CHAPTER – 1

INTRODUCTION

In present days in the entire world mobiles are used most usually. So the people are excepting some new technologies in mobiles as we know android is an open source.

As introductory to android application, there is a lot of scope to reach the people with one app approach. As per the idea of "Donate Food", we implemented with two modules in this application as Volunteer and Donator.

According to this project, those who like to Donate Food should post the request using "Donate Food" application and volunteer can register initially for viewing those requests and responds and Accepting and Commenting and Chatting to the respective donator.

1.1 Problem Statement

In present days in the entire world mobiles are using very rashly so the people are expecting some new technologies in mobiles as we know android is an open source.

Defined to the analysis, there is a lot of problem at present scenario on wastage of food, even the government had investing in the form of camps and awareness through media and cinemas.

The product aims at satisfying the requirement of needy organizations through donations over the act. The application shall ask the Donators to donate the Food and User/Volunteers to Register the Portal and verify the Food Donors nearby them ad Accept and Comment and Chat with them to get the Donated Food from Donors.

The application is developed using Android Studio and the languages using are core Java and XML. The main objectives of the proposed application include reduction in wastage of food, making food etc., available to orphanages, old age homes and other such organizations, which will also inculcate values of sharing and sensitivity among people.

1.2 Motivation

Most people don't realize how much food they throw away every day – from uneaten leftovers to spoiled produce. About 95 percent of the food we throw away ends up in landfills or combustion facilities.

In 2013, we disposed more than 35 million tons of food waste. Many people wish to donate things to needy organizations. Also, many organizations wish to ask for various things required by them such as food grains etc., but there is no source available through which they can satisfy their requirements.

Thereby, an Android application has been developed through which people can donate food as per their capacity and the application also allows organizations to put their requests i.e., item required by them, if any.

The majority of the population today uses smartphones with active internet connection, which is the basic requirements for this product to function properly.

1.3 Objective

The Android OS is roughly divided into five sections in four main layers:

- **Linux Kernel:**

  This is the kernel on which Android is based. This layer contains all the low level device drivers for the various hardware components of an Android device.

- **Libraries:**

  These contain all the code that provides the main features of an Android OS. For example, the SQLite library provides database support so that an application can use it for data storage. The Web kit library provides functionalities for web browsing.

- **Android runtime:**

  At the same layer as the libraries, the Android runtime provides a set of core libraries that enable developers to write Android apps using the Java programming language. The Android runtime also includes the Dalvik virtual machine, which enables every Android application to run in its own process; with its own instance of the Dalvik virtual machine (Android applications are compiled into Dalvik executables). Dalvik is a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU.

- **Application Framework:**

  Exposes the various capabilities of the Android OS to application developers so that they can make use of them in their applications.

- **Applications:**

  At this top layer, you will find applications that ship with the Android device (such as Phone, Contacts, Browser, etc.), as well as applications that you download and install from the Android Market. Any applications that you write are located at this layer.

1.3.1 Proposed System

The proposed application is android-based, developed on Android Studio version 2.0 using java and XML requires internet connection and will provide a platform for donors and volunteers after they successfully register into the system.

In this proposed system, donator can easily donate food using one single registration process. Volunteer can check details of donator and can call back for the particulars.

1.3.2 Advantages of Proposed System

Single request process can reach the maximum volunteers and get back call from volunteer to take food.

Maximum Volunteers can receive the requests.

If a user wishes to donate something, he/she can send a message in application. This message will be shown as notification in donations tab to other users. This message will be stored in backend in the database.

Once a notification is sent, the volunteers who wish to claim the donations can reply to the donor and contact him/her.

1.4 Literature Survey

This section takes critical review of existing system implemented, the success factor, challenges faced, technologies used and unresolved problems. This forms the basis for implementing the later version.

CHAPTER – 2

TECHNOLOGIES LEARNT

2.1 Technologies

Android:

Android is a Linux-based operating system for mobile devices such as smart phones and tablet computers. It is developed by the Open Handset Alliance, led by Google, and other companies.

Google purchased the initial developer of the software, Android Inc., in 2005. The unveiling of the Android distribution in 2007 was announced with the founding of the Open Handset Alliance, a consortium of 86 hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. Google releases the Android code as open-source, under the Apache License. The Android Open Source Project (AOSP) is tasked with the maintenance and further development of Android.

Android has a large community of developers writing application ("apps") that extended the functionality of the devices. Developers write primarily in a customized version of Java. Apps can be downloaded from third-party sites or through online stores such as Google Play (formerly Android Market), the app store run by Google. In October 2011, there were more than 500,000 apps available for Android, and the estimated number of applications downloaded from the Android market as of December 2011 exceeded 10 billion.

Android became the world's leading Smartphone platform at the end of 2010. For the first quarter of 2012, Android has a 59% Smartphone market share worldwide, with a 331 million devices installed base and 85 million activations or 934,000 per day. Analysts point to the advantage to Android of being a multi-channel, multi-carrier OS.

Google Firebase:

Google Firebase is a Google-backend application development software that enables developers to develop iOS, Android and Web apps. Firebase provides tools for tracking analytics reporting and fixing app crashes, creating marketing and product experiment.

Firebase offers a number of services, including:

Analytics – Analytics presents data about user behavior in iOS and Android apps, enabling better decision-making about improving performance and app marketing.

Authentication – Firebase Authentication makes it easy for developers to build secure authentication systems and enhances the sign-in and onboarding experience for users. This feature offers a complete identity solution, supporting email and password accounts, phone auth, as well as Google, Facebook, GitHub, Twitter login and more.

Cloud messaging – Firebase Cloud Messaging (FCM) is a cross-platform messaging tool that lets companies reliably receive and deliver messages n iOS, Android and the web at no cost.

Real time database – The Firebase Real time Database is a cloud-hosted NoSQL database that enables data to be stored and synced between users in real time. The data is synced across all clients in real time and is still available when an app goes offline.

Performance – Firebase Performance Monitoring service gives developers insight into the performance characteristics of their iOS and Android apps to help them determine where and when the performance of their apps can be improved.

2.2 Tools Used

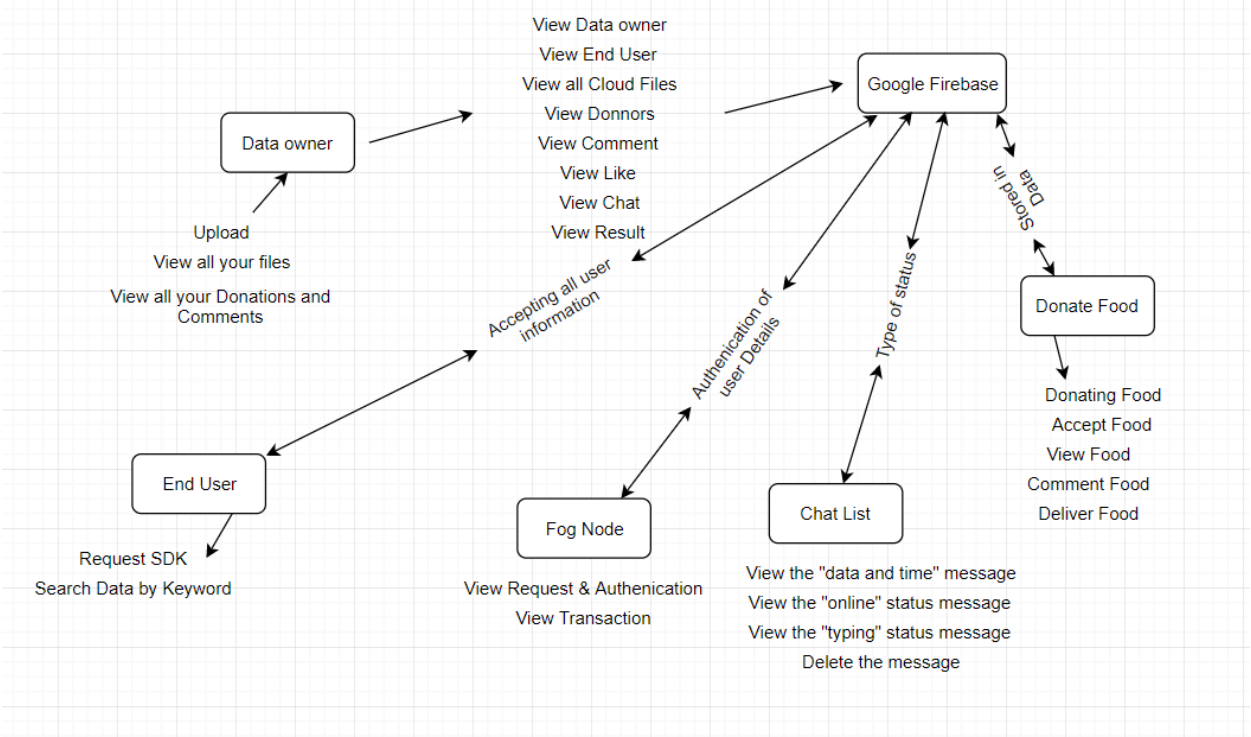
GPS System:

Google Maps Navigation is a mobile application developed by Google for the Android and iOS operating systems that was later integrated into the Google Maps mobile app. The application uses an Internet connection to a GPS navigation system to provide turn-by-turn voice-guided instructions on how to arrive at a given destination. The application requires connection to Internet data (e.g. 3G, 4G, WiFi etc.,) and normally uses a GPS satellite connection to determine its location. A user can enter a destination into the application, where will plot a path to it. The app displays the user's progress along the route and issues instructions for each turn.

CHAPTER – 3

SYSTEM DESIGN

3.1 System Architecture:



3.2 Modules Description:

- **Data owner:**

  In this module, the data owner performs operations such as Upload, View all your files and View all your Donations and Comments

- **End User:**

  In this module, the end user performs operations such as Request SDK and Search Data by keyword.

- **Fod Node:**

  In this module, the Fod Node performs operations such as View Request & Authenication and View Transaction

- **Donate Food:**

  In this module, the Donate Food performs operations such as Donating Food, Accept Food, View Food, Comment Food, Delivery Food

- **Chat list:**

  ` In this module, the Chatlist performs operations such as View the "**date and time**" message, View the **"online"** status message, View the **"typing"** status message and Delete the message

- **Google Firebase:**

  In this module, the Google Frebase operations such as Authentication of User, Database Storage, Images Storage.

3.3 System Specifications:

3.3.1 Software Requirements:

- Operating System : Android, Linux, Windows XP
- Software : Java/J2SE, ADT plug-in
- Development Tools: Android SDK, Android Emulator, Eclipse Helios.
- Front End : Java
- Back End : Google Firebase

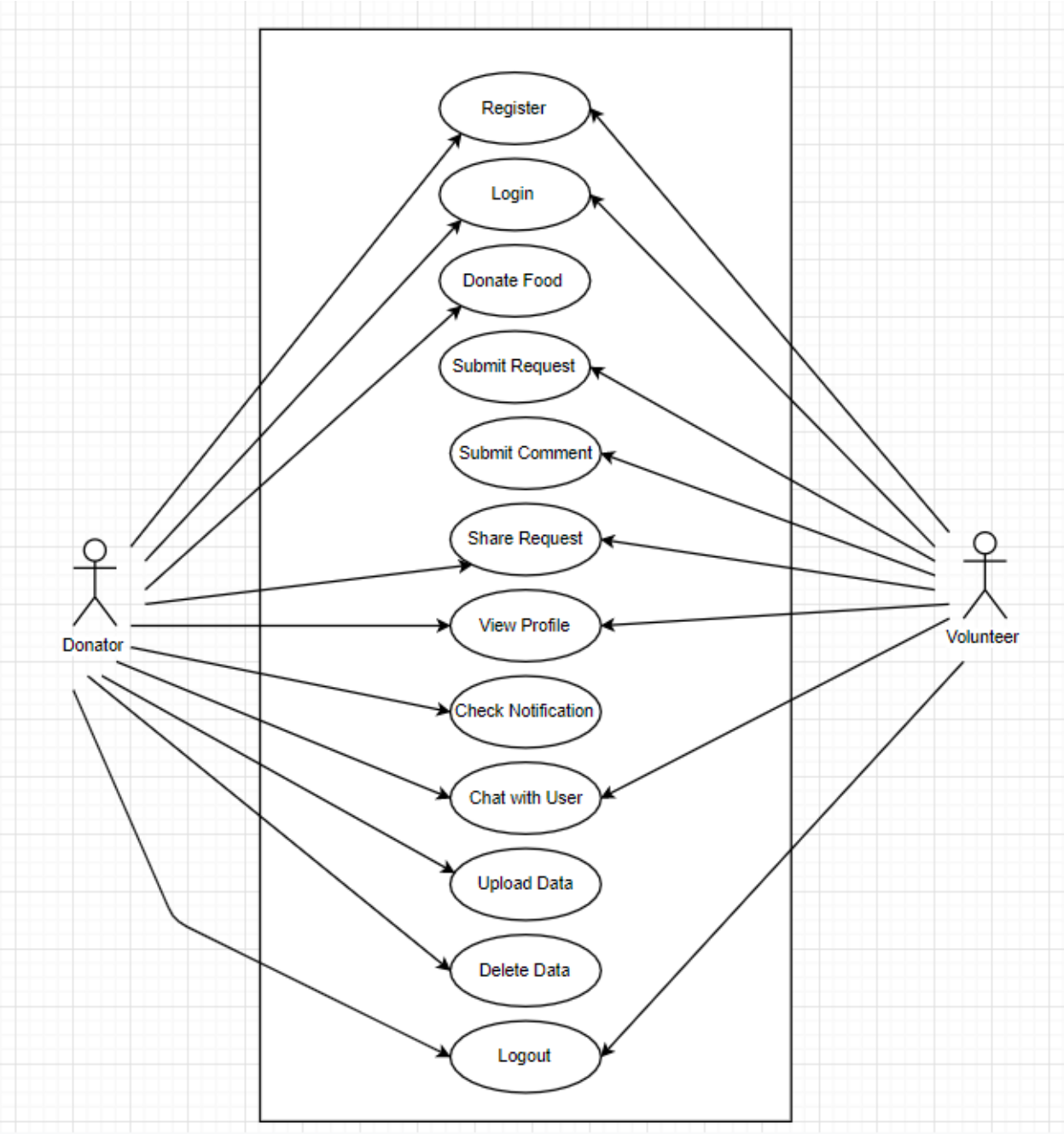
3.3.2 Hardware Requirements:

