowledge how to upload a database on the server. In the end, we used Firebase as external database to remote data.

Although we used Firebase solved the mql server problem but we have no knowledge of Firebase at all on how to implement and setup. It takes us a lot of times for research and study on how to setup and remote data from database or to database. But fortunately it finally successful to implement it on our application does some actions such as create, retrieve and delete data.

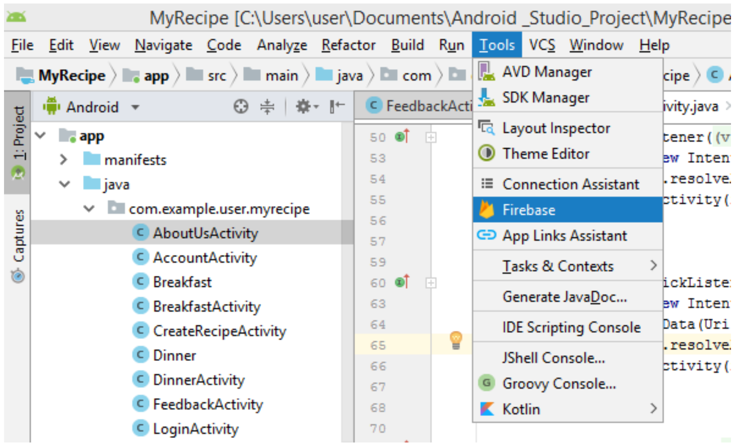
8.0 Firebase Tutorial

8.1Setup Firebase

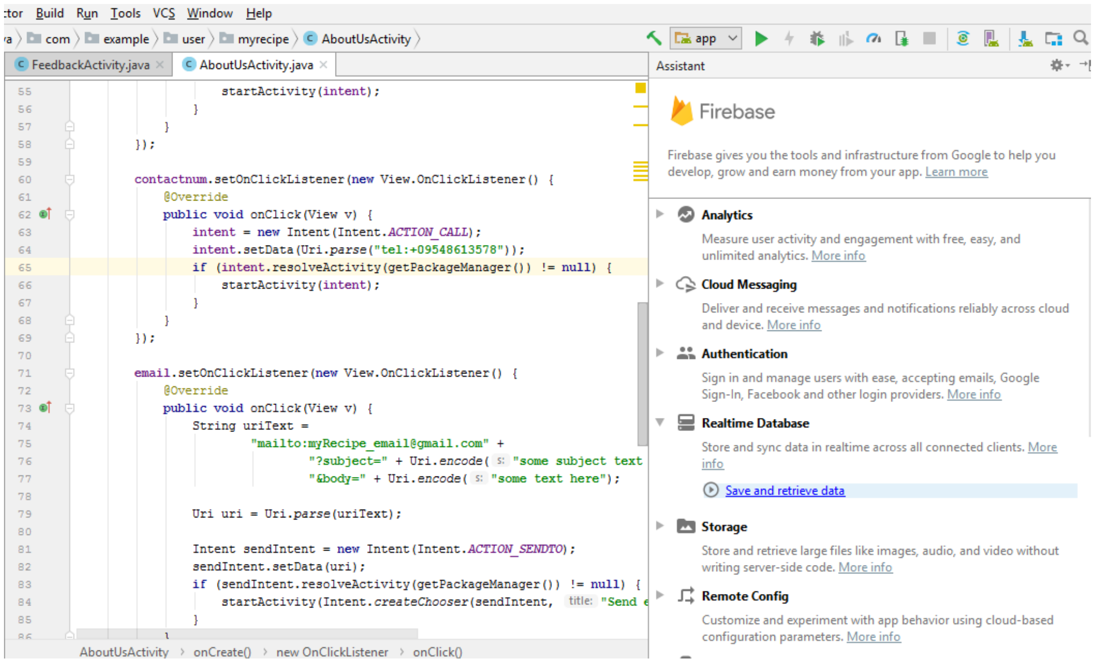
Add the dependency for Realtime Database to build.gradle file:

```
        implementation 'com.google.firebase:firebase-database:16.0.5'
```

Go to Android Studio click the Tools and click Firebase.



