PROJECT REPORT ON:

“AGRO BUDDY”

By,
Ashish Patel

ABSTRACT

India has been a primarily agricultural society since ancient times. Even now, though agriculture contributes a third highest share to the Gross Domestic Product of our nation, almost 60% of the population is engaged in some form of agricultural activity. This reflects a massive shortcoming in our agricultural process. Ideally India should be the biggest exporter of grains of pulses because of the farmer population. When one tries to look at the problems plaguing the Indian agricultural process, from top to bottom, we saw a lot of the causes like corruption, out-dated agricultural practices, and insufficient incentives from the government, landlords, middlemen etc. The application provides a new technique for both farmers and buyers to communicate over a single social medium. AGRO BUDDY aims at providing Freedom for farmers to sell their produce at the price they want to and also have a direct interaction with the consumers.

A Farmer can list and price his/her produce on our application to begin a new venture .The app provides very basic user interface for the farmers to work with. It also has many features withwhich a farmer can keep track of his daily affairs. The buyer or the consumer on the other hand will get a dedicated user interface for his/her affairs. A commercial consumer can list the crops which he/she wants to buy and start a dialogue with the farmers. Two types of consumers can register themselves, commercial and non-commercial. Therefore, AGRO BUDDY aims to bridge the gap between consumers and farmers by providing an easy to use android application for convenient agricultural commerce.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Android is one of the most popular operating systems for mobiles**.** AGRO BUDDY application is a flexible, easy to use and securely designed to benefit the farmers and consumers.It is used to digitize the market of retail selling and wholesale all in one single android mobile phone and manage them efficiently.

Because of third party vendors, farmers never get to price their produce or get any profits out of it. Consumers on the other hand have to pay more price to the third party vendors. With AGRO BUDDY the whole ecosystem of third party vendors will be restricted from entering one way business between consumers and farmers.

1.2 MOBILE APPLICATION DEVELOPMENT

Mobile app development is the act or process by which a mobile app is developed for mobile devices, such as personal digital assistants, enterprise digital assistants or mobile phones. These applications can be pre-installed on phones during manufacturing platforms, or delivered as web applications using server-side or client-side processing (e.g., JavaScript) to provide an "application-like" experience within a Web browser. Application software developers also must consider a long array of screen sizes, hardware specifications, and configurations because of intense competition in mobile software and changes within each of the platforms. Mobile app development has been steadily growing, in revenues and jobs created. A 2013 analyst report estimates there are 529,000 direct *app economy* jobs within the EU then 28 members (including the UK), 60 percent of which are mobile app developers.

1.3 OBJECTIVES

The objectives of this project are:

- To provide an easy to use way for farmers to sell their crops
- To restrict corruption and middle-men like APMC
- To enable consumers to buy fresh crops at low prices
- To ensure commercial consumers bulk produce at much cheaper rates
- To give a much needed boost to HOPCOMs department
- To introduce all farmers to technology and give them freedom

1.4 APPLICATIONS OF MOBILE APPLICATION DEVELOPMENT

FARMER:

- Adds the products to be sold and prices them.
- Can access the list of requests given by non-commercial consumers

COMMERCIAL CONSUMER:

- Can read the list of produce given by farmers
- Buy the products

NON COMMERCIAL CONSUMER:

- Can request the products farmers usually produce as they buy in small-scale
- Requests need to be accepted by farmer to complete trade

1.5 THE MOBILE APPLICATION DEVELOPMENT LIFECYCLE

There are two interlinked core components of a mobile application: 1) the mobile application "Front-End" that resides on the mobile device, and 2) the services "Back-End" that supports the mobile front-end.

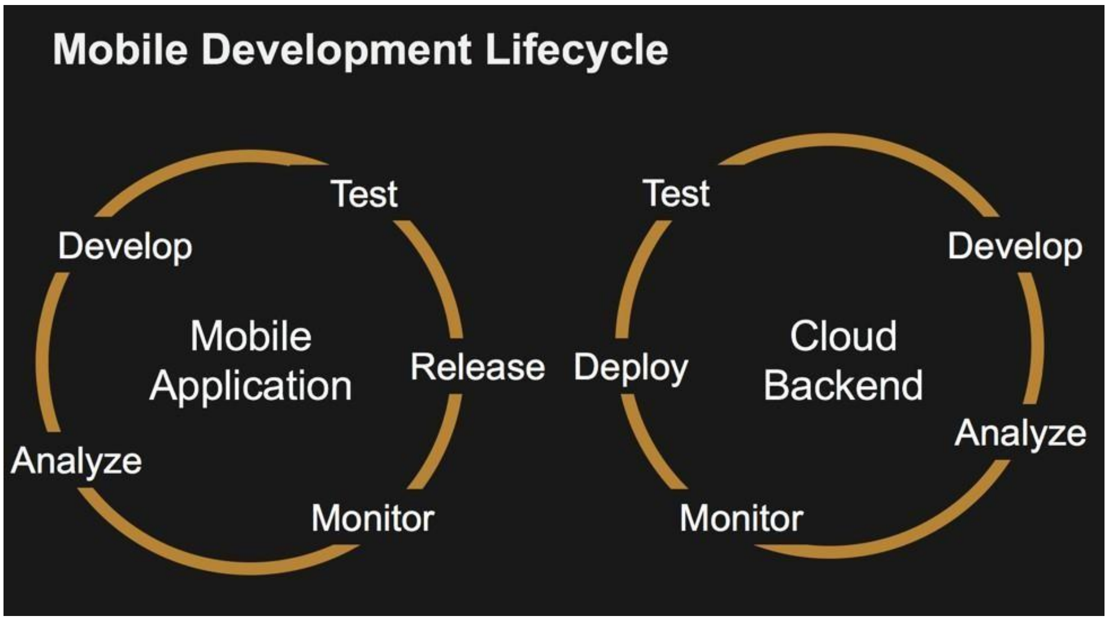


Fig 1.1 Mobile Development Cycle

Front-end v/s Back-end

In the early days of the modern smartphone applications era, mobile applications went through a similar evolution as first websites. At first, the applications and sites where wholly contained within themselves and acted as little more than static advertisements for the brand, company, product, or service.

However, as connectivity and network capabilities improved, the applications became increasingly connected to sources of data and information that lived outside of the app itself, and the apps became increasingly dynamic as they were able to update their UI and content with data received over the network from queries to data sources.

As a result, the mobile front-end applications increasingly rely on and integrated with back-end services which provide data to be consumed through the mobile front-end. Such data can include, for example, product information for e-commerce apps or flight info for travel and reservation apps. For a mobile game, the data may include new levels or challenges and scores or avatars from other players.

How Front-end 'Talks' to the Back-end:

The mobile front-end obtains the data from the back-end via a variety of service calls suchas APIs. In some cases, these APIs may be owned and operated by the same entity developing the mobile application. In other cases, the API may be controlled by a third party and access is granted to the mobile application via a commercial arrangement.

For example, a developer may obtain social media or advertising content by making calls to media or advertising company services. In this case, a developer may have to sign a contract in order to obtain credentials and a key that grants access to the API and governshow that developer can use it, how much it will cost, or how frequently it may be called, or how much data can be requested over what time period.

1.6 LITERATURE SURVEY

AUTHOR	PUBLICATION	YEAR	ABSTRACT
Soumalya Ghosh, A. B. Garg, Sayan Sarcar, P.S.V.S Sridhar, Ojasvi Maleyvar, and Raveesh kapoor	IEEE Students' Technology Symposium	2014	Krishi-Bharati is an interface in which users can only access their intended agricultural information from internet and local repository through icon based interaction. For this, user has to select or click the proper icon shown in the interface.
Divya Sawant, Anchal Jaiswal, Jyoti Singh, Payal Shah	IEEE Bombay Section Signature Conference (IBSSC)	2019	The common problem existing among the Indian farmers today is that they fail to choose the right crop based on their region specifications and yield history. Hence they face a serious setback in productivity.
Manav Singhal , Kshitij Verma , Anupam Shukla	IEEE Students' Technology Symposium	2016	Krishi-ville is Android based mobile application which would provide all the facilities to the farmers related to their agricultural activities. It would be helping them in getting the weather updates and they can also access the news related to agriculture and farms.

AUTHOR	PUBLICATION	YEAR	ABSTRACT
S. Mokshapathy	International Journal of Horticulture, Vol. 3, No. 20	2013	HOPCOMS is handling about 90-100 M T of fruits and vegetables every day, and has made a significant impact on both consumers and producers as all the payments are made immediately.
H. M. Chandrashekar	International NGO Journal Vol. 6 (5), pp. 122-132,	2011	94% of farmers strongly agreed to sell more fruits and vegetables at HOPCOMS. It had a great impact for farmers and consumers since both made profits.
Pranav Shriram, Sunil Mahamane	3rd International Conference on Inventive Computation Technologies (ICICT)	2018	The system lets the farmers to sell goods at a reasonable price and makes business even fair and transparent. Consumers are the opposite side of the same coin.

CHAPTER 2

REQUIREMENTS

To demonstrate the concept of AGRO BUDDY and demonstrate it's working using the following Hardware and Software Requirements:

2.1 SOFTWARE REQUIREMENTS

1. Operating System: Windows 7/8/10 (32-bit or 64-bit)
2. Android SDK
3. Android Studio
4. Firebase

2.1.2 ANDROID SDK

The Android SDK provides you the API libraries and developer tools necessary to build, test, and debug apps for Android. The ADT bundle includes the essential Android SDK components and a version of the Eclipse IDE with builtin Android Developer Tools to streamline the Android app development. ADT bundle consists of following components for developing the application II. Eclipse ADT plugin.

- Android SDK Tools
- Android Platform-tools
- The latest Android platform
- The latest Android system image for the emulator

2.1.3 ANDROID STUDIO

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on Jet Brains IntelliJ IDEA software and designed Specifically, for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. The current stable version is 3.3, which was released in January 2019.

2.1.4 FIREBASE

Firebase is a mobile and web application development platform developed by Firebase, Inc. in 2011, then acquired by Google in 2014.As of October 2018, the Firebase platform has 18 products, which are used by 1.5 million apps. Firebase Page | 10 provides a real-time database and backend as a service. The service provides application developers an API that allows application data to be synchronized across clients and stored on Firebase's cloud. Firebase Storage provides secure file uploads and downloads for Firebase apps, regardless of network quality. The developer can use it to store images, audio, video, or other user-generated content. Firebase Storage is backed by Google Cloud Storage.

2.2 HARDWARE REQUIREMENTS

1. Minimum 4 GB RAM (8GB recommended).
2. 5GB free disk space
3. USB 2.0 or higher
4. Android Device

CHAPTER 3

SYSTEM DESIGN

3.1 DATAFLOW DIAGRAM

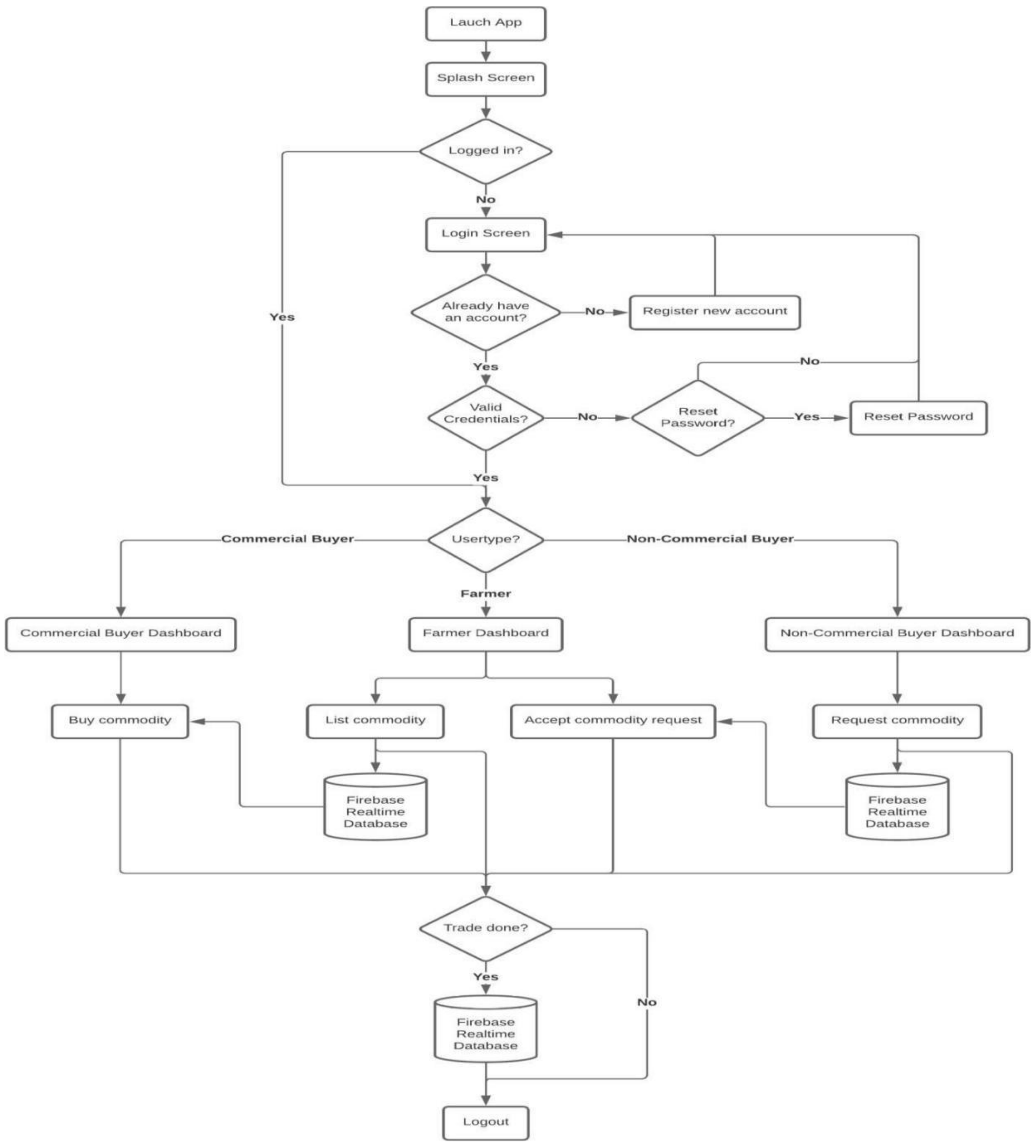


Fig 3.1 Dataflow Diagram

The figure shows the high level dataflow diagram of the AGRO BUDDY application. It shows the interactions of the two types of users with the database

When a user launches AGRO BUDDY app he/she will be able to see the home page that shows up when app opens. If the user is already having an existing account, they can directly login by providing valid credentials to use or else they have to register by creating a new account by providing valid details including which type of user he/she is preferring to be (Farmer/Commercial Buyer/Non- commercial buyer) and if the existing user has forgot his/her password they can reset the password by following required steps.