- Processor : Pentium IV with 2 GHZ
- Ram : 1GB Ram
- Hard Disk : 40GB Hard Drive
- OS : Android Phone (optional)

3.4 Detailed Design:

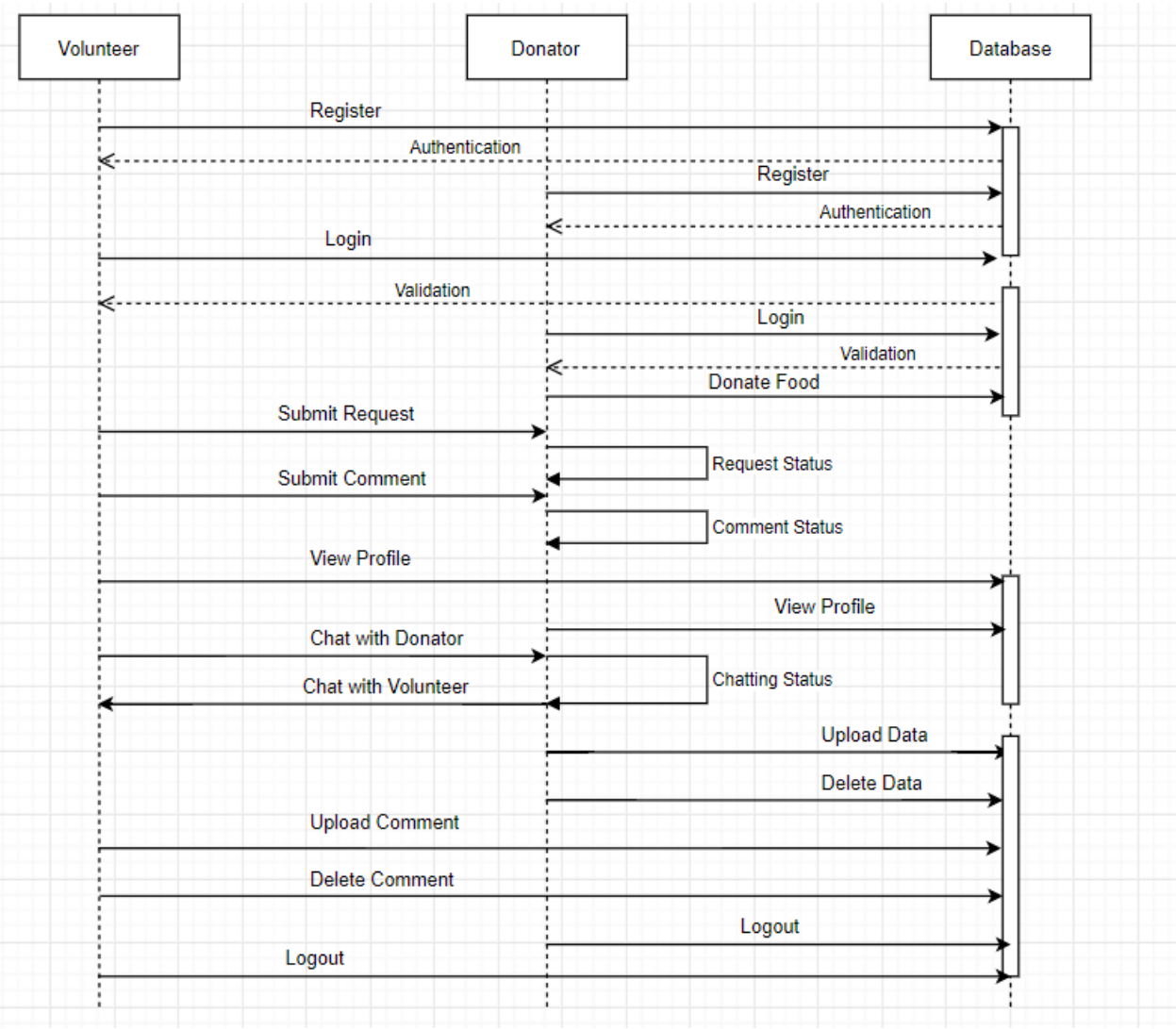
3.4.1 Use Case Diagram:

A use case diagram shows as set of use cases and actors and their relationships. Use case diagram are especially important in organizing and modelling behavior of a system

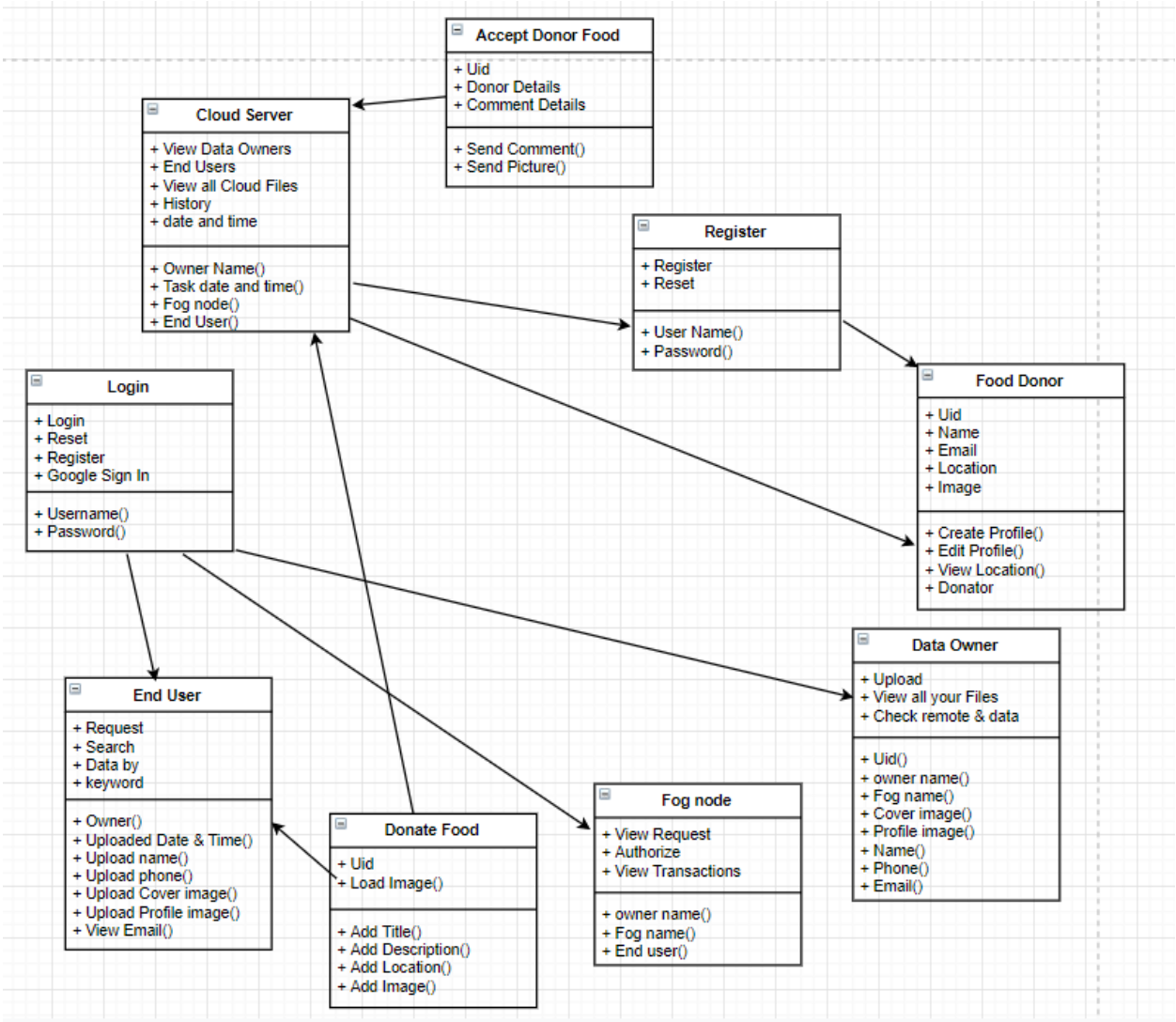


3.4.2 Sequence Diagram:

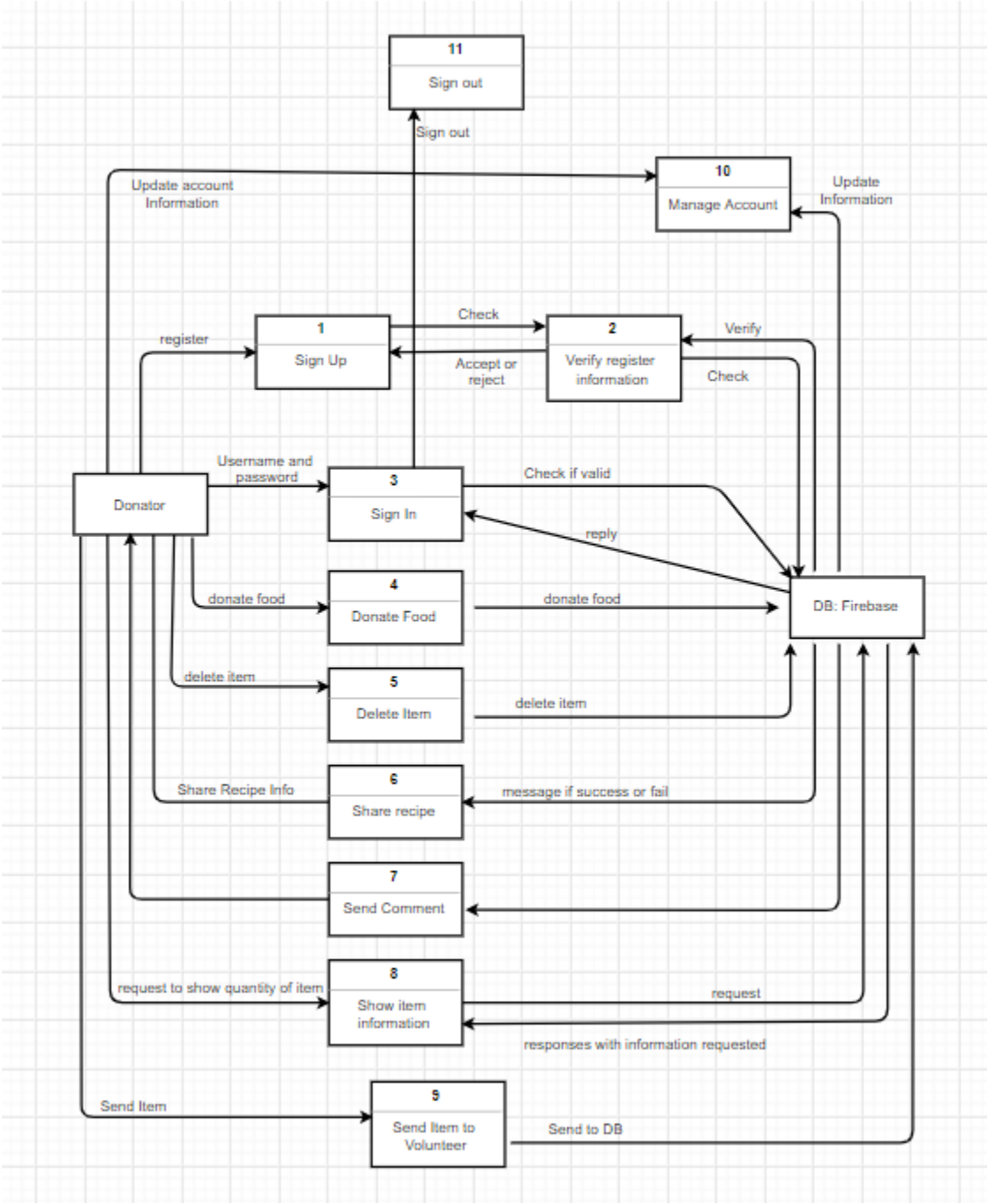
A sequence diagram is an interaction diagram that emphasizes the time ordering of messages. A sequence diagram shows a set of objects and messages sent and receive by those objects. The objects are typically named or anonymous instances of other things, such as collaborations, components and nodes. We can use sequence diagram to illustrate the dynamic view of a system.