It will show up Firebase Assistance on right hand side.


tor Build Run Tools VCS Window Help
com example user myrecipe AboutUsActivity
app
FeedbackActivity.java
AboutUsActivity.java
Assistant
AboutUsActivity > onCreate() > new OnClickListener > onClick()

```
                startActivity(intent);
            }
        }
    });

    contactnum.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            intent = new Intent(Intent.ACTION_CALL);
            intent.setData(Uri.parse("tel:+09548613578"));
            if (intent.resolveActivity(getPackageManager()) != null) {
                startActivity(intent);
            }
        }
    });

    email.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String uriText =
                    "mailto:myRecipe_email@gmail.com" +
                            "?subject=" + Uri.encode( s: "some subject text
                            "&body=" + Uri.encode( s: "some text here");

            Uri uri = Uri.parse(uriText);

            Intent sendIntent = new Intent(Intent.ACTION_SENDTO);
            sendIntent.setData(uri);
            if (sendIntent.resolveActivity(getPackageManager()) != null) {
                startActivity(Intent.createChooser(sendIntent,  title: "Send e
            }
```

Firebase

Firebase gives you the tools and infrastructure from Google to help you develop, grow and earn money from your app. Learn more

Analytics

Measure user activity and engagement with free, easy, and unlimited analytics. More info

Cloud Messaging

Deliver and receive messages and notifications reliably across cloud and device. More info

Authentication

Sign in and manage users with ease, accepting emails, Google Sign-In, Facebook and other login providers. More info

Realtime Database

Store and sync data in realtime across all connected clients. More info

Save and retrieve data

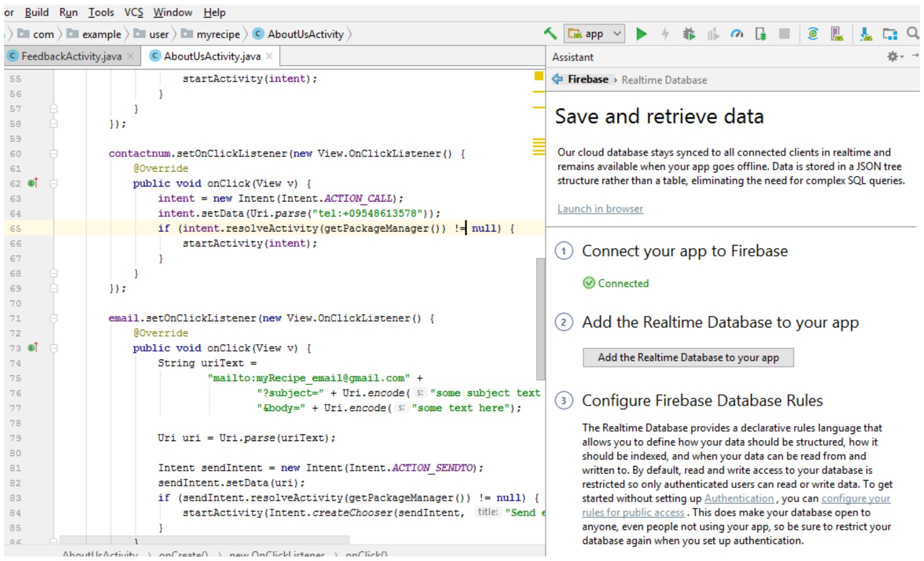
Storage

Store and retrieve large files like images, audio, and video without writing server-side code. More info

Remote Config

Customize and experiment with app behavior using cloud-based configuration parameters. More info

Click on Realtime Database and "Save and retrieve data" to activate firebase.


or Build Run Tools VCS Window Help
com › example › user › myrecipe › AboutUsActivity
app
FeedbackActivity.java
AboutUsActivity.java
Assistant
Firebase › Realtime Database
AboutUsActivity › onCreate() › new OnClickListener › onClick()

```java
                    startActivity(intent);
                }
            }
        });

        contactnum.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                intent = new Intent(Intent.ACTION_CALL);
                intent.setData(Uri.parse("tel:+09548613578"));
                if (intent.resolveActivity(getPackageManager()) != null) {
                    startActivity(intent);
                }
            }
        });

        email.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String uriText =
                        "mailto:myRecipe_email@gmail.com" +
                                "?subject=" + Uri.encode( s: "some subject text
                                "&body=" + Uri.encode( s: "some text here");

                Uri uri = Uri.parse(uriText);

                Intent sendIntent = new Intent(Intent.ACTION_SENDTO);
                sendIntent.setData(uri);
                if (sendIntent.resolveActivity(getPackageManager()) != null) {
                    startActivity(Intent.createChooser(sendIntent,  title: "Send e
                }
            }
```

Save and retrieve data

Our cloud database stays synced to all connected clients in realtime and remains available when your app goes offline. Data is stored in a JSON tree structure rather than a table, eliminating the need for complex SQL queries.