After logging in to their respective account they can see their details in the dashboard at the right top corner which includes their name, address, user type, Mobile number, email-id and also as an option where they can edit and update their profile.

The farmer is responsible to list his/her commodity so the buyer who is interested can buy the commodity and also when a non-commercial buyer requests for a commodity he can accept their request. The commercial buyer who usually buys the commodity in large quantity requests farmer by providing the commodity type, commodity name and preferred quantity and clicks on submit button. So, when the trade is complete after the buyer receives his/her commodity now can see their trade history which shows the details about the commodity name, quantity, amount of purchase and seller (Farmer) details.

The farmers homepage after he/she logs in to their account can see options such as list crop here the farmer lists the commodity which he/she is ready to sell to the buyers, list history-this talks about the commodity that farmer has listed previously to sell it, trade history talks about the commodity's which as been sold to the buyers.

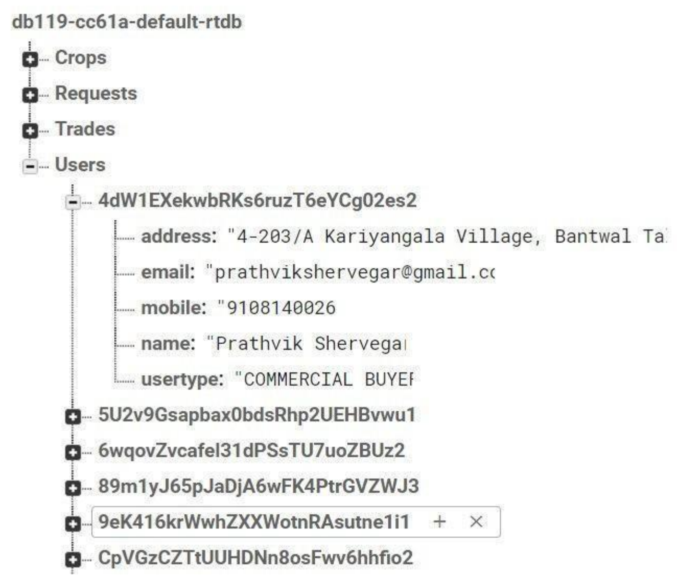
The non-commercial buyer when logins to the account can see his/her details in the dashboard. Here unlike the commercial buyer the non-commercial buyer cannot buy the commodity directly as the non-commercial buyers usually buys in less quantity have to request the farmer in order to buy the commodity. After the seeing the request from non-commercial buyer the farmer can trade if he/she is okay with it.

After the trade is done then data which includes all the details of the users and the trade details would be recorded in the firebase Realtime database. The user after the trade is done or not can logout of their respective account.

3.2 DATABASE DESIGN

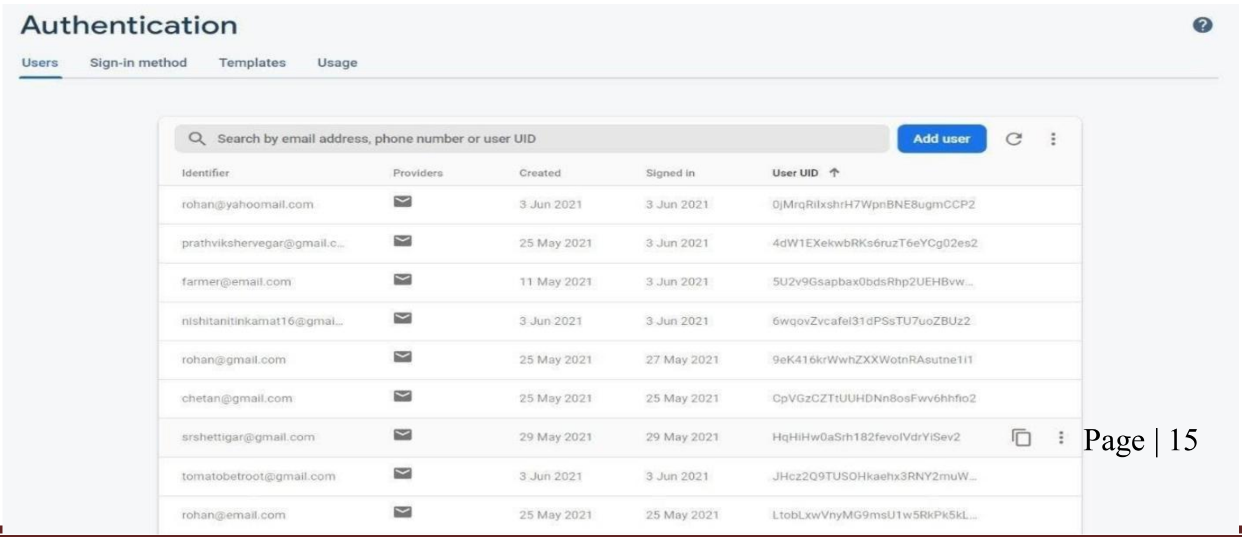
The database is designed using Google Firebase Console in which data is stored in popular data structure known as JSON tree (JavaScript Object Notation). Every time when the data transfer happens from client end, the information given to the UI is converted into JSON tree structure which is efficient and faster way to retrieve and store data

A snapshot of the data in the firebase database



```
db119-cc61a-default-rtdb
 ⊞ Crops
 ⊞ Requests
 ⊞ Trades
 ⊟ Users
    ⊟ 4dW1EXekwbRKs6ruzT6eYCg02es2
       address: "4-203/A Kariyangala Village, Bantwal Ta
       email: "prathvikshervegar@gmail.cc
       mobile: "9108140026
       name: "Prathvik Shervegar
       usertype: "COMMERCIAL BUYEF
    ⊞ 5U2v9Gsapbax0bdsRhp2UEHBvwu1
    ⊞ 6wqovZvcafel31dPSsTU7uoZBUz2
    ⊞ 89m1yJ65pJaDjA6wFK4PtrGVZWJ3
    ⊞ 9eK416krWwhZXXWotnRAsutne1i1  +  ×
    ⊞ CpVGzCZTtUUHDNn8osFwv6hhfio2
```

A snapshot of the data authenticated



Identifier	Providers	Created	Signed in	User UID ↑
rohan@yahoomail.com	✉	3 Jun 2021	3 Jun 2021	0jMrqRilxshrH7WpnBNE8ugmCCP2
prathvikshervegar@gmail.c...	✉	25 May 2021	3 Jun 2021	4dW1EXekwbRKs6ruzT6eYCg02es2
farmer@email.com	✉	11 May 2021	3 Jun 2021	5U2v9Gsapbax0bdsRhp2UEHBvw...
nishitanitinkamat16@gmai...	✉	3 Jun 2021	3 Jun 2021	6wqovZvcafel31dPSsTU7uoZBUz2
rohan@gmail.com	✉	25 May 2021	27 May 2021	9eK416krWwhZXXWotnRAsutne1i1
chetan@gmail.com	✉	25 May 2021	25 May 2021	CpVGzCZTtUUHDNn8osFwv6hhfio2
srshettigar@gmail.com	✉	29 May 2021	29 May 2021	HqHiHw0aSrh182fevolVdrYiSev2
tomatobetroot@gmail.com	✉	3 Jun 2021	3 Jun 2021	JHcz2Q9TUSOHkaehx3RNY2muW...
rohan@email.com	✉	25 May 2021	25 May 2021	LtobLxwVnyMG9msU1w5RkPk5kL...

CHAPTER 4
IMPLEMENTATION

4.1 SOLUTION APPROACH/METHODOLOGY

We are here using xml and java for the front end and firebase for the backend as a server.

4.1.1 FIREBASE

Firebase is a Backend-as-a-Service(BaaS) which started as a YC11 startup. It grew up into a next-generation app-development platform on Google Cloud Platform. Firebase (a NoSQLjSON database) is a real-time database that allows storing a list of objects in the form of a tree. We can synchronize data between different devices.

Google Firebase is Google-backed application development software which allows developers to develop **Android, IOS,** and **Web apps**. For reporting and fixing app crashes, tracking analytics, creating marketing and product experiments, firebase provides several tools.

- Firebase manages real-time data in the database. So, it easily and quickly exchanges the data to and from the database. Hence, for developing mobile apps such as live streaming, chat messaging, etc., we can use Firebase.
- Firebase allows syncing real-time data across all devices - iOS, Android, and Web - without refreshing the screen.
- Firebase provides integration to Google Advertising, AdMob, Data Studio, BigQuery DoubleClick, Play Store, and Slack to develop our apps with efficient and accurate management and maintenance.
- Everything from databases, analytics to crash reports are included in Firebase. So, the app development team can stay focused on improving the user experience.
- Firebase applications can be deployed over a secured connection to the firebase server.
- Firebase offers a simple control dashboard.
- It offers a number of useful services to choose from.

Firebase has a lot of pros or advantages. Apart from the advantages, it has disadvantages too. Let's take a look at these advantages and disadvantages:

Firebase real-time database feature is very easy to use. Once the Firebase and database dependency is added to the app, unstructured data can be added to database.

4.1.2 STORAGE

Image files can be stored in the app. The data stored is highly secured and is robust in nature, means it resumes from the last point if any network error occurs. The steps below are to be followed to use storage feature in Android application: added to the application.

4.1.3 FIREBASE AND ANDROID APP

An Android application has been developed for the demonstration of Firebase. In this app images along with strings are loaded to Firebase and retrieved from Firebase similar to Instagram. For the development of an Android app to demonstrate the use of Firebase, prototyping model has been followed. Steps for connecting App to Firebase

Step1: An account in the Firebase Login has to be created at https://www.firebase.com/login/ using the Google account.

Step2: Creating a new application on Firebase. Firebase creates a new application when one logs in for the first time. Also, at the bottom left corner, one can find an option to create a new application on the Firebase server. The app url has to be unique among all applications deployed on Firebase.

Step3: Next step is to add Firebase as a project dependency. Make changes to the following lines to the build.gradle file, which is located in the app’s project folder, and not the root folder. After adding any dependency, one has to make sure to sync the application. If there is any build error complaint about duplicate files then one can choose to exclude those files by adding the packaging Options directive to the build.gradle file: android

Step4: Next, add permissions to Android application, add network permission to the app, the same way it has been done for parse earlier. Now add the following line to the AndroidManifest.xml file: Firebase is a Backend-as-a-Service— BaaS— that started as an YC11 start up and grew up into a next-generation app-development platform on Google Cloud Platform.

4.1.4 JAVA

There are several ways to create apps for Android devices, but the recommended method for most developers is to write native apps using Java and the Android SDK. Java for Android apps is both similar and quite different from other types of Java applications. If you have experience with Java (or a similar language) then you'll probably feel comfortable diving right into the code and learning how to use the Android SDK to make your app run. But if you're new to programming or object oriented languages then you'll probably want to get familiar with the syntax of the Java language and how to accomplishbasic programming tasks before learning how to use the Android SDK.

4.2 INTRODUCTION TO ANDROID STUDIO

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013, at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0.

On May 7, 2019, Kotlin replaced Java as Google's preferred language for Android app development. Java is still supported, as is C++.
The following features are provided in the current stable version:

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- ProGuard integration and app-signing capabilities
- Template-based wizards to create common Android designs and components
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations

- Support for building Android Wear apps
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine

- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Android Studio supports all the same programming languages of IntelliJ (and CLion) e.g. Java, C++, and more with extensions, such as Go and Android Studio 3.0 or later supports Kotlin and "all Java 7 language features and a subset of Java 8 language features that vary by platform version." External projects backport some Java 9 features. While IntelliJ states that Android Studio supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android.

Once an app has been compiled with Android Studio, it can be published on the Google Play Store. The application has to be in line with the Google Play Store developer content policy These features includes requirements for IDE + Android SDK + Android Emulator.

- Windows: x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for a Windows Hypervisor;
- x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for a Windows Hypervisor;
- MacOS: ARM-based chips, or 2nd generation Intel Core or newer with support for Hypervisor.Framework;
- Linux: x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD processor with support for AMD Virtualization (AMD-V) and SSSE3;
- Windows: CPU with UG (unrestricted guest) support;
- Intel Hardware Accelerated Execution Manager (HAXM) 6.2.1 or later (HAXM 7.2.0 or later recommended).
-

The use of hardware acceleration has additional requirements on Windows and Linux:

- Intel processor on Windows or Linux: Intel processor with support for Intel VT-x, Intel EM64T (Intel 64), and Execute Disable (XD) Bit functionality;

- AMD processor on Linux: AMD processor with support for AMD Virtualization (AMD-V) and Supplemental Streaming SIMD Extensions 3 (SSSE3);
- AMD processor on Windows: Android Studio 3.2 or higher and Windows 10 April 2018 release or higher for Windows Hypervisor Platform (WHPX) functionality.
- For an attached webcam to work with Android 8.1 (API level 27) and higher system images, it must have the capability to capture 720p frames.