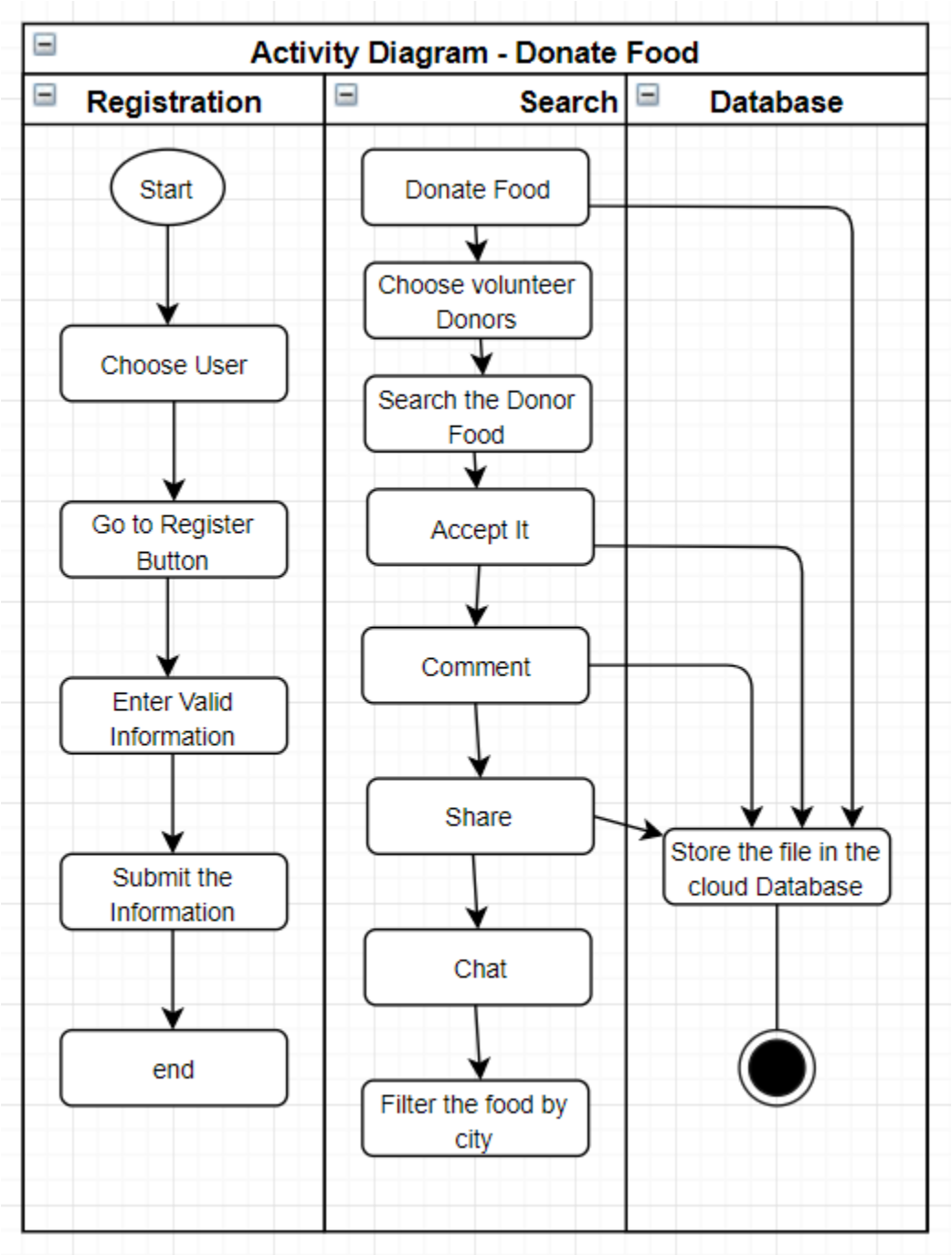
3.4.3 Class Diagram:



3.4.4 Dataflow Diagram:



3.4.5 Activity Diagram:



CHAPTER - 4

IMPLEMENTATION

XML – Layouts (.xml):

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:padding="20dp"
   tools:context=".MainActivity">

   <ImageView
      android:layout_width="250dp"
      android:layout_height="250dp"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="20dp"
      android:layout_centerVertical="true"
      android:src="@drawable/fooddonate" />

   <Button
      android:id="@+id/register_btn"
      android:text="Register"
      style="@style/Base.Widget.AppCompat.Button.Colored"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_above="@+id/login_btn"
      android:layout_centerHorizontal="true"
      android:drawableStart="@drawable/ic_register"
      android:drawableLeft="@drawable/ic_register"
      android:minWidth="230dp"/>

   <Button
      android:id="@+id/login_btn"
      android:text="Login"
      android:layout_alignParentBottom="true"
      android:layout_marginBottom="20dp"
      style="@style/Base.Widget.AppCompat.Button.Colored"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      android:drawableStart="@drawable/ic_login"
      android:drawableLeft="@drawable/ic_login"
      android:minWidth="230dp"/>

</RelativeLayout>
```

activity_login.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="20dp"
    tools:context=".LoginActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAlignment="center"
        android:text="Login"
        android:layout_marginTop="150dp"
        android:textColor="#000"
        android:textSize="25sp" />

    <com.google.android.material.textfield.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:id="@+id/emailTIL">
        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/emailEt"
            android:inputType="textEmailAddress"
            android:hint="Email"/>
    </com.google.android.material.textfield.TextInputLayout>

    <com.google.android.material.textfield.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:id="@+id/passwordTIL"
        android:layout_below="@+id/emailTIL"
        app:passwordToggleEnabled="true">
        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/passwordET"
            android:inputType="textPassword"
            android:hint="Password"/>
    </com.google.android.material.textfield.TextInputLayout>

    <Button
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Login"
        style="@style/Base.Widget.AppCompat.Button.Colored"
        android:layout_below="@+id/passwordTIL"
        android:layout_centerHorizontal="true"
        android:minWidth="120dp"
        android:drawableLeft="@drawable/ic_login"
        android:drawableStart="@drawable/ic_login"
        android:id="@+id/loginBtn"/>

    <TextView
        android:id="@+id/recoverPassTv"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAlignment="center"
        android:text="Forgot Password? Recover"
        android:layout_below="@+id/loginBtn"
        android:textColor="@color/colorBlack"/>

    <com.google.android.gms.common.SignInButton
        android:id="@+id/googleLoginBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="10dp"
        android:layout_below="@+id/recoverPassTv">
    </com.google.android.gms.common.SignInButton>

    <TextView
        android:id="@+id/nothave_acccountTv"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Not have account? Register"
        android:textAlignment="center"
        android:layout_alignParentBottom="true"
        android:layout_marginTop="30dp"
        android:textColor="@color/colorBlack"
        android:layout_marginBottom="20dp"/>

</RelativeLayout>
```

Java Folder Files (.java):

MainActivity.java:

```
package com.bhargav.verifyproject;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

    Button mRegisterBtn, mLoginBtn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mRegisterBtn = findViewById(R.id.register_btn);
        mLoginBtn = findViewById(R.id.login_btn);

        mRegisterBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(MainActivity.this,RegisterActivity.class));
                finish();
            }
        });

        mLoginBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(MainActivity.this,LoginActivity.class));
                finish();
            }
        });
    }
}
```

LoginActivity.java:

```
package com.bhargav.verifyproject;

import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;
```

```java
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.text.InputType;
import android.util.Patterns;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.auth.api.signin.GoogleSignIn;
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;
import com.google.android.gms.auth.api.signin.GoogleSignInClient;
import com.google.android.gms.auth.api.signin.GoogleSignInOptions;
import com.google.android.gms.common.SignInButton;
import com.google.android.gms.common.api.ApiException;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthCredential;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.auth.GoogleAuthProvider;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import java.util.HashMap;

public class LoginActivity extends AppCompatActivity {

    private static final int RC_SIGN_IN = 100;
    GoogleSignInClient mGoogleSignInClient;
    EditText mEmail,mPassword;
    Button mLogin;
    TextView mForgot,mRegister;
    SignInButton mGoogleLoginBtn;

    private FirebaseAuth mAuth;

    ProgressDialog pd;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
```

```
    ActionBar actionBar = getSupportActionBar();
    actionBar.setTitle("Login...");

    actionBar.setDisplayHomeAsUpEnabled(true);
    actionBar.setDisplayShowHomeEnabled(true);

    // Configure Google Sign In
    GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
            .requestIdToken(getString(R.string.default_web_client_id))
            .requestEmail()
            .build();
    mGoogleSignInClient = GoogleSignIn.getClient(this,gso);

    mAuth = FirebaseAuth.getInstance();

    mEmail = findViewById(R.id.emailEt);
    mPassword = findViewById(R.id.passwordET);
    mLogin = findViewById(R.id.loginBtn);
    mForgot = findViewById(R.id.recoverPassTv);
    mRegister = findViewById(R.id.nothave_acccountTv);
    mGoogleLoginBtn = findViewById(R.id.googleLoginBtn);

    mLogin.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String email = mEmail.getText().toString().trim();
            String passw = mPassword.getText().toString().trim();

            if(!Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
                mEmail.setError("Invalid Email");
                mEmail.setFocusable(true);
            }
            else {
                loginUser(email,passw);
            }
        }
    });

    mRegister.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent i = new Intent(LoginActivity.this,RegisterActivity.class);
            startActivity(i);
        }
    });

    mForgot.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
```

```
            showRecoveryPasswordDialog();
        }
    });

    mGoogleLoginBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent signInIntent = mGoogleSignInClient.getSignInIntent();
            startActivityForResult(signInIntent, RC_SIGN_IN);
        }
    });

    pd = new ProgressDialog(this);
}

private void showRecoveryPasswordDialog() {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Recover Password");

    LinearLayout linearLayout = new LinearLayout(this);
    final EditText emailEt = new EditText(this);
    emailEt.setHint("Email");
    emailEt.setInputType(InputType.TYPE_TEXT_VARIATION_EMAIL_ADDRESS);

    emailEt.setMinEms(16);

    linearLayout.addView(emailEt);
    linearLayout.setPadding(10,10,10,10);

    builder.setView(linearLayout);

    builder.setPositiveButton("Recover", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            String email = emailEt.getText().toString().trim();
            beginRecovery(email);
        }
    });

    builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.dismiss();
        }
    });

    builder.create().show();
}

private void beginRecovery(String email) {
    pd.setMessage("Sending email...");
```

```
        pd.show();
        mAuth.sendPasswordResetEmail(email).addOnCompleteListener(new
OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                pd.dismiss();
                if(task.isSuccessful()) {
                    Toast.makeText(LoginActivity.this,"Email Sent",Toast.LENGTH_SHORT).show();
                }
                else {
                    Toast.makeText(LoginActivity.this,"Failed.",Toast.LENGTH_SHORT).show();
                }
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                pd.dismiss();
                Toast.makeText(LoginActivity.this,""+e.getMessage(),Toast.LENGTH_SHORT).show();
            }
        });
    }

    private void loginUser(String email, String passw) {
        pd.setMessage("Logging In...");
        pd.show();
        mAuth.signInWithEmailAndPassword(email, passw)
                .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (task.isSuccessful()) {
                            pd.dismiss();
                            // Sign in success, update UI with the signed-in user's information
                            FirebaseUser user = mAuth.getCurrentUser();
                            startActivity(new Intent(LoginActivity.this, DashboardActivity.class));
                            finish();
                        } else {
                            pd.dismiss();
                            // If sign in fails, display a message to the user.
                            Toast.makeText(LoginActivity.this, "Authentication failed.",
Toast.LENGTH_SHORT).show();
                        }

                        // ...
                    }
                }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                pd.dismiss();
                Toast.makeText(LoginActivity.this,""+e.getMessage(),Toast.LENGTH_SHORT).show();
            }
        });
```

```
}

@Override
public boolean onSupportNavigateUp() {
    onBackPressed();
    return super.onSupportNavigateUp();
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    // Result returned from launching the Intent from GoogleSignInApi.getSignInIntent(...);
    if (requestCode == RC_SIGN_IN) {
        Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);
        try {
            // Google Sign In was successful, authenticate with Firebase
            GoogleSignInAccount account = task.getResult(ApiException.class);
            firebaseAuthWithGoogle(account);
        } catch (ApiException e) {
            // Google Sign In failed, update UI appropriately
            Toast.makeText(this,""+e.getMessage(),Toast.LENGTH_SHORT).show();
            // ...
        }
    }
}

private void firebaseAuthWithGoogle(GoogleSignInAccount acct) {

    AuthCredential credential = GoogleAuthProvider.getCredential(acct.getIdToken(), null);
    mAuth.signInWithCredential(credential)
            .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()) {
                        // Sign in success, update UI with the signed-in user's information
                        FirebaseUser user = mAuth.getCurrentUser();

                        if(task.getResult().getAdditionalUserInfo().isNewUser()) {
                            String email = user.getEmail();
                            String uid = user.getUid();

                            HashMap<Object, String> hashMap = new HashMap<>();

                            hashMap.put("email", email);
                            hashMap.put("uid", uid);
                            hashMap.put("name", "");
                            hashMap.put("onlineStatus", "online");
                            hashMap.put("typingTo", "noOne");
                            hashMap.put("phone", "");
                            hashMap.put("image", "");
```

```
                    hashMap.put("cover", "");

                    FirebaseDatabase database = FirebaseDatabase.getInstance();

                    DatabaseReference reference = database.getReference("Users");
                    reference.child(uid).setValue(hashMap);
                }


Toast.makeText(LoginActivity.this,""+user.getEmail(),Toast.LENGTH_SHORT).show();
                    startActivity(new Intent(LoginActivity.this, DashboardActivity.class));
                    finish();
                    //updateUI(user);
                } else {
                    // If sign in fails, display a message to the user.
                    Toast.makeText(LoginActivity.this,"Login
Failed...",Toast.LENGTH_SHORT).show();
                    //updateUI(null);
                }
            }
        }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Toast.makeText(LoginActivity.this,""+e.getMessage(),Toast.LENGTH_SHORT).show();
        }
    });
  }
}
```

AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  package="com.bhargav.verifyproject">

  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission android:name="android.permission.CAMERA" />
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

  <application
    android:allowBackup="true"
    android:icon="@drawable/fooddonate"
    android:label="@string/app_name"
    android:roundIcon="@drawable/fooddonate"
    android:supportsRtl="true"
    android:theme="@style/AppTheme"
    tools:ignore="GoogleAppIndexingWarning"
    tools:replace="android:allowBackup">
    <activity android:name=".PostDetailActivity"></activity>
```

```xml
        <activity android:name=".TheresProfileActivity" />
        <activity android:name=".AddPostActivity" />
        <activity
            android:name=".ChatActivity"
            android:theme="@style/AppThemeNo" />
        <activity android:name=".DashboardActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".RegisterActivity" />
        <activity android:name=".LoginActivity" />
        <activity android:name=".MainActivity" />

        <service
            android:name=".notifications.FirebaseService"
            android:enabled="true"
            android:exported="true">
            <intent-filter>
                <action android:name="com.googee.firebase.INSTANCE_ID_SERVICE" />
            </intent-filter>
        </service>
        <service
            android:name=".notifications.FirebaseMessaging"
            android:enabled="true"
            android:exported="true">
            <intent-filter>
                <action android:name="com.google.firebase.MESSAGING_EVENT" />
            </intent-filter>
        </service>

        <provider
            android:authorities="com.bhargav.verifyproject.fileprovider"
            android:name="androidx.core.content.FileProvider"
            android:exported="false"
            android:grantUriPermissions="true">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/paths" />
        </provider>
    </application>