Launch in browser

1. Connect your app to Firebase

   Connected

2. Add the Realtime Database to your app

   Add the Realtime Database to your app

3. Configure Firebase Database Rules

   The Realtime Database provides a declarative rules language that allows you to define how your data should be structured, how it should be indexed, and when your data can be read from and written to. By default, read and write access to your database is restricted so only authenticated users can read or write data. To get started without setting up Authentication, you can configure your rules for public access. This does make your database open to anyone, even people not using your app, so be sure to restrict your database again when you set up authentication.

Choose the database u preferred or create a new database. Once it synced, your project is connected to database.

There a lot of available libraries for the various Firebase features. For this project we used some of the libraries such as following:

Gradle Dependency Line	**Service**
com.google.firebase:firebase-database:16.0.5	Realtime Database
com.google.firebase:firebase-storage:16.0.5	Storage
com.google.firebase:firebase-auth:16.1.0	Authentication
com.firebaseui:firebase-ui-storage:4.1.0	Firebase User Interface for Storage
com.firebaseui:firebase-ui-database:0.4.1	Firebase User Interface for Realtime Database

Table 5: Firebase libraries

8.2 Write data into Firebase

To retrieve an instance database using getInstance() ang getReference() to location to write to.

```
mDatabaseGuest =
FirebaseDatabase.getInstance().getReference().child("Feedback").child("Guest");
```

Or

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("Feedback").child("Guest");

myRef.setValue("Hello!");
```

Or using addValueEventListener() to insert the data.

```
mDatabase.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

        mDatabase.child("Rating").setValue(progress);
        mDatabase.child("Comment").setValue(comments)
                .addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        if (task.isSuccessful()) {
                            Toast.makeText(FeedbackActivity.this, "Your feedback has
been sent. Thank you very much.", Toast.LENGTH_SHORT).show();
                            intent = new Intent(FeedbackActivity.this,
MainActivity.class);
                            startActivity(intent);
                        }
                    }
                });
    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {
    }
});
```

8.3 Read data from Firebase

To read data from database we need a listener which is ValueEventListener to the reference we created.

```
BDatabase.child(Bpost_key).addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

        post_recipename = (String) dataSnapshot.child("RecipeName").getValue();
        post_serving = (String) dataSnapshot.child("Serving").getValue();
        post_duration = (String) dataSnapshot.child("Duration").getValue();
        post_ingredient = (String) dataSnapshot.child("Ingredient").getValue();
        post_preparation = (String) dataSnapshot.child("Preparation").getValue();
        post_user = (String) dataSnapshot.child("Username").getValue();
        post_image = (String) dataSnapshot.child("Image").getValue();
        post_id = (String) dataSnapshot.child("UserID").getValue();

    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {

    }
});
```

OnDataChange() method is called once with the initial value and again, whenever data at this location is updated. Using dataSnapshot will snap the every data in the database. onCancelled() happens when the read value is failed.

8.4 Delete data in Firebase

The simplest way to delete data is by using removeValue() on a reference to the location of that data.

```
if(Bpost_key!=null){
    BDatabase.child(Bpost_key).removeValue();
    Toast.makeText(context, "Recipe deleted", Toast.LENGTH_SHORT).show();

}
```

Delete function usually attached with button, once user click on the button the data will be removed.

9.0 Codes and Functions

9.1 MainActivity

9.1.1 Accelerometer Sensor

```
sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
sensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
sensorManager.registerListener(this, sensor, sensorManager.SENSOR_DELAY_NORMAL);


 //Method of accelerometer sensor
 @Override
 public void onSensorChanged(SensorEvent event) {
     float x = event.values[0];
     float y = event.values[1];

     if(Math.abs(x)>Math.abs(y)){
         if(x>15){//Left
             intent = new Intent(MainActivity.this, LoginActivity.class);
             startActivity(intent);
         }
         if(x<-15){//Right
             mAuth.signOut();
             Toast.makeText(this, "Logged out", Toast.LENGTH_SHORT).show();
         }
     }
 }
 @Override
 public void onAccuracyChanged(Sensor sensor, int accuracy) {

