A

Project Report

on

Farmer's Assistant

Developed By

Nayan Satani [190511201740]

Saharsh Modi [190511201719]

as

Partial Fulfillment of Semester V of

Master of Computer Applications

for A.Y. 2020 - 2021

Under the Guidance of

Dr. Priya Swaminarayan

Submitted To

Department of MCA
Faculty of IT & Computer Science
PARUL University

CERTIFICATE

This is to certify that **Mr. Nayan Satani, Enrollment No. 190511201740 and Mr. Saharsh Modi, Enrollment No. 190511201719** student of Master of Computer Applications has satisfactorily completed the Minor Project on **"Farmer's Assistant"** at **Department of MCA, Parul University** as partial fulfillment of MCA Semester V.

Seat No. _____ Date of Submission: _____

_____ _____

_____

Internal Guide Project Coordinator Director - MCA

Department of MCA
Faculty of IT & Computer Science
PARUL University, Vadodara

Preface

Project “Farmer’s Assistant” is an Android Mobile App. This is very helpful for Farmers of Gujarat. Which is used to get updates of new government’s subsidies, schemas on farming. Using this, a farmer can get some important tips to improve crop productions. Also, Farmers can ask a query or any equations to the top-notch Experts.

This app is provided in two languages, English and Gujarati. So, it is very easy to use for any farmers who are using smartphones.

Our system is helpful for farmers to keep updated from the daily Market (Mandi) Rate of Gujarat. It will give awareness to the farmers about new technologies that are helpful in farming.

We dedicate this app to Farmers of Gujarat.

Acknowledgement

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along with the completion of our project. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

We owe our deep gratitude to our project guide ***Dr. Priya Swaminarayan*** (Director, MCA), who took a keen interest in our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good application.

We also like to thank ***Prof. Vivek Dave*** (Head of Department, MCA), ***Prof. Garima Pathak*** (Project Coordinator), ***Prof. Kaushal Gor*** for guiding, helping and providing the expert lectures whenever needed.

We are thankful for and fortunate enough to get constant encouragement, support, and guidance from our Parents, all Teaching staff of the MCA Department which helped us in successfully completing our project work. Also, we would like to extend our sincere esteems to all staff in the laboratory for their timely support.

Nayan Satani – 190511201740

Saharsh Modi – 190511201719

INDEX

1. About Department of MCA

PARUL University

Parul University is a legitimate university established under Gujarat Private University Act 2009, after legislation passed by the Government of Gujarat on 26th March 2015 giving University status to Parul Group of Institutes functioning under the aegis of Parul Arogya Seva Mandal Trust.

Faculty of IT & Computer Science

Faculty of IT and Computer Science, Parul University has materialized as one of the prime IT education providers at global level. Various departments under Faculty of IT and Computer Science strive in preparing IT-industry ready professionals by means of various skill development courses, vocational courses, co-curricular &extra-curricular activities, industry visits and expert lectures.

MCA Department

The Department of Master of Computer Application at Parul University emphasizes on building professionals in the domain of computer applications by providing necessary environment by means of facilitating suitable blend of technical and non-technical learning experience. The department cultivates students in various curricular, co-curricular and extra-curricular activities in order to produce future system analysts, system designers, system programmers, application programmers, testing professionals, system managers, project managers, researchers and other leading positions in systems/IT department.

The department offers various subjects from diversified technical/non-technical areas such as – core IT domain, management, communication skills, mathematics & logic building and rich pool of elective subjects.

The department of MCA focuses on project-based learning, and hence students are motivated to work on tiny hands-on projects in practical oriented subjects to get better exposure. Moreover, throughout their MCA studies, students are required to work on around 3 mini/major projects in individual/team to get enough confidence on software-development and thereby become industry-ready.

2. Project Profile

2.1. Project Definition:

Farmer's Assistant Mobile App

2.2. Project Description:

Farmer's Assistant is an Android based mobile application. It provides complete information to farmers on crop production, crop protection, diseases of crop, soil requirement with agriculture services. This app is also use for giving awareness of the governments subsidies and it has feature of sending notification to all registered user about latest subsidies/schemas details. It is an agriculture app for farmers that will be greatly beneficial to farming community. This app helps farmer with full information about crop, fruit, vegetables etc. growing. The information details the climate and soil requirements, how to harvest crops. This app has also concept of Market Rate in which farmers can get instant access to Market prices for their produce, market status and prevailing trade prices along with quantities. Farmers can also view price trends for their product and plan sale of their produce. With this app, farmers can find the information what they need to improve their agriculture practices in the right format. This app complements technical assistance, plan improvements. We are trying to give application in regional language for customers experience.

2.3. Existing System:

There are many farmer helping related applications and websites are available in the market. But this websites and applications are only accessible by any government officers or any Gram Panchayat Manager. It cannot accessible directly by farmers. Sometimes because of some corrupt officers Farmers are not able to get all the subsidies and helps which are granted by government. We are trying to give solution of this problem. Because of less knowledge of Market rate or not proper knowledge of current market rate, Farmers are always robbed by some broker and businessman.

2.4. Problem Statements:

- There is no any feature like notification of any government subsidies.
- Before this it will be only accessible by any Government Officers or any Gram Panchayat Person.
- Because of long waiting that is more time consuming for getting any subsidies information.
- In old system no any technological changes help and technical changes in Farming.
- Some officers are corrupting the system. So, no proper details of any government schemes/updates.

2.5. Proposed System:

Our proposed system gives knowledge of all the latest updates/ schemes to the registered farmers directly via notification. Our system aware farmers about latest technology in Farming sector. System is also giving recommended Seeds, fertilizers, crop guide, and knowledge of how to use technical machines in good manners. This system also tracks the daily records of updated market prices of all the crops.

2.6. Scope:

- It is developed for effective production of the crop production activity.
- Identify challenges through self-assessment.
- Plan improvements in their farms.
- Farmers can get help for drip irrigation system in their farm.
- Farmers can learn new farming techniques to prevent attacks on their crops.

2.7. Outcomes:

- Farmers can access this system from anywhere in the world.
- Farmers can save time using this app.
- Farmers can create new opportunities in their small farm.

2.8. Tools & Technology Used:

I. Tools:

a) Android Studio

b) Firebase Real-time Database

c) Android Mobile Phone/Android Virtual Device

II. Technology:

a) Front End:

- **Java:**

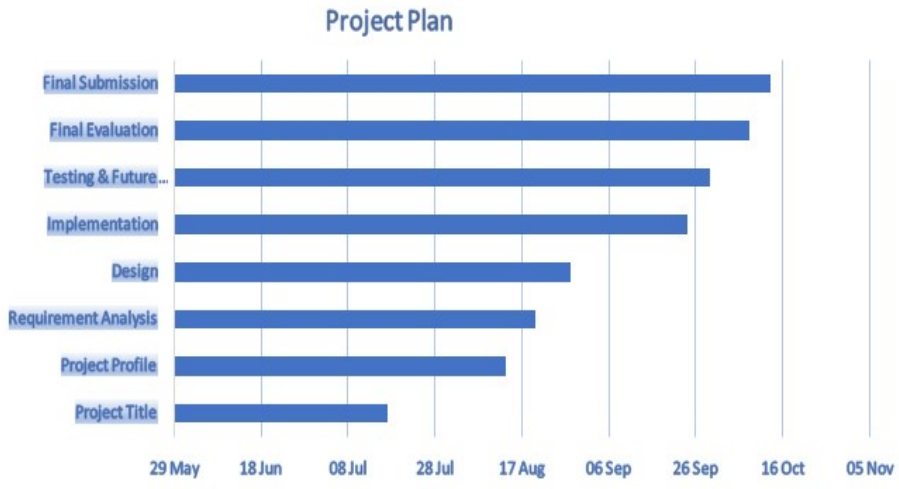
  Java is a general proposed programming language developed by Sun Microsystems to create web application, Mobile application, Desktop GUI application etc. It works on a different platform and it is easy to learn and use. It is totally object-oriented language.

b) Back End:

- **Firebase Real-time Database:**

  Firebase Real-time Database is free online Real-time Database system. It is widely famous and most used database for mobile computing like Android, iOS, Blackberry, etc. It is free of cost to use for small projects.

2.9. Project Plan:



3. Requirement Analysis

3.1. Feasibility Study:

As the name implies, a feasibility study is used to determine the viability of an idea, such as ensuring a project is legally and technically feasible as well as economically justifiable. It tells us whether a project is worth the investment—in some cases, a project may not be doable. There can be many reasons for this, including requiring too many resources, which not only prevents those resources from performing other tasks but also may cost more than an organization would earn back by taking on a project that isn't profitable.

There are different types of feasibility study.

I. Technical Feasibility:

This study focuses on the technical resources available in application. This app is built on android platform which is worldwide use standard. Therefore, the technical risk is not so high and hence it is feasible. Users require only smart phone and internet connectivity to use this app.

II. Economical Feasibility:

This application is free of cost for the customer to use they don't need to purchase application from anywhere. The system will follow the freeware software standard. Other than this in the application, customers have don't need to pay any single rupees for taking any helps.

III. Operational Feasibility:

This study analyses how a project plan satisfies the requirements in the requirement phase of the development. The current mode of operation provides adequate throughput and response time. The system offers effective controls to protect against fraud and to guarantee accuracy and security of data and information. This application helps farmers to know daily markets rate of nearest market yard. Application is providing the best UI which is easily operable & understandable by user.

3.2. Users of the System:

I. Admin:

- Admin can login to the system.
- After login to the system, admin can manage the experts and view the farmers.
- Admin can also change the information of new Government subsidies, schemas, and other technology related information.

II. **Expert:**

- Expert can login to the system.
- Expert can view the farmer's queries and questions.
- Expert can give responses of farmer's queries and questions.

III. **Farmer:**

- Farmer can register and login to the system.
- After login to the system, farmer can view current Government subsidies, schemes, view daily market rate of nearest government market yard etc.
- Farmer can talk (chat) with experts about their queries and crop problems.
- Farmer can also take suggestion for fertilizer, seeds, and crop improvements tips.

3.3. Modules:

I. **Farmer Management:**

- This module views all the farmer of the system.

II. **Expert Management:**

- This module manages all the experts of the system.

III. **Market Rate Management:**

- This module manages market price record.

IV. **Subsidies/Schemas Information Management:**

- This module manages all the information of new subsidies and schemas.

V. **Report:**

- This module generates reports of all the farmer who got notification and current farmers of the system.

3.4. Hardware & Software Requirements:

1. **Developer Side:**

A. **Software Requirement:**

- Front-End: Android Studio with Java JDK
- Emulator: Genny-Motion or Inbuilt Emulator
- Back-End: Android SQLite Database

B. **Hardware Requirement:**

- RAM: 8GB
- SSD/HHD: 120GB Or Above
- Operating System: Windows 7 Or Above (32 or 64)
- Screen Resolution: 1280*800 screen resolution

2. **User Side:**

   A. **Software Requirement:**

   - Android Version: 4.0(Ice Cream Sandwich) Or Above
   - Google Play Version: 7.5 Or Above

   B. **Hardware Requirement:**

   - Device (Operating System): Android Device
   - RAM: 512MB or Above
   - Storage: 100MB or Above

3.5. Use Cases:

I. **Admin:**

- **Actions:**

  1. Login
  2. View Farmer
  3. Manage Expert
  4. Manage Government Subsidies/Schemas
  5. Manage Queries/Questions

II. **Expert:**

- **Actions:**

  1. Login
  2. View Queries/Questions
  3. Response of Queries/Questions

III. **Farmer:**

- **Actions:**

  1. Login
  2. View Government Subsidies/Schemas
  3. View Daily Market Rate
  4. Post Queries/Questions
  5. View Solutions