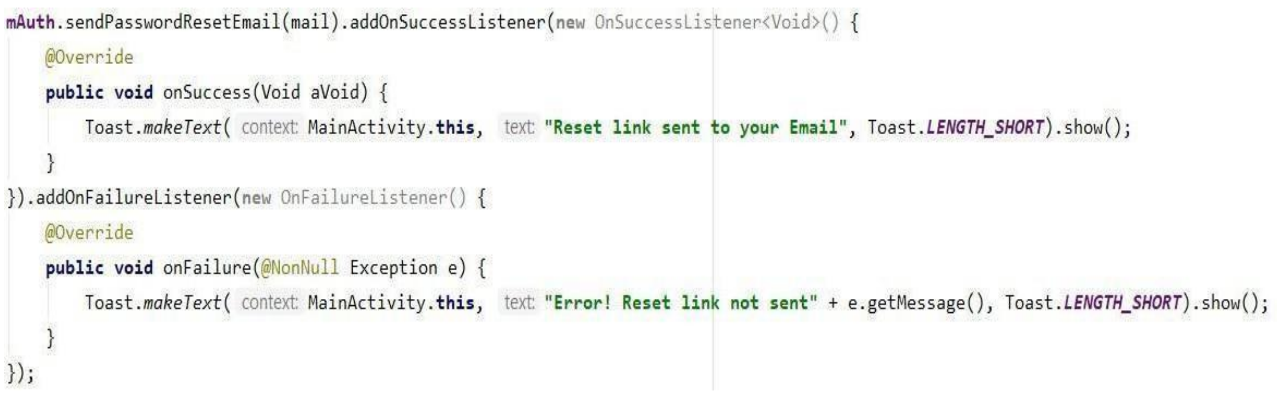
4.3 MODULES USED

android.os.Bundle

Bundle is a utility class that lets you store a set of name-value pairs. You will always find this import along with the import for Activity class because both onCreate() and onFreeze() methods take Bundle as a parameter. Into a Bundle object, you can put integers, longs, strings, arrays, etc along with the keys to identify them. When needed, these values can be obtained by using those keys.

So, as you can see, a bundle object is ideal for storing a set of state variables in the onFreeze method; and the same state variables can be read back in the onCreate method.

So, for example, in the snake game a series of state variables (direction, move delay, score, snake trail, etc) can be stored in a Bundle object (whether they are arrays, integers, strings, etc.).



```
mAuth.sendPasswordResetEmail(mail).addOnSuccessListener(new OnSuccessListener<Void>() {
    @Override
    public void onSuccess(Void aVoid) {
        Toast.makeText( context: MainActivity.this,  text: "Reset link sent to your Email", Toast.LENGTH_SHORT).show();
    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        Toast.makeText( context: MainActivity.this,  text: "Error! Reset link not sent" + e.getMessage(), Toast.LENGTH_SHORT).show();
    }
});
```

Fig 4.1 Code snippet of forgot password activity

The forgot password activity is used in case a user forgets their password. In case the password is forgotten, the reset link is sent to their registered email.

com.google.firebase

We can set up a Firebase project and register apps in the Firebase console (or, for advanced use cases, via the Firebase Management REST API or the Firebase CLI). When you set up a project and register apps, you need to make some organizational decisions and add Firebase-specific configuration information to your local projects.

For production apps, you need to set up a clear development workflow, which usually involves using multiple environments. Review our documentation on developer workflows, including general best practices and general security guidelines for setting up Firebase projects and registering apps to create your development workflow.

com.google.firebase.auth.FirebaseAuth

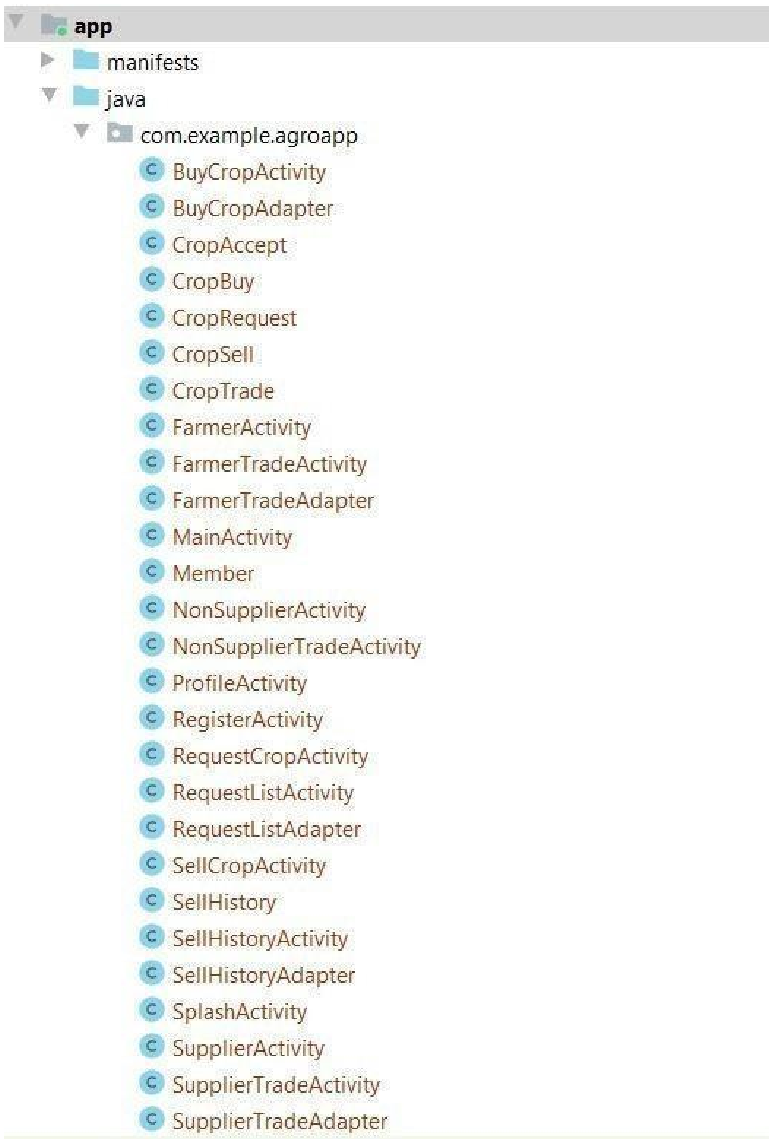
A DataSnapshot instance contains data from a Firebase Database location. Any time you read Database data, you receive the data as a DataSnapshot. DataSnapshots are passed to the methods in listeners that you attach with addValueEventListener (ValueEventListener) ,addChildEventListener(ChildEventListener) , or addListenerForSingleValueEvent(ValueEventListener) . They are efficiently-generated immutable copies of the data at a Firebase Database location. They can't be modified and will never change. To modify data at a location, use a DatabaseReference reference (e.g. with setValue(Object)).

java.util.ArrayList

Resizable-array implementation of the List interface. Implements all optional list operations, and permits all elements, including null. In addition to implementing the List interface, this class provides methods to manipulate the size of the array that is used internally to store the list. (This class is roughly equivalent to Vector, except that it is unsynchronized.)
The size, isEmpty, get, set, iterator, and listIterator operations run in constant time. The add operation runs in amortized constant time, that is, adding n elements requires O(n) time. All of the other operations run in linear time (roughly speaking). The constant factor is low compared to that for the LinkedList implementation.

4.4 IMPLEMENTATION CODE

Overview of all the project files:

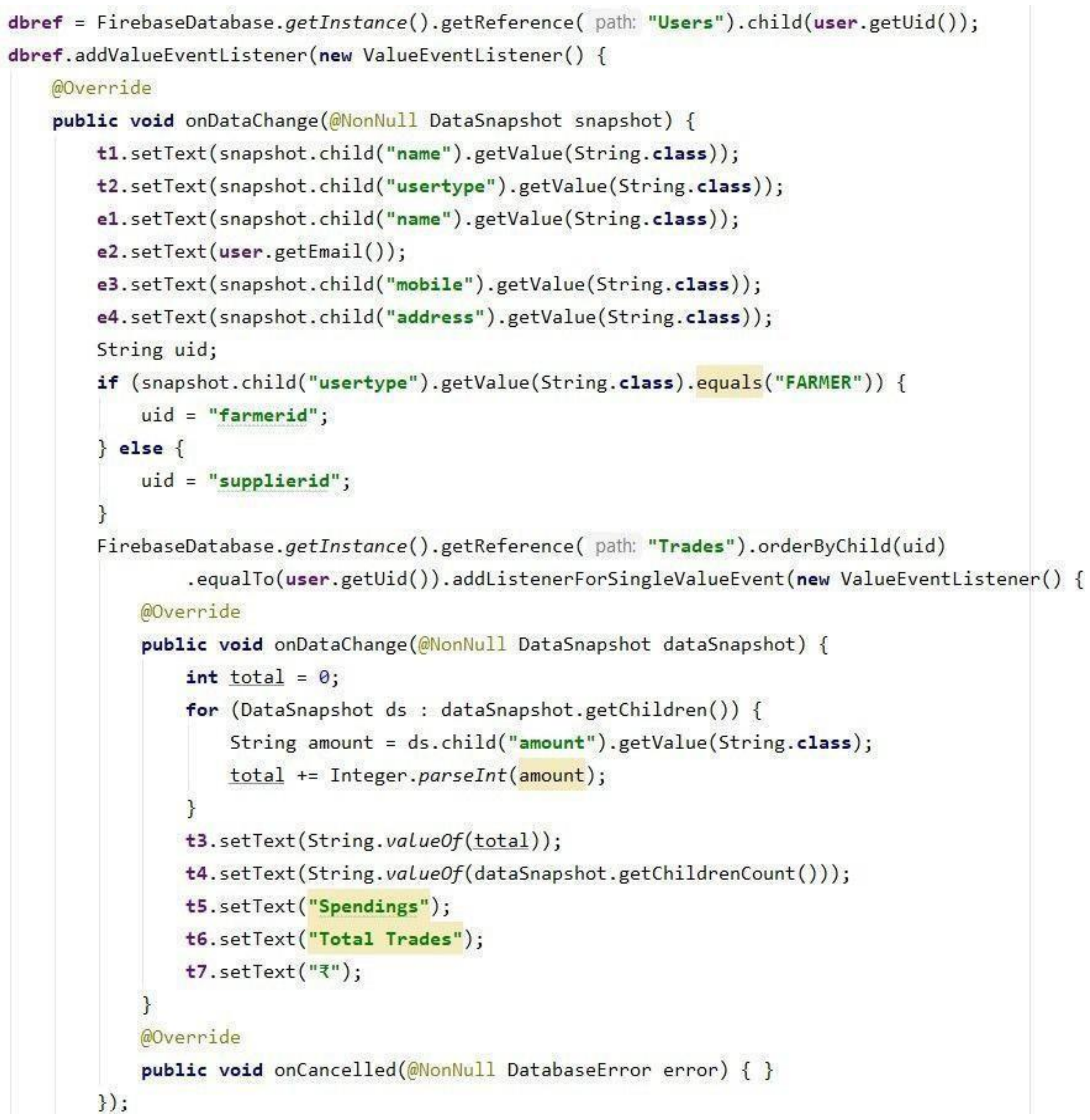


```
app
  manifests
  java
    com.example.agroapp
      BuyCropActivity
      BuyCropAdapter
      CropAccept
      CropBuy
      CropRequest
      CropSell
      CropTrade
      FarmerActivity
      FarmerTradeActivity
      FarmerTradeAdapter
      MainActivity
      Member
      NonSupplierActivity
      NonSupplierTradeActivity
      ProfileActivity
      RegisterActivity
      RequestCropActivity
      RequestListActivity
      RequestListAdapter
      SellCropActivity
      SellHistory
      SellHistoryActivity
      SellHistoryAdapter
      SplashActivity
      SupplierActivity
      SupplierTradeActivity
      SupplierTradeAdapter
```

Fig 4.2 List of all project files

This is the list of all the java activities which have been used in the AGRO BUDDY application. The classes have all been declared explicitly to avoid confusion and enable further improvements.