 }
```

Accelerometer sensor is our project 1st sensor, once the user swings the mobile device to left. It will call the intent function jump to LoginActivity. On the other hand, swing to the right will logout user account.

9.1.2 Menu

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main_menu, menu);
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if(item.getItemId() == R.id.action_account){
        intent = new Intent(MainActivity.this, AccountActivity.class);
        startActivity(intent);
    }
    if(item.getItemId() == R.id.action_search){
        intent = new Intent(MainActivity.this, SearchActivity.class);
        startActivity(intent);
    }
    if(item.getItemId() == R.id.action_feedback){
        intent = new Intent(MainActivity.this, FeedbackActivity.class);
        startActivity(intent);
    }
    if(item.getItemId() == R.id.action_about){
        intent = new Intent(MainActivity.this, AboutUsActivity.class);
        startActivity(intent);
    }
    return super.onOptionsItemSelected(item);
}
```

The optional menu has 4 items which are account, search, feedback and about us. Once the user clicks either one it will guide user to that activity.

9.2 SplashActivity

9.2.1 Splash screen

```
private static int SPLASH_TIME_OUT = 2000;
new Handler().postDelayed(new Runnable() {
    @Override
    public void run() {
        intent = new Intent(SplashActivity.this, MainActivity.class);
        startActivity(intent);
        finish();
    }
},SPLASH_TIME_OUT);
```

The splash activity activated once the application is open. The time of splash screen is 2 seconds, after 2 seconds it will jump to Main Activity to perform another function.

9.3 BreakfastActivity/ LunchActivity/ DinnerActivity

9.3.1 FirebaseRecyclerViewAdapter

```
@Override
protected void onStart() {
    super.onStart();

    FirebaseRecyclerAdapter<Breakfast, BreakfastViewHolder> firebaseRecyclerAdapter;

    firebaseRecyclerAdapter = new FirebaseRecyclerAdapter<Breakfast, BreakfastViewHolder>(Breakfast.class,
            R.layout.list_of_recipe, BreakfastViewHolder.class, mDatabase) {
        @Override
        protected void populateViewHolder(BreakfastViewHolder viewHolder, Breakfast model, int position) {

            final String post_key = getRef(position).getKey();

            viewHolder.setRecipename(model.getRecipeName());
            viewHolder.setDuration(model.getDuration());
            viewHolder.setServing(model.getServing());
            viewHolder.setImage(getApplicationContext(), model.getImage());

            viewHolder.mView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {

                    Intent intent = new Intent(BreakfastActivity.this, SingleRecipeActivity.class);
                    intent.putExtra("breakfast_id",post_key);
                    startActivity(intent);
                }
            });
        }
    };
    breakfast_List.setAdapter(firebaseRecyclerAdapter);
}

private static class BreakfastViewHolder extends RecyclerView.ViewHolder {
    View mView;

    public BreakfastViewHolder(@NonNull View itemView) {
        super(itemView);
        mView = itemView;
    }

    public void setRecipename(String recipename) {
        TextView post_recipename = (TextView)mView.findViewById(R.id.list_recipename);
        post_recipename.setText("Recipe Name: "+recipename);
    }

    public void setDuration(String duration) {
       TextView post_duration = (TextView)mView.findViewById(R.id.list_duration);
       post_duration.setText("Duration: " +duration +" minutes");
    }

    public void setServing(String serving) {
        TextView post_serving = (TextView)mView.findViewById(R.id.list_serving);
        post_serving.setText("Serving " +serving);
    }

    public void setImage(Context ctx,String image){

        ImageView post_image = (ImageView)mView.findViewById(R.id.list_image);
        Glide.with(ctx).load(image).into(post_image);
    }
}
```

FirebaseRecyclerAdapter is an adapter to retrieve the data from firebase and display on recycler view. In the Breakfast, Lunch and Dinner Activity will retrieve recipe name, duration, serving, image from database and display on the card view.