3.6. Use Case Diagram:

I. Combine:

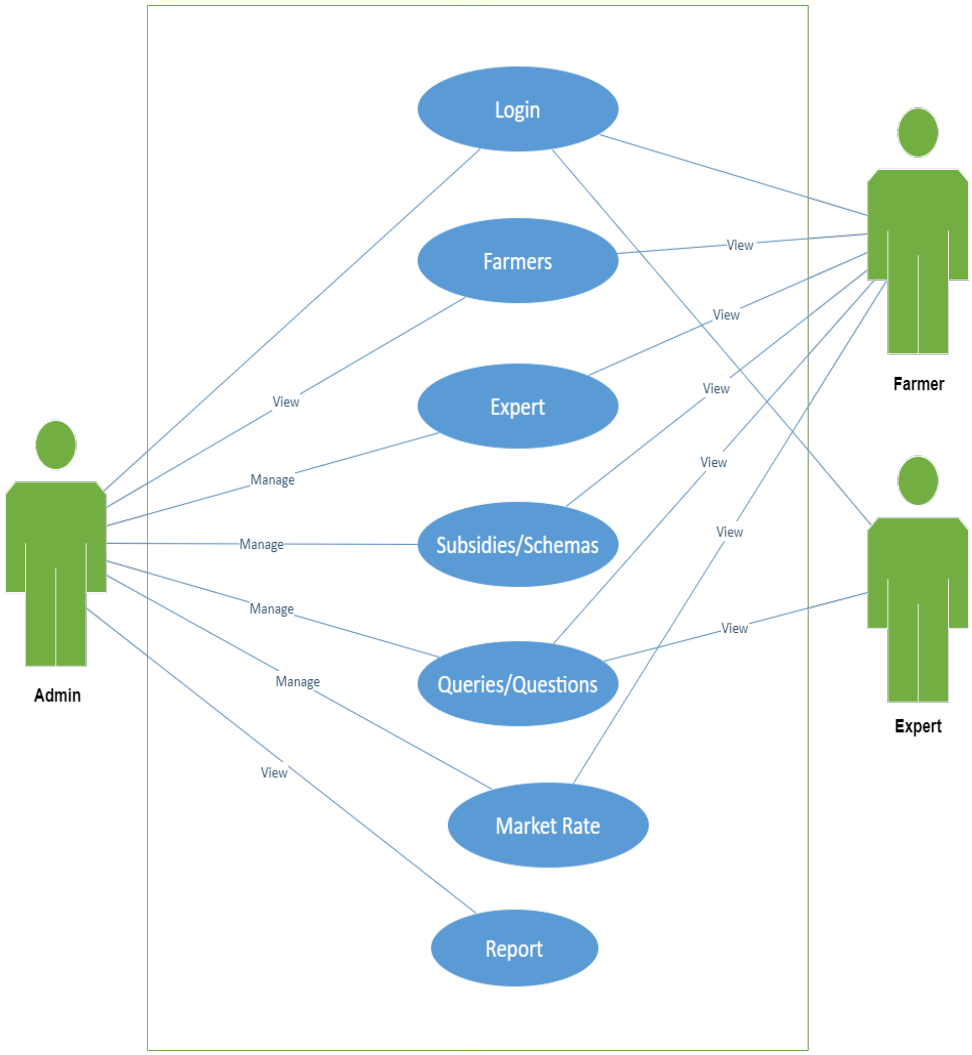


Figure 1 : Combine Use Case Diagram

II. Admin:

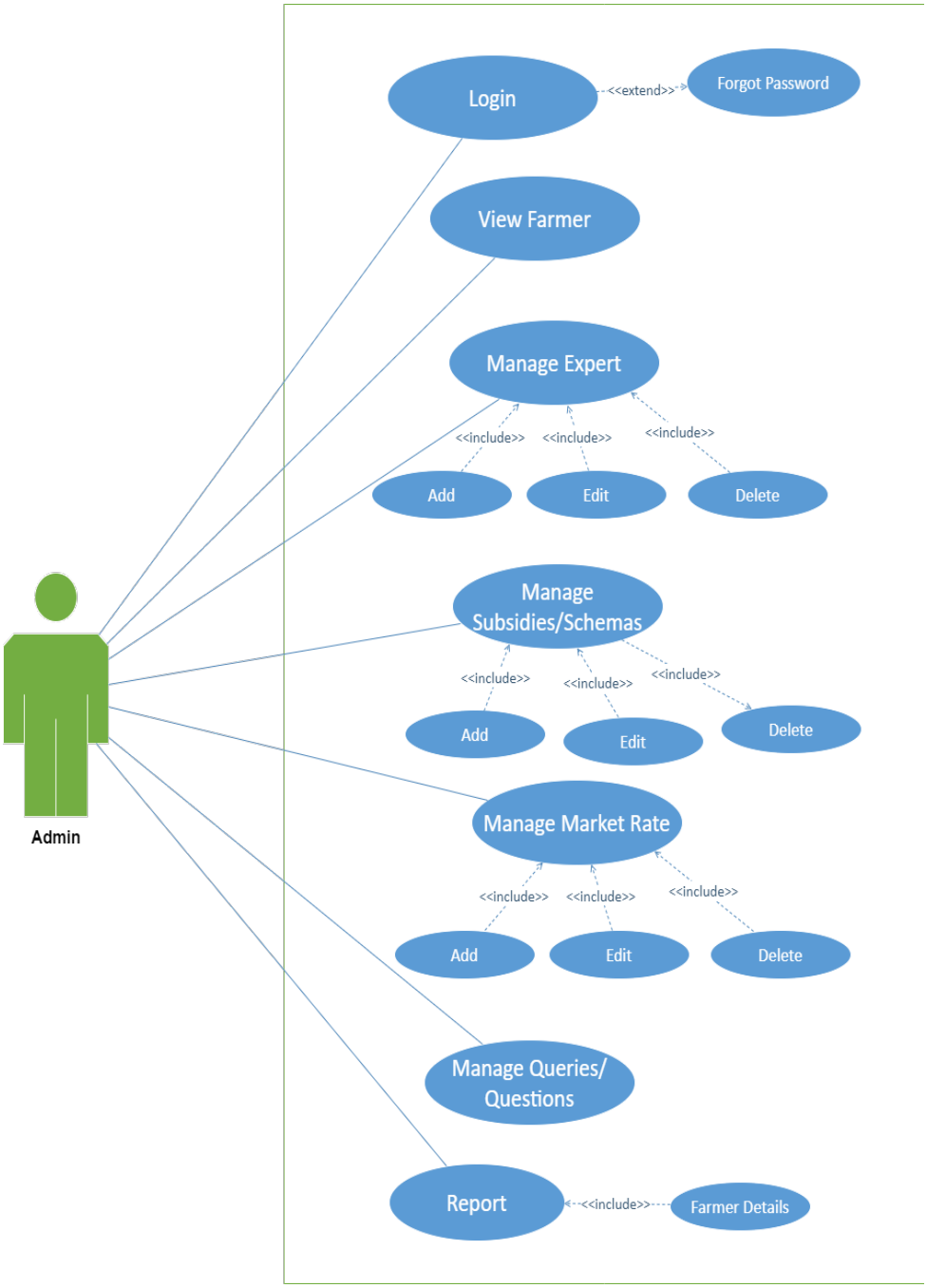


Figure 2 : Use Case Diagram of Admin

III. Expert:

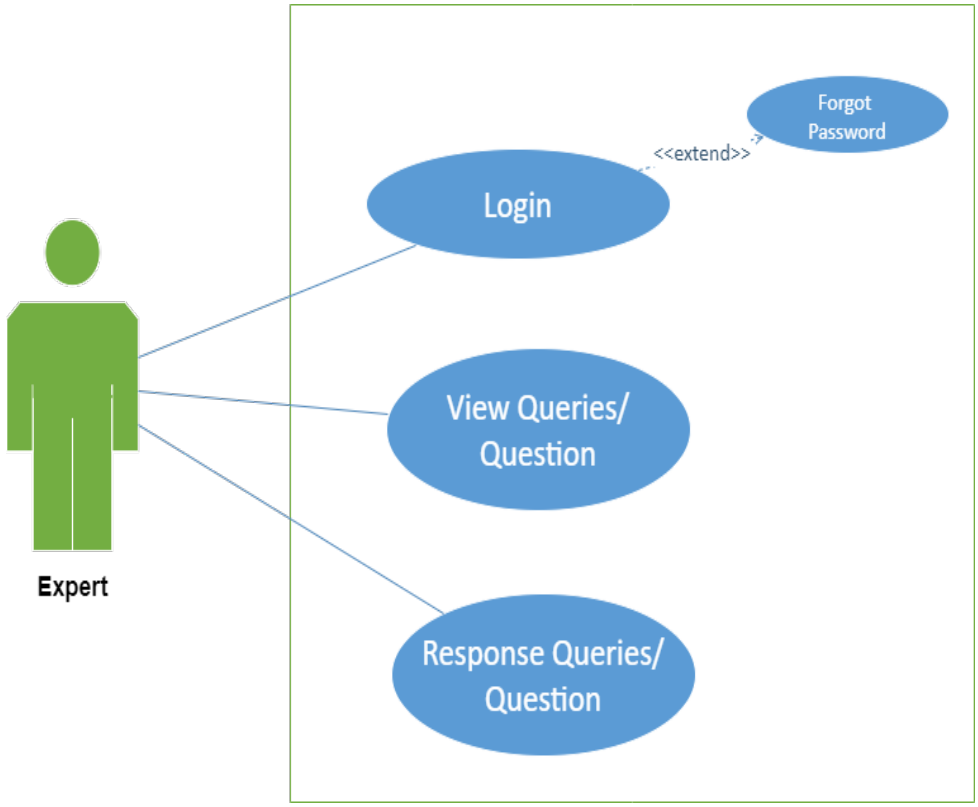


Figure 3 : Use Case Diagram of Expert

IV. Farmer:

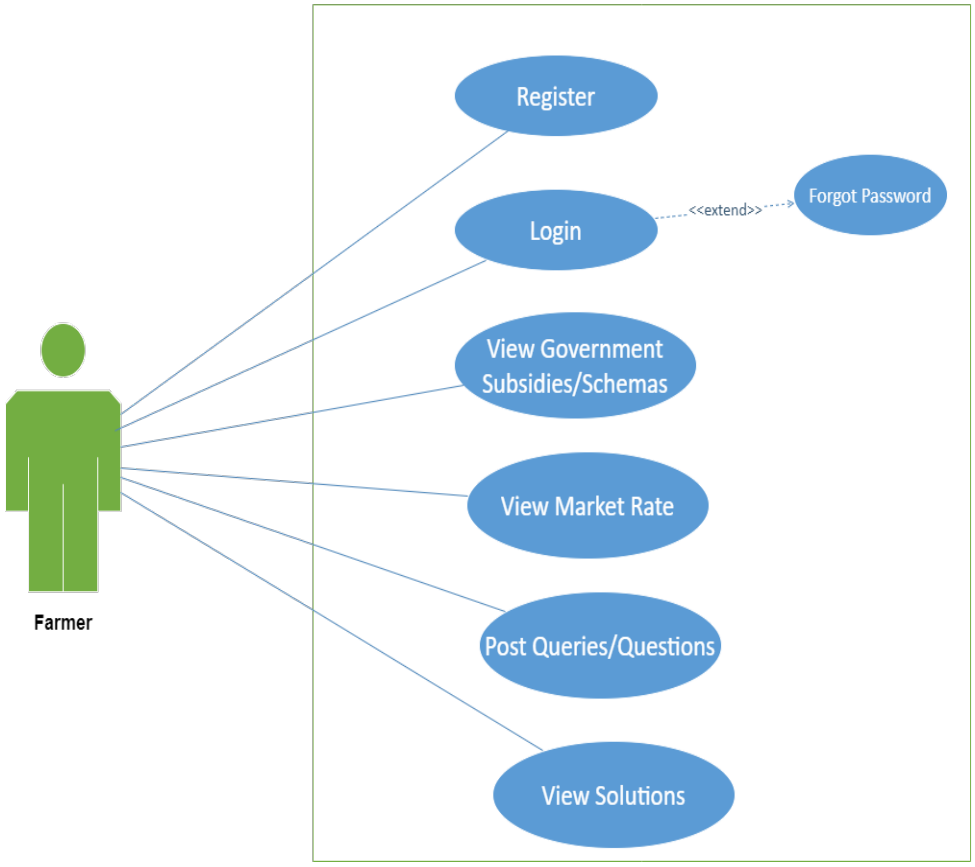


Figure 4 : Use Case Diagram of Farmer

4. Design

4.1 Use Case Scenarios:

1. Login:

Use case ID	UC01
Use Case Name	Login
Primary Actor	Admin
Pre-Condition	Admin must have been registered.
Basic Flow of System	The application can be accessed through mobile number and password.
Alternative Way	There is no alternative way for login.

Table 1: Use Case Scenario for Login

2. View Farmer:

Use case ID	UC02
Use Case Name	View Farmer
Primary Actor	Admin
Pre-Condition	Admin must be logged in.
Basic Flow of System	The admin can view the farmers.
Alternative Way	There is no alternative way for viewing farmer.

Table 2: Use Case Scenario for View Farmer

3. Manage Expert:

Use case ID	UC03
Use Case Name	Manage Expert
Primary Actor	Admin
Pre-Condition	Admin must be logged in.
Basic Flow of System	The admin can add, delete and update the expert.
Alternative Way	There is no alternative way for managing expert.

Table 3: Use Case Scenario for Manage Expert

4. Manage Subsidies/Schemas:

Use case ID	UC04
Use Case Name	Manage Subsidies/Schemas
Primary Actor	Admin
Pre-Condition	Admin must be logged in.
Basic Flow of System	The admin can add, delete and update the subsidies/schemas.
Alternative Way	There is no alternative way for managing subsidies/schemas.

Table 4: Use Case Scenario for Manage Subsidies/Schemas

5. Farmer Login:

Use case ID	UC05
Use Case Name	Farmer Login
Primary Actor	Farmer
Pre-Condition	Farmer must be registered in system.
Basic Flow of System	The farmer can login with their mobile number and password.
Alternative Way	There is no alternative way for farmer login.

Table 5: Use Case Scenario for Farmer Login

6. View Subsidies/Schemas:

Use case ID	UC06
Use Case Name	View Subsidies/Schemas
Primary Actor	Farmer
Pre-Condition	Farmer must be logged in.
Basic Flow of System	The farmer can view subsidies/schemas.
Alternative Way	There is no alternative way for view subsidies/schemas.

Table 6: Use Case Scenario for View Subsidies/Schemas

7. View Daily Market Rate:

Use case ID	UC07
Use Case Name	View Daily Market Rate
Primary Actor	Farmer
Pre-Condition	Farmer must be logged in.
Basic Flow of System	The farmer can view daily market rate of nearest market yard.
Alternative Way	There is no alternative way for view daily market rate.

Table 7: Use Case Scenario for View Daily Market Rate

8. Post Queries/Questions:

Use case ID	UC08
Use Case Name	Post Queries/Question
Primary Actor	Farmer
Pre-Condition	Farmer must be logged in.
Basic Flow of System	The farmer can post Queries/Question for suggestion.
Alternative Way	There is no alternative way for post queries/question.

Table 8: Use Case Scenario for Post Queries/Questions

9. View Solutions :

Use case ID	UC9
Use Case Name	View Solutions
Primary Actor	Farmer
Pre-Condition	Farmer must be logged in.
Basic Flow of System	The farmer can view the solutions of their queries/questions.
Alternative Way	There is no alternative way for view solutions.

Table 9: Use Case Scenario for View Solutions

10. Expert Login:

Use case ID	UC10
Use Case Name	Expert Login
Primary Actor	Expert
Pre-Condition	Expert must be registered in system.
Basic Flow of System	The expert can login with their mobile number and password.
Alternative Way	There is no alternative way for expert login.

Table 10: Use Case Scenario for Expert Login

11. View Queries/Questions :

Use case ID	UC11
Use Case Name	View Queries/Questions
Primary Actor	Expert
Pre-Condition	Expert must be logged in.
Basic Flow of System	The expert can view the queries/questions of farmer's.
Alternative Way	There is no alternative way for view queries/questions.

Table 11: Use Case Scenario for View Queries/Questions

12. Response of Queries/Questions:

Use case ID	UC12
Use Case Name	Response of Queries/Question
Primary Actor	Expert
Pre-Condition	Expert must be logged in.
Basic Flow of System	The expert can give response of queries/question of farmer's.
Alternative Way	There is no alternative way for giving response of farmer's queries/questions.

Table 12: Use Case Scenario for Response of Queries/Questions

13. Forgot Password:

Use case ID	UC13
Use Case Name	Forgot Password
Primary Actor	Admin, Expert, Farmer
Pre-Condition	They must be registered in system.
Basic Flow of System	They can forget their password through mobile number.
Alternative Way	There is no alternative way for forgot password.