Profile Activity:

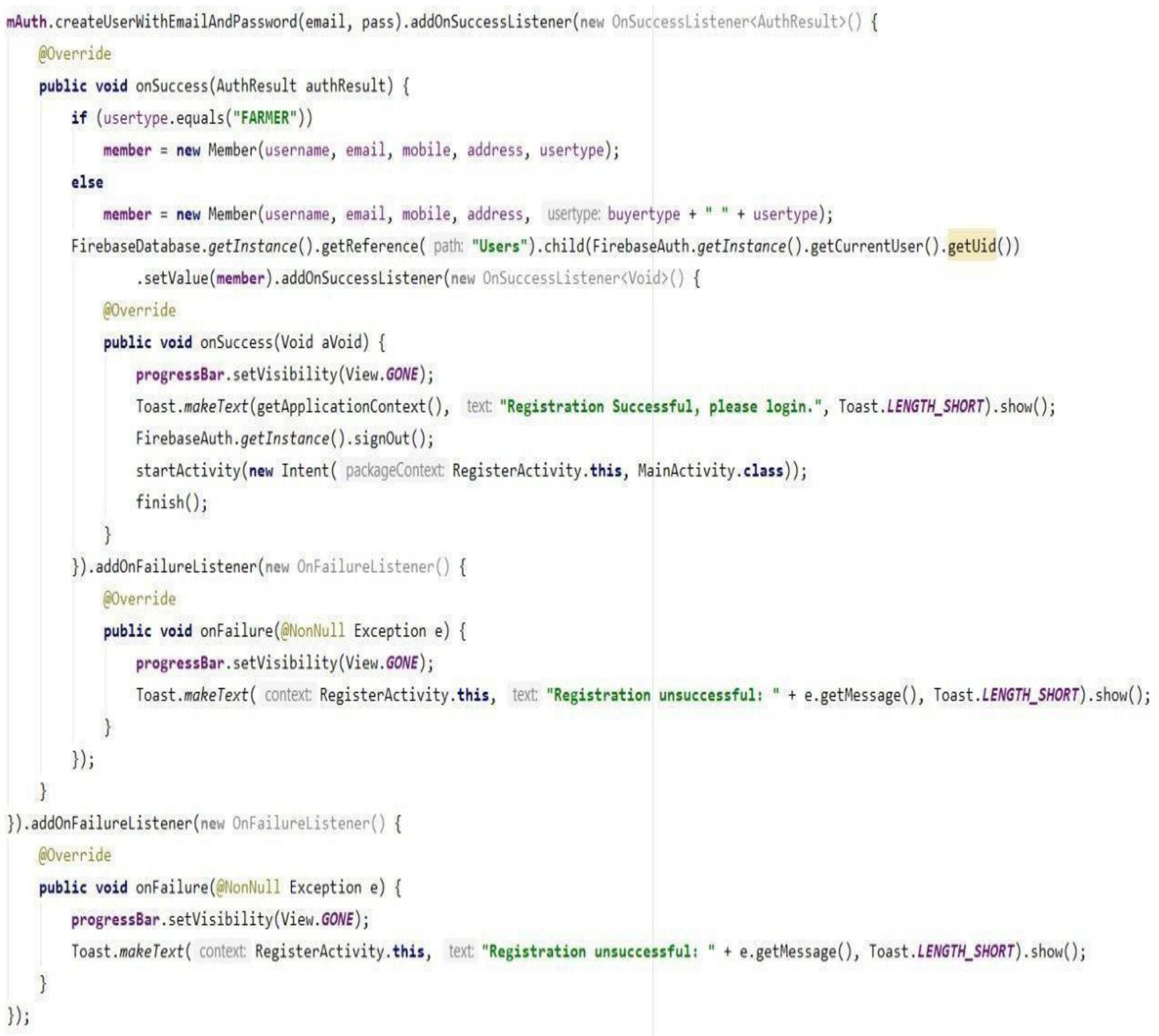


```
dbref = FirebaseDatabase.getInstance().getReference( path: "Users").child(user.getUid());
dbref.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        t1.setText(snapshot.child("name").getValue(String.class));
        t2.setText(snapshot.child("usertype").getValue(String.class));
        e1.setText(snapshot.child("name").getValue(String.class));
        e2.setText(user.getEmail());
        e3.setText(snapshot.child("mobile").getValue(String.class));
        e4.setText(snapshot.child("address").getValue(String.class));
        String uid;
        if (snapshot.child("usertype").getValue(String.class).equals("FARMER")) {
            uid = "farmerid";
        } else {
            uid = "supplierid";
        }
        FirebaseDatabase.getInstance().getReference( path: "Trades").orderByChild(uid)
                .equalTo(user.getUid()).addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                int total = 0;
                for (DataSnapshot ds : dataSnapshot.getChildren()) {
                    String amount = ds.child("amount").getValue(String.class);
                    total += Integer.parseInt(amount);
                }
                t3.setText(String.valueOf(total));
                t4.setText(String.valueOf(dataSnapshot.getChildrenCount()));
                t5.setText("Spendings");
                t6.setText("Total Trades");
                t7.setText("₹");
            }
            @Override
            public void onCancelled(@NonNull DatabaseError error) { }
        });
```

Fig 4.3 Profile Activity

This is the main java file for showing a user's profile in the dashboard section. Since none of the data is stored locally, appropriate calls have been made to the database to retrieve information.

Registration Page:

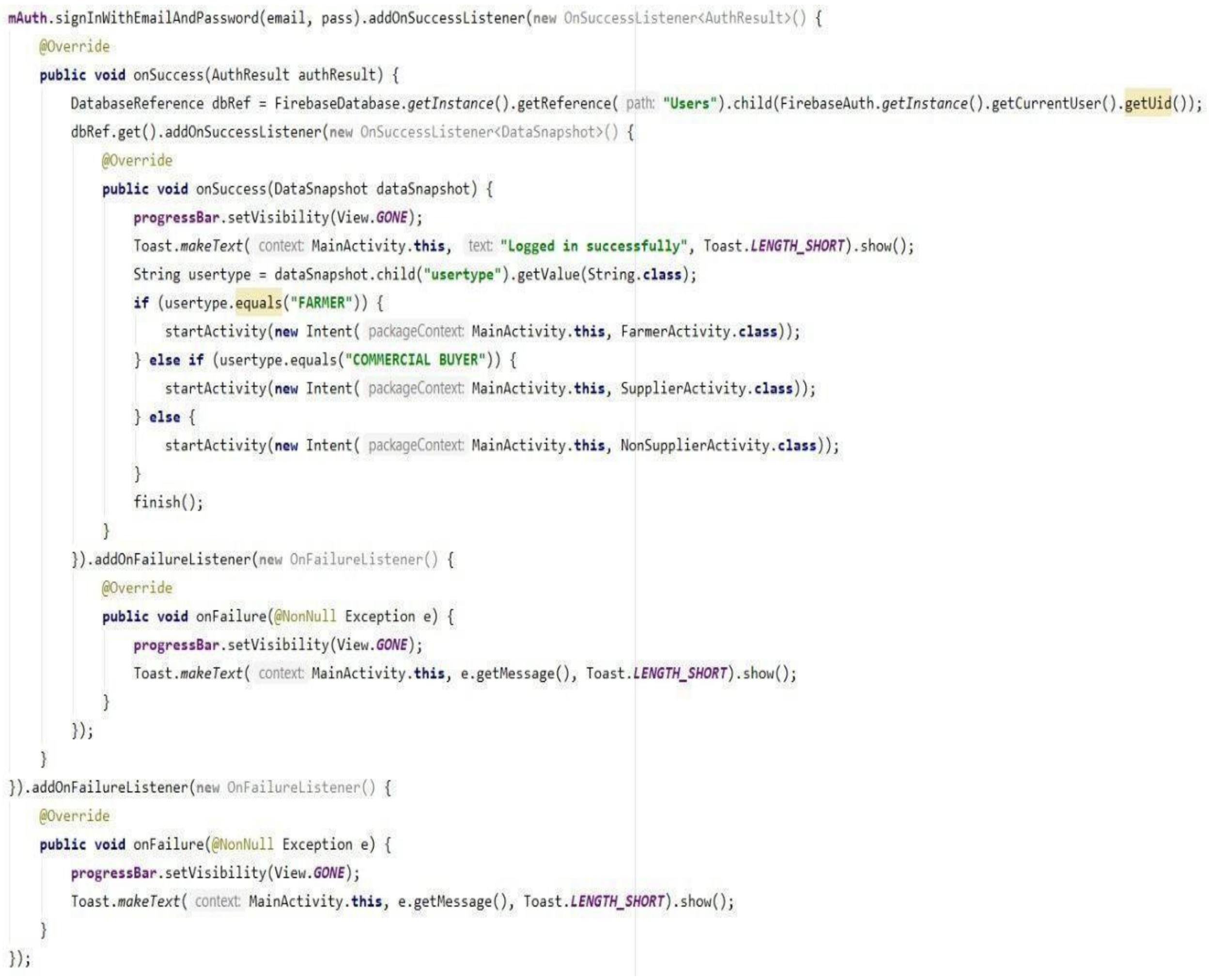


```
mAuth.createUserWithEmailAndPassword(email, pass).addOnSuccessListener(new OnSuccessListener<AuthResult>() {
    @Override
    public void onSuccess(AuthResult authResult) {
        if (usertype.equals("FARMER"))
            member = new Member(username, email, mobile, address, usertype);
        else
            member = new Member(username, email, mobile, address,  usertype: buyertype + " " + usertype);
        FirebaseDatabase.getInstance().getReference( path: "Users").child(FirebaseAuth.getInstance().getCurrentUser().getUid())
                .setValue(member).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                progressBar.setVisibility(View.GONE);
                Toast.makeText(getApplicationContext(),  text: "Registration Successful, please login.", Toast.LENGTH_SHORT).show();
                FirebaseAuth.getInstance().signOut();
                startActivity(new Intent( packageContext: RegisterActivity.this, MainActivity.class));
                finish();
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                progressBar.setVisibility(View.GONE);
                Toast.makeText( context: RegisterActivity.this,  text: "Registration unsuccessful: " + e.getMessage(), Toast.LENGTH_SHORT).show();
            }
        });
    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        progressBar.setVisibility(View.GONE);
        Toast.makeText( context: RegisterActivity.this,  text: "Registration unsuccessful: " + e.getMessage(), Toast.LENGTH_SHORT).show();
    }
});
```

Fig 4.4 Registration Activity

This code shows the registration code for the users. The user’s information is collected and sent to the database to be stored. In case of connection issues or mistakes, errors are displayed

Login Page:

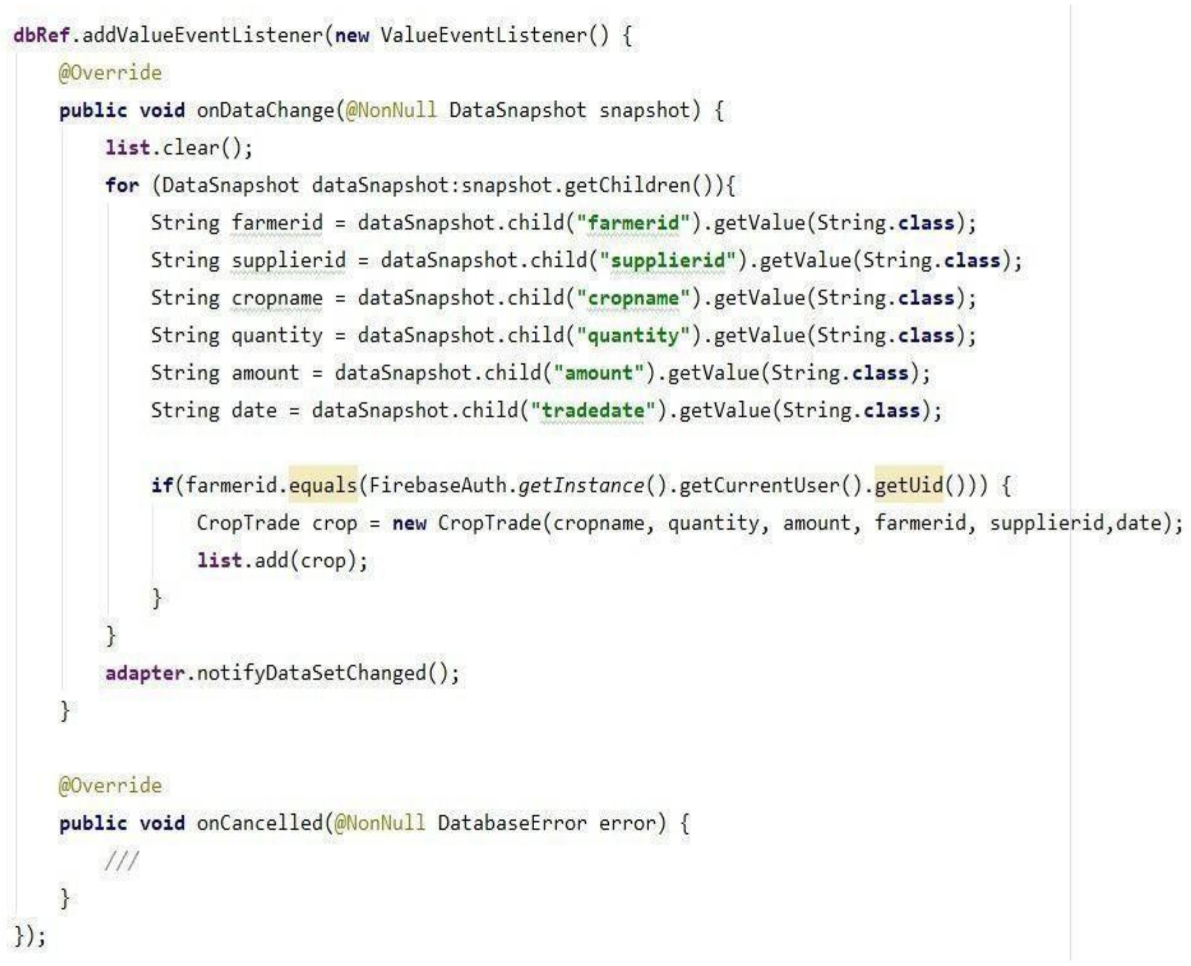


```
mAuth.signInWithEmailAndPassword(email, pass).addOnSuccessListener(new OnSuccessListener<AuthResult>() {
    @Override
    public void onSuccess(AuthResult authResult) {
        DatabaseReference dbRef = FirebaseDatabase.getInstance().getReference( path: "Users").child(FirebaseAuth.getInstance().getCurrentUser().getUid());
        dbRef.get().addOnSuccessListener(new OnSuccessListener<DataSnapshot>() {
            @Override
            public void onSuccess(DataSnapshot dataSnapshot) {
                progressBar.setVisibility(View.GONE);
                Toast.makeText( context: MainActivity.this,  text: "Logged in successfully", Toast.LENGTH_SHORT).show();
                String usertype = dataSnapshot.child("usertype").getValue(String.class);
                if (usertype.equals("FARMER")) {
                    startActivity(new Intent( packageContext: MainActivity.this, FarmerActivity.class));
                } else if (usertype.equals("COMMERCIAL BUYER")) {
                    startActivity(new Intent( packageContext: MainActivity.this, SupplierActivity.class));
                } else {
                    startActivity(new Intent( packageContext: MainActivity.this, NonSupplierActivity.class));
                }
                finish();
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                progressBar.setVisibility(View.GONE);
                Toast.makeText( context: MainActivity.this, e.getMessage(), Toast.LENGTH_SHORT).show();
            }
        });
    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        progressBar.setVisibility(View.GONE);
        Toast.makeText( context: MainActivity.this, e.getMessage(), Toast.LENGTH_SHORT).show();
    }
});
```

Fig 4.5 Login Activity

The shown code is used to login a user to the application. It checks their credentials and then if they are a farmer or a consumer. Accordingly their dashboard and interface changes.