9.4 AccountActivity

9.4.1 Check user login

```
private DatabaseReference mDatabaseUser;
private FirebaseUser mCurrentUser;
private FirebaseAuth mAuth;
private FirebaseAuth.AuthStateListener mAuthListener;

mAuth = FirebaseAuth.getInstance();

    mAuthListener = new FirebaseAuth.AuthStateListener() {
        @Override
        public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {

            if(firebaseAuth.getCurrentUser() == null){

                recipebtn.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        Toast.makeText(AccountActivity.this,"Please log in", Toast.LENGTH_SHORT).show();
                    }
                });

                loginbtn.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        intent = new Intent(AccountActivity.this, LoginActivity.class);
                        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                        startActivity(intent);
                    }
                });
            }
            else{
                mCurrentUser = mAuth.getCurrentUser();
                mDatabaseUser =
FirebaseDatabase.getInstance().getReference().child("Users").child(mCurrentUser.getUid());
                mDatabaseUser.addValueEventListener(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                        String email = (String) dataSnapshot.child("Email").getValue();
                        String name = (String) dataSnapshot.child("Name").getValue();
                        username.setText(name);
                        useremail.setText(email);
                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError databaseError) {
                    }
                });


                recipebtn.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        intent = new Intent(AccountActivity.this, MyRecipeActivity.class);
                        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                        startActivity(intent);
                    }
                });
                loginbtn.setVisibility(View.INVISIBLE);
                logoutbtn.setVisibility(View.VISIBLE);
            }
        }
    };
}
```

In the Account Activity, the application will check is there any user log in into application. If no, user can't go through my recipe part and it will display a message user need to login to perform this function else user logged in able to go my recipe part. In this activity will retrieve user logged in's username and user email and display in textview.

9.4.2 User log out

```
logoutbtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mAuth.signOut();

        intent = new Intent(AccountActivity.this, MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        startActivity(intent);

    }
});
```

The logout button is invisible if user not in login status, otherwise if user logged in the logout button will visible. User can logout account by using mAuth.signOut() method.

9.5 LoginActivity

9.5.1 Check login

```
private void checkLogin() {
    String email = Login_email.getText().toString().trim();
    String password = Login_password.getText().toString().trim();

    if(!TextUtils.isEmpty(email) && !TextUtils.isEmpty(password)){
        pDialog.setMessage("Login-ing....");
        pDialog.show();
        mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if(task.isSuccessful()){
                    checkUserExist();
                    pDialog.dismiss();
                }
                else{
                    Toast.makeText(LoginActivity.this, "Incorrect email and password. Please try it again.",
                            Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
    else{
        Toast.makeText(LoginActivity.this, "Fill in the blank.", Toast.LENGTH_SHORT).show();
    }
}
```

The activity will compare email and password is null or not, if null it will display an error message else it will check the user exist.

9.5.2 Check user exist

```
private void checkUserExist(){

    final String user_id = mAuth.getCurrentUser().getUid();

    mDatabase.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

            if(dataSnapshot.hasChild(user_id)){
                intent = new Intent(LoginActivity.this, MainActivity.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                startActivity(intent);
            }
            else{
                Toast.makeText(LoginActivity.this, "This email does not exist. Sign up an account.",
                        Toast.LENGTH_SHORT).show();
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
        }
    });
}
```

It will check it database based on the user ID which automatically generated by firebase. If the database does have the user ID then it will jump to Main activity else display an error message.

9.6 SignupActivity

9.6.1 Sign up Account

```
private void StartSignUp(){

    final String username = Signup_username.getText().toString().trim();
     String email = Signup_email.getText().toString().trim();
    String password = Signup_password.getText().toString().trim();

    final String useremail = email;

    if(!TextUtils.isEmpty(username) && !TextUtils.isEmpty(email) && !TextUtils.isEmpty(password)
            && password.length()>=8 && password.length()<=20){

        mProgress.setMessage("Signing Up...");
        mProgress.show();

       mAuth.createUserWithEmailAndPassword(email,password).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()){

                        String user_id = mAuth.getCurrentUser().getUid();

                        DatabaseReference current_user_id = mDatabase.child(user_id);

                        current_user_id.child("Name").setValue(username);
                        current_user_id.child("Email").setValue(useremail);

                        mProgress.dismiss();

                        Intent intent = new Intent(SignupActivity.this, SplashActivity.class);
                        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                        startActivity(intent);
                    }
                }
            });
        }
    }
```

Sign up activity will verify the email and password meets the requirements. If success then will call the mAuth.createUserWithEmailAndPassword() function available in firebase authentication library. If the task is success it will create a new user account in database.