Table 13: Use Case Scenario for Forgot Password

4.2. UML Diagrams :

4.2.1. Class Diagram :

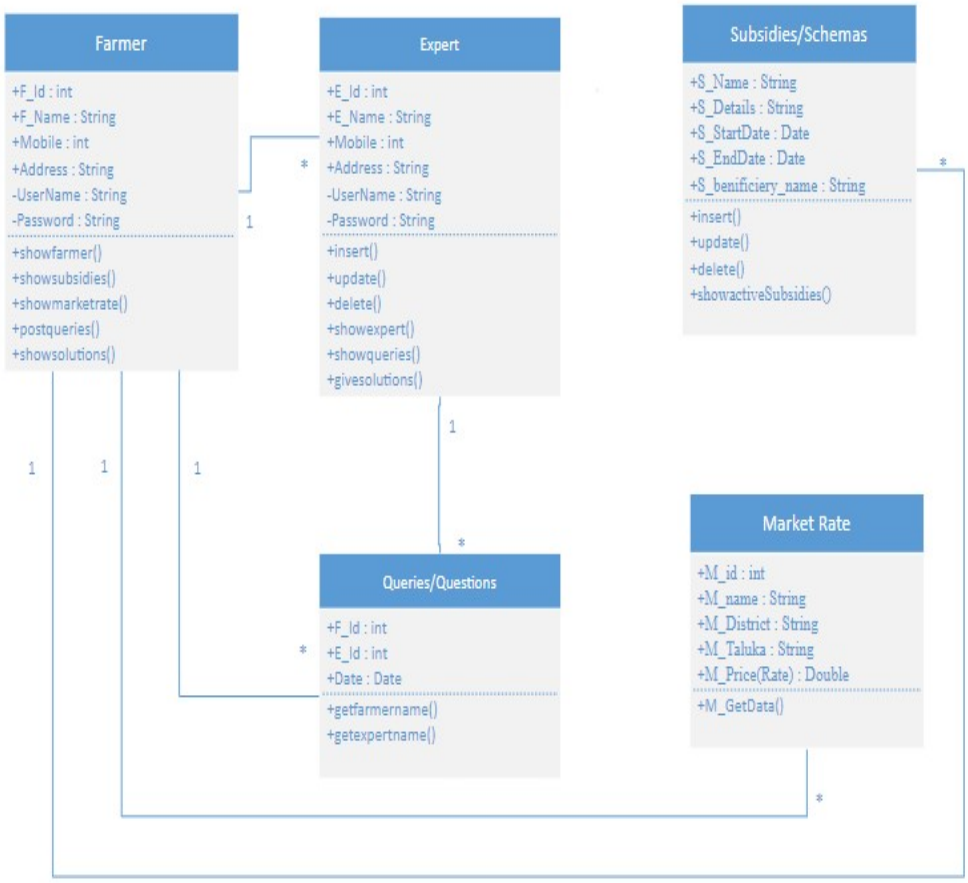


Figure 5 : Class Diagram for Farmer's Assistant

4.2.2. Sequence Diagram :

I. Admin :

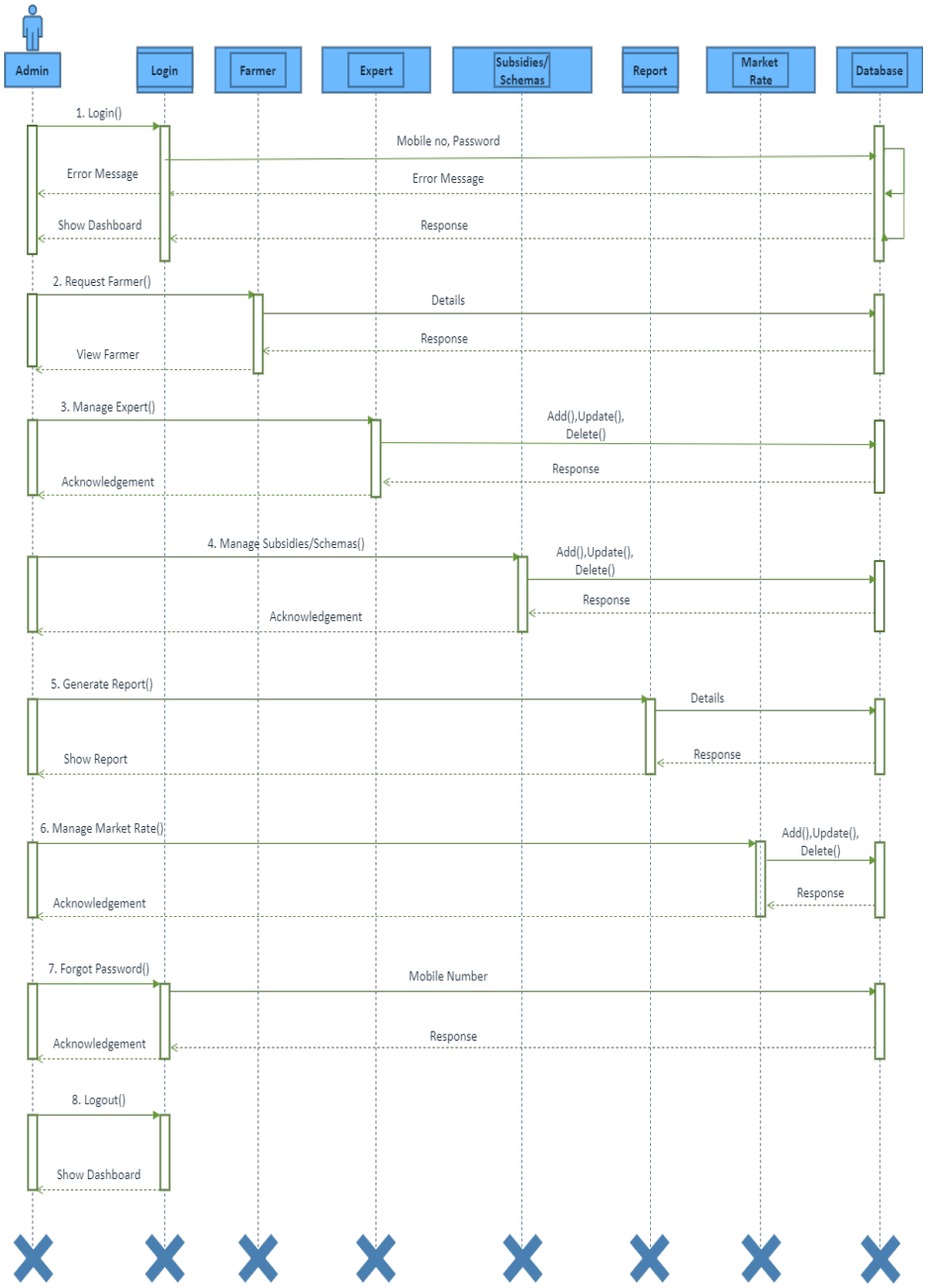


Figure 6 : Sequence Diagram of Admin

II. Expert :

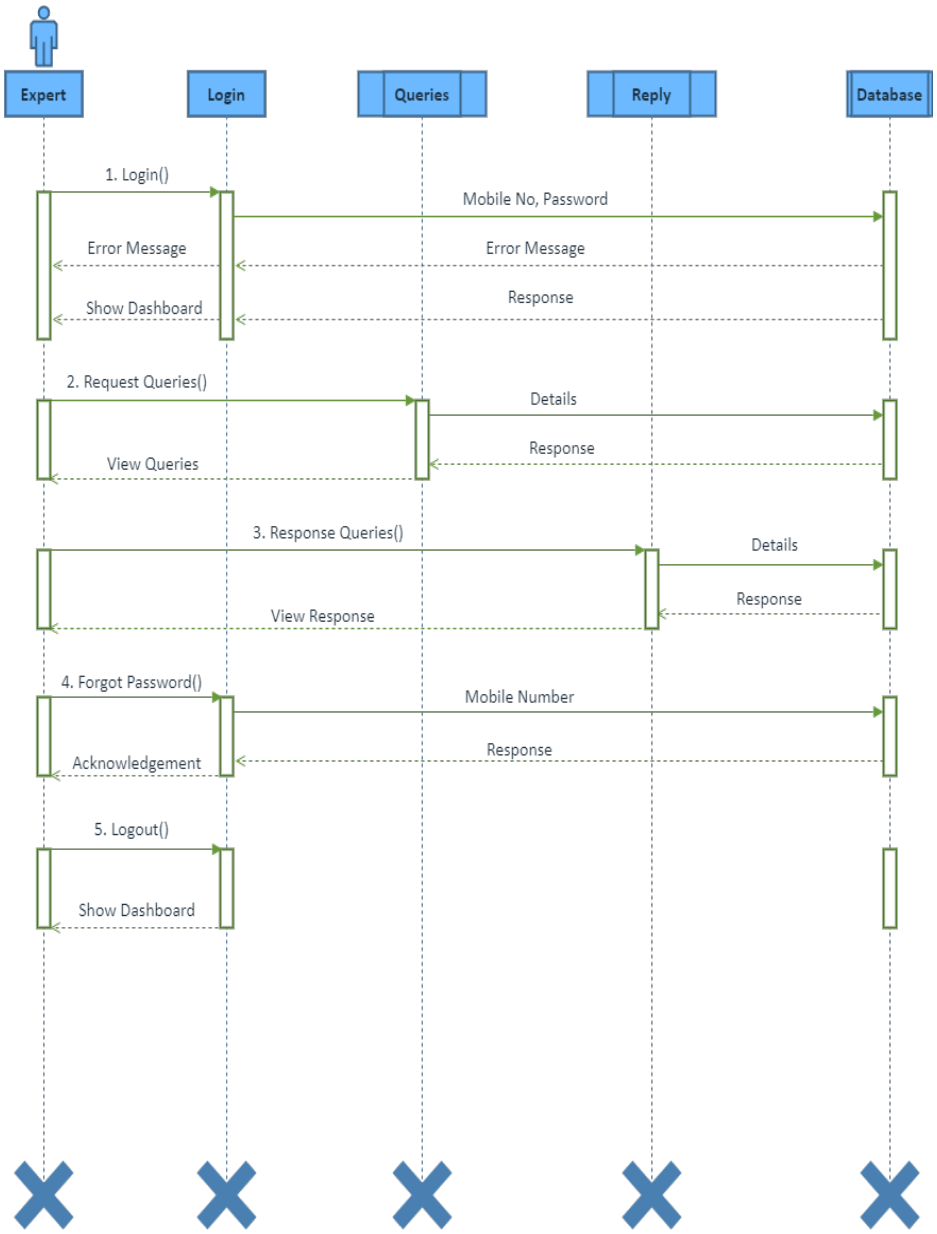


Figure 7 : Sequence Diagram of Expert

III. Farmer :

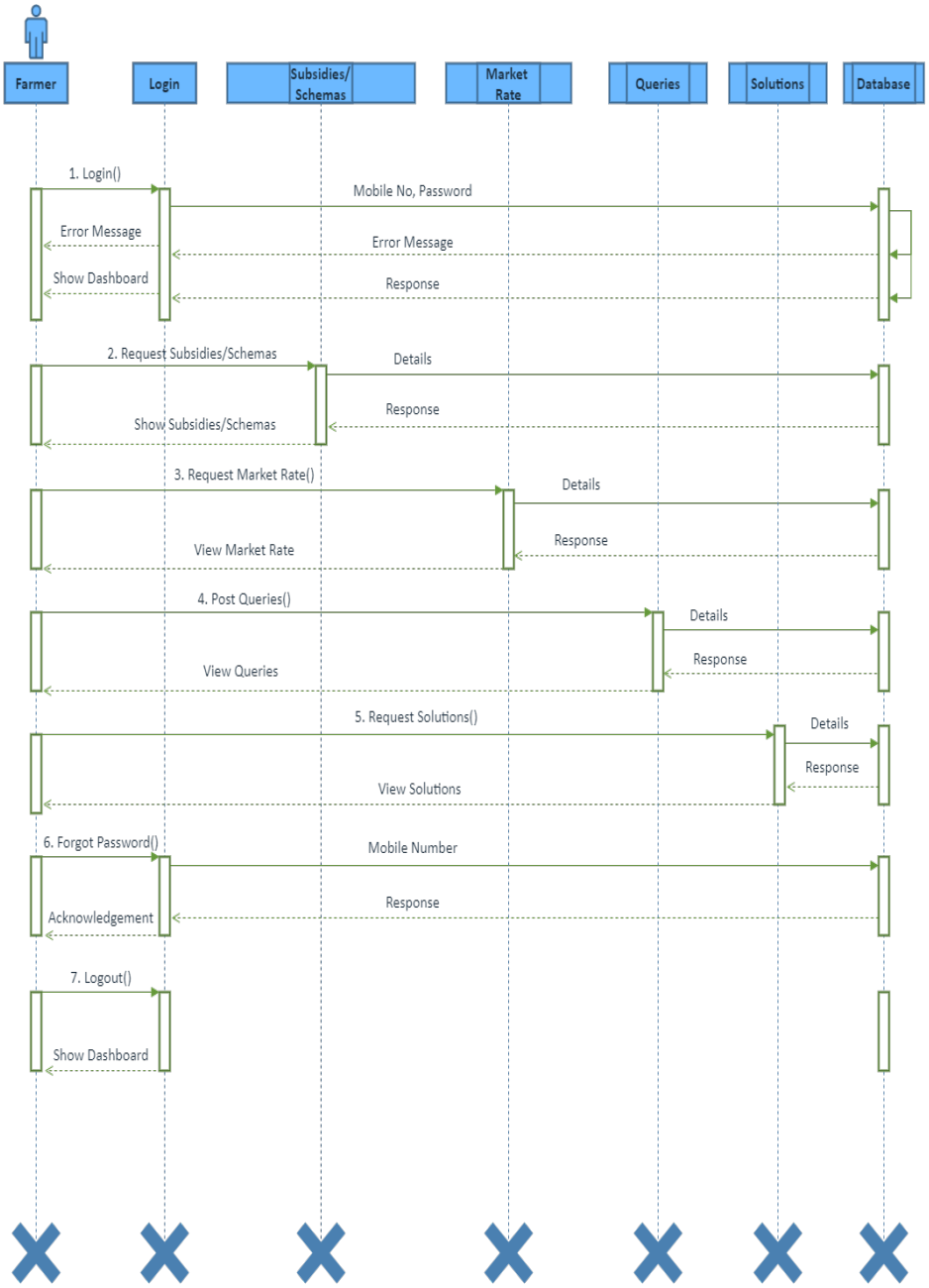


Figure 8 : Sequence Diagram of Farmer

4.2.3. Activity Diagram :

I. Admin :

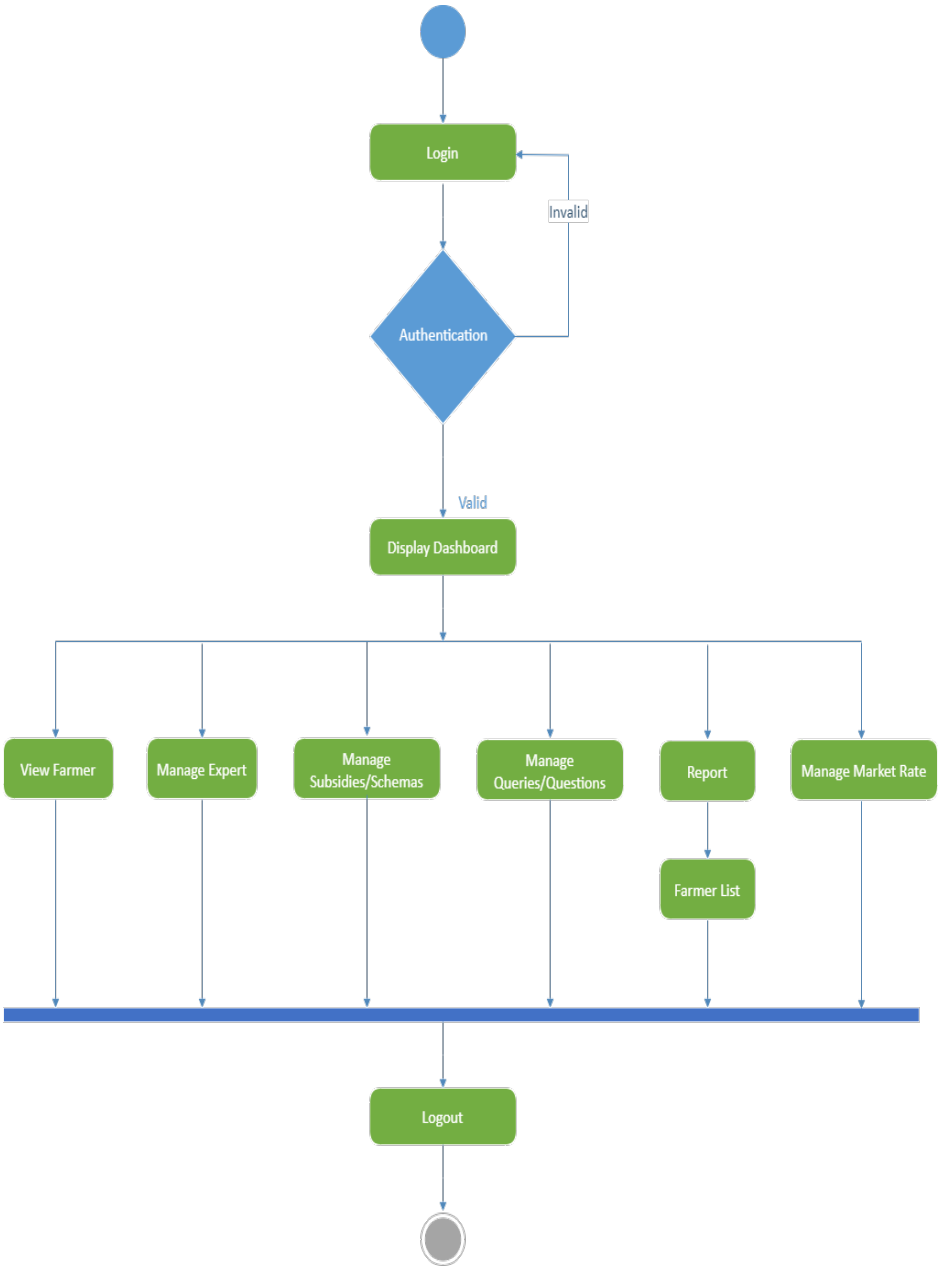


Figure 9 : Activity Diagram of Admin

II. Expert :

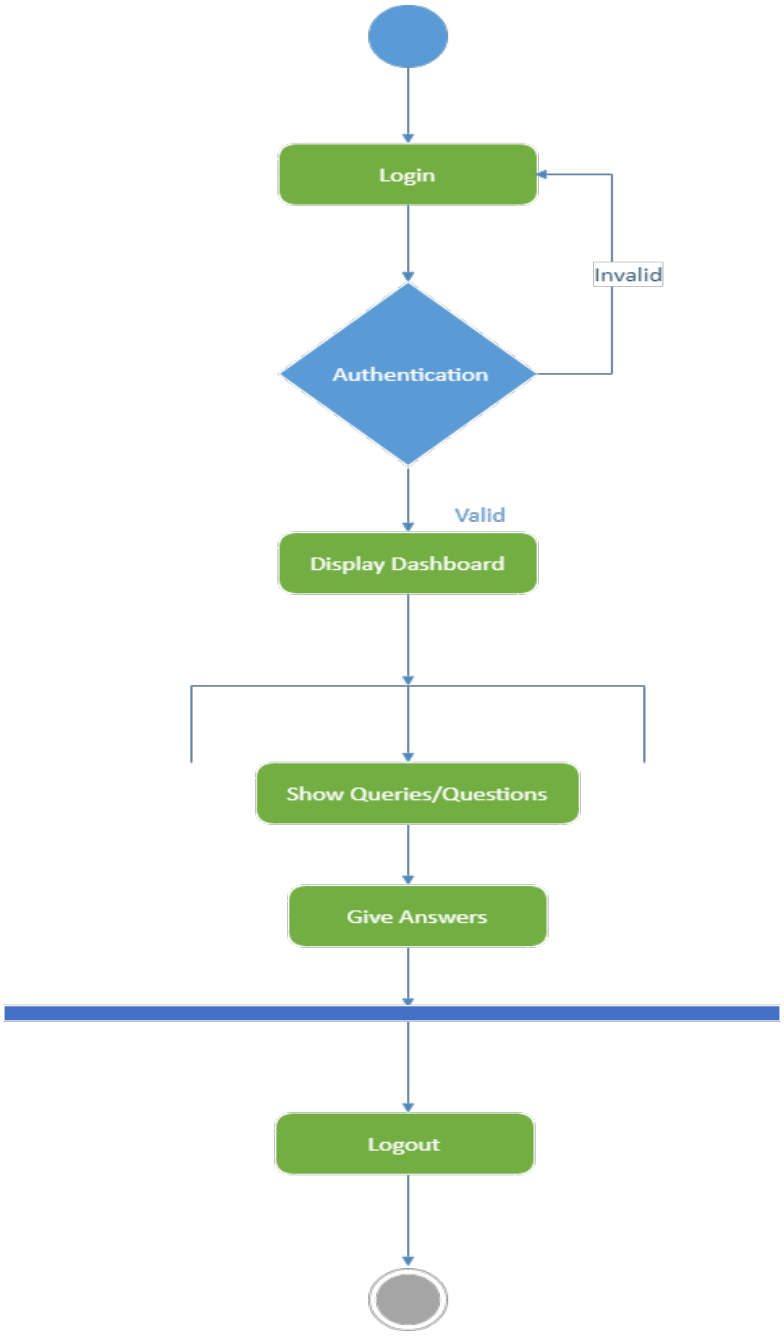


Figure 10 : Activity Diagram of Expert

III. Farmer :

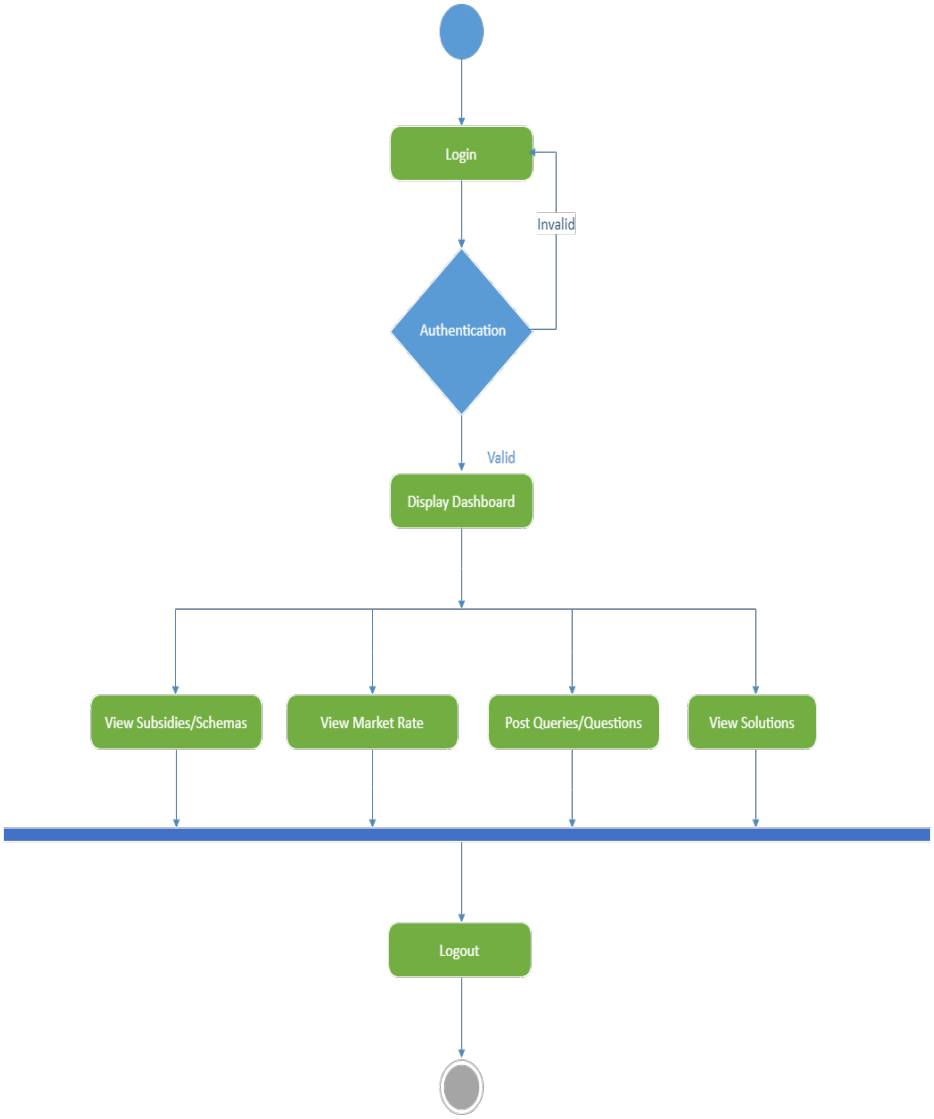


Figure 11 : Activity Diagram of Farmer

4.3. Data Dictionary :

4.3.1. For Farmer Registration :

```
"Users": {
   numeric: {
    "email": "alphanumeric",
    "firstname": "text",
    "lastname": "text",
    "mobile": "numeric",
    "password": "alphanumeric"
   },
   "numeric": {
    "email": "alphanumeric",
    "firstname": "text",
    "lastname": “text",
    "mobile": "numeric",
    "password": "alphanumeric"
   }
  }
}
```

