

Project Report

*on*

AgriBook

*by*

Naman Sharma (AU19B1002)

*Under the guidance of*

Atul Choudhary and Satyajit Pangaonkar

Design Project: Mobile Application (CS4201)



School of Computer Engineering

Avantika University, Ujjain

**2020-2021**

## **Abstract**

Nowadays Farmers are facing problems for selling their crops like long queues have been observed outside the warehouses which leads to the loss of time, energy, other resources and also no proper system for registering their crops to sell. Due to this there are losses to farmers to their crops and they have to long wait for their turn to sell.

The problem which has been faced by the farmers in the current scenario is that the procedure is very manual like user has to go to the warehouse for getting the form and then filling it manually and then resubmit it to the warehouse which takes too much time of the farmer and also every time he has to visit to the warehouse for doing it.

AgriBook is the solution to all major problems regarding to their crop selling needs as we are providing user the proper solution to select schedule the preferred warehouse at the preferred time. And also, he can check the live market rates of the crops to plan his selling accordingly.

## Introduction

---

### 1.1 Motivation

The motivation for doing this project was primarily an interest in undertaking a challenging project in an interesting area of research. And also, the domain we had worked has still some problems which are still unsolved. So, coming up with the solution for it is somewhat self-motivating. And the opportunity to learn about a new area of mobile-app development in the environment of Android Studio and Java and also integrate it with a real-world problem and also giving it a digital solution to it.

### 1.2 Problem Identification

Nowadays Farmers are also a part of the changing evolution in this digital era, so we are coming up with an app which provides them registering their crops on their preferred warehouses and to sell them on the live market rates.

### 1.3 Aim

Our aim was to create a functional, user-friendly and aesthetically good application for most of the solutions farmers nowadays facing related to their crop selling.

### 1.4 Objectives

Objective of making this whole domain is to solve the problem of farmers regarding their crop stock selling in a very digitalized way and also giving them portability to get their crops registered on the warehouses and also to book the date at which they will deliver it.

### Ideation

---

For the ideation of this concept we have followed a particular method in which starting it from brainstorming the concept than doing both the researches i.e. primary research and secondary research after gathering all the inputs finalizing the functionalities.

### Need of this Project

Solution to this problem is necessary because looking at the current solution so it results a lot of physical efforts like first going to office for registration than waiting in the long queues for their chance to sell which makes ending up with a loss of the time and money. And here we are making this process quite more digital where farmers are registering their crops than selecting their preferred warehouse and also able to see the rates at which the crop will be sold. So this whole thing makes the need of this solution

### Scope of the project

The scope of this project has been set to some limited extent and will lead to the following functionalities that are given below -