9.7 AboutUsActivity

9.7.1 Implicit Gmail

```
email.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String uriText =
                "mailto:myRecipe_email@gmail.com" +
                        "?subject=" + Uri.encode("some subject text here") +
                        "&body=" + Uri.encode("some text here");

        Uri uri = Uri.parse(uriText);

        Intent sendIntent = new Intent(Intent.ACTION_SENDTO);
        sendIntent.setData(uri);
        if (sendIntent.resolveActivity(getPackageManager()) != null) {
            startActivity(Intent.createChooser(sendIntent, "Send email"));
        }
```

User able to contact us by using Gmail, once he/she clicked the “email us” it will ask user prefer send the email by which method.

9.8 UserBreakfastActivity/ UserLunchActivity/ UserDinnerActivity

9.8.1 Get user created recipe

```
firebaseRecyclerAdapterU = new FirebaseRecyclerAdapter<Breakfast, UBreakfastViewHolder>(
        Breakfast.class,
        R.layout.list_of_recipe,
        UserBreakfastActivity.UBreakfastViewHolder.class,
        mQuery) {
    @Override
    protected void populateViewHolder(UBreakfastViewHolder viewHolder, Breakfast model, int position) {
        final String post_key = getRef(position).getKey();

        viewHolder.setRecipename(model.getRecipeName());
        viewHolder.setDuration(model.getDuration());
        viewHolder.setServing(model.getServing());
        viewHolder.setImage(getApplicationContext(), model.getImage());

        viewHolder.mView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(UserBreakfastActivity.this, SingleRecipeActivity.class);
                intent.putExtra("breakfast_id", post_key);
                startActivity(intent);
            }
        });
    }
  };
  breakfast_List.setAdapter(firebaseRecyclerAdapterU);
}
```

This activity is according the user and display recipe created by user own self. That mean other users recipe will not display in this activity.

9.9 CreateRecipeActivity

9.9.1 Create recipe

```
private void UploadtoBreakfastStorage(final String recipename_val, final String serving_val, final String
duration_val,
                                      final String ingredient_val, final String preparaton_val){

    final StorageReference filepath;

    final ProgressDialog progressDialog = new ProgressDialog(this);
    progressDialog.setTitle("Uploading...");
    progressDialog.show();

    filepath = mStorage.child("Breakfast/"+UUID.randomUUID().toString());

    filepath.putFile(imageUri).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
        @Override
        public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
            progressDialog.dismiss();
            Toast.makeText(CreateRecipeActivity.this, "Upload Successful", Toast.LENGTH_SHORT).show();


        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Toast.makeText(CreateRecipeActivity.this, "Upload Failed"+e.getMessage(),
Toast.LENGTH_SHORT).show();
        }
    }).addOnProgressListener(new OnProgressListener<UploadTask.TaskSnapshot>() {
        @Override
        public void onProgress(UploadTask.TaskSnapshot taskSnapshot) {
            double progress = (100.0*taskSnapshot.getBytesTransferred()/taskSnapshot
                    .getTotalByteCount());
            progressDialog.setMessage("Uploaded "+(int)progress+"%");
        }
    });

    Task<Uri> urlTask = filepath.putFile(imageUri).continueWithTask(new Continuation<UploadTask.TaskSnapshot,
Task<Uri>>() {
        @Override
        public Task<Uri> then(@NonNull Task<UploadTask.TaskSnapshot> task) throws Exception {
            if(!task.isSuccessful()){

                Toast.makeText(CreateRecipeActivity.this, ""+task.getException(), Toast.LENGTH_SHORT).show();
            }

            return filepath.getDownloadUrl();
        }
    }).addOnCompleteListener(new OnCompleteListener<Uri>() {
        @Override
        public void onComplete(@NonNull Task<Uri> task) {
            if(task.isSuccessful()){
                final Uri downloaduri = task.getResult();

                final DatabaseReference BDatabase = mDatabase.child("Breakfast").push();

                mDatabaseUser.addValueEventListener(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                        BDatabase.child("Image").setValue(downloaduri.toString());
                        BDatabase.child("RecipeName").setValue(recipename_val);
                        BDatabase.child("Serving").setValue(serving_val);
                        BDatabase.child("Duration").setValue(duration_val);
                        BDatabase.child("Ingredient").setValue(ingredient_val);
                        BDatabase.child("Preparation").setValue(preparaton_val);
                        BDatabase.child("UserID").setValue(mCurrentUser.getUid());
                        BDatabase.child("Username").setValue(dataSnapshot.child("Name").getValue())
                                .addOnCompleteListener(new OnCompleteListener<Void>() {
                                    @Override
                                    public void onComplete(@NonNull Task<Void> task) {
                                        if(task.isSuccessful()){
                                            intent = new Intent(CreateRecipeActivity.this, MyRecipeActivity.class);
                                            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                            startActivity(intent);
                                        }
                                    }
                                });
                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError databaseError) {
                    }
                });
            }
            else{
                Toast.makeText(CreateRecipeActivity.this, "Error", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
```