4.3.2. For Tips :


```
"Tips": {
  "text": {
    "tipdesc": "text",
    "tiptitle": "text"
  },
  "text": {
    "tipdesc": "text",
    "tiptitle": "text"
  }
},
```


4.3.3. For Subsidies/Schemas :


```
"text": {
    "subdetails": "text",
    "subendingdate": "date",
    "sublimitperyear": "numeric",
    "subname": "text",
    "subrequiredform": "alphanumeric",
    "substartingdate": "date",
    "subwhocanget": "text"
  }
},
```

5. Implementation

5.1. Form Layouts:

5.1.1. All Module Login:

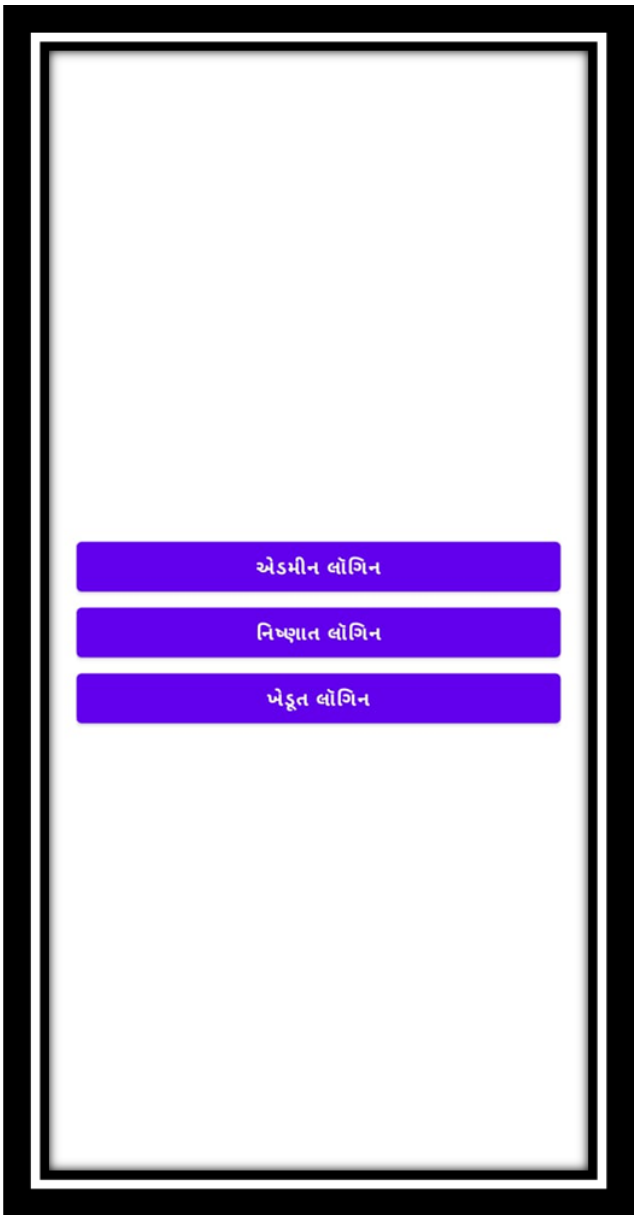


Figure 12: Login Page of All Modules

5.1.2. Admin Home:

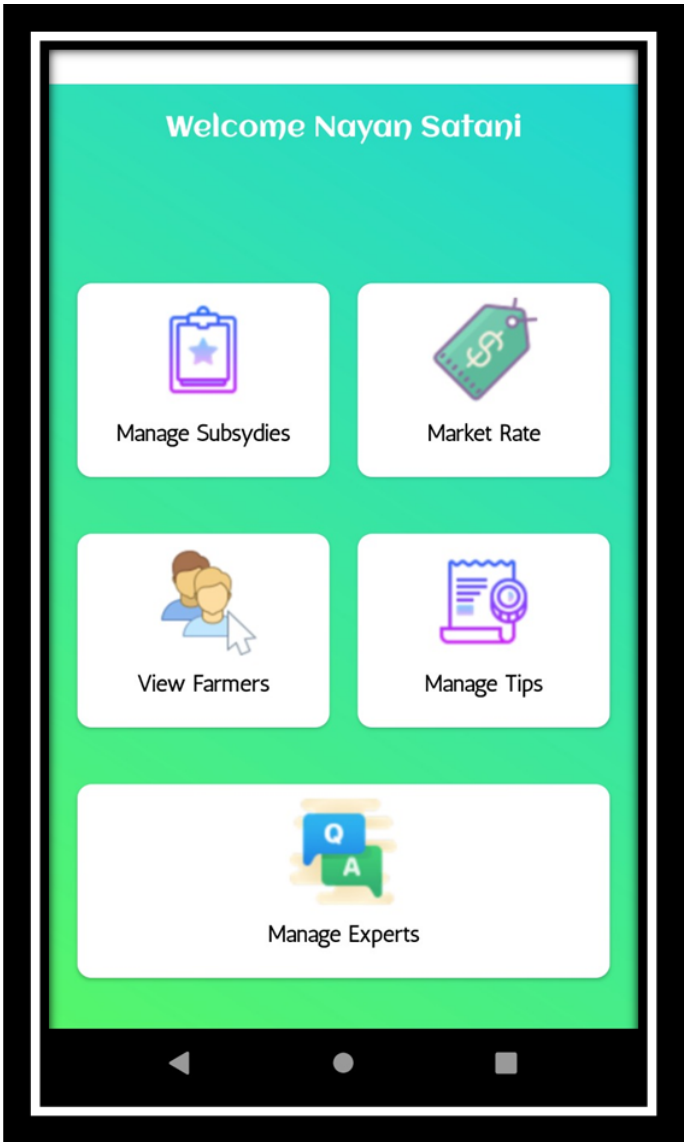


Figure 13: Home Page of Admin

5.1.3. Admin Login:

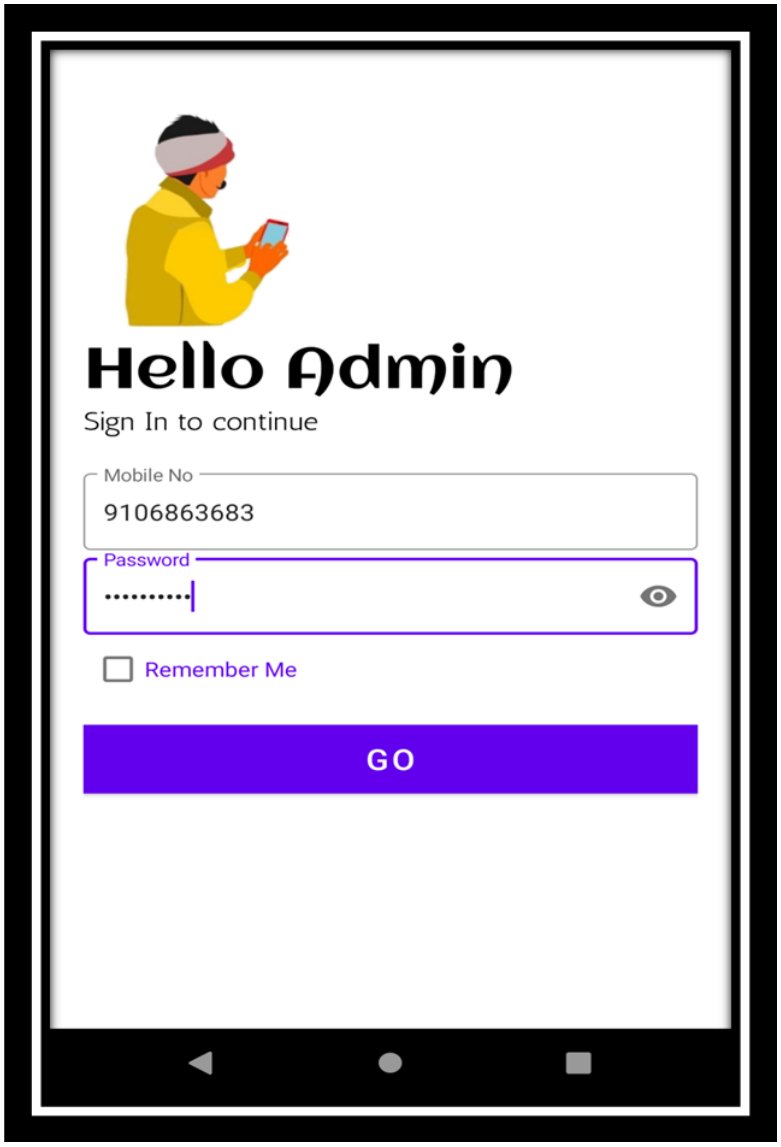


Figure 14: Login Page of Admin

5.1.4. Manage Expert:

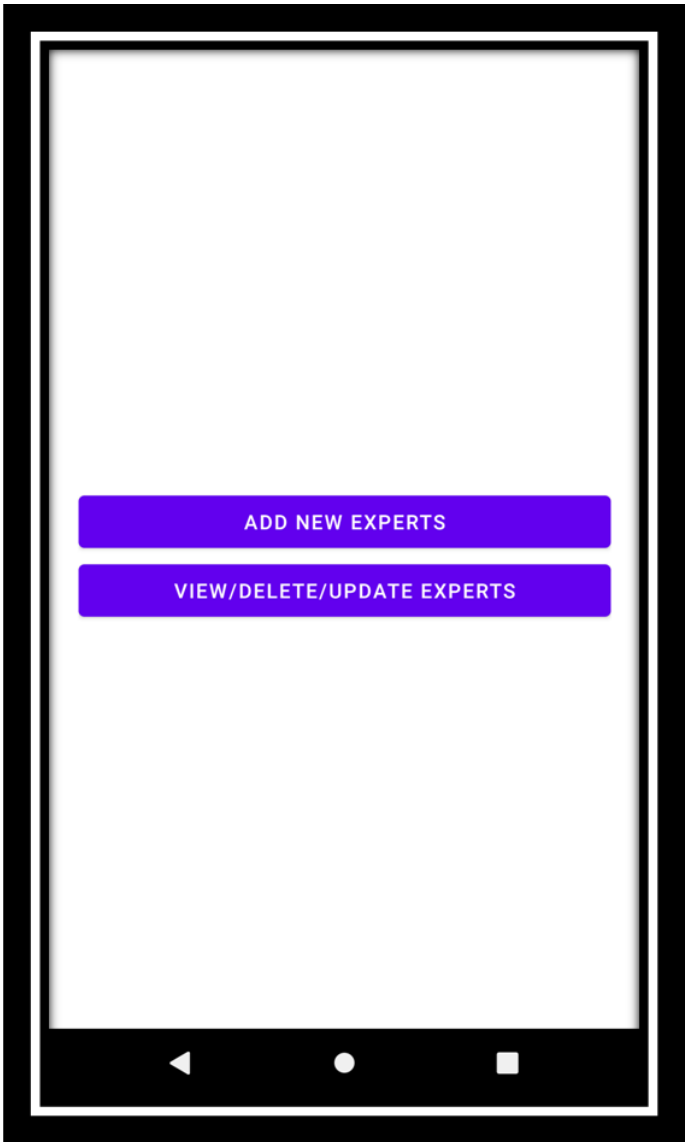


Figure 15: Home Page of Admin Managing Expert

5.1.5. Manage Subsidies:

Figure 16: Home Page of Admin Managing Subsidies

5.1.6. Manage Tips:

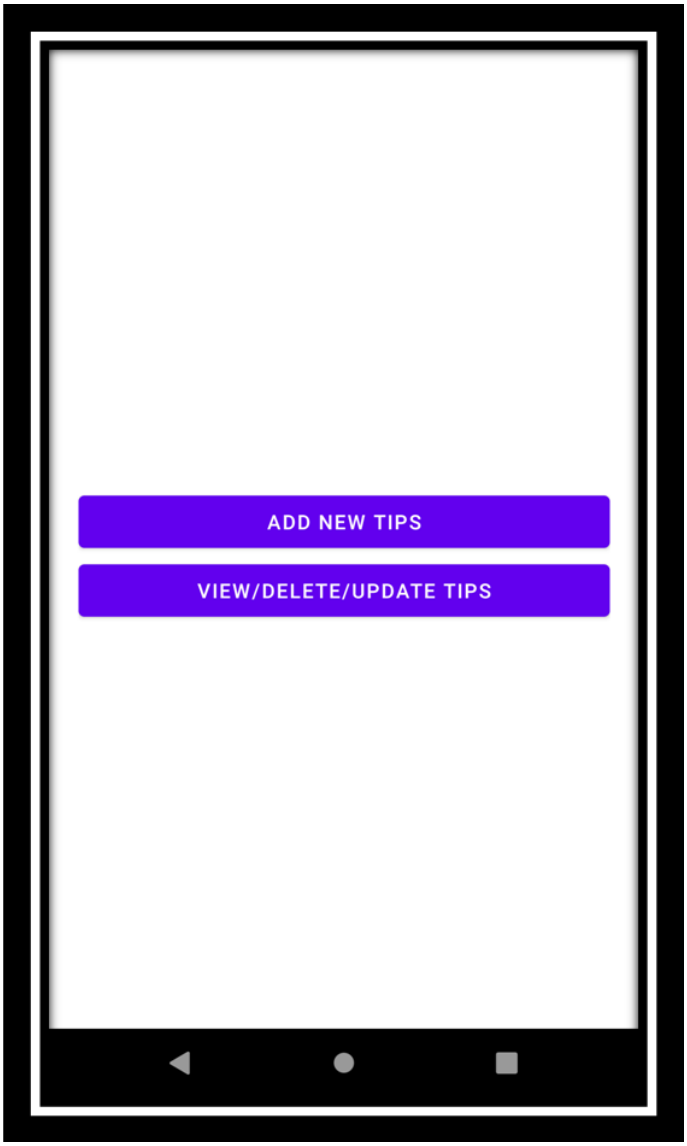


Figure 17: Home Page of Admin Managing Tips

5.1.7. Manage Market Rate:

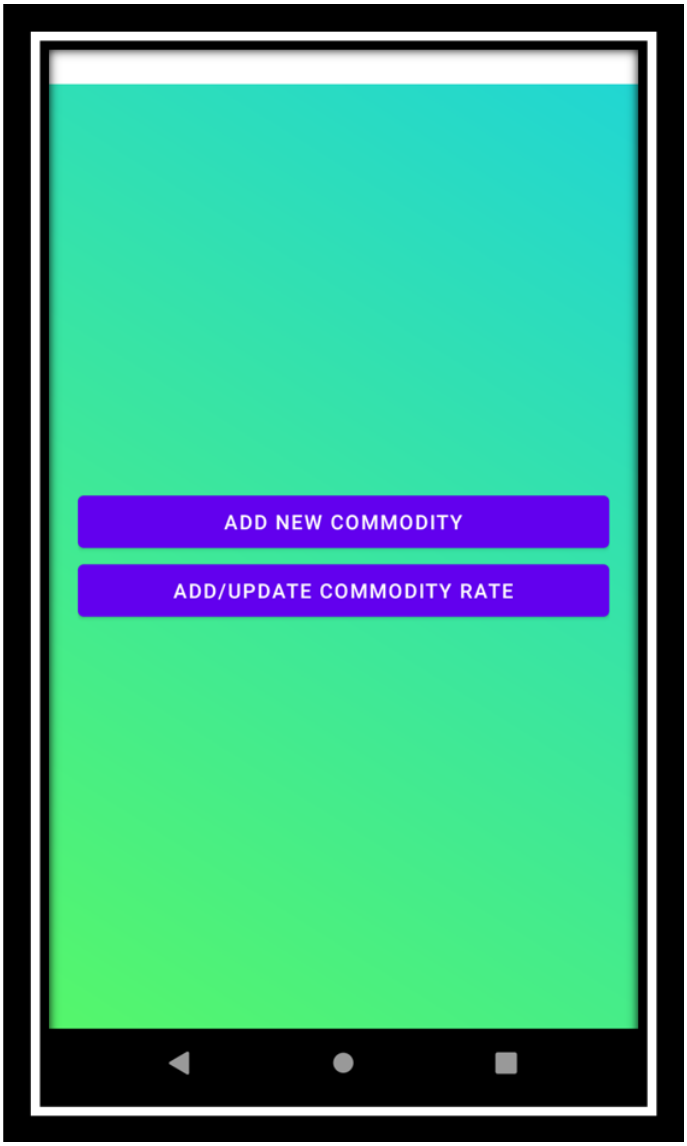


Figure 18: Home Page of Admin Managing Market Rate

5.1.8. Expert Login:

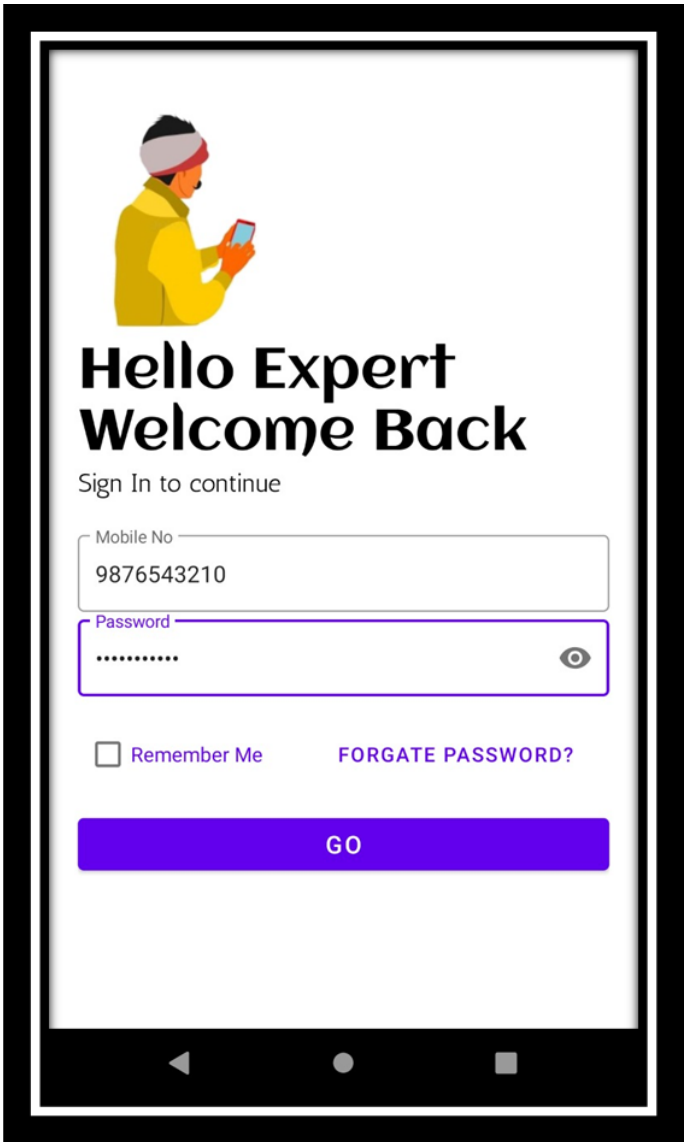


Figure 19: Login Page of Expert

5.1.9. Expert Home:

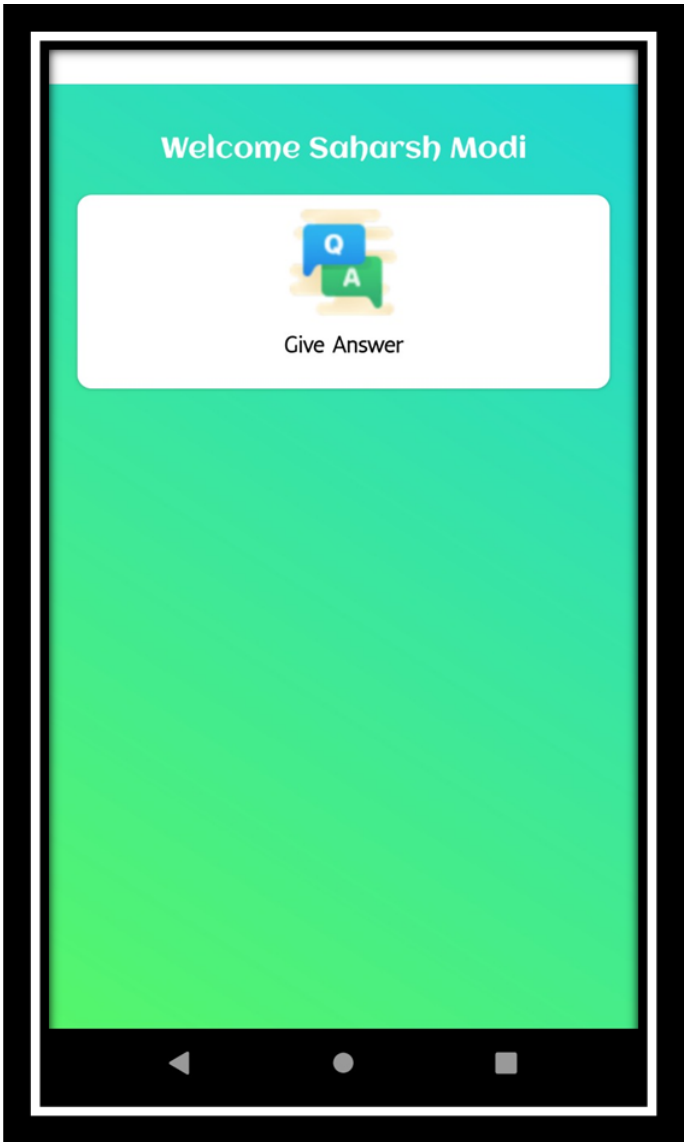


Figure 20: Home Page of Expert

5.1.10. Farmer Login:

Figure 21: Login Page of Farmer

5.1.11. Farmer Registration:

Figure 22: Registration Page of Farmer

5.1.12. Farmer Home:

Figure 23: Home Page of Farmer

5.1.13. Chatting with Expert Home:

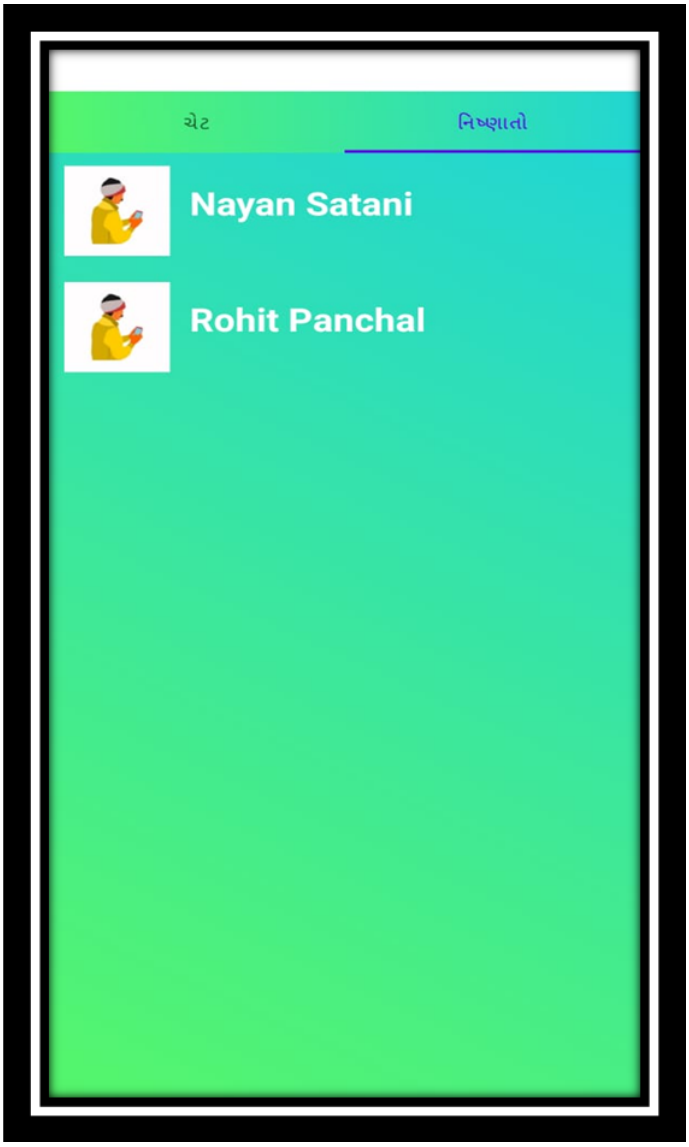


Figure 24: Home Page of Farmer Chatting with Expert

5.1.14. View Market Rate Home:

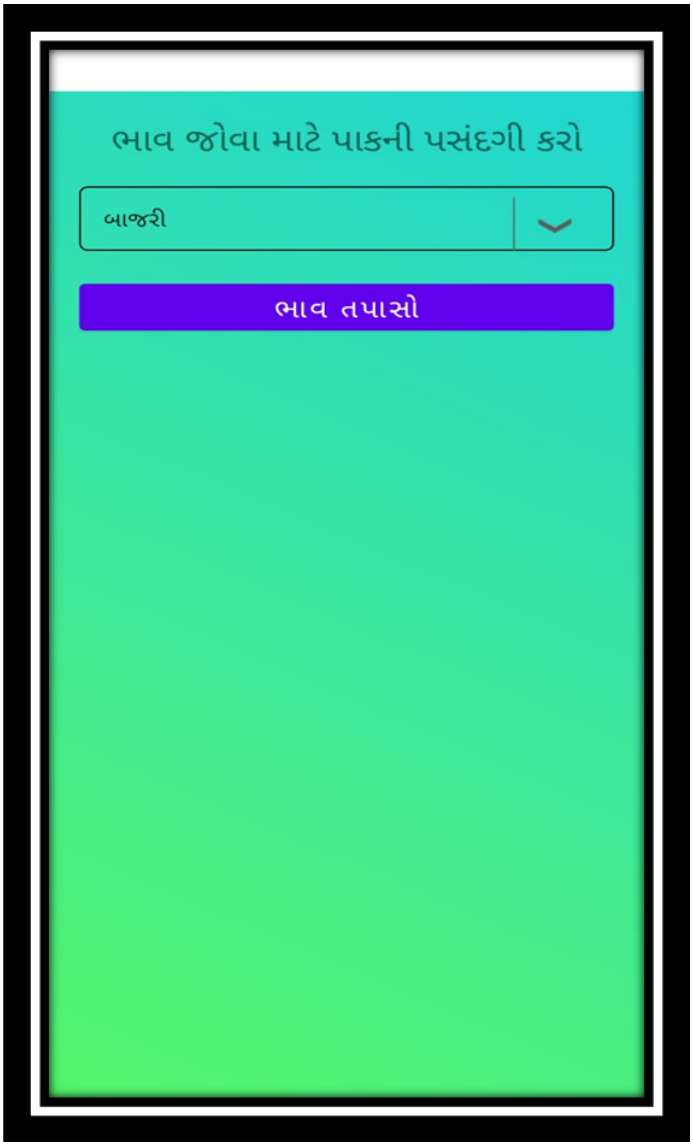


Figure 25: Home Page of Farmer View Market Rate

5.1.15. Forgot Password:

Figure 26: Forgot Password Page

5.2. Report Layouts:

5.2.1. Farmer List:

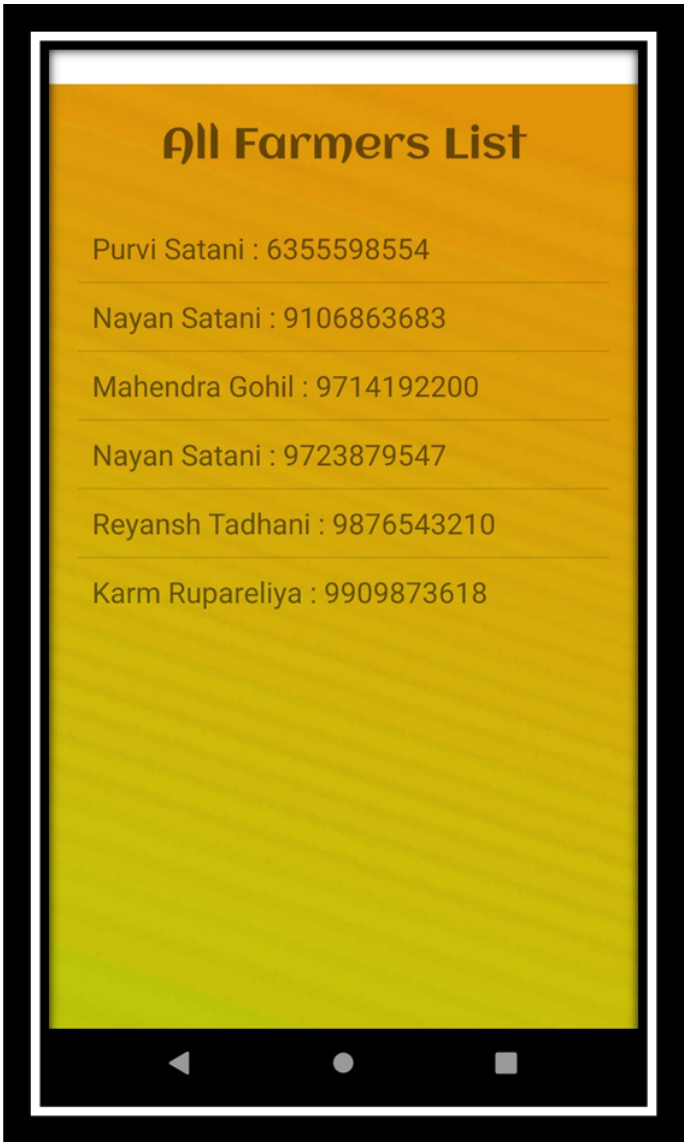


Figure 27: All Farmers list page

5.2.2. Add Expert:

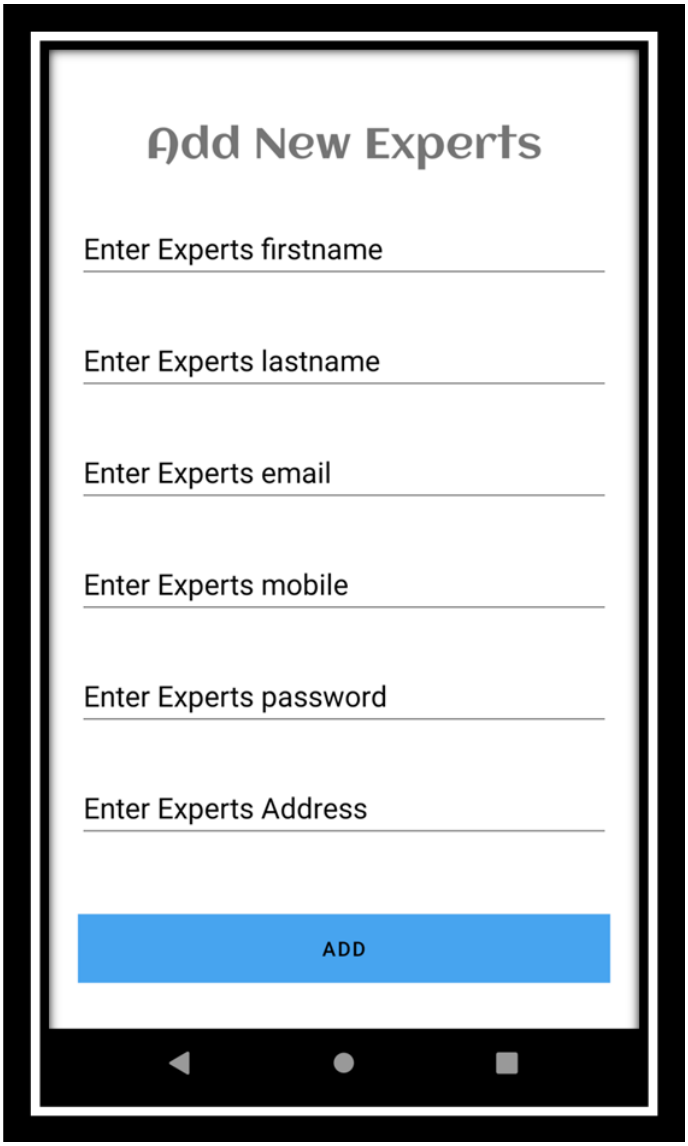


Figure 28: Admin Adding new Expert Page

5.2.3. Delete/Update Expert:

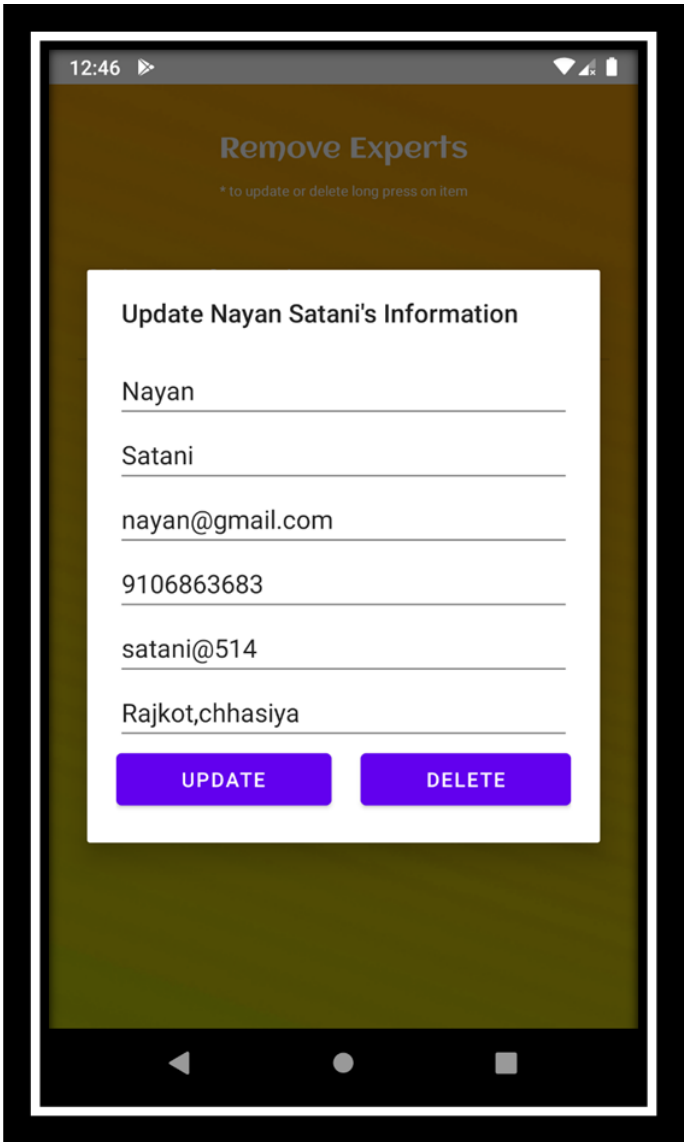


Figure 29: Admin Updating or Deleting Expert Page

5.2.4. Add Subsidy:

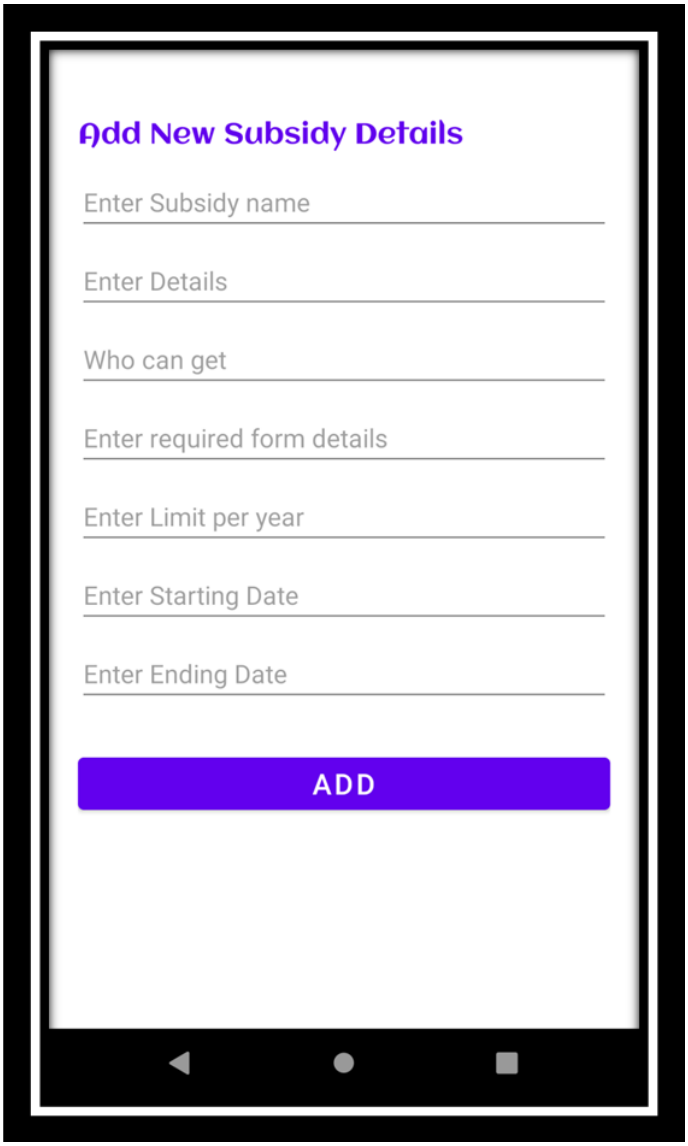


Figure 30: Admin Adding new Subsidy Page

5.2.5. Delete/Update Subsidy:

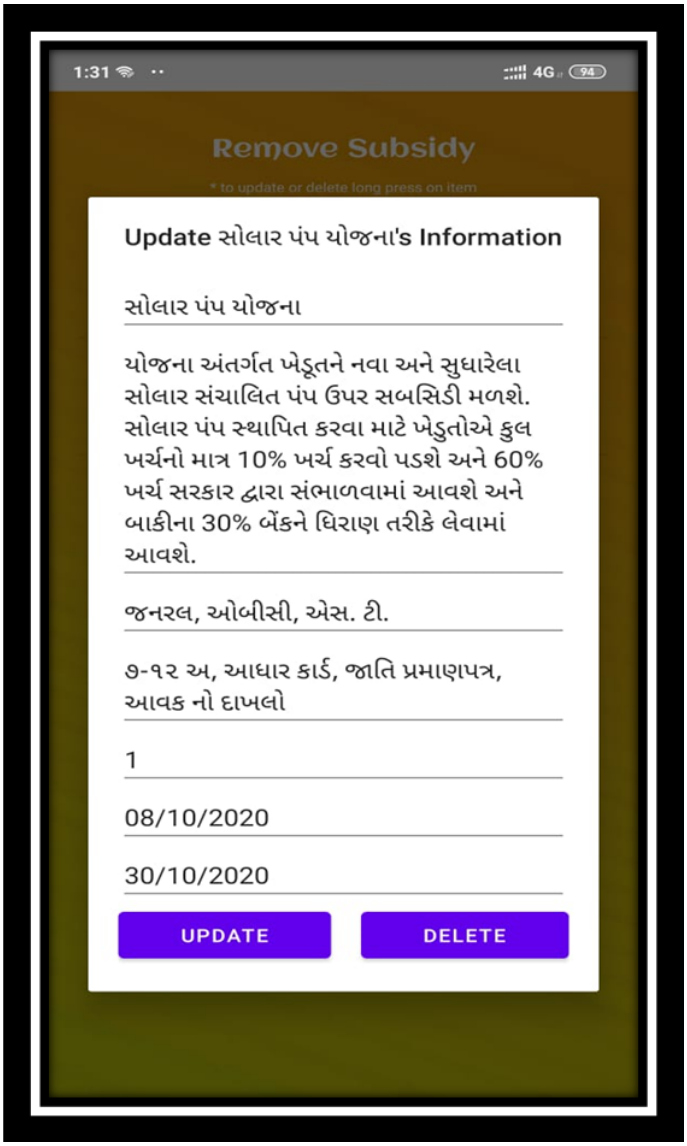


Figure 31: Admin Updating or Deleting Expert Page

5.2.6. Add Tips:

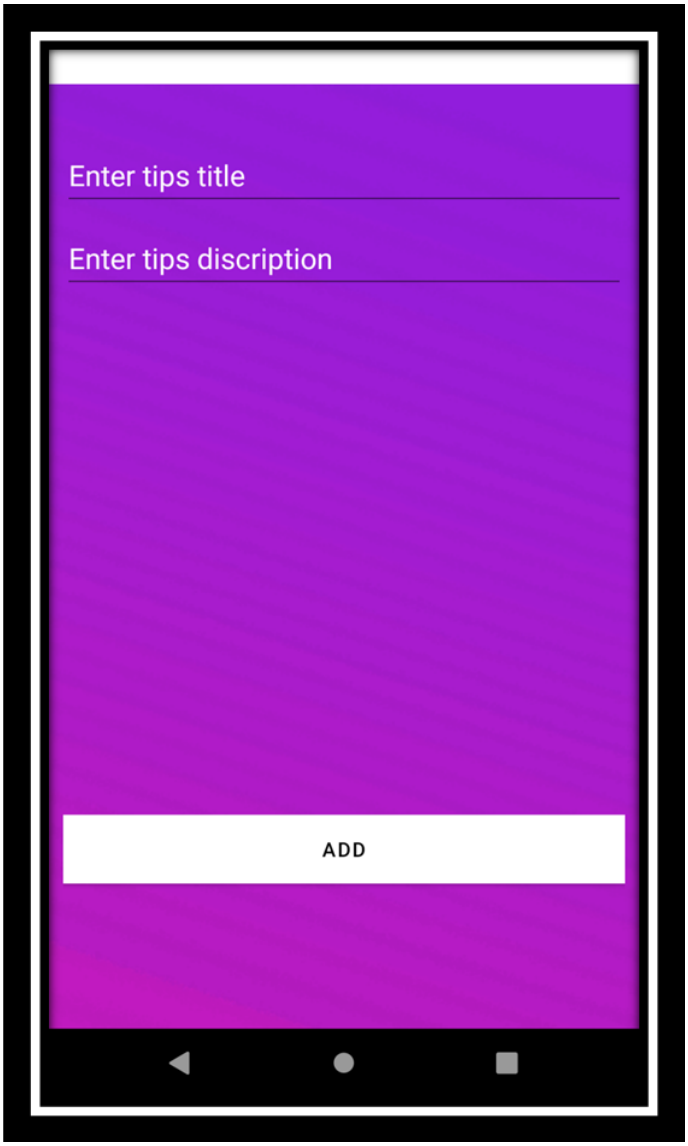


Figure 32: Admin Adding new Tips Page

5.2.7. Delete/Update Tips:

Figure 33: Admin Updating or Deleting Tips Page

5.2.8. Add Commodity:

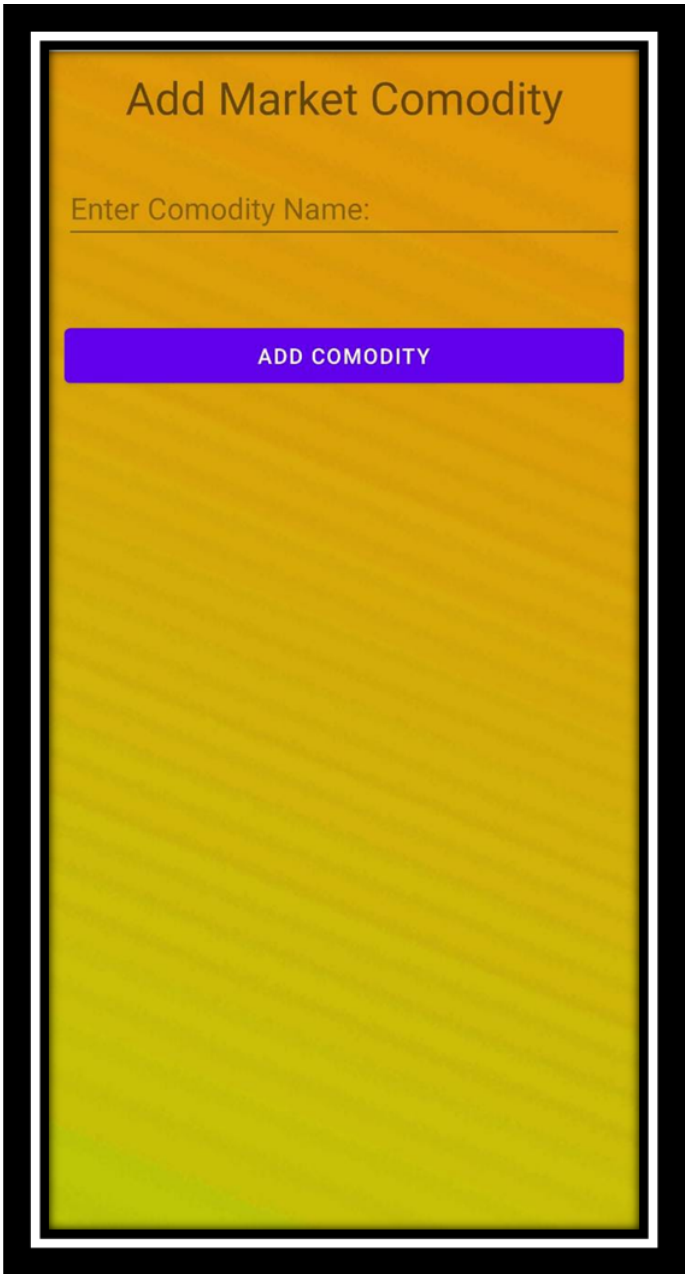


Figure 34: Admin Adding new Commodity Page

5.2.9. Add Commodity Market Rate:

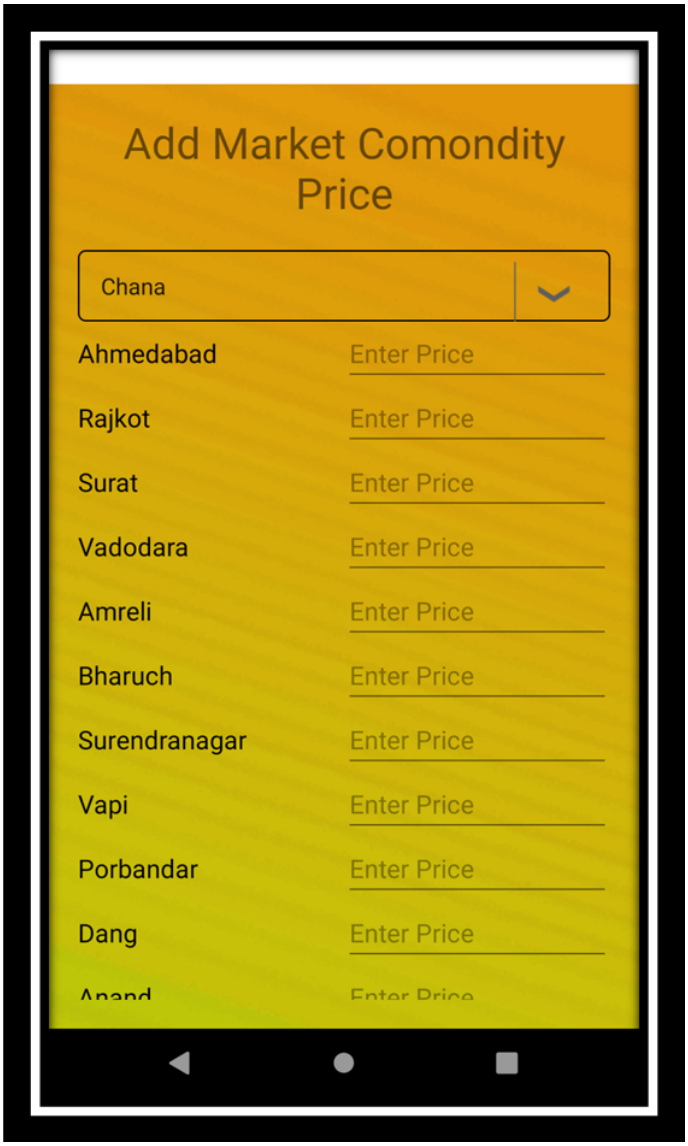


Figure 35: Admin Adding Commodity Market Rate Page

5.2.10. Farmer View Subsidy:

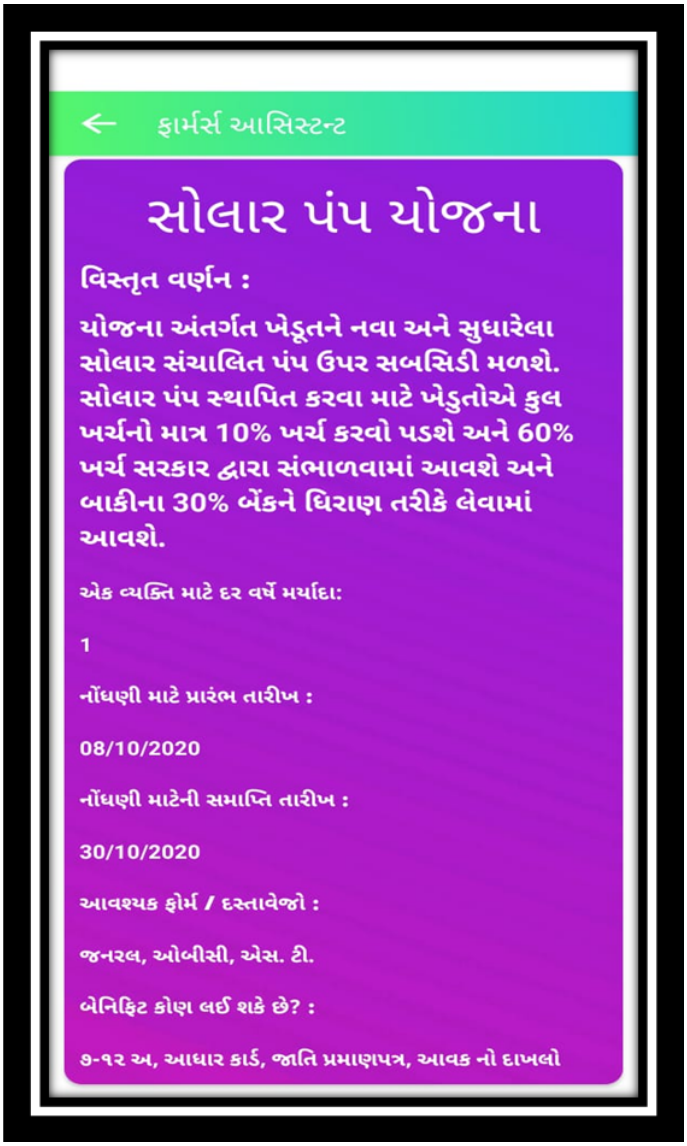


Figure 36: Farmer Viewing Subsidy Page

5.2.11. Farmer View Market Rate:

Figure 37: Farmer Viewing Selected Commodity Market Rate Page

5.2.12. Farmer View Tips:

Figure 38: Farmer Viewing Tips Page

5.2.13. OTP Verification:

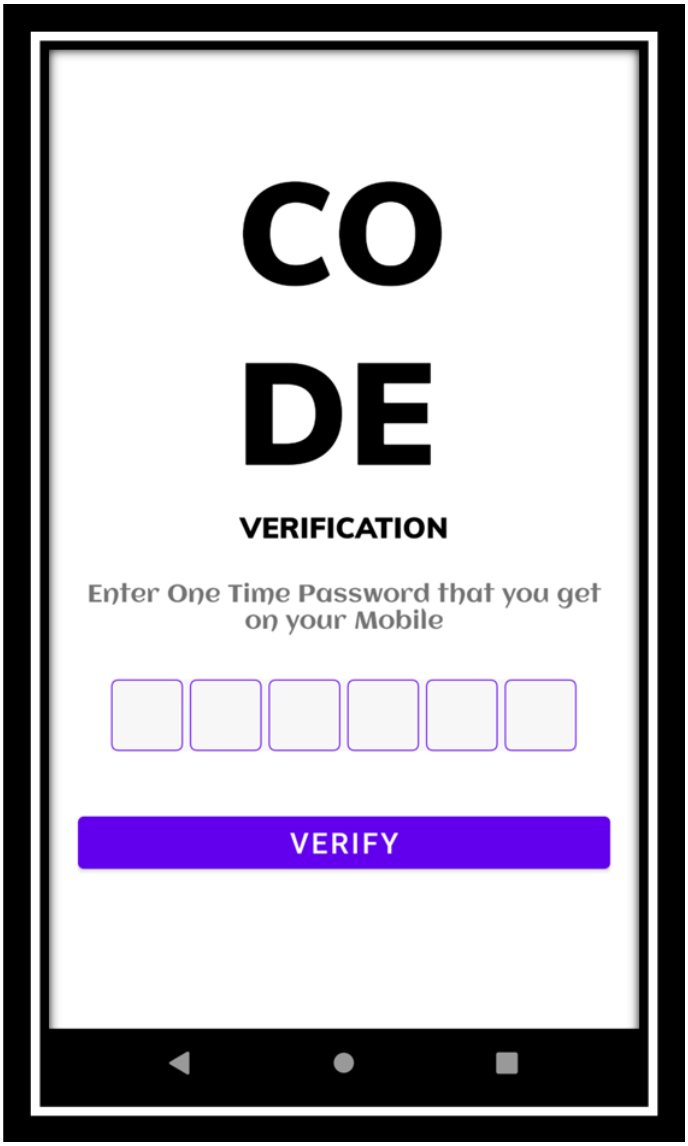


Figure 39: Forgot Password OTP Verification Page

5.3. Coding Conventions:

5.3.1. Add Expert:

```
public void addExpertClick(View view)
{
    String efirstname=expfirstname.getText().toString();
    String elastname=explastname.getText().toString();
    String eemail=expemail.getText().toString();
    String emobile=expmobile.getText().toString();
    String epassword=exppassword.getText().toString();
    String eaddress=expaddress.getText().toString();

    firebaseDatabase = FirebaseDatabase.getInstance();
    reference = firebaseDatabase.getReference( path: "Experts");

    AdminAddExpertHelper helperClass = new AdminAddExpertHelper(efirstname,elastname,eemail,emobile,epassword,eaddress);

    if (TextUtils.isEmpty(efirstname)||TextUtils.isEmpty(elastname)||TextUtils.isEmpty(eemail)||TextUtils.isEmpty(emobile)||
            TextUtils.isEmpty(epassword)||TextUtils.isEmpty(eaddress))
    {
        Toast.makeText( context: AdminAddNewUser.this,  text: "Field must not be empty", Toast.LENGTH_LONG).show();
    }
    else {
        reference.child(emobile).setValue(helperClass);

        Toast.makeText( context: AdminAddNewUser.this,  text: "Expert Added Succsessfully", Toast.LENGTH_LONG).show();
        Intent intent = new Intent( packageContext: AdminAddNewUser.this, AdminManageExpertHome.class);
        startActivity(intent);
        finish();
    }
}
```

5.3.2. Delete/Update Expert:

```
buttonUpdate.setOnClickListener((view) -> {
        sfname = edFirstName.getText().toString().trim();
        slname = edLastName.getText().toString().trim();