Trade History Page:



```java
dbRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        list.clear();
        for (DataSnapshot dataSnapshot:snapshot.getChildren()){
            String farmerid = dataSnapshot.child("farmerid").getValue(String.class);
            String supplierid = dataSnapshot.child("supplierid").getValue(String.class);
            String cropname = dataSnapshot.child("cropname").getValue(String.class);
            String quantity = dataSnapshot.child("quantity").getValue(String.class);
            String amount = dataSnapshot.child("amount").getValue(String.class);
            String date = dataSnapshot.child("tradedate").getValue(String.class);

            if(farmerid.equals(FirebaseAuth.getInstance().getCurrentUser().getUid())) {
                CropTrade crop = new CropTrade(cropname, quantity, amount, farmerid, supplierid,date);
                list.add(crop);
            }
        }
        adapter.notifyDataSetChanged();
    }

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

Fig 4.6 Trade History Activity

The above activity shows the trade history of a consumer. It calculates the previous trade amounts and displays it in the form of distinct boxes with all the relevant information.

CHAPTER 5

RESULT

5.1 SNAPSHOTS

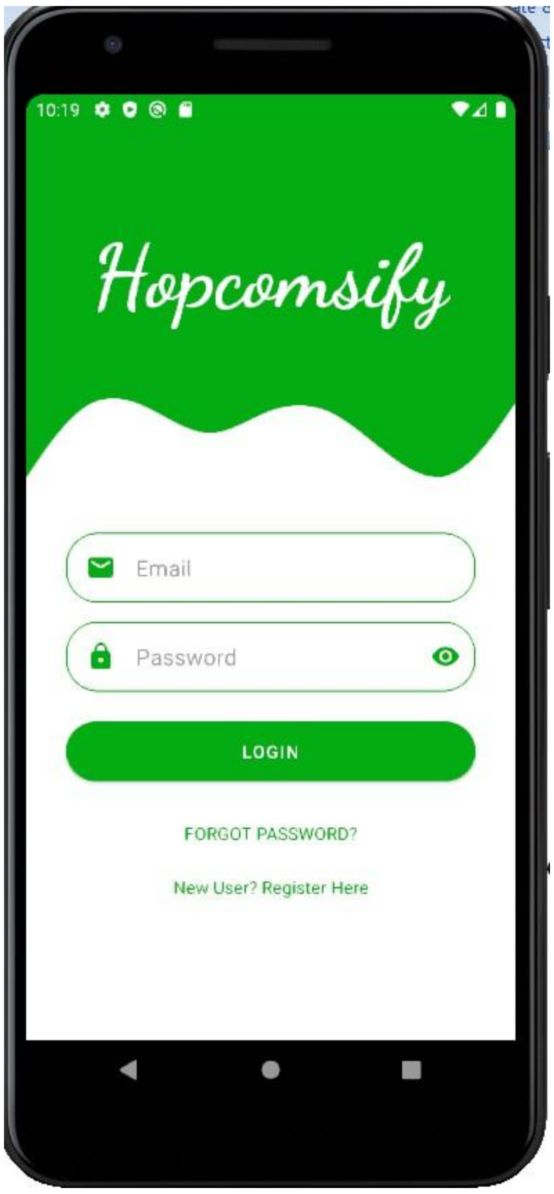


Fig 5.1 Home Page

This is the home page that shows up when the app opens. Farmers and both types of consumers may login here. In case a user forgets their password, forgot password option is also present.

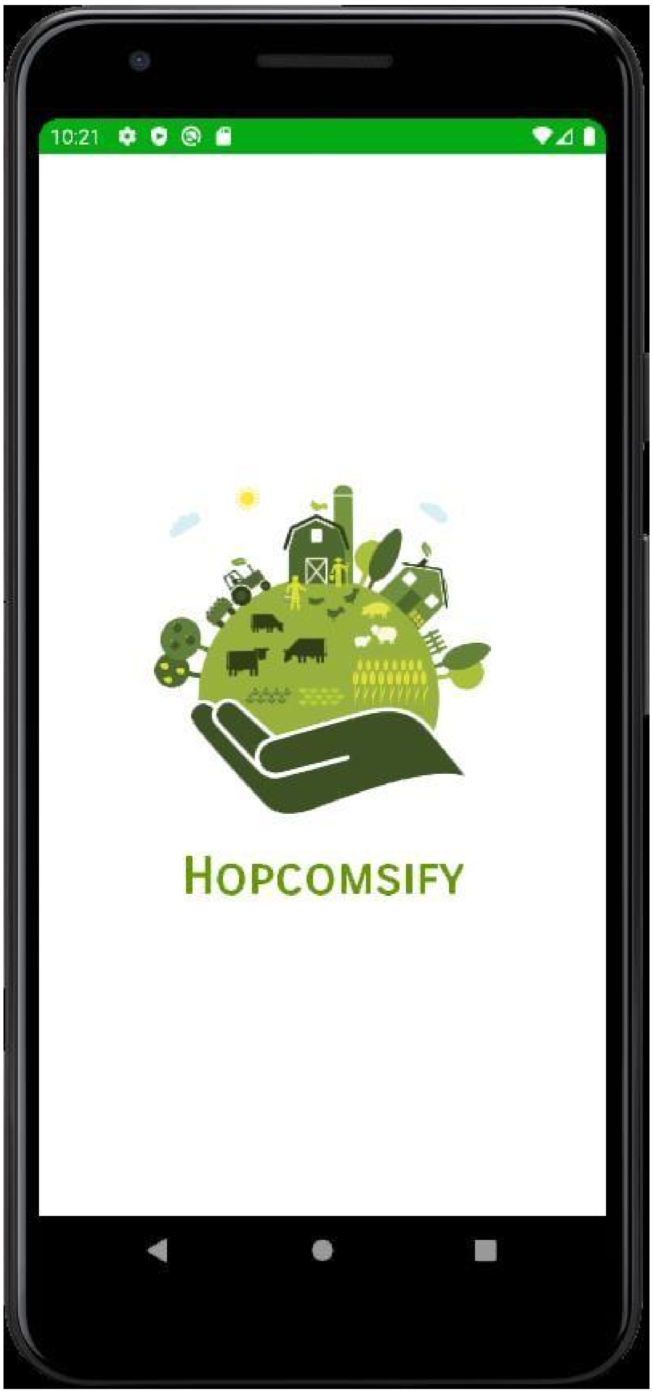


Fig 5.2 App Logo

This is the app logo that shows during loading. The logo was chosen to represent the intrinsic role a farmer plays in India, and how the whole country revolves around their efforts.

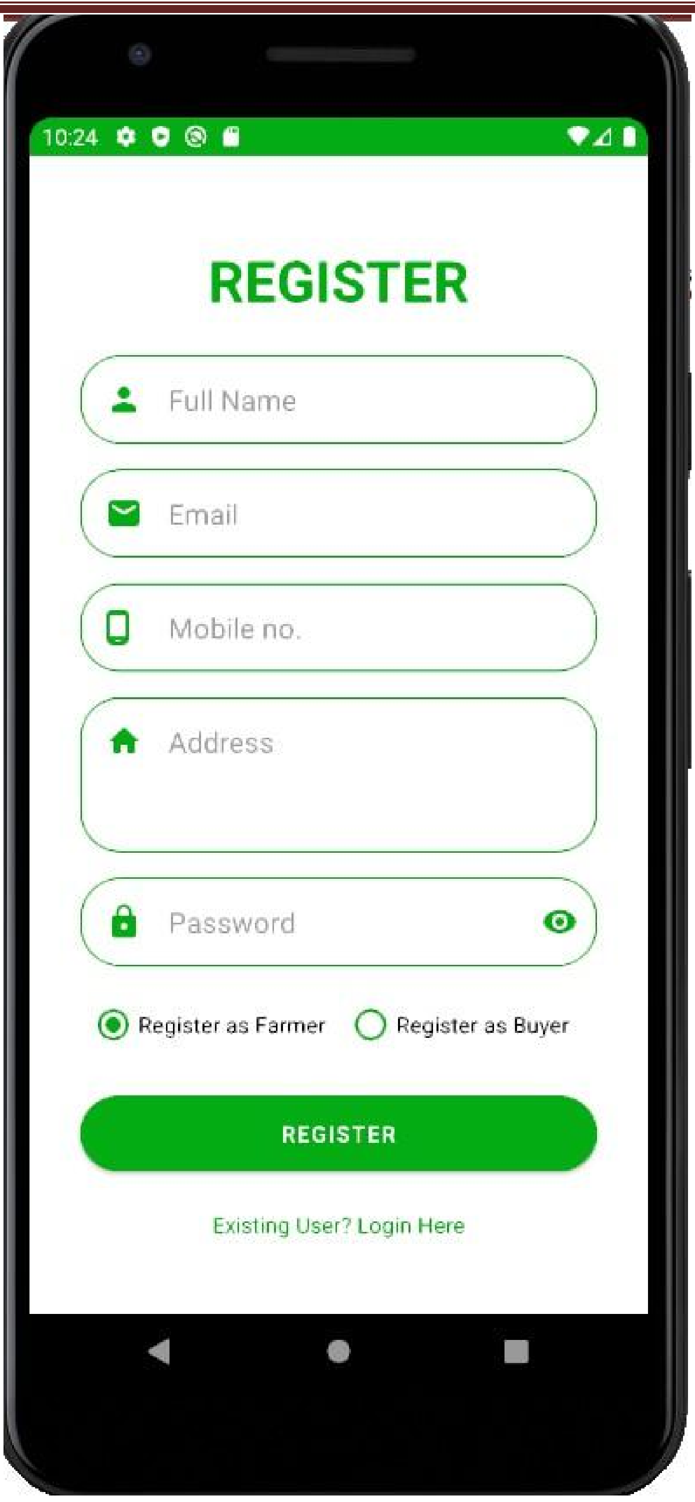
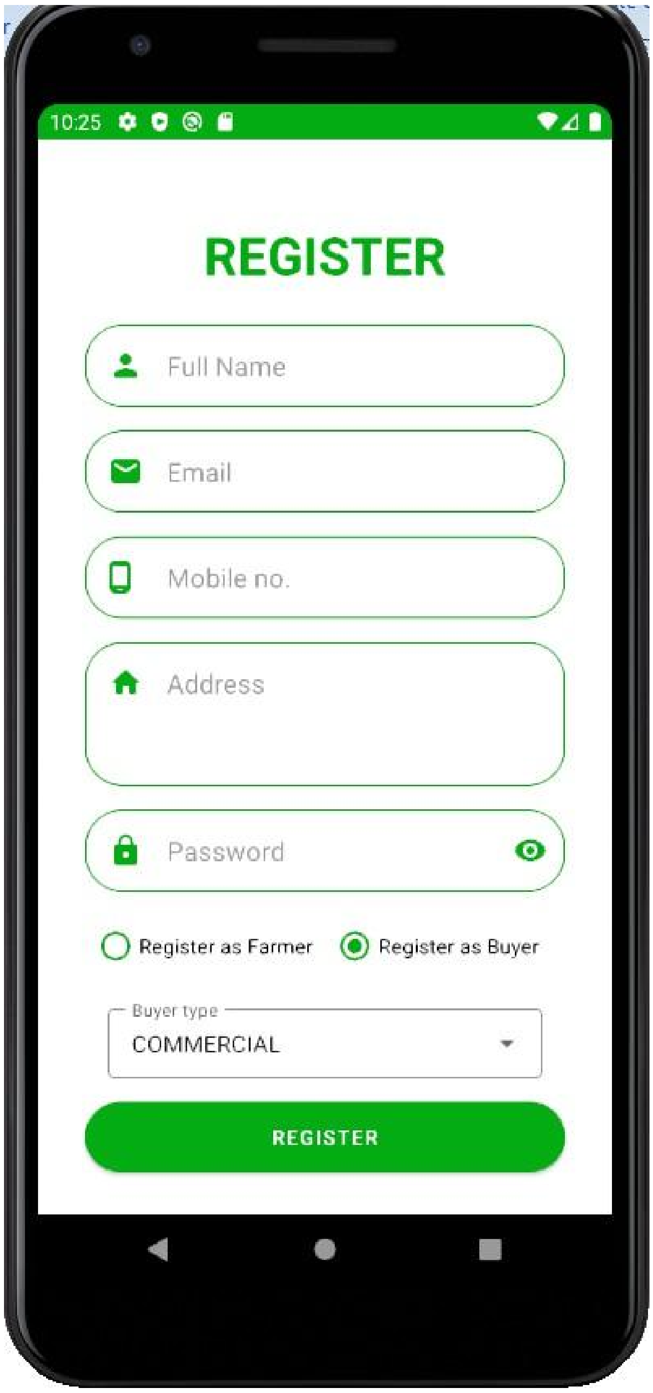

10:24
REGISTER
Full Name
Email
Mobile no.
Address
Password
[✓ Register as Farmer] Register as Buyer
REGISTER
Existing User? Login Here

Fig 5.3 Registration Page for Farmer

This is the page in which the farmer can register. It contains all of the personal data fields with validation. All of the data is not stored locally but is sent to the cloud database.

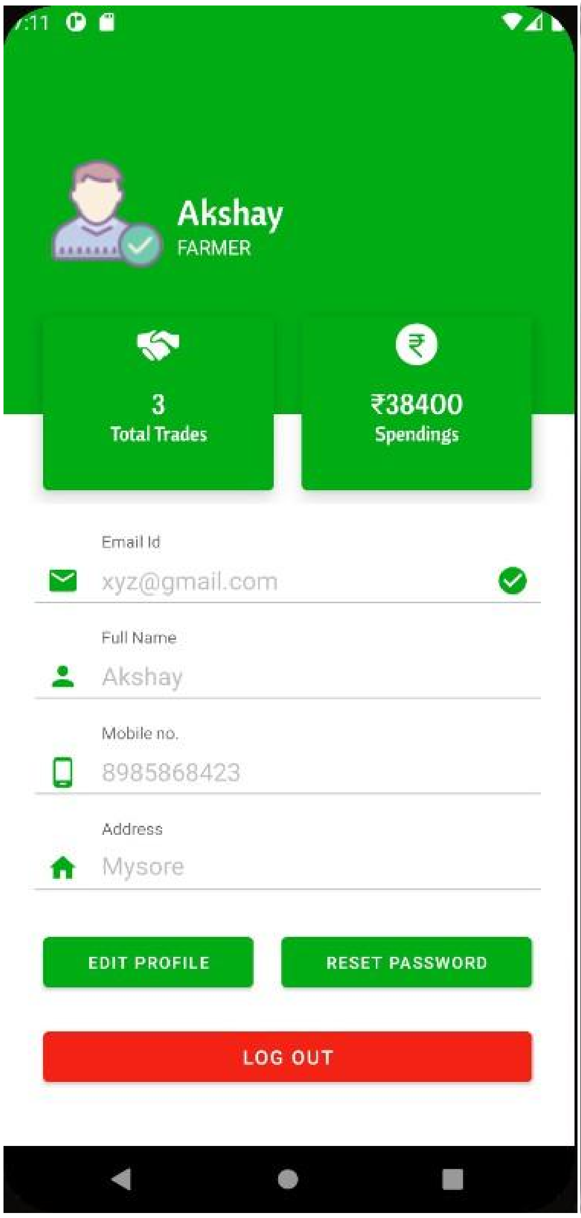


Register as Farmer [✓ Register as Buyer]

Buyer type: COMMERCIAL

Fig 5.4 Registration Page for Buyer

This is the page in which the buyer can register. It contains all of the personal data fields of a consumer, with relevant validation. All of the data is not stored locally but is sent to the cloud database.



Email Id
xyz@gmail.com

Full Name
Akshay

Mobile no.
8985868423

Address
Mysore

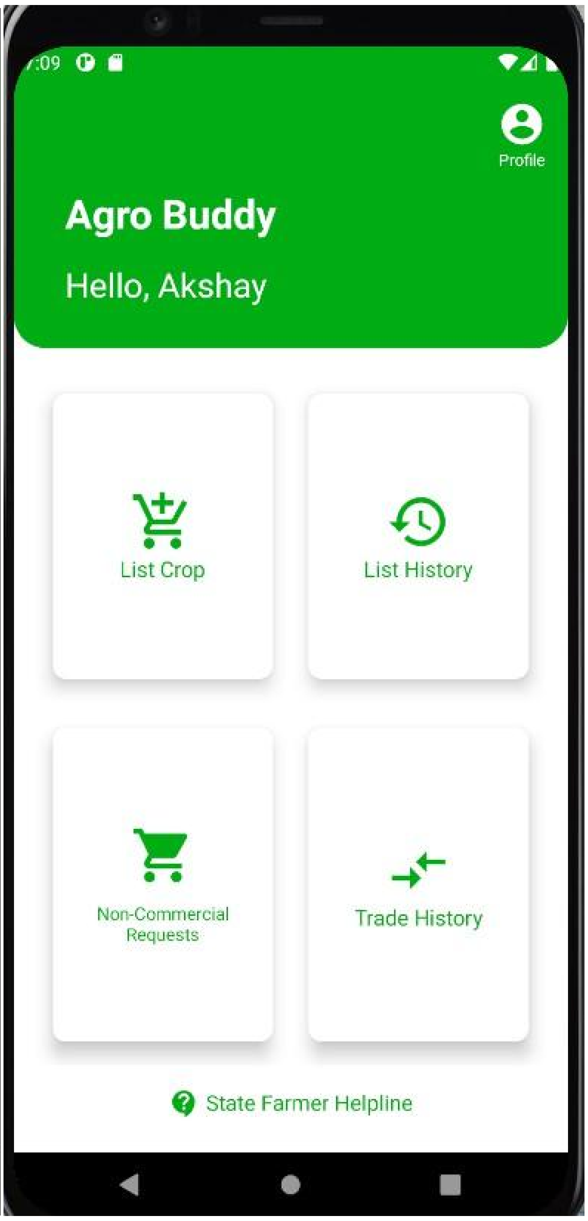