In this activity will retrieve user name and user ID, once user click create button the activity will receive the recipe name, duration, serving, ingredient, preparation, user ID and username write into database.

9.10 SingleRecipeActivity

9.10.1 Delete recipe

```
deletebtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(final View v) {
        AlertDialog.Builder alertbuilder = new AlertDialog.Builder(context);
        alertbuilder.setTitle("Remove Recipe");
        alertbuilder
                .setMessage("Are you sure want do remove the recipe?")
                .setCancelable(false)
                .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {

                        if(Bpost_key!=null){
                            BDatabase.child(Bpost_key).removeValue();
                            Toast.makeText(context, "Recipe deleted", Toast.LENGTH_SHORT).show();

                        }

                        if(Lpost_key!=null){
                            LDatabase.child(Lpost_key).removeValue();
                            Toast.makeText(context, "Recipe deleted", Toast.LENGTH_SHORT).show();
                        }

                        if(Dpost_key!=null){
                            DDatabase.child(Dpost_key).removeValue();
                            Toast.makeText(context, "Recipe deleted", Toast.LENGTH_SHORT).show();
                        }
                        finish();
                    }
                })
                .setNegativeButton("No", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.cancel();
                    }
                });
        AlertDialog alertDialog = alertbuilder.create();
        alertDialog.show();
    }
});
```

To able perform delete recipe is when the recipe creator click the delete button on the bottom of the recipe. Only the recipe creator can to perform the delete recipe action.

9.10.2 Display selected recipe details

```
if(Bpost_key!=null){
    BDatabase.child(Bpost_key).addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

            post_recipename = (String) dataSnapshot.child("RecipeName").getValue();
            post_serving = (String) dataSnapshot.child("Serving").getValue();
            post_duration = (String) dataSnapshot.child("Duration").getValue();
            post_ingredient = (String) dataSnapshot.child("Ingredient").getValue();
            post_preparation = (String) dataSnapshot.child("Preparation").getValue();
            post_user = (String) dataSnapshot.child("Username").getValue();
            post_image = (String) dataSnapshot.child("Image").getValue();
            post_id = (String) dataSnapshot.child("UserID").getValue();

            setDisplay();

            if(mAuth.getCurrentUser().getUid() .equals(post_id)){

                deletebtn.setVisibility(View.VISIBLE);
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
        }
    });
```

Once user or guest clicks the view, it will receive the clicked view position and display more details in single recipe page.

9.11 SearchActivity

9.11.1 Speech to text sensor

```
//Speech to text sensor
private void speak() {
    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,"Hi,speak something");

    try{
        startActivityForResult(intent,REQUEST_CODE_SPEECH_INPUT);

    }catch(Exception e) {
        Toast.makeText(this,""+e.getMessage(),Toast.LENGTH_SHORT).show();
    }
}
```

Speak() is another sensor: speech to text sensor. We using speech to voice out recipe name and it will transform to text format and display in the search text.