        semail = edEmail.getText().toString().trim();
        smobile = edMobile.getText().toString().trim();
        spassword = edPassword.getText().toString().trim();
        saddress = edAddress.getText().toString().trim();

        if (TextUtils.isEmpty(sfname)||TextUtils.isEmpty(slname)||TextUtils.isEmpty(semail)||TextUtils.isEmpty(spassword)
                || TextUtils.isEmpty(smobile)||TextUtils.isEmpty(saddress))
        {
            Toast.makeText( context: AdminRemoveExpert.this,  text: "Field must not be empty", Toast.LENGTH_LONG).show();
        }
        else {
            updateExpert(expertMobile, sfname, slname, semail, spassword, saddress);
            b.dismiss();
        }
});

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

            deleteExpert(expertMobile);
            b.dismiss();
    }
});
```

5.3.3. Forgot Password:

```
public void setNewPassword(View view)
{
    if (!validateNewPassword()||!validateConfirmPassword()||!samePassword())
    {
        return;
    }
    String _newPassword=txtnewpass.getEditText().getText().toString().trim();
    String _phoneNumber=confirmMobile;

    DatabaseReference reference= FirebaseDatabase.getInstance().getReference( path: "Users");
    reference.child(_phoneNumber).child("password").setValue(_newPassword);

    startActivity(new Intent(getApplicationContext(),ForgetSuccessMessage.class));
    finish();
}
```

5.3.4. Add Commodity:

```
public void addComClick(View view)
{
    firebaseDatabase = FirebaseDatabase.getInstance();
    reference = firebaseDatabase.getReference( path: "Commodity");

    String coName = edtAddCom.getText().toString();

    CommodityHelper commodityHelper=new CommodityHelper(coName);

    if (TextUtils.isEmpty(coName))
    {
        Toast.makeText( context: AdminAddMarketRate.this, text: "Commondity Field Must not be Empty !",Toast.LENGTH_LONG).show();
    }
    else {
        reference.push().setValue(commodityHelper);
        Toast.makeText( context: this,  text: "Commodity Added !", Toast.LENGTH_LONG).show();
        Intent intent = new Intent( packageContext: AdminAddMarketRate.this, AdminAddPrice.class);
        startActivity(intent);
        finish();
    }

}
```

5.3.5. Add Commodity Market Rate:

```
btnAddPrice.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //Toast.makeText(AdminAddPrice.this,selectedCom,Toast.LENGTH_LONG).show();
        reference=FirebaseDatabase.getInstance().getReference( path: "Price").child(selectedCom);

        reference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {

                String ahmedabad =getText(R.string.Ahmedabad)+ahmedabad1.getText().toString();
                String rajkot =getText(R.string.Rajkot)+rajkot1.getText().toString();
                String surat = getText(R.string.Surat)+surat1.getText().toString();
                String vadodara =getText(R.string.Vadodara)+vadodara1.getText().toString();
                String amreli =getText(R.string.Amreli)+amreli1.getText().toString();
                String bharuch =getText(R.string.Bharuch)+bharuch1.getText().toString();
                String surendranagar =getText(R.string.Surendranagar)+surendranagar1.getText().toString();
                String vapi =getText(R.string.Vapi)+ vapi1.getText().toString();
                String porbandar =getText(R.string.Porbandar)+ porbandar1.getText().toString();
                String dang =getText(R.string.Dang)+ dang1.getText().toString();
                String anand =getText(R.string.Anand)+ anand1.getText().toString();
                String bhavnagar =getText(R.string.Bhavnagar)+ bhavnagar1.getText().toString();
                String junagadh =getText(R.string.Junagadh)+ junagadh1.getText().toString();

                CityHelper cityHelper=new CityHelper(ahmedabad,rajkot,surat,vadodara,amreli,bharuch,surendranagar,vapi,
                        porbandar,dang,anand,bhavnagar,junagadh);
```

```
                if (TextUtils.isEmpty(ahmedabad1.getText().toString())||(TextUtils.isEmpty(rajkot1.getText().toString())
                        ||(TextUtils.isEmpty(surat1.getText().toString()))|| (TextUtils.isEmpty(vadodara1.getText().toString())
                        ||(TextUtils.isEmpty(amreli1.getText().toString()))||(TextUtils.isEmpty(bharuch1.getText().toString())
                        ||(TextUtils.isEmpty(surendranagar1.getText().toString()))||(TextUtils.isEmpty(vapi1.getText().toString())
                        ||(TextUtils.isEmpty(porbandar1.getText().toString()))||(TextUtils.isEmpty(dang1.getText().toString())
                        ||(TextUtils.isEmpty(anand1.getText().toString()))||(TextUtils.isEmpty(bhavnagar1.getText().toString())
                        ||(TextUtils.isEmpty(junagadh1.getText().toString()))))))))))))