</manifest>
```

5. TEST RESULTS

Testing:

The Android testing framework, an integral part of the development environment, provides architecture and powerful tools that help you test every aspect of your application at every level from unit to framework.
The testing framework has these key features:

- We can use palin Junit to test a class that doesn't call the Android API, or Android's Junit extensions to test Android components.
- Test suites are contained in test packages that are similar to main application packages, so you don't need to learn a new set of tools or techniques for designing and building tools.
- The SDK also provides monkey runner, an API tesing devices with Python programs, and UI/Application exercise runner, a command-line tool for stress-testing UIs by sending pseudo-random events to a device.

5.1 Test cases:

Test Case Id: TST_01

Test Title: Select the Register on owner main

STEP	TEST STEPS	EXCEPTED RESULT	ACTUAL RESULT	STATUS (PASS/FAIL)
1	Check the registration	The Register must be a new user	Click the register button	Pass
2	The list of Data owners	From the list of Data owners the details of the each login person	Data owner should be choosen	Pass

Test Case Id: TST_02

Test Title: Upload the data

STEP	TEST STEPS	EXCEPTED RESULT	ACTUAL RESULT	STATUS (PASS/FAIL)
1	Select the cloud server	It should be login	It should successful login with no issues	Pass
2	Select the fod node	Login to fog node	The login should be wrong	Pass

3	Uploading the user details	Uploading the Food, User details, Accept, Comment etc.,	It should upload the Food, User details, Accept, Comment etc.,	Pass

Test Case Id: TST_03

Test Title: Editing the Data

STEP	TEST STEPS	EXCEPTED RESULT	ACTUAL RESULT	STATUS (PASS/FAIL)
1	Select the owner info	It should be verify owner user info and edit details	It should be edit the details after verification of owner info	Pass
2	Select the owner info and upload Cover photo	It should be verify owner info and upload the Cover photo	It should be upload the Cover photo	Pass
3	Select the owner info and upload Profile photo	It should be verify owner info and upload the Profile photo	It should be upload the Profile photo	Pass