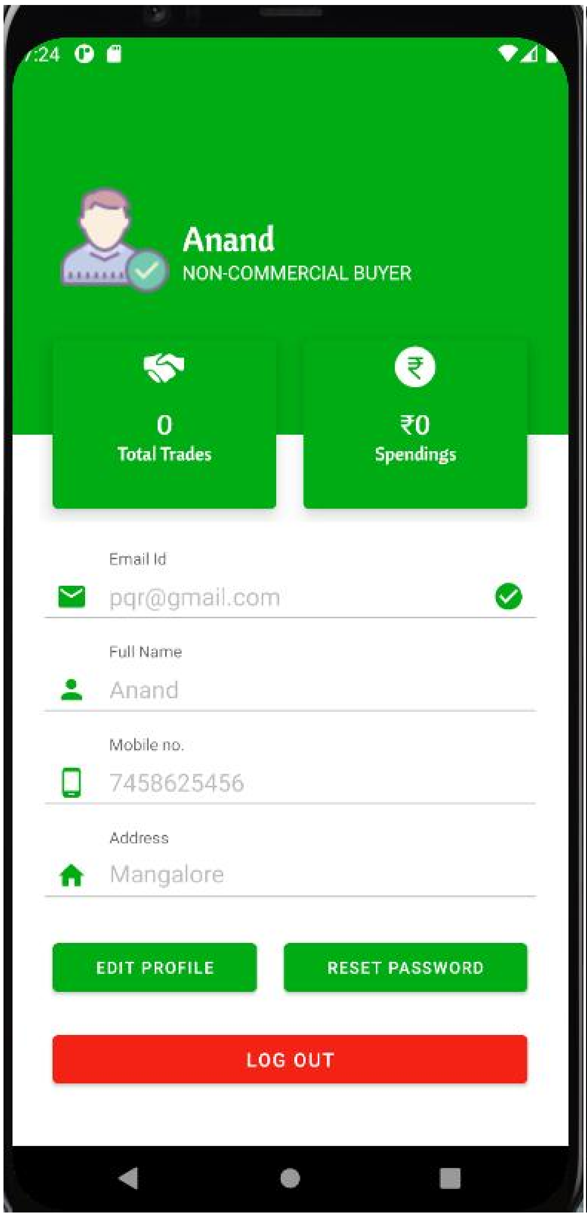


Fig 5.5 Farmer Details

Fig 5.6 Farmer Dashboard

The above two figures show the farmer details and the farmer dashboard respectively. The details show important information like personal data, trades completed and total earnings. The dashboard is the main way for the farmer to list crops, sell crops and see their past trades.



Anand
NON-COMMERCIAL BUYER

0
Total Trades

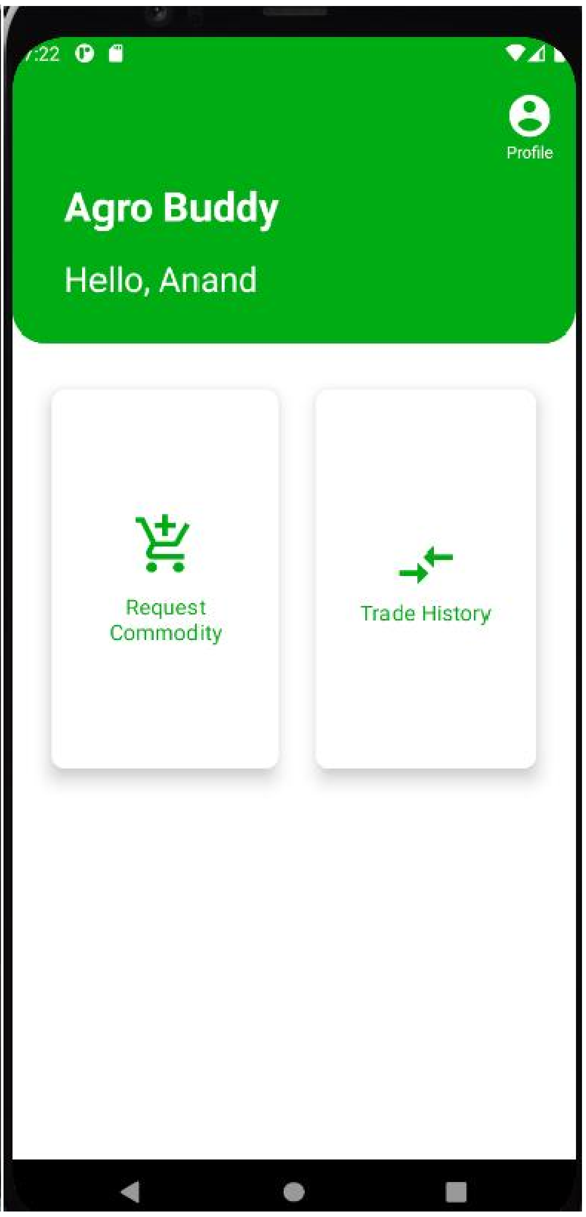
₹0
Spendings

Email Id
pqr@gmail.com

Full Name
Anand

Mobile no.
7458625456

Address
Mangalore

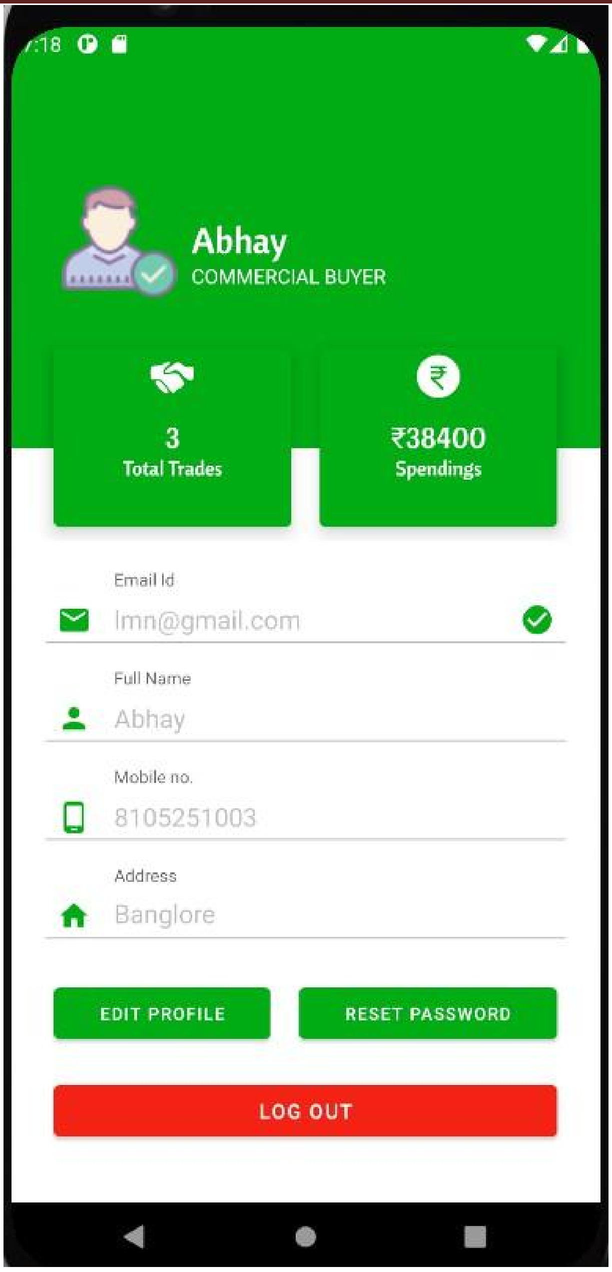


Agro Buddy

Hello, Anand

Fig 5.7 NonCom Buyer Details **Fig 5.8 NonCom Buyer Dashboard**

The above two figures show the non commercial consumer details and the non commercial consumer dashboard respectively. The details show important information like personal data, trades completed and total spending. The dashboard is the main way for the consumer to request commodities and see their past trades.



:18

Abhay
COMMERCIAL BUYER

3
Total Trades

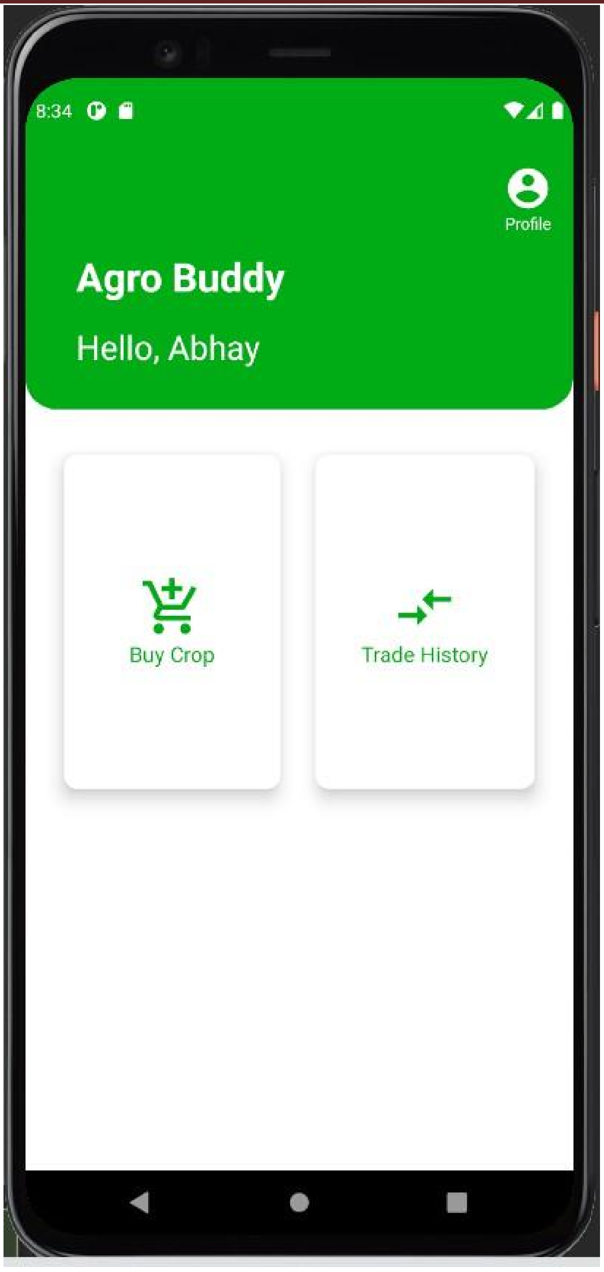
₹38400
Spendings

Email Id
lmn@gmail.com

Full Name
Abhay

Mobile no.
8105251003

Address
Banglore

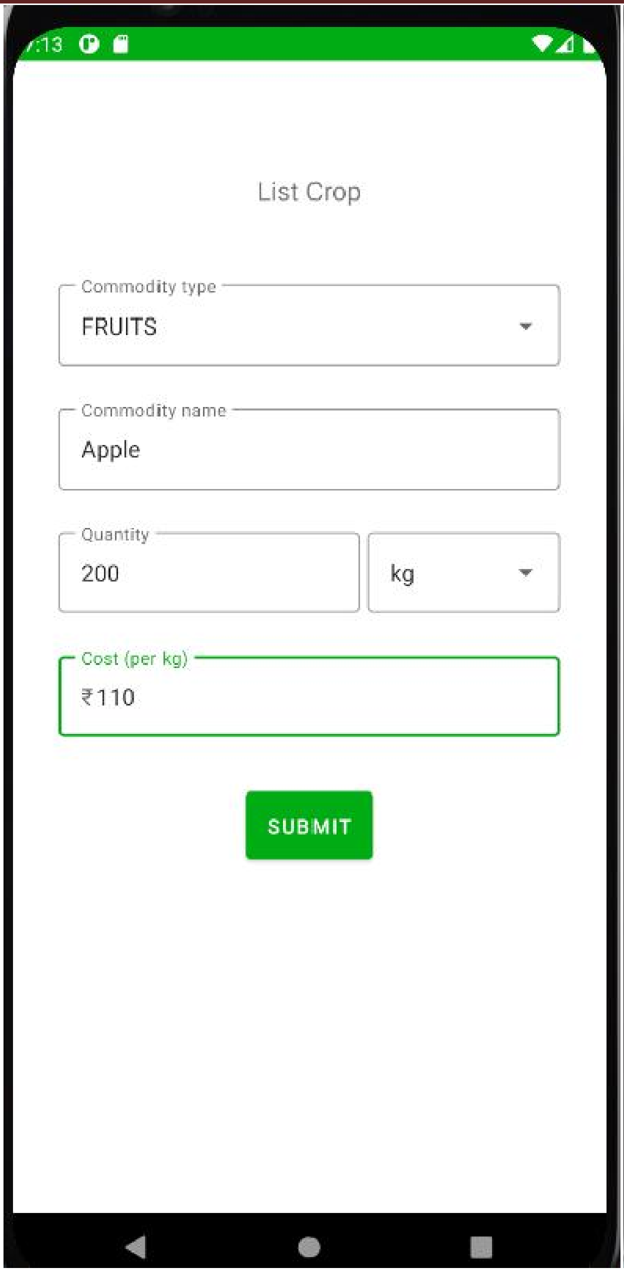


8:34

Agro Buddy
Hello, Abhay

Fig 5.9 Com Buyer Details **Fig 5.10 Com Buyer Dashboard**

The above two figures show the commercial consumer details and the commercial consumer dashboard respectively. The details show important information like personal data, trades completed and total spending. The dashboard is the main way for the consumer to buy crops and see their past trades.



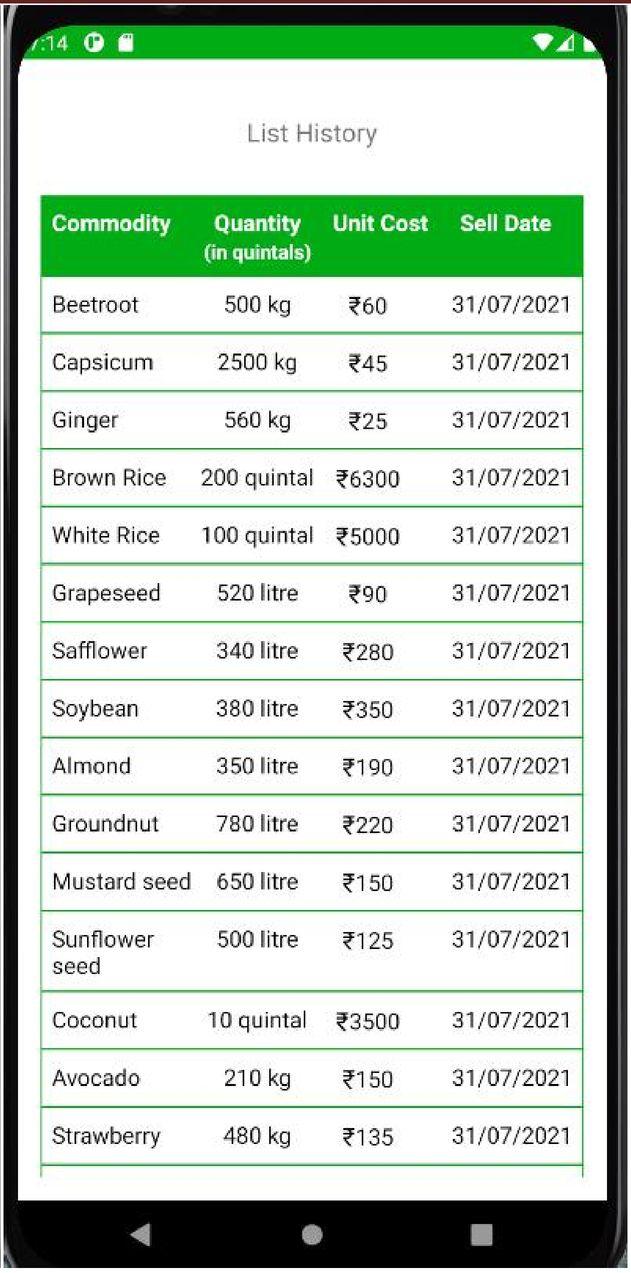
List Crop

Commodity type: FRUITS

Commodity name: Apple

Quantity: 200 kg

Cost (per kg): ₹110

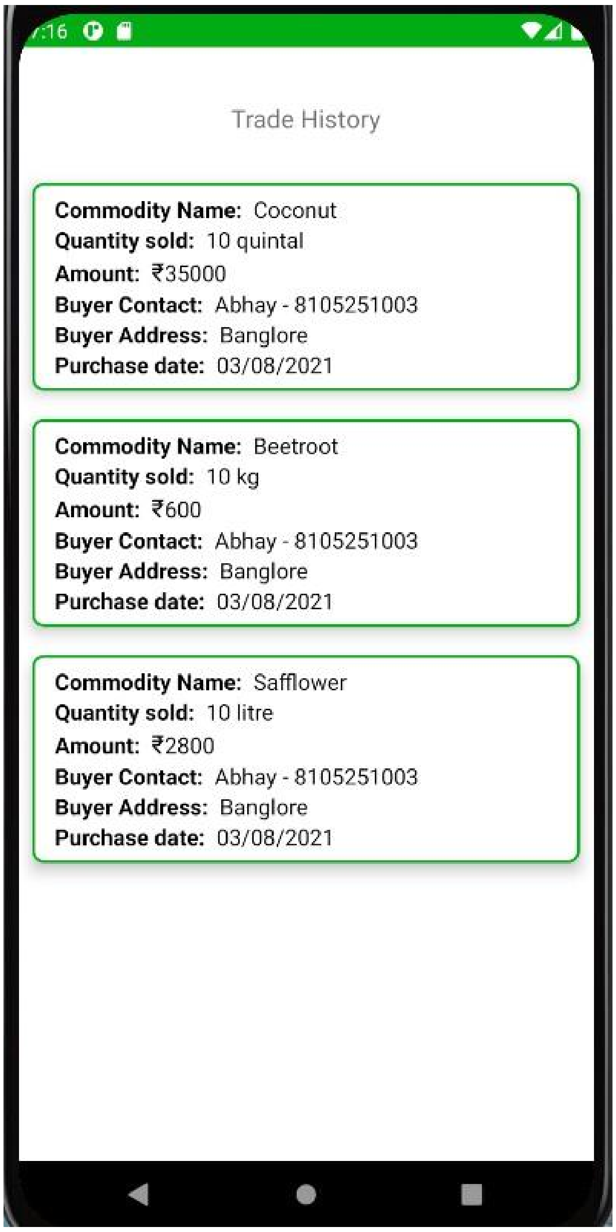


List History

Commodity	Quantity (in quintals)	Unit Cost	Sell Date
Beetroot	500 kg	₹60	31/07/2021
Capsicum	2500 kg	₹45	31/07/2021
Ginger	560 kg	₹25	31/07/2021
Brown Rice	200 quintal	₹6300	31/07/2021
White Rice	100 quintal	₹5000	31/07/2021
Grapeseed	520 litre	₹90	31/07/2021
Safflower	340 litre	₹280	31/07/2021
Soybean	380 litre	₹350	31/07/2021
Almond	350 litre	₹190	31/07/2021
Groundnut	780 litre	₹220	31/07/2021
Mustard seed	650 litre	₹150	31/07/2021
Sunflower seed	500 litre	₹125	31/07/2021
Coconut	10 quintal	₹3500	31/07/2021
Avocado	210 kg	₹150	31/07/2021
Strawberry	480 kg	₹135	31/07/2021

Fig 5.11 Farmer List Crop **Fig 5.12 Farmer List History**

This shows the list of crops the farmer has uploaded. The farmer can specify details like quantity and can enter their own price. The commodity names are present in the database. The list history shows all of their past produce and transactions.

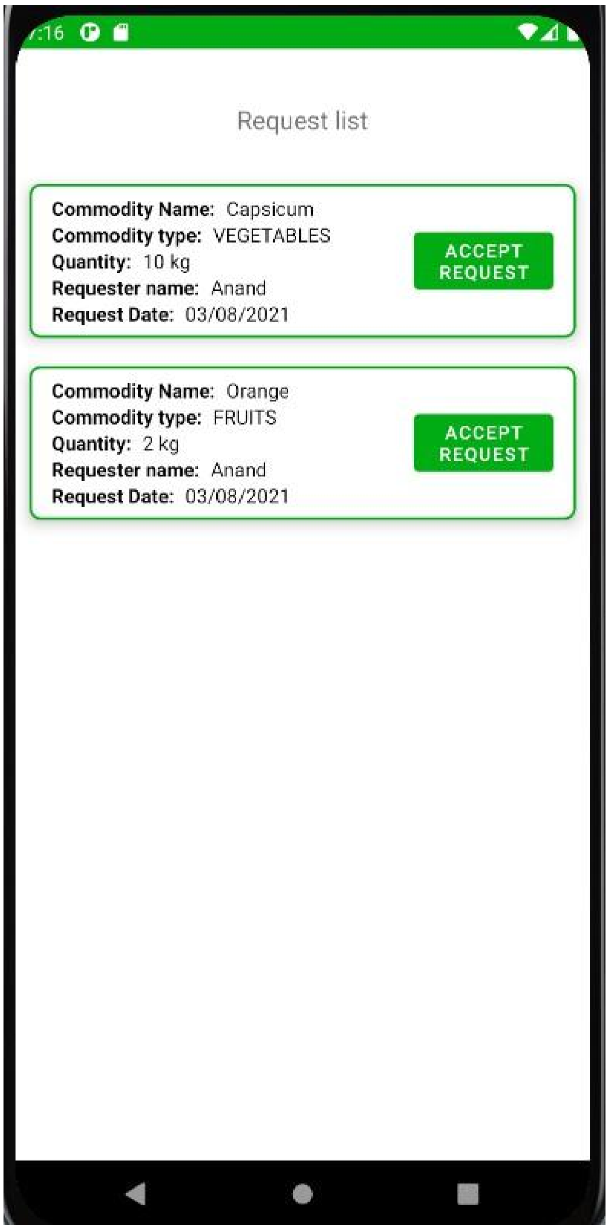