                {
                    Toast.makeText( context: AdminAddPrice.this, text: "All Fields must not be empty !",Toast.LENGTH_LONG).show();
                }
                else {
                    reference.setValue(cityHelper);
                    Toast.makeText( context: AdminAddPrice.this, text: selectedCom+" Price Added !", Toast.LENGTH_LONG).show();
                }
                  Intent intent=new Intent(AdminAddMarketRate.this,AdminAddPrice.class);
                  startActivity(intent);
            }

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

            }
        });
    }
});
```

5.3.6. Add Subsidy:

```
public void addSubClick(View view)
{
..
    String sname=edSubname.getText().toString();
    String sdetails=edSubDetails.getText().toString();
    String swhocanget=edWhocanget.getText().toString();
    String srequiredform=edRequiredform.getText().toString();
    String slimitperyear=edLimitperyear.getText().toString();
    String sstartingdate=edStartingdate.getText().toString();
    String sendingdate=edEndingdate.getText().toString();

    firebaseDatabase = FirebaseDatabase.getInstance();
    reference = firebaseDatabase.getReference( path: "Subsidy");

    AdminSubHelperClass helperClass = new AdminSubHelperClass(sname,sdetails,swhocanget,srequiredform,
            slimitperyear,sstartingdate,sendingdate);

    if (TextUtils.isEmpty(sname)||TextUtils.isEmpty(sdetails)||TextUtils.isEmpty(swhocanget)||TextUtils.isEmpty(srequiredform
            || TextUtils.isEmpty(slimitperyear)||TextUtils.isEmpty(sstartingdate)||TextUtils.isEmpty(sendingdate))
    {
        Toast.makeText( context: AdminAddSubsidy.this, text: "Field must not be empty", Toast.LENGTH_LONG).show();
    }
    else {
        reference.child(sname).setValue(helperClass);

        Toast.makeText( context: AdminAddSubsidy.this, text: "Subsidy Added Succsessfully", Toast.LENGTH_LONG).show();
        Intent intent = new Intent( packageContext: AdminAddSubsidy.this, AdminHome.class);
        startActivity(intent);
        finish();
    }
}
```

5.3.7. Delete/Update Subsidy:

```
private void showUpdateDeleteDialog(final String subsidyName, final String subsidydetails,
                                    final String requiredform,final String whocanget,final String limitperyear,
                                    final String startingdate,final String endingdate) {

    AlertDialog.Builder dialogBuilder = new AlertDialog.Builder( context: this);
    LayoutInflater inflater = getLayoutInflater();
    final View dialogView = inflater.inflate(R.layout.update_dialog_subsidy,  root: null);
    dialogBuilder.setView(dialogView);

    final EditText edSubName = (EditText) dialogView.findViewById(R.id.editTextSubsidyName);
    final EditText edSubDetails = (EditText) dialogView.findViewById(R.id.editTextSubsidyDetails);
    final EditText edRequiredForm = (EditText) dialogView.findViewById(R.id.editTextRequiredForm);
    final EditText edWhocanget = (EditText) dialogView.findViewById(R.id.editTextWhocanGet);
    final EditText edLimitperyear = (EditText) dialogView.findViewById(R.id.editTextLimitPerYear);
    edStartingDate= (EditText) dialogView.findViewById(R.id.editTextStartingDate);
    eedEndingDate = (EditText) dialogView.findViewById(R.id.editTextEndingDate);
    final Button buttonUpdate = (Button) dialogView.findViewById(R.id.buttonUpdateSubsidy);
    final Button buttonDelete = (Button) dialogView.findViewById(R.id.buttonDeleteSubsidy);


    edSubName.setText(subsidyName);
    edSubDetails.setText(subsidydetails);
    edRequiredForm.setText(requiredform);
    edLimitperyear.setText(limitperyear);
    edWhocanget.setText(whocanget);
    edStartingDate.setText(startingdate);
    eedEndingDate.setText(endingdate);

    //Datepicker dialog
```

```
    dialogBuilder.setTitle("Update " + subsidyName+ "'s Information");
    final AlertDialog b = dialogBuilder.create();
    b.show();

    buttonUpdate.setOnClickListener((view) → {
            String sSubname = subsidyName;
            String sSubdetails = edSubDetails.getText().toString().trim();
            String sRequiredForm = edRequiredForm.getText().toString().trim();
            String sWhocanget = edWhocanget.getText().toString().trim();
            String sLimitperyear = edLimitperyear.getText().toString().trim();
            String sStartingDate = edStartingDate.getText().toString().trim();
            String sEndingDate = eedEndingDate.getText().toString().trim();

            if (TextUtils.isEmpty(sSubdetails)||TextUtils.isEmpty(sRequiredForm)||TextUtils.isEmpty(sWhocanget)||
                    TextUtils.isEmpty(sLimitperyear)|| TextUtils.isEmpty(sStartingDate)||TextUtils.isEmpty(sEndingDate))
            {