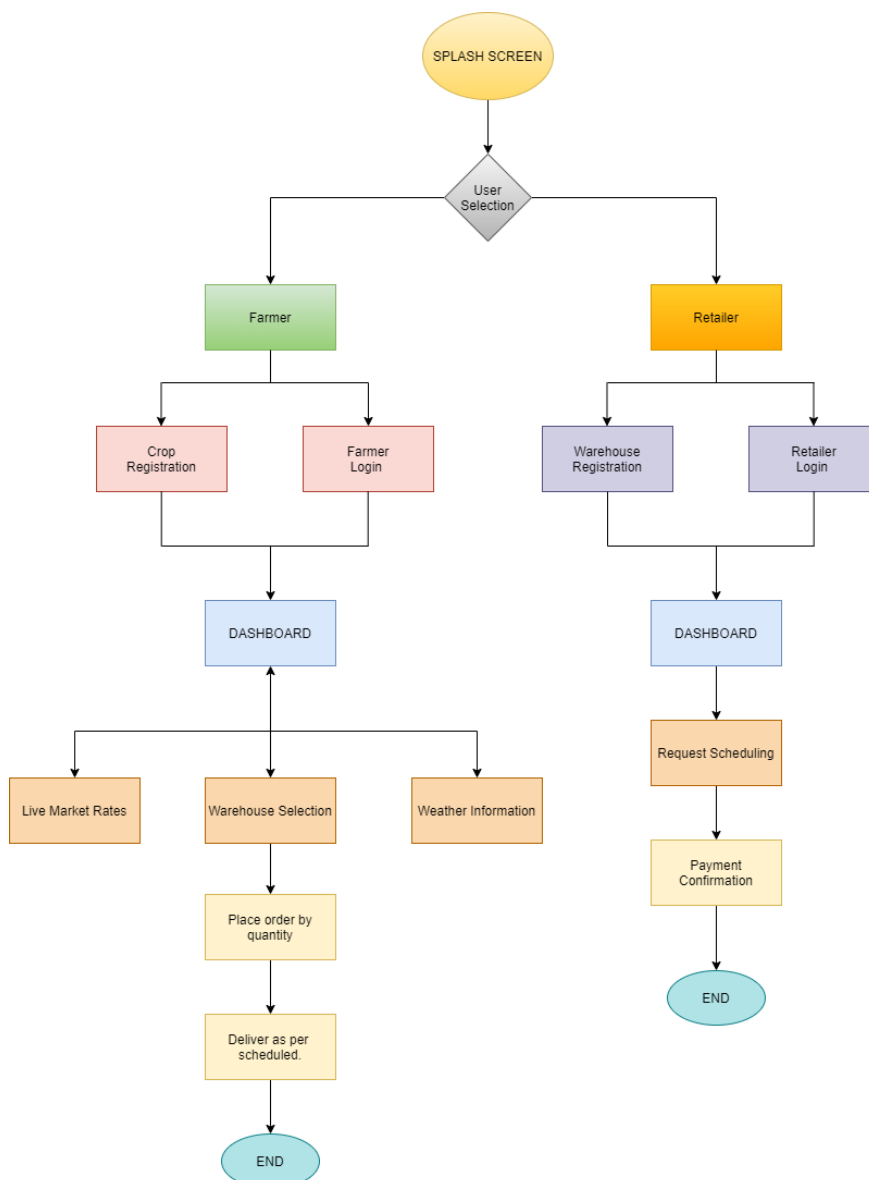
1. Registration of crops.
2. Listing of all the nearby Warehouses.
3. Current rates of crops at which it can be sold.
4. Booking of the crop by quantity.
5. Payment processing after delivery.
6. Live weather information.

### Design and Methodology

---

Design phase is the early phase of this project where we are designing the project's key features, structure in the form of flowchart diagram and wireframe for the same also we are going to design the constraints for it.

Flowchart for the ideated app flow is given below.



Also, we have created high-fidelity wireframe design as a replica to better understand this app.



COMPLETE PROFILE

Farmer Registration

Full Name

Email ID

Mobile No.

Password

State City

Pin Code

SUBMIT

AgriBook

FARMER LOGIN

Enter Your Mobile No.

Submit

SIGN UP / REGISTER



Mr. Mayank Sharma 7772010704

NCML Warehouse  
Vikram Udhogpuri Road, Ujjain

Select Crop

Crop Quantity per Quintal

Aadhar No.