Trade History

Commodity Name: Coconut
Quantity sold: 10 quintal
Amount: ₹35000
Buyer Contact: Abhay - 8105251003
Buyer Address: Banglore
Purchase date: 03/08/2021

Commodity Name: Beetroot
Quantity sold: 10 kg
Amount: ₹600
Buyer Contact: Abhay - 8105251003
Buyer Address: Banglore
Purchase date: 03/08/2021

Commodity Name: Safflower
Quantity sold: 10 litre
Amount: ₹2800
Buyer Contact: Abhay - 8105251003
Buyer Address: Banglore
Purchase date: 03/08/2021



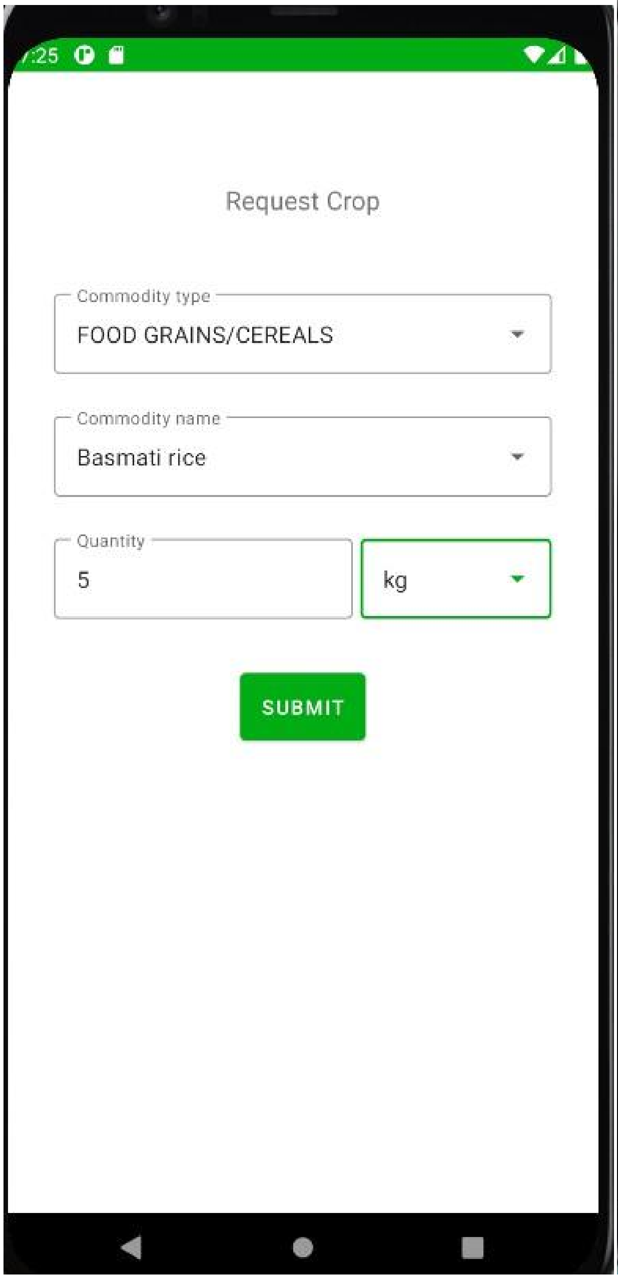
Request list

Commodity Name: Capsicum
Commodity type: VEGETABLES
Quantity: 10 kg
Requester name: Anand
Request Date: 03/08/2021

Commodity Name: Orange
Commodity type: FRUITS
Quantity: 2 kg
Requester name: Anand
Request Date: 03/08/2021

Fig 5.13 Farmer Trade History **Fig 5.14 Farmer Request List**

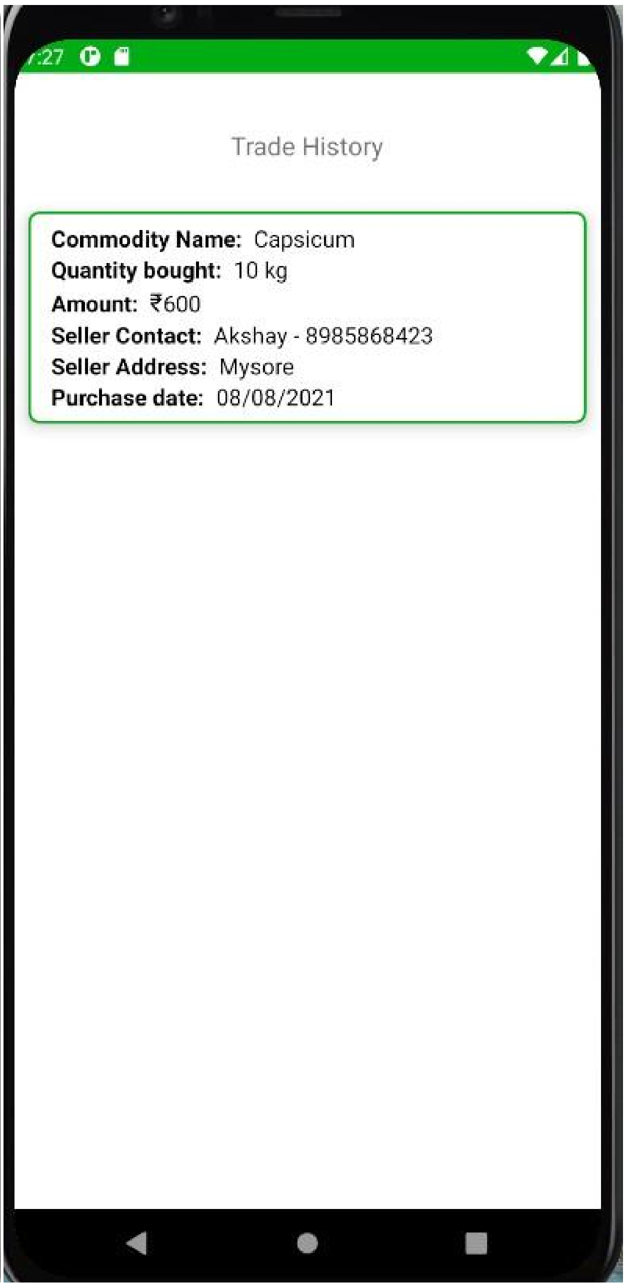
The above two images show the farmer completed trades and all of the request he receives from non commercial buyers. Trade history shows all the details of the buyer for complete documentation. The farmer can only receive requests for non commercial buyers in small quantities.


Request Crop

Commodity type: FOOD GRAINS/CEREALS

Commodity name: Basmati rice

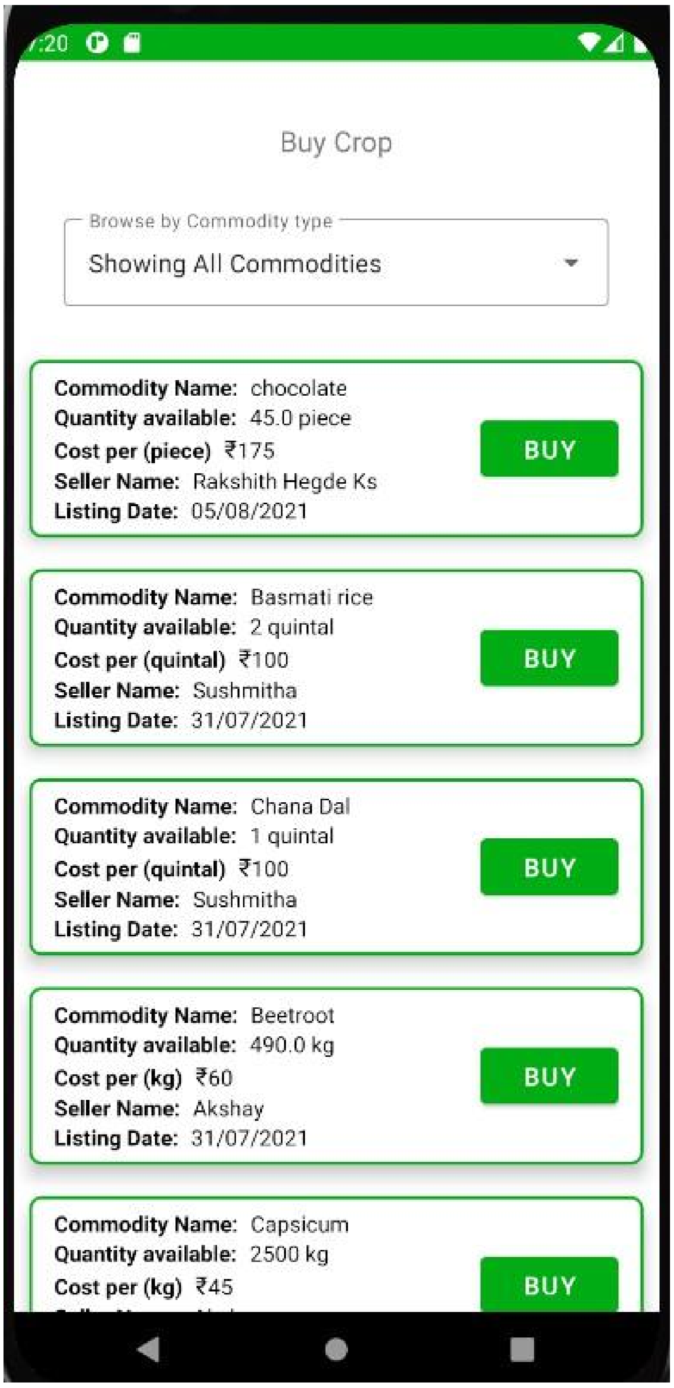
Quantity: 5 kg


Trade History

Commodity Name: Capsicum
Quantity bought: 10 kg
Amount: ₹600
Seller Contact: Akshay - 8985868423
Seller Address: Mysore
Purchase date: 08/08/2021

Fig 5.15 NonCom Request List **Fig 5.16 NonCom Trade History**

The above two images show non commercial consumer request page and all of their past transactions. The requests are shown to every farmer and they can choose whether to accept it or not. The complete farmer details are shown in trade history section to reduce confusion on which farmer sold what.



Buy Crop

Browse by Commodity type
Showing All Commodities

Commodity Name: chocolate
Quantity available: 45.0 piece
Cost per (piece) ₹175
Seller Name: Rakshith Hegde Ks
Listing Date: 05/08/2021

Commodity Name: Basmati rice
Quantity available: 2 quintal
Cost per (quintal) ₹100
Seller Name: Sushmitha
Listing Date: 31/07/2021

Commodity Name: Chana Dal
Quantity available: 1 quintal
Cost per (quintal) ₹100
Seller Name: Sushmitha
Listing Date: 31/07/2021

Commodity Name: Beetroot
Quantity available: 490.0 kg
Cost per (kg) ₹60
Seller Name: Akshay
Listing Date: 31/07/2021

Commodity Name: Capsicum
Quantity available: 2500 kg
Cost per (kg) ₹45

Fig 5.17 Com Buyer Trade History

This page shows all the commercial buyer transactions and their whole sale transactions. Commercial consumers were created to facilitate bulk orders. They can't request crops but they get a list of all commodities that farmers have listed. They can buy what they require from this page.