Test Case Id: TST_04

Test Title: Sharing the Food Message

STEP	TEST STEPS	EXCEPTED RESULT	ACTUAL RESULT	STATUS (PASS/FAIL)
1	We can select or non-select the owner info	It should be share the donator food details	It should be share the donator food details to any one	Pass

Test Case Id: TST_05

Test Title: Deleting the message

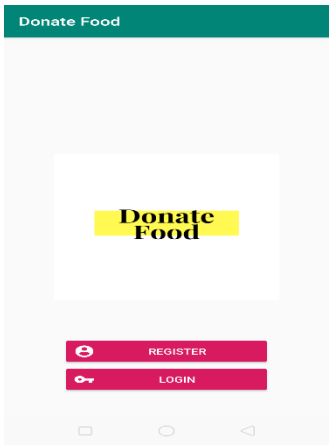
STEP	TEST STEPS	EXCEPTED RESULT	ACTUAL RESULT	STATUS (PASS/FAIL)
1	Select the volunteer user and delete the particular message	It should be delete the message if it is sent by mistake	It should be delete the message	Pass

CHAPTER – 6

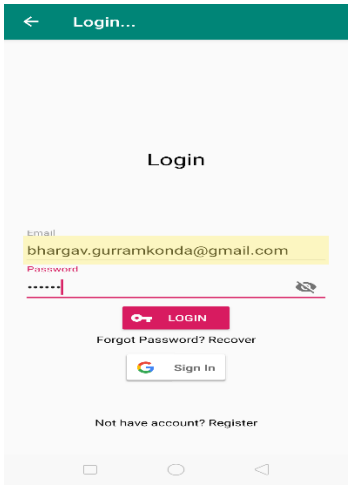
RESULT AND DISCUSSION

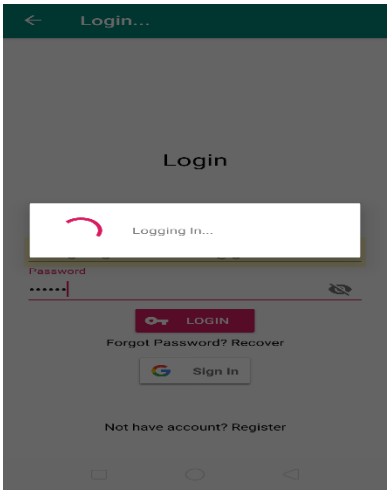
Application Output:

Main Page:

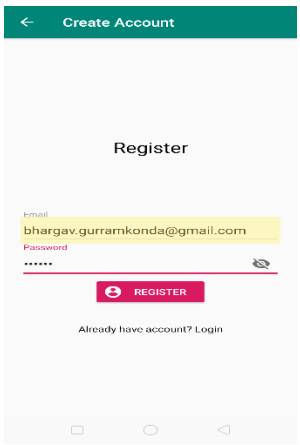


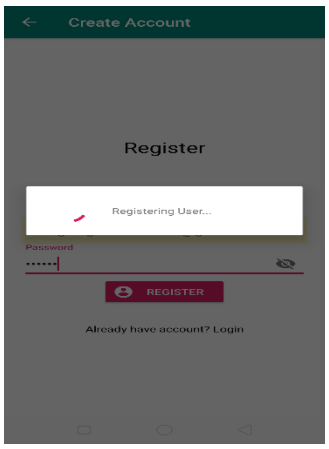
Login Page:



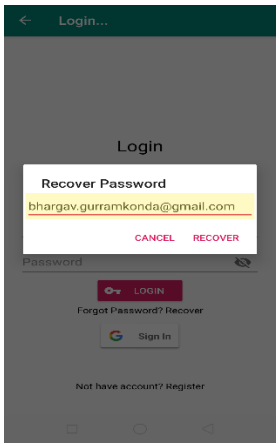


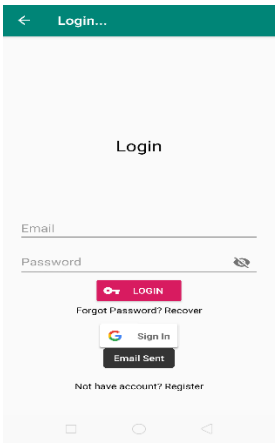
Register Page:

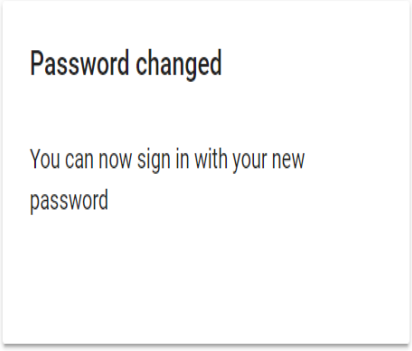


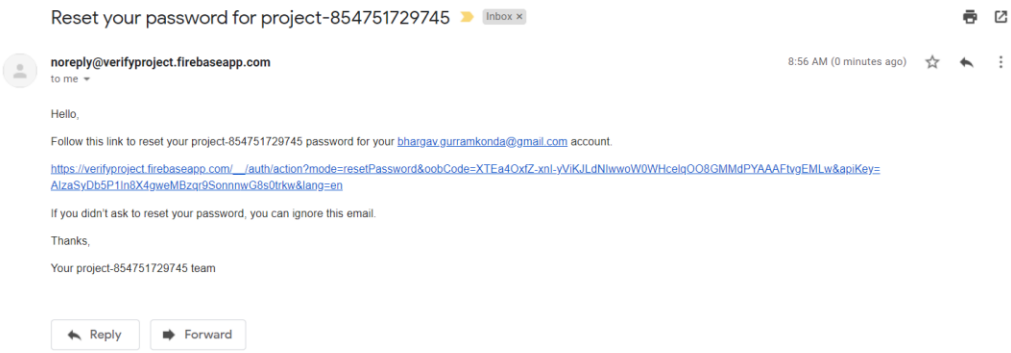


Reset Password Page:

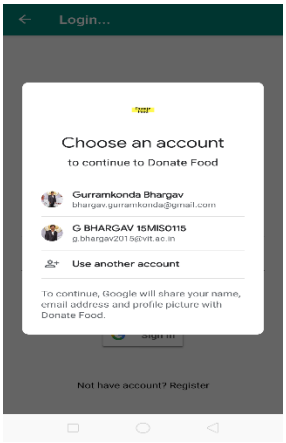


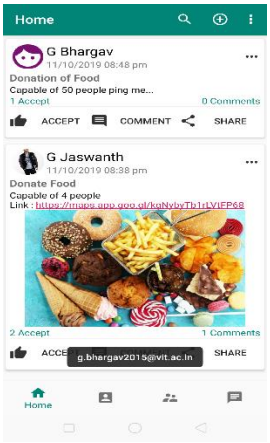




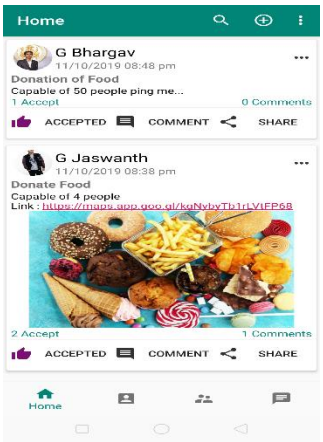


Google SignIn Page:

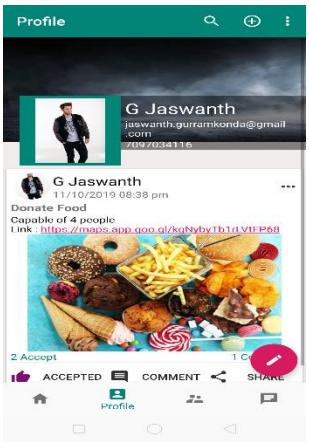




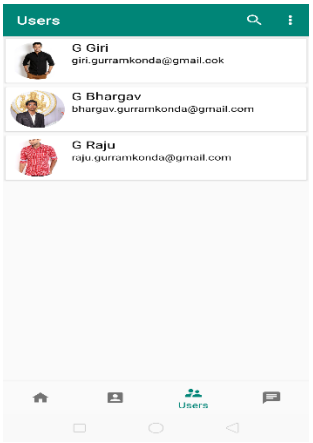
Home Page:



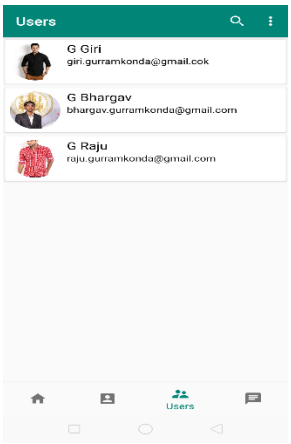
Profile Page:

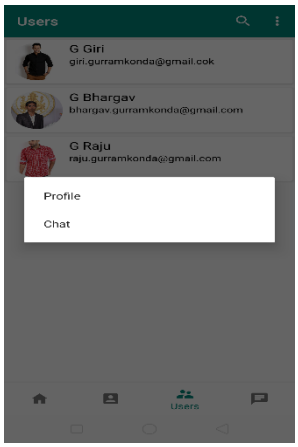


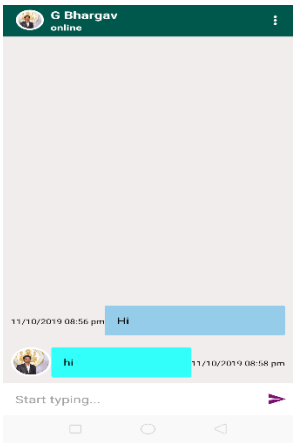
Users Page:

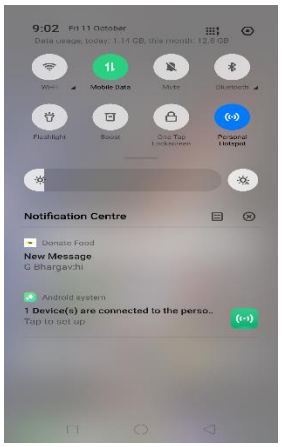


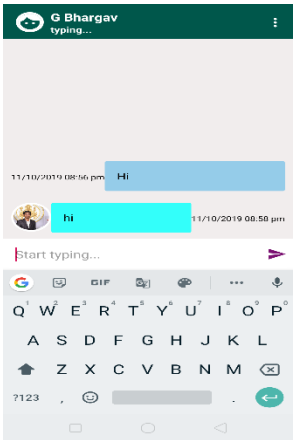
Chats Page:

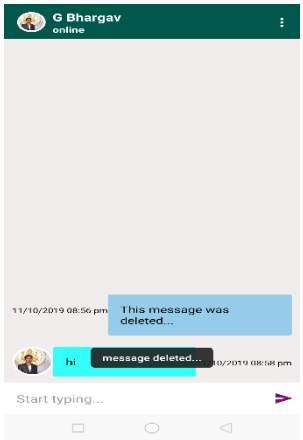


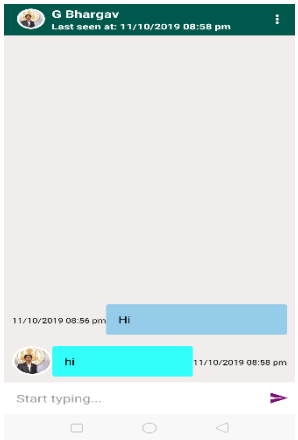




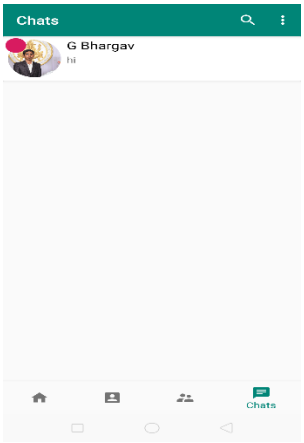




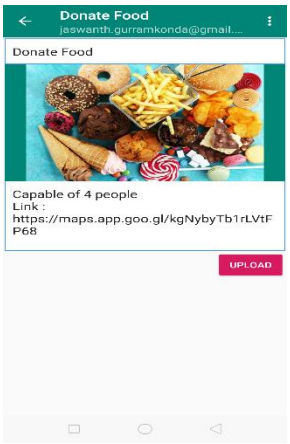


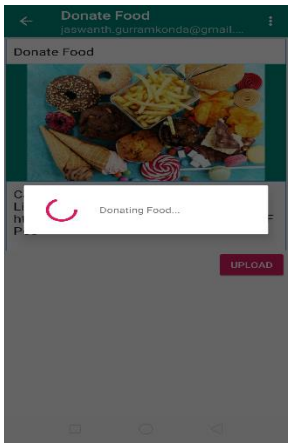


Chat List Page:



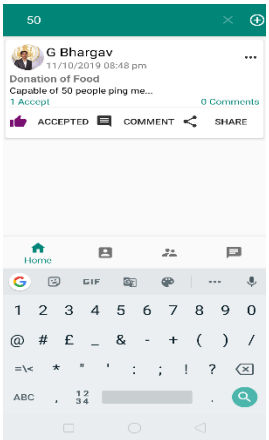
Donate Food Page:



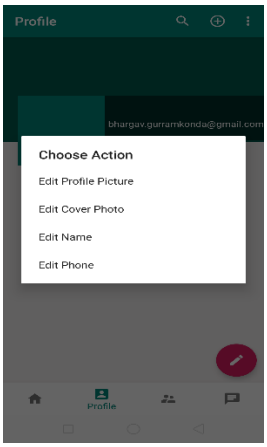


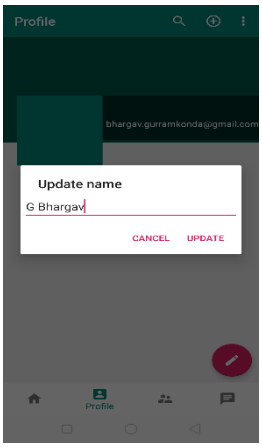
Logout Page:

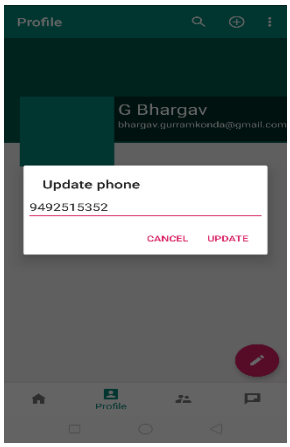
Search View:

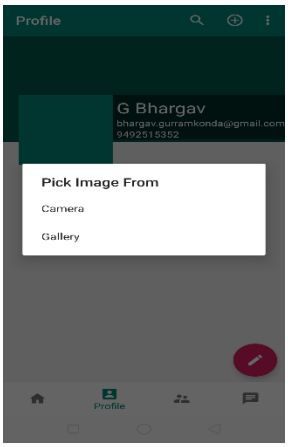


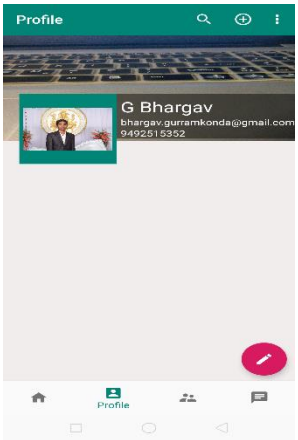