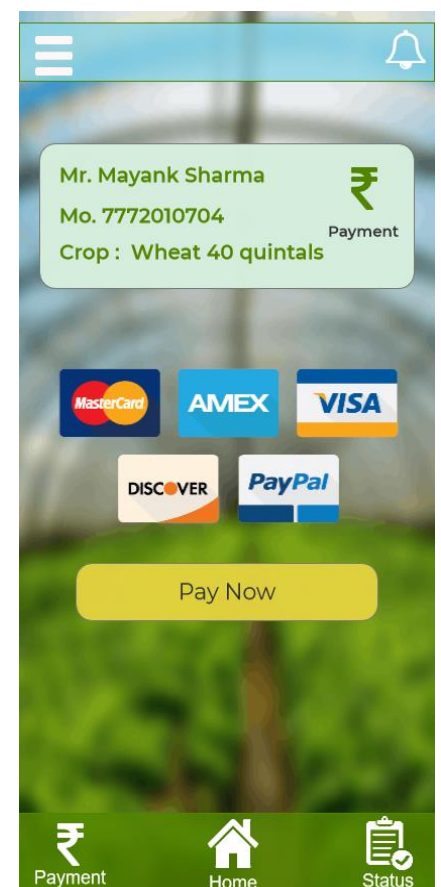
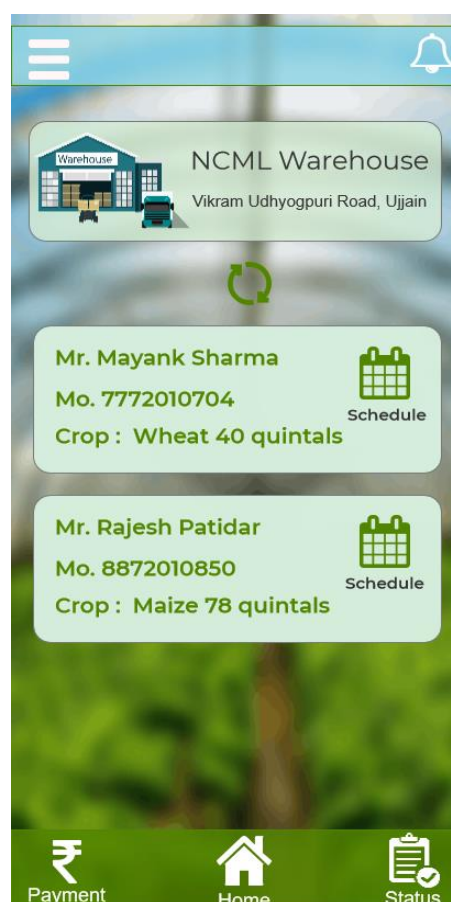
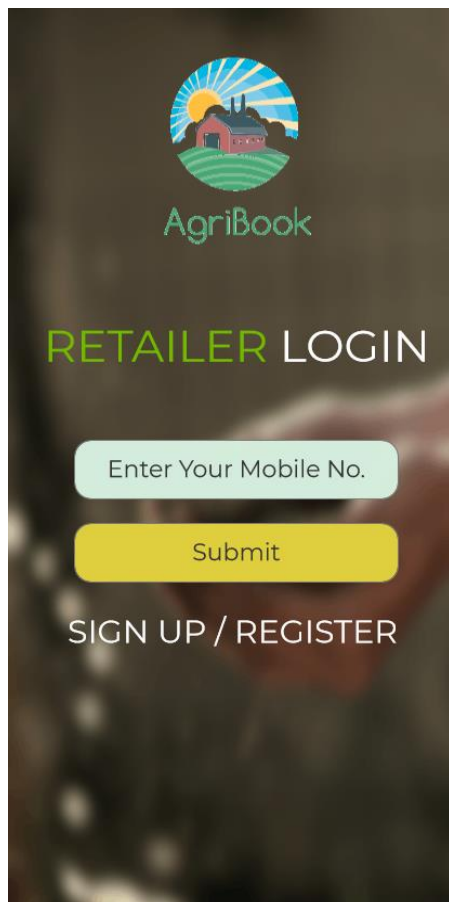
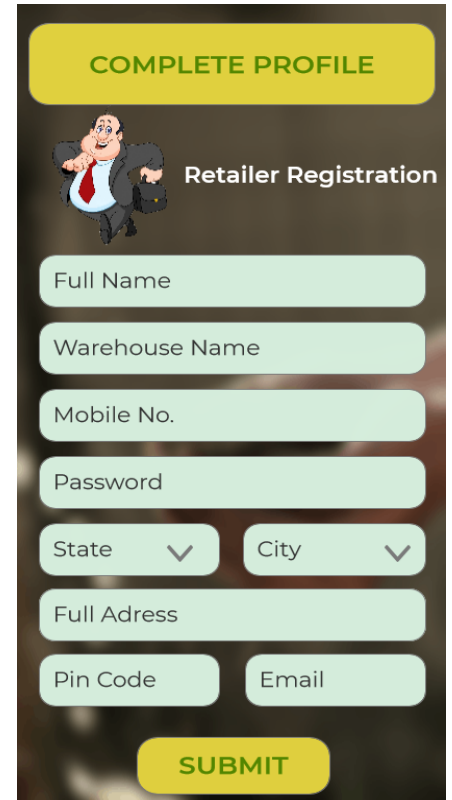
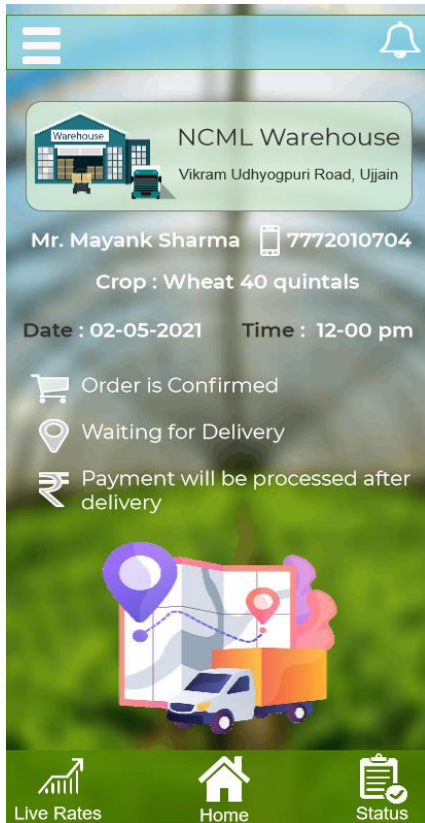
Village Name