9.11.2 Search recipe

```
private void firebaseSearchB(String searchtext) {

    SearchQuery = BDatabase.orderByChild("RecipeName").startAt(searchtext).endAt(searchtext+"\uf8ff");

    FirebaseRecyclerAdapter<Breakfast, SearchActivity.SearchViewHolder> firebaseRecyclerAdapter;

    firebaseRecyclerAdapter = new FirebaseRecyclerAdapter<Breakfast,
SearchActivity.SearchViewHolder>(Breakfast.class,
            R.layout.list_of_recipe, SearchActivity.SearchViewHolder.class, SearchQuery) {
        @Override
        protected void populateViewHolder(SearchActivity.SearchViewHolder viewHolder, Breakfast model, int
position) {

            final String post_key = getRef(position).getKey();

            viewHolder.setDetails(getApplicationContext(),model.getRecipeName(), model.getServing(),
model.getServing(),model.getImage());

            viewHolder.mView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {

                    Intent intent = new Intent(SearchActivity.this, SingleRecipeActivity.class);
                    intent.putExtra("breakfast_id",post_key);
                    startActivity(intent);
                }
            });
        }
    };

    search_list.setAdapter(firebaseRecyclerAdapter);
}
```

To search a recipe we need a search text search for recipe name. By using Query we order the data by recipe name and search for starting and ending text. Once the search result successfully the data will display as recycler view.

9.12 FeedbackActivity

The activity will check user status to determine user or guest and use which method to store rate and comment into database.

9.12.1 Check for guest

```
mAuthListener = new FirebaseAuth.AuthStateListener() {
    @Override
    public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
        if (firebaseAuth.getCurrentUser() == null) {

            submitbtn.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    final String progress = tv.getText().toString();
                    mDatabase = mDatabaseGuest.push();
                    mDatabase.addValueEventListener(new ValueEventListener() {
                        @Override
                        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                            mDatabase.child("Rating").setValue(progress);
                            mDatabase.child("Comment").setValue(comments)
                                    .addOnCompleteListener(new OnCompleteListener<Void>() {
                                        @Override
                                        public void onComplete(@NonNull Task<Void> task) {
                                            if (task.isSuccessful()) {
                                                Toast.makeText(FeedbackActivity.this, "Your feedback has been sent. Thank you
very much.", Toast.LENGTH_SHORT).show();
                                                intent = new Intent(FeedbackActivity.this, MainActivity.class);
                                                startActivity(intent);
                                            }
                                        }
                                    });
                        }

                        @Override
                        public void onCancelled(@NonNull DatabaseError databaseError) {
                        }
                    });
                }
            });
```

9.12.2 Check for user

```
mCurrentUser = mAuth.getCurrentUser();
mDatabaseUser = FirebaseDatabase.getInstance().getReference().child("Users").child(mCurrentUser.getUid());

mDatabaseUser.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

        String email = (String) dataSnapshot.child("Email").getValue();
        String name = (String) dataSnapshot.child("Name").getValue();

        submitbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                final String progress = tv.getText().toString();
                mDatabase = FirebaseDatabase.getInstance().getReference().child("Feedback").child("User").push();
                mDatabaseUser.addValueEventListener(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                        mDatabase.child("Rating").setValue(progress);
                        mDatabase.child("Comment").setValue(comments);
                        mDatabase.child("UserID").setValue(mCurrentUser.getUid());
                        mDatabase.child("Username").setValue(dataSnapshot.child("Name").getValue())
                                .addOnCompleteListener(new OnCompleteListener<Void>() {
                                    @Override
                                    public void onComplete(@NonNull Task<Void> task) {
                                        if (task.isSuccessful()) {
                                            Toast.makeText(FeedbackActivity.this, "Your feedback has been sent. Thank you very
much.", Toast.LENGTH_SHORT).show();
                                            intent = new Intent(FeedbackActivity.this, MainActivity.class);
                                            startActivity(intent);
                                        }
                                    }
                                });
                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError databaseError) {
                    }
                });
            }
        });
    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {

    }
});
```

10.0 Reference

Add Firebase to Your Android Project. (2018, December 19). Retrieved from Firebase: https://firebase.google.com/docs/android/setup

[1] Leng, Y. L. (2010, May -). -. Retrieved from -: http://umpir.ump.edu.my/id/eprint/2640/1/YAP_LEE_LENG.PDF

Read and Write Data on Android. (2018, December 21). Retrieved from Firebase: https://firebase.google.com/docs/database/android/read-and-write

samtstern. (2016). *FirebaseUI-Android*. Retrieved from GitHub: https://github.com/firebase/FirebaseUI-Android

Upload Files on Android. (2018, December 21). Retrieved from Firebase: https://firebase.google.com/docs/storage/android/upload-files