Edit Profile Page:





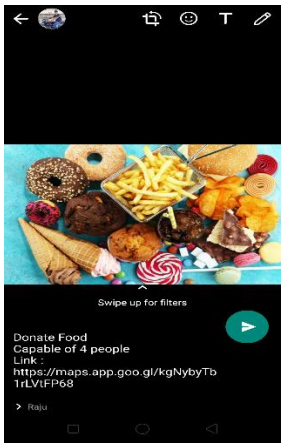




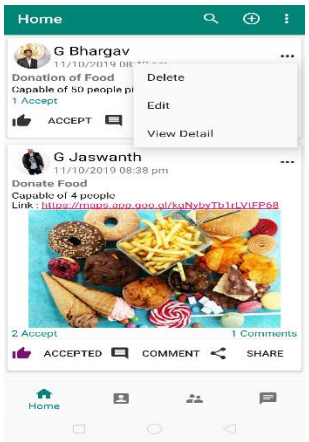


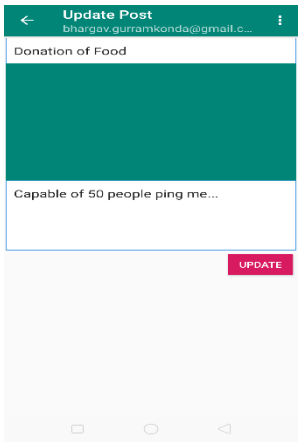
Comment Page:

Share Page:

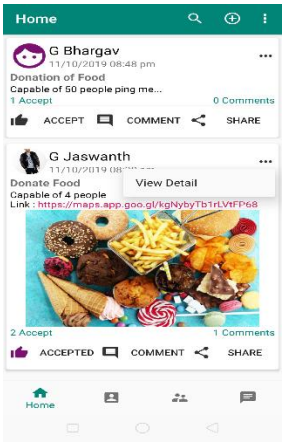


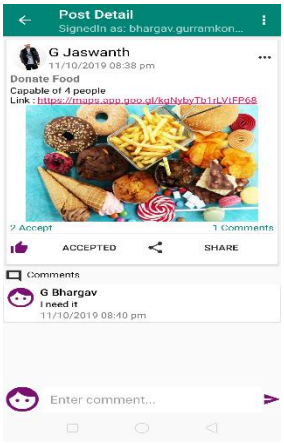
Edit Post:



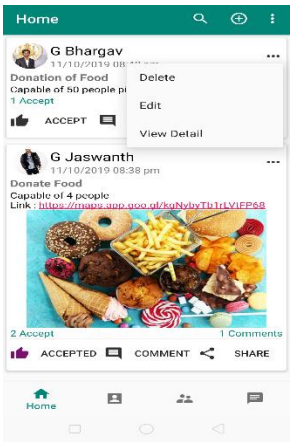


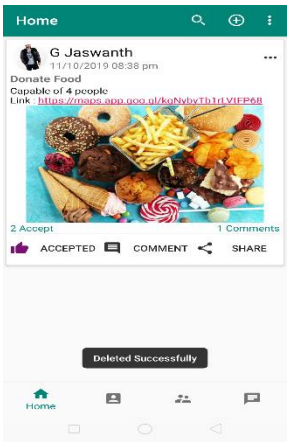
View Detail:



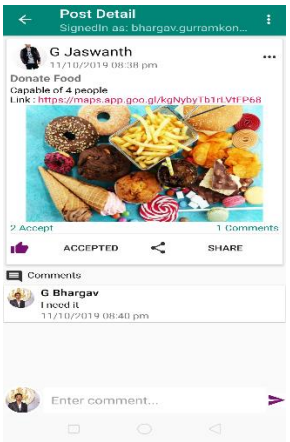


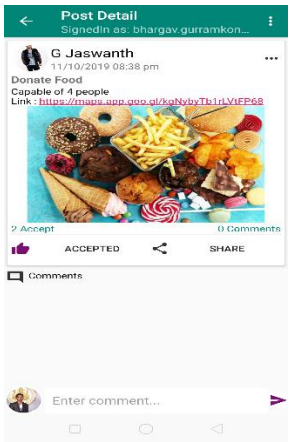
Delete Post:





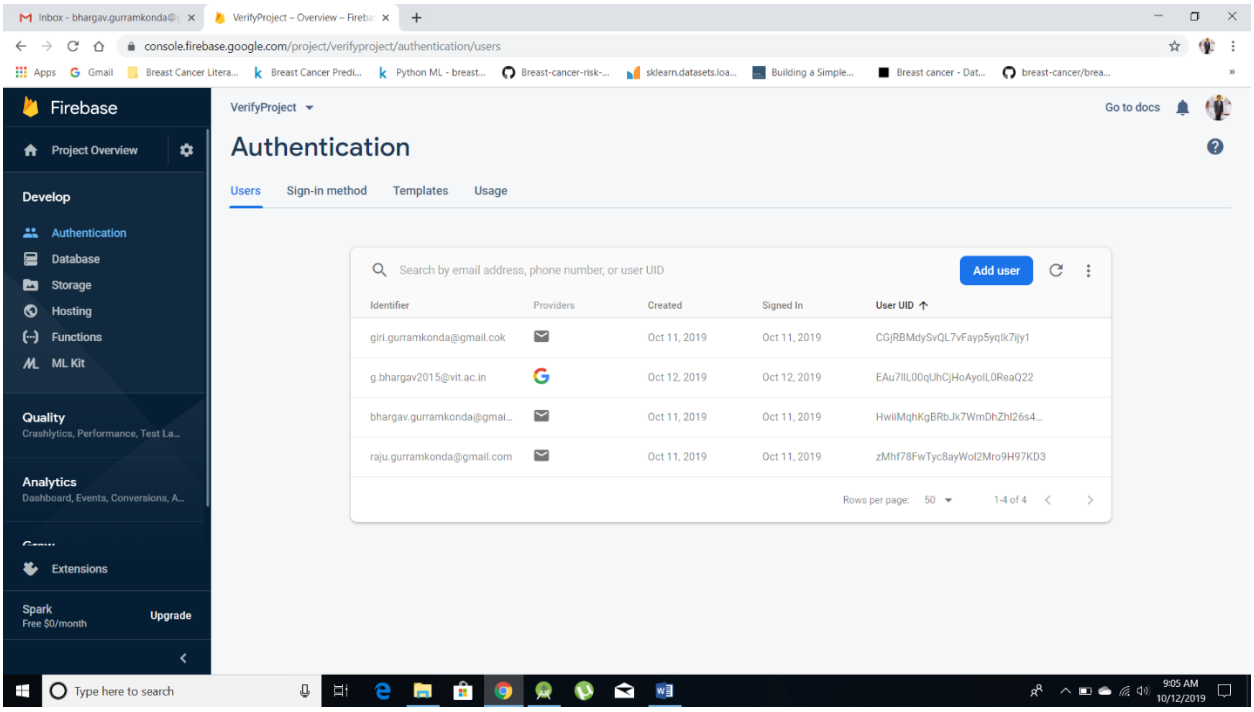
Delete Comment:



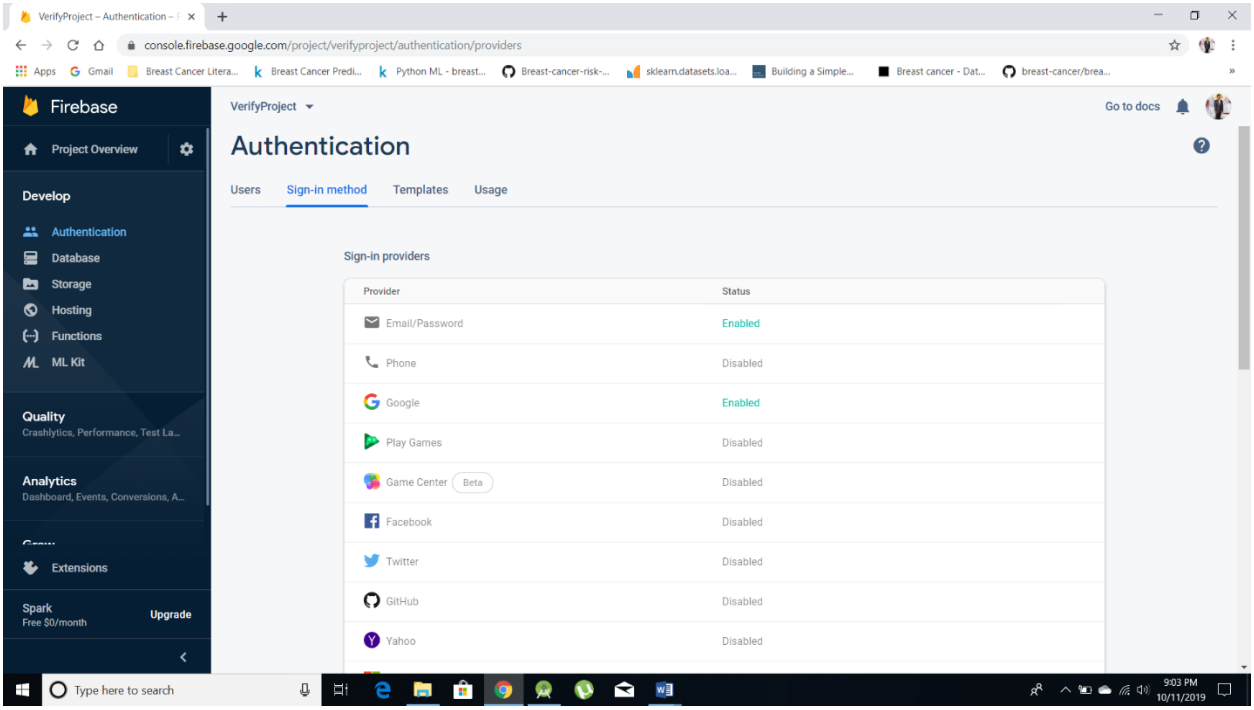


Database Output:

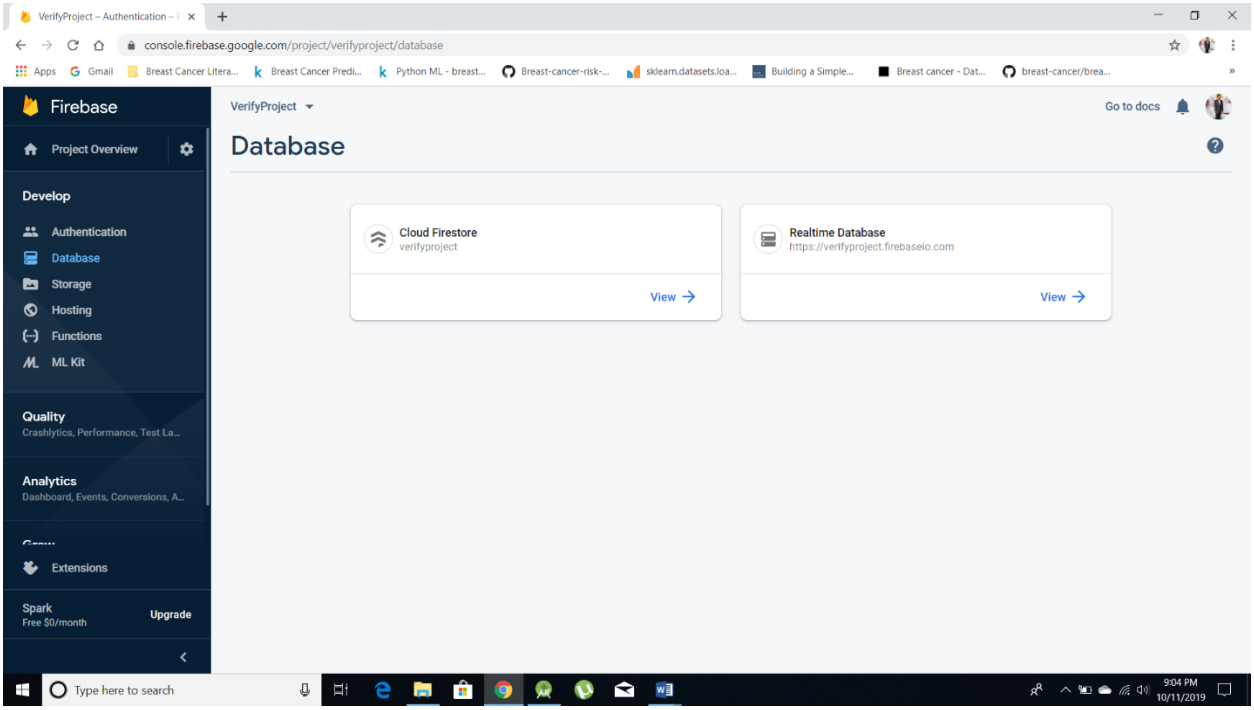
Registered Users :



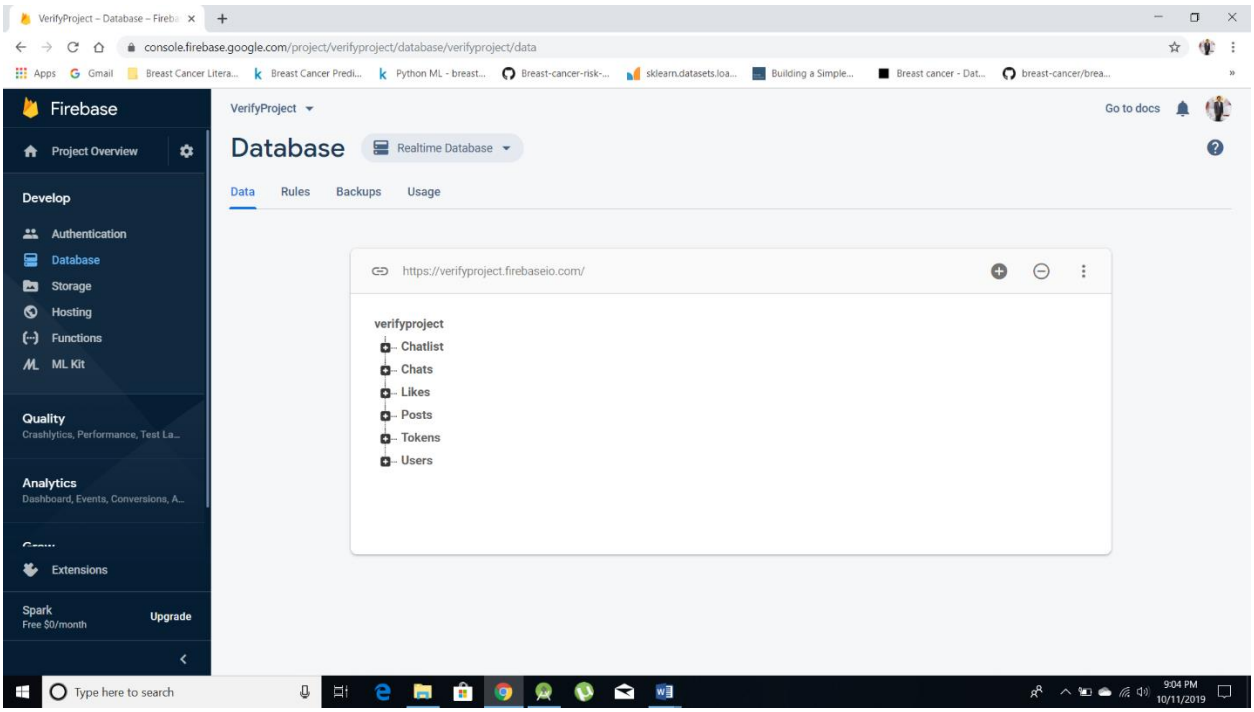
Authentication Rules:



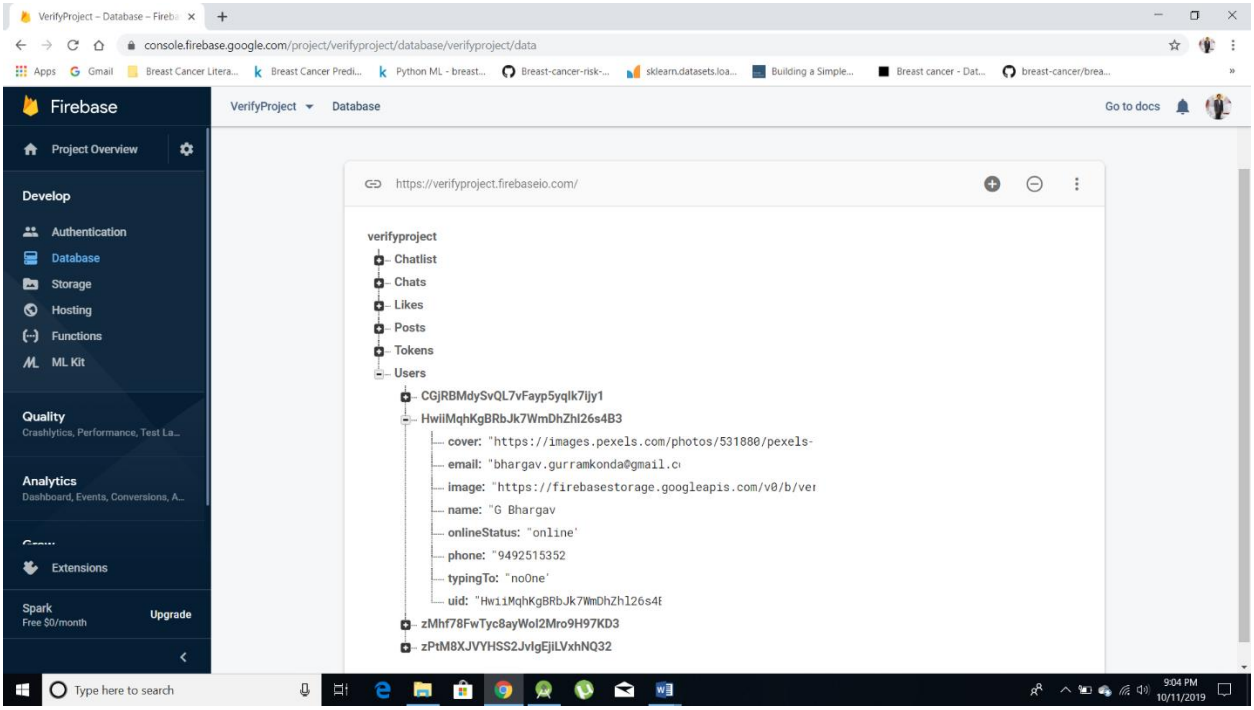
Database Types (avaliable):



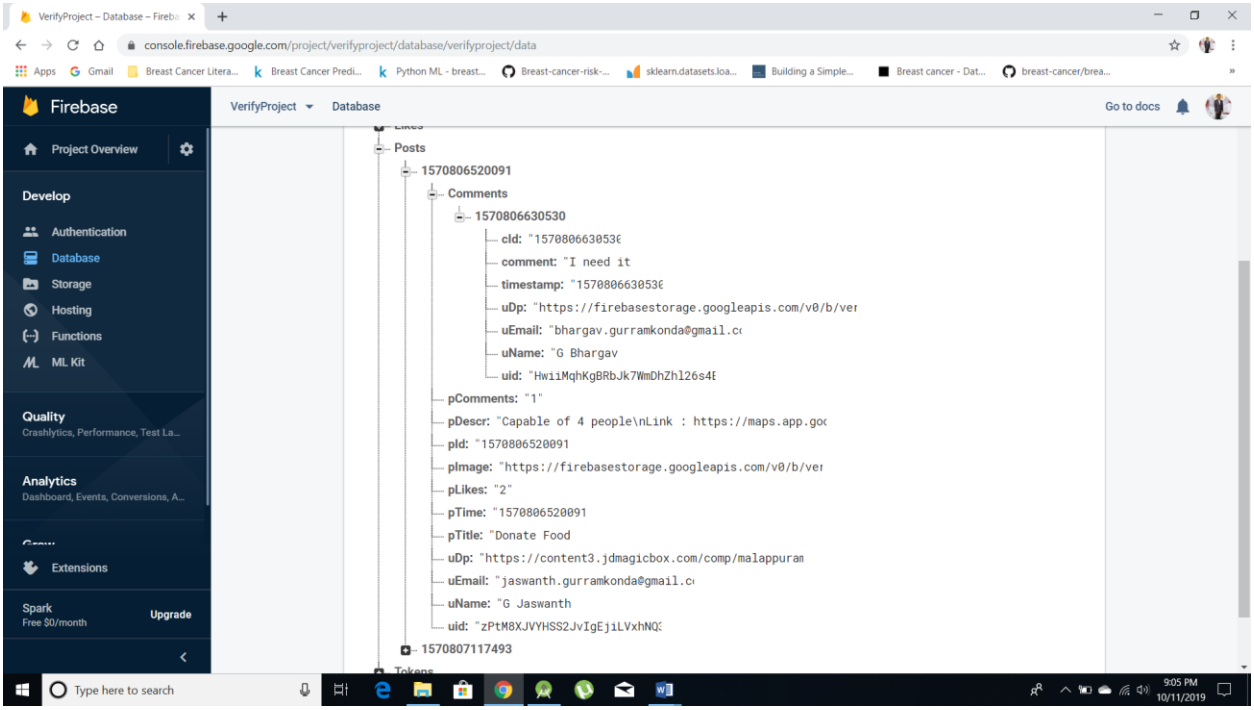
Databases:



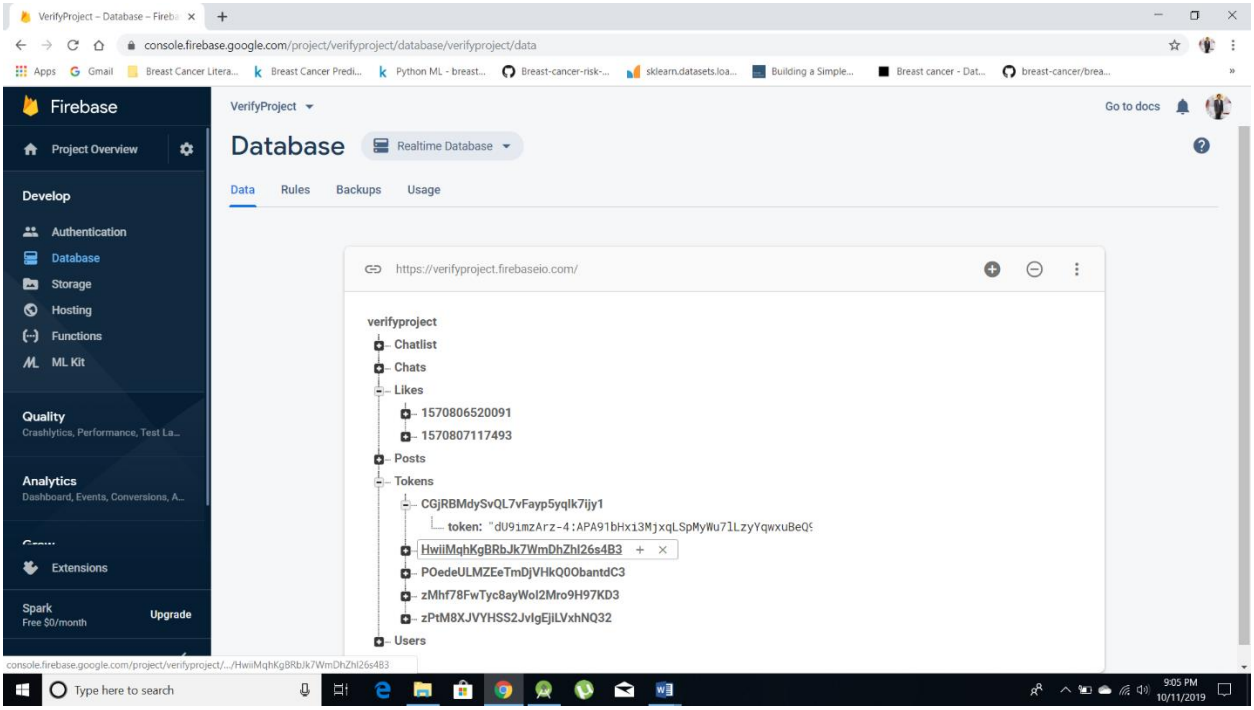
Databases – Registered Users:



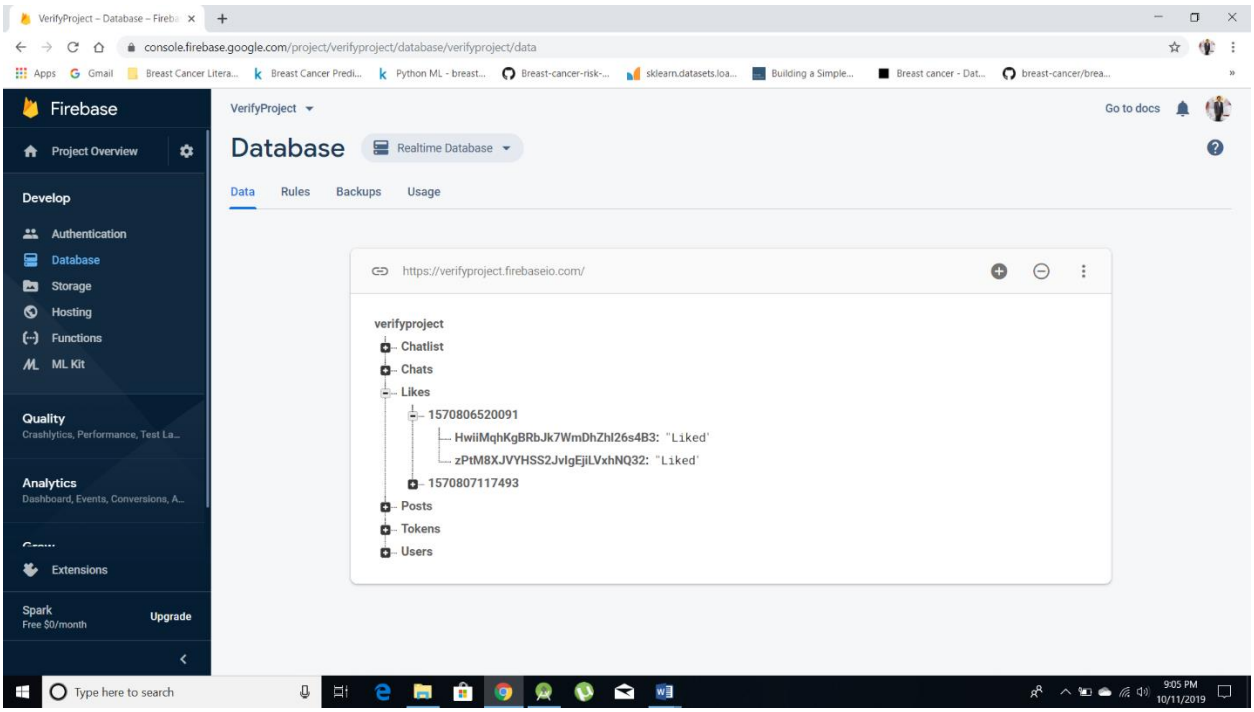
Databases – Posts and Comments:



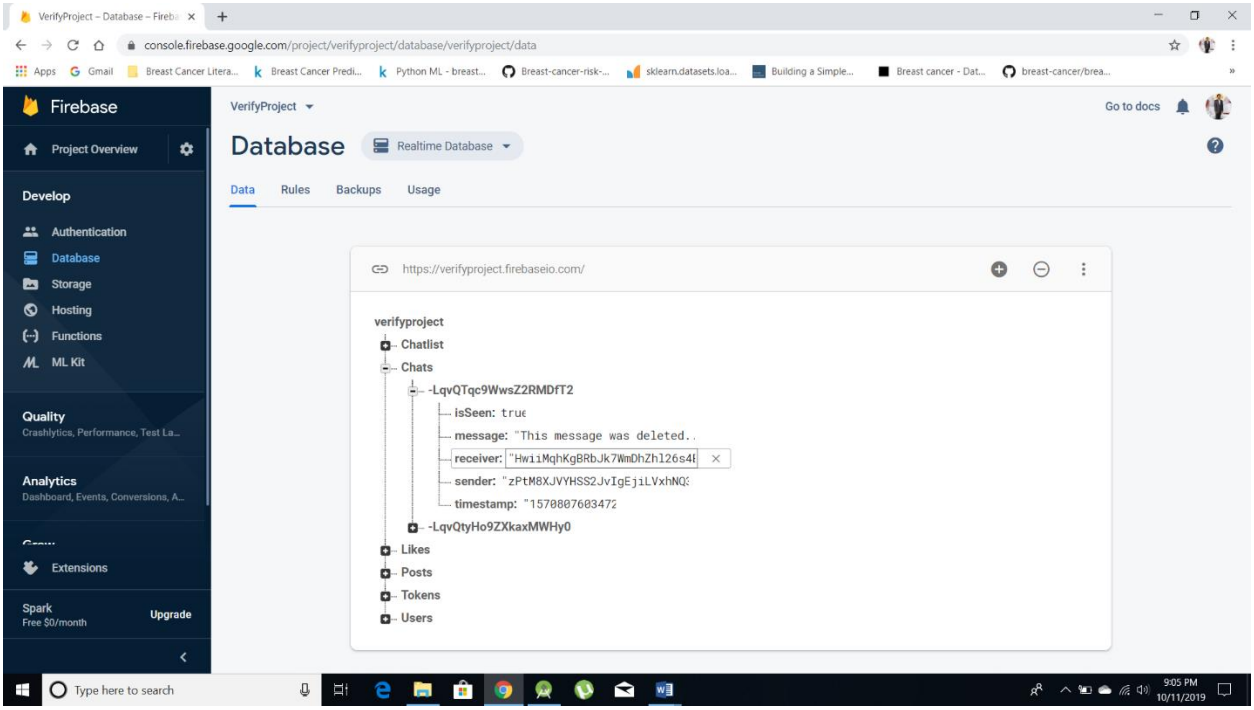
Databases – Tokens:



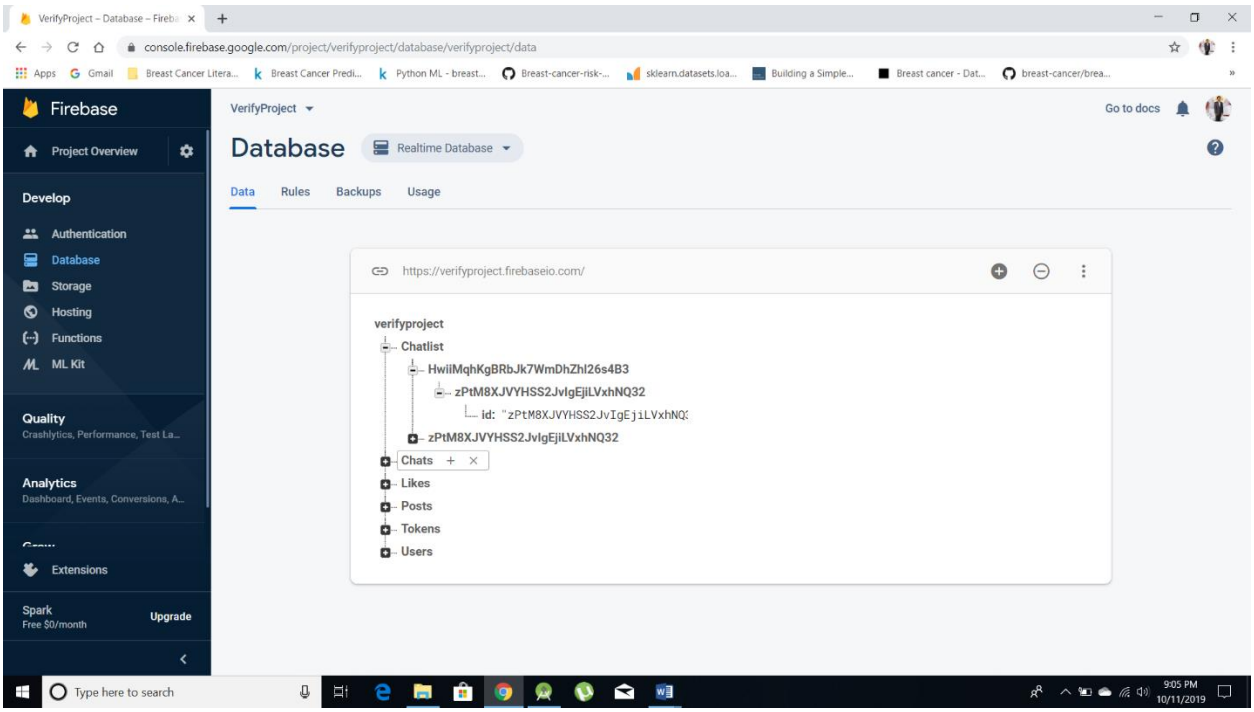
Databases – Accept:



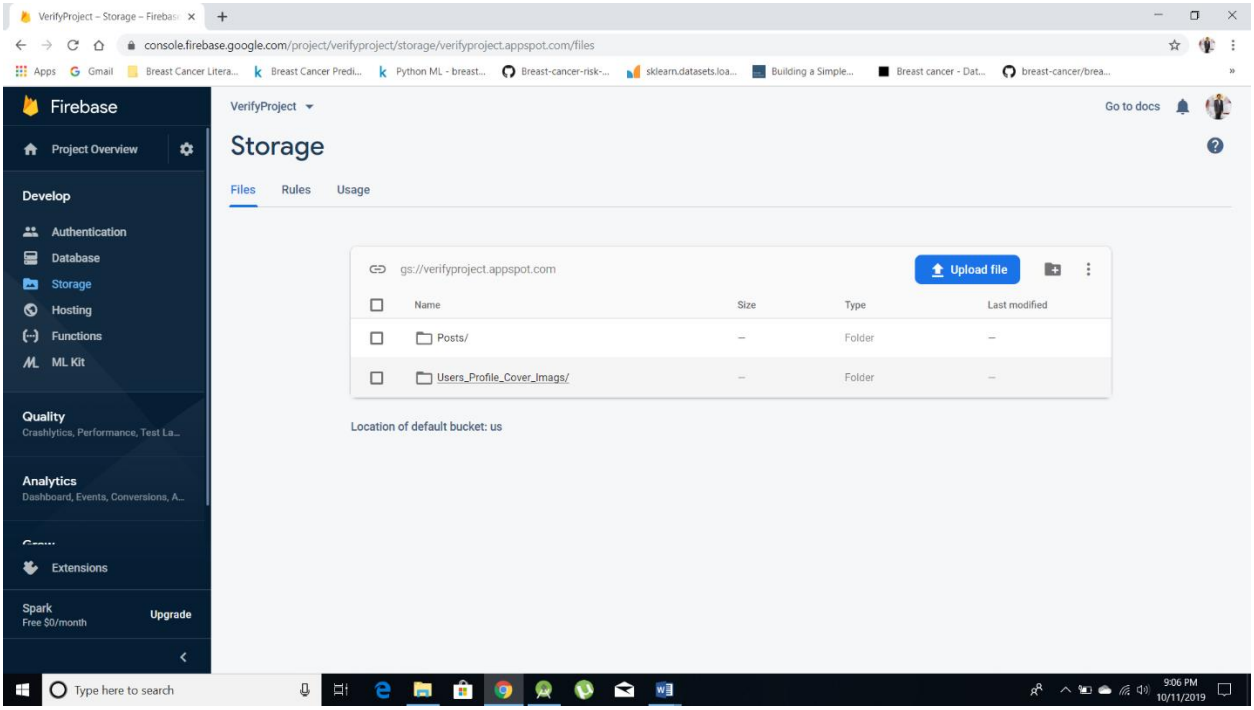
Databases – Chats:



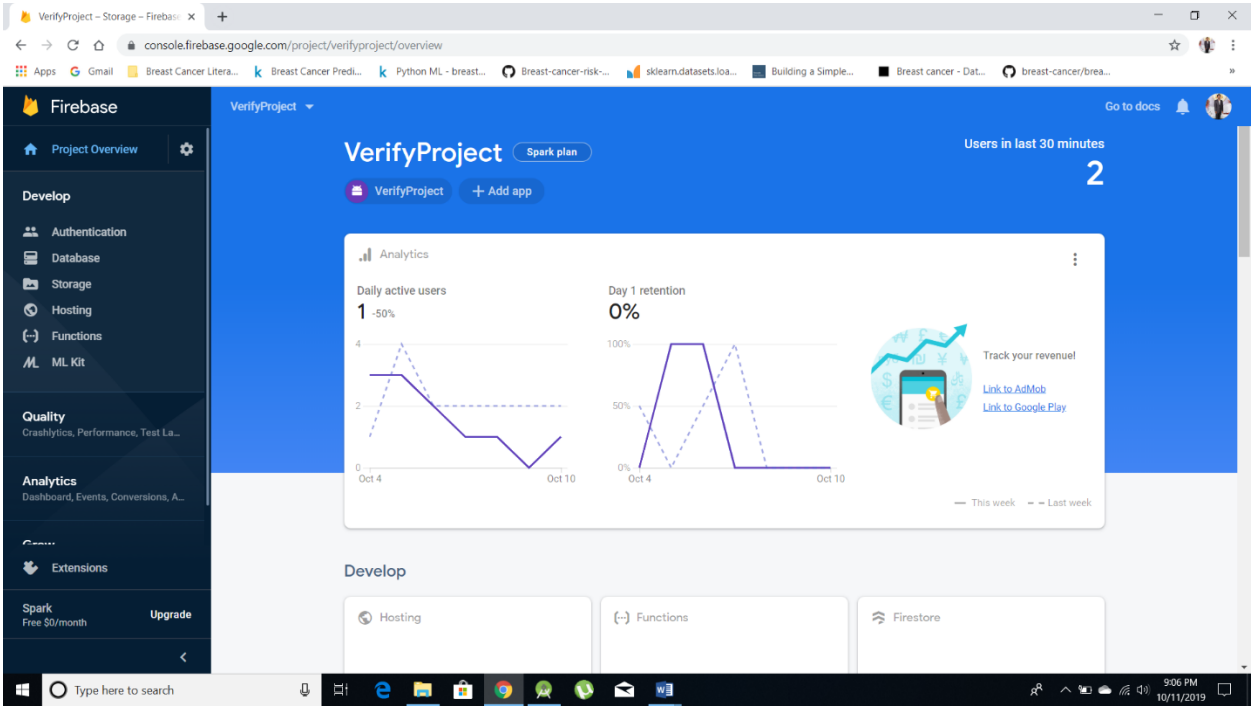
Databases – Chatlist:



Storage (storing of images):



Analytics:



CHAPTER – 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion:

Here concluding that "**Donate Food**" is a helpful android application which can avails free and useful thing if it is go for publishing and reaches maximum Donators.

The proposed application shall reduce food wstage and also fulfil other requirements like food etc., of needy organizations.

The development of this product surely prompts many new areas of investigation. This product has wide scope of implementation by making it live. Moreover this product creates many benefits for the business and the community. By taking it online it will help many people throughout the city by donating food daily.