                Toast.makeText( context: AdminRemoveSubsidy.this,  text: "Field must not be empty", Toast.LENGTH_LONG).show();
            }
            else {
                updateSubsidy(subsidyName, sSubdetails, sRequiredForm, sWhocanget, sLimitperyear, sStartingDate, sEndingDate);
                b.dismiss();
            }
    });

    buttonDelete.setOnClickListener((view) → {
            deleteSubsidy(subsidyName);
            b.dismiss();
    });
```

5.3.8. Add Tips:

```
public void addTipClick(View view)
{
    String tname=txttiptitle.getText().toString();
    String ttitle=txttipdesc.getText().toString();


    firebaseDatabase = FirebaseDatabase.getInstance();
    reference = firebaseDatabase.getReference( path: "Tips");

    AdminTipHelperClass helperClass = new AdminTipHelperClass(tname,ttitle);

    if (TextUtils.isEmpty(tname)||TextUtils.isEmpty(ttitle))
    {
        Toast.makeText( context: AdminAddGetTips.this, text: "All Field Must not be Empty !",Toast.LENGTH_LONG).show();
    }
    else {
        reference.child(tname).setValue(helperClass);

        Toast.makeText( context: AdminAddGetTips.this,  text: "Tips Added Succsessfully", Toast.LENGTH_LONG).show();
        Intent intent = new Intent( packageContext: AdminAddGetTips.this, AdminHome.class);
        startActivity(intent);
        finish();
    }
}
```

5.3.9. Delete/Update Tips:

```
AlertDialog.Builder dialogBuilder = new AlertDialog.Builder( context: this);
LayoutInflater inflater = getLayoutInflater();
final View dialogView = inflater.inflate(R.layout.update_dialog_tips,  root: null);
dialogBuilder.setView(dialogView);

final EditText edtipsName = (EditText) dialogView.findViewById(R.id.editTextTipsName);
final EditText edtipsDesc = (EditText) dialogView.findViewById(R.id.editTextTipsDetails);
final Button buttonUpdate = (Button) dialogView.findViewById(R.id.buttonUpdateTips);
final Button buttonDelete = (Button) dialogView.findViewById(R.id.buttonDeleteTips);

edtipsName.setText(tipName);
edtipsDesc.setText(tipsdesc);

dialogBuilder.setTitle("Update " + tipName+ "'s Information");
final AlertDialog b = dialogBuilder.create();
b.show();

buttonUpdate.setOnClickListener((view) -> {
        String sTipsname = edtipsName.getText().toString().trim();
        String sTipsdetails = edtipsDesc.getText().toString().trim();

        if (TextUtils.isEmpty(sTipsname)||TextUtils.isEmpty(sTipsdetails))
        {
            Toast.makeText( context: AdminRemoveTips.this, text: "All Field Must not be Empty !",Toast.LENGTH_LONG).show();
        }
        else {
            updateTips(tipName, sTipsdetails);
            b.dismiss();
        }
});
```

```
    buttonDelete.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            deleteTips(tipName);
            b.dismiss();
        }
    });
}

private boolean updateTips( String tiname, String tidetails)
{
    DatabaseReference dbReference=FirebaseDatabase.getInstance().getReference( path: "Tips").child(tiname);
    AdminTipHelperClass adminTipHelperClass = new AdminTipHelperClass(tiname,tidetails);
    dbReference.setValue(adminTipHelperClass);
    Toast.makeText(getApplicationContext(),  text: "Tit Updated", Toast.LENGTH_LONG).show();
    return true;
}

private void deleteTips(String sTipsName)
{
    DatabaseReference drExpert=FirebaseDatabase.getInstance().getReference( path: "Tips").child(sTipsName);
    drExpert.removeValue();
    Toast.makeText( context: this, text: "Tips Deleted",Toast.LENGTH_LONG).show();
}
```

5.3.10. Farmer Registration:

```
public void regUser(View view) {

    if (!validateFirstname() || !validateLastname() || !validateEmail() || !validateMobile() || !validatePassword())
    {
        return;
    }
    String sfname = firstname.getEditText().getText().toString();
    String slname = lastname.getEditText().getText().toString();
    String semail = email.getEditText().getText().toString();
    String smobile = mobile.getEditText().getText().toString();
    String spassword = password.getEditText().getText().toString();
    String completeMobileNumber="+91"+smobile;

    rootNode = FirebaseDatabase.getInstance();
    reference = rootNode.getReference( path: "Users");

    UserHelperClass helperClass = new UserHelperClass(sfname, slname, semail, smobile, spassword);

    reference.child(smobile).setValue(helperClass);
    //Toast.makeText(this,"Registred !",Toast.LENGTH_LONG).show();
    Intent intent=new Intent( packageContext: Register.this,OTPActivity.class);

    intent.putExtra( name: "fullmobile",completeMobileNumber);
    startActivity(intent);
}

private Boolean validateFirstname() {...}
private Boolean validateLastname() {...}
private Boolean validateEmail() {...}
private Boolean validateMobile() {...}
private Boolean validatePassword() {...}
```