CONCLUSION AND FUTURE ENHANCEMENTS

AGRO BUDDY attempts to ease some of the burden on the government regarding HOPCOMs outlets and the terrible losses they are facing. This has been attempted by digitizing the entire platform and ensuring the removal of middle men such as APMC. It also improves the farmer and community outreach massively, making it possible to buy and sell using just their phones. AGRO BUDDY comes with many advantages like remote transactions, price guarantee, automatic deductions, better prices, easier outreach and so on.

However, after conducting a comprehensive survey among farmers, they were all under the opinion that such an application would be extremely beneficial and the amount of wrong-doing would be minimal. This was the main motivation behind this app. Furthermore, to ensure fairness in crop prices, the app allows every farmer to set their own price for a crop. A certificate from APMC was also procured to monitor daily fruit/vegetable prices so that there would be disparities like hoarding and shorting in the app. This has culminated in famer-friendly interfaced app with robust functionalities while retaining a simple user interface.

Further improvements exist to add a translate feature to enable every farmer access to this application. We are planning to approach the Horticulture Department, Government of Karnataka regarding this application and will request their permission to implement a trial run in the near future confining to small scale like a village or a city and then implement it fully based on the initial performance.

REFERENCES

1 Soumalya Ghosh, A. B. Garg , Sayan Sarcar , P.S.V.S Sridhar , Ojasvi Maleyvar , and Raveesh kapoor : Proceeding of the 2014 IEEE Students' Technology Symposium

2 S. Revathi , S. Sathya Prasad : 2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP)

3 Divya Sawant , Anchal Jaiswal , Jyoti Singh , Payal Shah : 2019 IEEE Bombay Section Signature Conference (IBSSC)

4 Manav Singhal , Kshitij Verma , Anupam Shukla : Proceeding of the 2016 IEEE Students' Technology Symposium

5 S. Mokshapathy : International Journal of Horticulture, 2013, Vol. 3, No. 20

6 H. M. Chandrashekar : International NGO Journal Vol. 6 (5), pp. 122-132, May 2011

7 Kalyani Khodaskar : 2015 Fifth International Conference on Communication Systems and Network Technologies

8 Pranav Shriram , Sunil Mhamane : 2018 3rd International Conference on Inventive Computation Technologies (ICICT)

9 “Overview Guides Reference Samples Libraries Support Go to console” Documentation Firebase, https://firebase.google.com /docs/

10 Stack Overflow, https://stackoverflow.com