Hunderds of thousands of tons of food are either lost or wasted while millions of people suffer from malnutrition. A plausible intitative is the food donation portal in which large retail chains and potentially other organizations can donate food. This food is collected and delivered by Third party vendor in need. Food donation portal will help thousand of people that suffer from starvation and also consume food that are wasted with no reason. As consequence, research and actions are needed to improve the efficiency of food donation portal.

7.2 Future Work:

In Future we give the formal security analysis and comprehensive performance analysis. Specifially, the main contributions of our paper are shown as follows:

- Multiple Language Support
- Better UI Design
- Security and Privacy
- Inbuilt GPS Design

CHAPTER – 8

REFERENCES

[1] Federal Information & News Dispatch, Inc. (20 December 2011). Congressmen Davis and Levin introduce bipartisan food donation bill – Rep. Geoff Davis (R-KY) News Release. Congressional Documents & Publications.

[2] U.S. Congressional Record. (March 18, 2011). House of Representatives Ways and Means Subcommittee on Select Revenue Measures.

[3] U.S. House of Representatives, Committee on Ways and Means, Subcommittee on Human Resources & Subcommittee on Select Revenue Measures, 107th Congress, First Session. (14 June 2001). Hearing on H.R. 7, The "Community Solutions Act of 2001," Statement of Bill Reighard, President, Food Donation Connection, Newport, Virginia.

[4] National Restaurant Association. (2011). Sustainability/ Recycling brochure.

[5] States News Service. (Sue Hensley & Annika Stensson). (14 September 2009). National Restaurant Association announces partnership to relieve hunger and reduce food waste in America.

[6] Harvest Support Network website. (2012).

[7] National Restaurant Association. (14 April 2011). Donating excess food can save dollars.

[8] Darden Restaurants website (2012). "Employee Engagement". Darden Restaurants.

[9] Bloom, J. (15 September 2009). Now serving: More donations. Wasted Food blog.

[10] National Restaurant Association website. NRA Show 2012.

[11] Larson, J. (28 October 2009). Turn surplus food into tax savings to help end hunger. National Restaurant Association webinar presentation.