5.3.11. Farmer Login:

```
public void isUser() {
    final String userEnteredMobile=mobile.getEditText().getText().toString().trim();
    final String userEnteredPassword=password.getEditText().getText().toString().trim();
    //Remember Me
    if (rememberMe.isChecked())
    {
        SessionManagement sessionManagement=new SessionManagement( _context: Login.this,SessionManagement.SESSION_REMEMBERME);
        sessionManagement.createRememberMeSession(userEnteredMobile,userEnteredPassword);
    }

    DatabaseReference reference= FirebaseDatabase.getInstance().getReference( path: "Users");
    Query checkUser=reference.orderByChild("mobile").equalTo(userEnteredMobile);
    checkUser.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

            if (dataSnapshot.exists())
            {
                mobile.setError(null);
                mobile.setErrorEnabled(false);
                String passwordFromDB=dataSnapshot.child(userEnteredMobile).child("password").getValue(String.class);
                if (passwordFromDB.equals(userEnteredPassword))
                {
                    String firstnameFromDB=dataSnapshot.child(userEnteredMobile).child("firstname").getValue(String.class);
                    String lastnameFromDB=dataSnapshot.child(userEnteredMobile).child("lastame").getValue(String.class);
                    String emailFromDB=dataSnapshot.child(userEnteredMobile).child("email").getValue(String.class);
                    String mobileFromDB=dataSnapshot.child(userEnteredMobile).child("mobile").getValue(String.class);

                    //for session management of the user
                    SessionManagement sessionManagement=new SessionManagement( _context: Login.this,SessionManagement.SESSION_USERSESSION);
                    sessionManagement.createLoginSession(firstnameFromDB,lastnameFromDB,emailFromDB,mobileFromDB,passwordFromDB);
```

5.3.12. Farmer Home:

```
@Override
public boolean onNavigationItemSelected(@NonNull MenuItem item)     {

    switch (item.getItemId())
    {
        case R.id.nav_home:
            break;
        case R.id.nav_check_subsydies:
            Intent intent=new Intent( packageContext: MainActivity.this,subsidy.class);
            startActivity(intent);
            break;
        case R.id.nav_market_rate:
            intent=new Intent( packageContext: MainActivity.this,FarmerMarketRate.class);
            startActivity(intent);
            break;
        case R.id.nav_query:
            intent=new Intent( packageContext: MainActivity.this,Queries.class);
            startActivity(intent);
            break;
        case R.id.nav_get_tips:
            intent=new Intent( packageContext: MainActivity.this,GetTips.class);
            startActivity(intent);
            break;
        case R.id.nav_share:
            Toast.makeText( context: this, text: "Share",Toast.LENGTH_SHORT).show();
            break;
        case R.id.nav_rate:
            AlertDialog.Builder builder = new AlertDialog.Builder( context: MainActivity.this);
            View layout= null;
            LayoutInflater inflater = (LayoutInflater)getSystemService(Context.LAYOUT_INFLATER_SERVICE);
```

```
            layout = inflater.inflate(R.layout.rating,  root: null);
            final RatingBar ratingBar = (RatingBar)layout.findViewById(R.id.ratingBar);
            builder.setTitle("Rate Us");
            builder.setMessage("Thank you for rating us , it will help us to provide yo...");
            builder.setPositiveButton("OK", (dialog, id) → {
                    Float value = ratingBar.getRating();
                    Toast.makeText( context: MainActivity.this, text: "Rating is : "+value,Toast.LENGTH_LONG).show();
            });
            builder.setNegativeButton("No,Thanks", (dialog, which) → {
                    dialog.dismiss();
            });
            builder.setCancelable(false);
            builder.setView(layout);
            builder.show();
            break;
        case R.id.nav_logout:
            FirebaseAuth.getInstance().signOut();
            intent=new Intent( packageContext: MainActivity.this,Login.class);
            startActivity(intent);
            finish();
            return true;

    }
    drawerLayout.closeDrawer(GravityCompat.START);
    return true;
```

5.3.13. Chatting Home:

```
class ViewPagerAdapter extends FragmentPagerAdapter
{
    private ArrayList<Fragment> fragments;
    private ArrayList<String> titles;

    ViewPagerAdapter(FragmentManager fm)
    {
        super(fm);
        this.fragments=new ArrayList<>();
        this.titles=new ArrayList<>();
    }

    @NonNull
    @Override
    public Fragment getItem(int position) { return fragments.get(position); }

    @Override
    public int getCount() { return fragments.size(); }
    public void addFragment(Fragment fragment,String title)
    {
        this.fragments.add(fragment);
        this.titles.add(title);
    }

    @Nullable
    @Override
    public CharSequence getPageTitle(int position) { return titles.get(position); }
}
```

5.3.14. Expert List for Chatting:

```
private void chatList()
{
    //Getting all users/previus users
    mUsers=new ArrayList<>();
    reference=FirebaseDatabase.getInstance().getReference( path: "Experts");
    reference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {

            mUsers.clear();
            for (DataSnapshot snapshot1:snapshot.getChildren())
            {
                experts=snapshot1.getValue(AdminAddExpertHelper.class);
                for (ChatList chatLis:expertList)
                {
                    if (experts.getExpertMobile().equals(chatLis.getMnumber()))
                    {
                        mUsers.add(experts);
                    }
                }
            }
            expertAdapter=new ExpertAdapter(getContext(),mUsers);
            recyclerView.setAdapter(expertAdapter);
        }

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

        }
    });
}
```

6. Testing

6.1. Test Strategy:

- In our scenario test strategy is used to test the functionality of our system. We have to use to cover all scenarios. Main focus is on Functional Testing. In Functional Testing test case are used to test the application interface.
- In our system testing is going to be done at individual module level. Each module will be undergone to Unit Testing and expected result is supposed to be same as actual result.

6.2. Test Cases:

6.2.1. Admin Login:

Test Case ID	TC001
Test Case Summary	This test case was conducted for checking admin login.
Related Requirement	RS001
Prerequisite	Admin must be registered in system.
Test Procedure	1. Enter the mobile number and password. 2. Check the mobile number and password is correct or not. 3. Click on login button.
Test Data	Mobile Number: 7096617883 Password: ********
Expected Result	Admin enters into the system.
Actual Result	Admin can view the home page.
Status	Pass
Remarks	This test case is for login into the system.
Created By	Nayan Satani, Modi Saharsh
Date of Creation	05/10/2020
Executed By	Nayan Satani, Modi Saharsh
Date of Execution	05/10/2020
Test Environment	Android Emulator

Table 14: Test Case of Admin Login

6.2.2. Add Expert:

Test Case ID	TC002
Test Case Summary	This test case was conducted for add expert.
Related Requirement	RS002
Prerequisite	Admin must be logged into the system.
Test Procedure	1. Enter First Name, Last Name, Email, Mobile Number, and Password. 2. Click on add button.
Test Data	First Name: Saharsh Last Name: Modi Email: saharshmodi2480@gmail.com Mobile number: 7096617883 Password: ********
Expected Result	Add expert in system.
Actual Result	Expert data can be Added successfully.
Status	Pass
Remarks	This test case for add expert.
Created By	Nayan Satani, Modi Saharsh
Date of Creation	05/10/2020
Executed By	Nayan Satani, Modi Saharsh
Date of Execution	05/10/2020
Test Environment	Android Emulator

Table 15: Test Case of Admin Adding Expert

6.2.3. Delete/Update Expert:

Test Case ID	TC003
Test Case Summary	This test case was conducted for delete or update expert.
Related Requirement	RS003
Prerequisite	Admin must be logged into the system.
Test Procedure	1. Enter First Name, Last Name, Email, Mobile Number, and Password. 2. Click on delete or update button.
Test Data	First Name: Saharsh Last Name: Modi Email: saharshmodi2480@gmail.com Mobile number: 7096617883 Password: ********
Expected Result	Delete or Update expert in system.
Actual Result	Expert data can be deleted or updated successfully.
Status	Pass
Remarks	This test case for delete or update expert.
Created By	Nayan Satani, Modi Saharsh
Date of Creation	05/10/2020
Executed By	Nayan Satani, Modi Saharsh
Date of Execution	05/10/2020
Test Environment	Android Emulator

Table 16: Test Case of Admin Deleting or Updating Expert

6.2.4. Add Subsidy:

Test Case ID	TC004
Test Case Summary	This test case was conducted for add subsidy.
Related Requirement	RS004
Prerequisite	Admin must be registered in system.
Test Procedure	1. Enter Subsidy Name, Subsidy Detail. 2. Click on add button.
Test Data	Subsidy Name: Solar System Subsidy Detail: Solar System Subsidy benefits
Expected Result	Add Subsidy in system.
Actual Result	Subsidy can be added successfully.
Status	Pass
Remarks	This test case for add Subsidy.
Created By	Nayan Satani, Modi Saharsh
Date of Creation	05/10/2020
Executed By	Nayan Satani, Modi Saharsh
Date of Execution	05/10/2020
Test Environment	Android Emulator

Table 17: Test Case of Admin Adding Subsidy

6.2.5. Delete/Update Subsidy:

Test Case ID	TC005
Test Case Summary	This test case was conducted for delete or update subsidy.
Related Requirement	RS005
Prerequisite	Admin must be registered in system.
Test Procedure	1. Enter Subsidy Name, Subsidy Detail. 2. Click on delete or update button.
Test Data	Subsidy Name: Solar System Subsidy Detail: Solar System Subsidy benefits
Expected Result	Delete or Update Subsidy in system.
Actual Result	Subsidy can be deleted or updated successfully.
Status	Pass
Remarks	This test case for delete or update Subsidy.
Created By	Nayan Satani, Modi Saharsh
Date of Creation	05/10/2020
Executed By	Nayan Satani, Modi Saharsh
Date of Execution	05/10/2020
Test Environment	Android Emulator

Table 18: Test Case of Admin Deleting or Updating Subsidy

6.2.6. Add Commodity:

Test Case ID	TC006
Test Case Summary	This test case was conducted for add commodity.
Related Requirement	RS006
Prerequisite	Admin must be registered in system.
Test Procedure	1. Enter Commodity Name. 2. Click on add Commodity button.
Test Data	Commodity Name: Chana
Expected Result	Add Commodity in system.
Actual Result	Commodity can be added successfully.
Status	Pass
Remarks	This test case for add Commodity.
Created By	Nayan Satani, Modi Saharsh
Date of Creation	05/10/2020
Executed By	Nayan Satani, Modi Saharsh
Date of Execution	05/10/2020
Test Environment	Android Emulator

Table 19: Test Case of Admin Adding Commodity

6.2.7. Add Commodity Market Rate:

Test Case ID	TC007
Test Case Summary	This test case was conducted for add commodity market rate.
Related Requirement	RS007
Prerequisite	Admin must be registered in system.
Test Procedure	1. Enter Commodity Market Rate. 2. Click on add price button.
Test Data	Ahmedabad: 375 Surat: 425 Vadodara: 475
Expected Result	Add Commodity Rate in system.
Actual Result	Commodity Market Rate can be added successfully.
Status	Pass
Remarks	This test case for add Commodity Market Rate.
Created By	Nayan Satani, Modi Saharsh
Date of Creation	05/10/2020
Executed By	Nayan Satani, Modi Saharsh
Date of Execution	05/10/2020
Test Environment	Android Emulator

Table 20: Test Case of Admin Adding Commodity Market Rate

6.2.8. Expert Login:

Test Case ID	TC008
Test Case Summary	This test case was conducted for checking expert login.
Related Requirement	RS008
Prerequisite	Expert must be registered in system.
Test Procedure	1. Enter the mobile number and password. 2. Check the mobile number and password is correct or not. 3. Click on login button.
Test Data	Mobile number: 7096617883 Password: ********
Expected Result	Expert enters into the system.
Actual Result	Expert can view the home page.
Status	Pass
Remarks	This test case for expert login into the system.
Created By	Nayan Satani, Modi Saharsh
Date of Creation	05/10/2020
Executed By	Nayan Satani, Modi Saharsh
Date of Execution	05/10/2020
Test Environment	Android Emulator

Table 21: Test Case of Expert Login

6.2.9. Farmer Registration:

Test Case ID	TC009
Test Case Summary	This test case was conducted for checking farmer registration.
Related Requirement	RS009
Prerequisite	NA
Test Procedure	1. Enter First Name, Last Name, Email, Mobile Number, and Password. 2. Check the Validation. 3. Click on Sign-up button.
Test Data	First Name: Saharsh Last Name: Modi Email: saharshmodi2480@gmail.com Mobile number: 7096617883 Password: ********
Expected Result	Farmer registers into the system.
Actual Result	Farmer registers into the system.
Status	Pass
Remarks	This test case for farmer registers into the system.
Created By	Nayan Satani, Modi Saharsh
Date of Creation	05/10/2020
Executed By	Nayan Satani, Modi Saharsh
Date of Execution	05/10/2020
Test Environment	Android Emulator

Table 22: Test Case of Farmer Registration

6.2.10. Farmer Login:

Test Case ID	TC010
Test Case Summary	This test case was conducted for checking farmer login.
Related Requirement	RS010
Prerequisite	Farmer must be registered in system.
Test Procedure	1. Enter the mobile number and password. 2. Check the mobile number and password is correct or not. 3. Click on login button.
Test Data	Mobile number: 7096617883 Password: ********
Expected Result	Farmer enters into the system.
Actual Result	Farmer can view the home page.
Status	Pass
Remarks	This test case for farmer login into the system.
Created By	Nayan Satani, Modi Saharsh
Date of Creation	05/10/2020
Executed By	Nayan Satani, Modi Saharsh
Date of Execution	05/10/2020
Test Environment	Android Emulator

Table 23: Test Case of Farmer Login

6.2.11. View Subsidies:

Test Case ID	TC011
Test Case Summary	This test case was conducted for Viewing new subsidies.
Related Requirement	RS011
Prerequisite	Farmer must be logged into the system.
Test Procedure	1. Click on check subsidies button.
Test Data	NA
Expected Result	Farmer can view latest subsidies.
Actual Result	Farmer can view latest subsidies.
Status	Pass
Remarks	This test case for Viewing subsidies
Created By	Nayan Satani, Modi Saharsh
Date of Creation	05/10/2020
Executed By	Nayan Satani, Modi Saharsh
Date of Execution	05/10/2020
Test Environment	Android Emulator

Table 24: Test Case of Farmer Viewing Subsidy

6.2.12. Post Queries:

Test Case ID	TC012
Test Case Summary	This test case was conducted for posting queries.
Related Requirement	RS012
Prerequisite	Farmer must be logged into the system.
Test Procedure	1. Enter a query. 2. Click on Post Query button.
Test Data	Query: Give a suggestion for crop protection.
Expected Result	Post a query for suggestion.
Actual Result	Query posted in system.
Status	Pass
Remarks	This test case for posting queries.
Created By	Nayan Satani, Modi Saharsh
Date of Creation	05/10/2020
Executed By	Nayan Satani, Modi Saharsh
Date of Execution	05/10/2020
Test Environment	Android Emulator

Table 25: Test Case of Farmer Posting Queries

6.2.13. View Market Rate:

Test Case ID	TC013
Test Case Summary	This test case was conducted for Viewing a Market Rate.
Related Requirement	RS013
Prerequisite	Farmer must be logged into the system.
Test Procedure	1. Select commodity. 2. Click on check button.
Test Data	Commodity: Chana
Expected Result	Showing a result.
Actual Result	Show the market rate of selected commodity.
Status	Pass
Remarks	This test case for view market rate.
Created By	Nayan Satani, Modi Saharsh
Date of Creation	05/10/2020
Executed By	Nayan Satani, Modi Saharsh
Date of Execution	05/10/2020
Test Environment	Android Emulator

Table 26: Test Case of Farmer Viewing Selected Commodity Market Rate

6.2.14. Forgot Password:

Test Case ID	TC014
Test Case Summary	This test case was conducted for forgot password.
Related Requirement	RS014
Prerequisite	Farmer must be registered into the system.
Test Procedure	1. Navigate to a particular page. 2. Farmer input a valid mobile number. 3. Click on Forgot Password Button.
Test Data	Mobile number: 7096617883
Expected Result	Display message.
Actual Result	Display message and get OTP on registered Mobile Number.
Status	Pass
Remarks	This test case for forgot password.
Created By	Nayan Satani, Modi Saharsh
Date of Creation	05/10/2020
Executed By	Nayan Satani, Modi Saharsh
Date of Execution	05/10/2020
Test Environment	POCO F1 Phone (Real Device)

Table 27: Test Case of Forgot Password

7. Future Enhancement

- In future we will improve Market Rate functionality for this we will use live market rate tracking that will directly connected with government markets.
- We will improve the Questions/Answers Section and make it more reliable.
- We will add notification alert for new updates on Subsidy, Tips and Daily Market Rate.
- In Future we will add solution based on photo of any crops that will improve productivity.

8. Bibliography

- https://www.taimoorsikander.com/category/android/android-material-design-tutorials/
- https://www.simplifiedcoding.net/firebase-realtime-database-crud/
- https://www.udemy.com/course/complete-android-n-developer-course/
- https://www.udemy.com/course/the-complete-android-10-developer-course-mastering-android/
- https://firebase.google.com/docs/android/setup
- https://console.firebase.google.com/u/0/project/farmerapp-9f62f/