I confirm that above details are correct

Submit

Live Rates Home Status





## **Design Constraints**

Design constraints are limitations on a design. These include imposed limitations that you don't control and limitations that are self-imposed as a way to improve a design.

According to it I had identified the following constraints in this app which are given above:

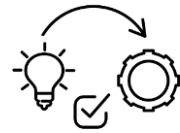
1. While registration of farmers and retailers inputs fields has been certain validations in it.
2. Crops registration needs the farmer personal information like Aadhar number in which KYC can't be verified.
3. Scheduling the date of delivering for the farmers will be done.
4. Payment will be done on the basis of amount which will be displayed on the live rates.
5. Just in case if farmer does not deliver the crop after registration on scheduled date and time the request will be deleted.



### Implementation

---

The implementation phase involves putting the design concept into action to meet the objectives of the defined problem. Here we actually do the project work to produce the deliverables. The term “deliverable” means anything our project delivers. The deliverables for our project include all of the products or services that we are working to learn, practice and achieving it.



### Functionalities

Here are all the major functionality requirements that were initiated before.

Major Functionalities	Status
Login and Registration of Farmer & Retailer.	Completed ✓
Warehouse List on farmer Dashboard	Completed ✓
Crop Registration and date scheduling	Completed ✓
Weather information through API	Completed ✓
Live Market rates through websites	Completed ✓
Farmers Booking on retailer dashboard	Completed ✓
Payment Gateway Integration	Completed ✓

# 1. Splash Screen

Splash screen is the introductory UI page where all the data been loaded in the backend.

In the code at onResponse we put a condition for checking for pre-login and session wait of 3.5sec.

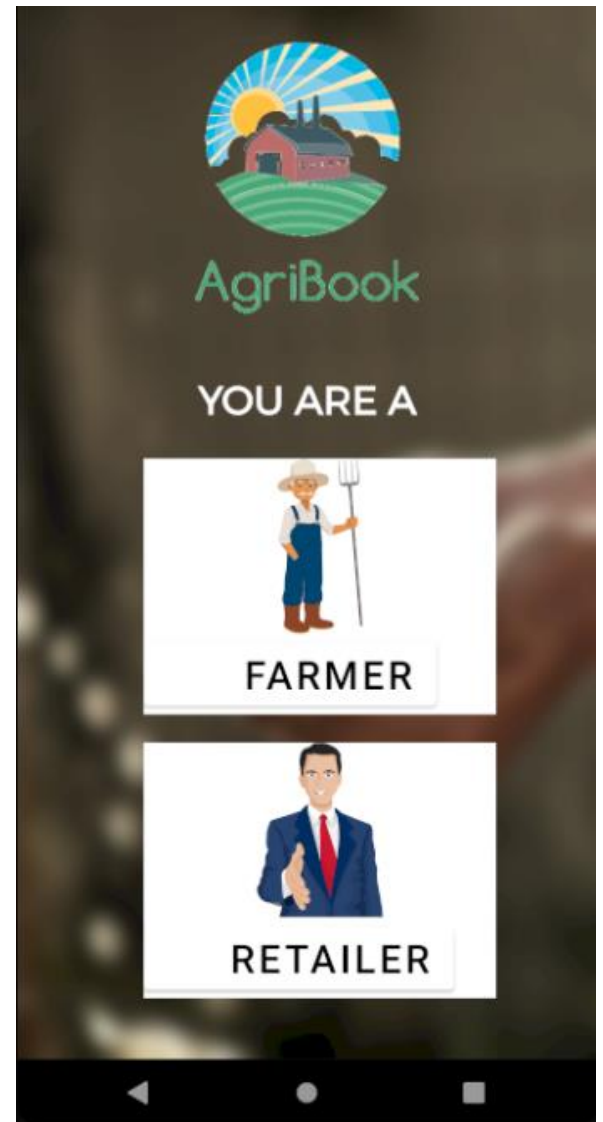


```
@Override
protected void onResume() {
    super.onResume();
    new Handler().postDelayed(new Runnable() {
        @Override
        public void run() {
            Intent intent;
            if(!email.equals("") && !password.equals(""))
            {
                intent = new Intent(getApplicationContext(), DashBoard.class);
            }else {
                intent = new Intent(getApplicationContext(), LoginAct.class);
            }
            startActivity(intent);
            finish();
        }
    }, delayMillis: 3500);
}
```

## 2. User Selection

User Selection is the page where the end User needs to select that whether it should be accessed as farmer or retailer.

In the code we have create two buttons and set button on click listener on it.

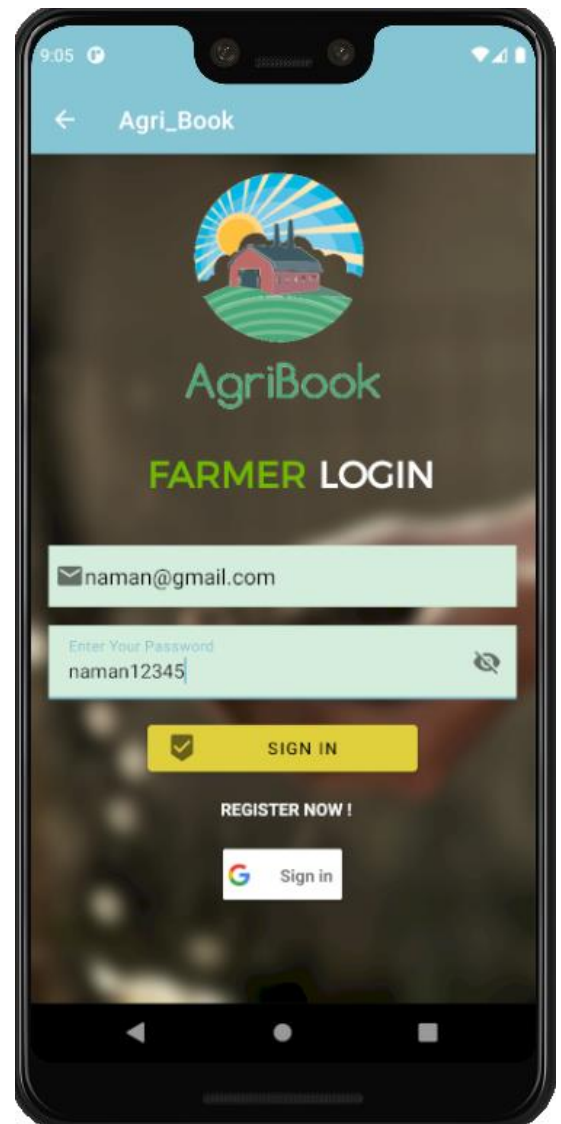


```
@Override
protected void onResume() {
    super.onResume();
    farmerBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent( packageContext: UserSel.this, LoginAct.class);
            startActivity(intent);
        }
    });
    retailerBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent( packageContext: UserSel.this, Retailer_LoginAct.class);
            startActivity(intent);
            finish();
        }
    });
}
```

### 3. Farmer Login

Farmer Login is the login page where user will be put his credentials and will get access by registering or through Google sign in.

Firstly, we had created Database for storing the data and accessing it through PHP connection than we had used volleySingleton and put validation over the code.

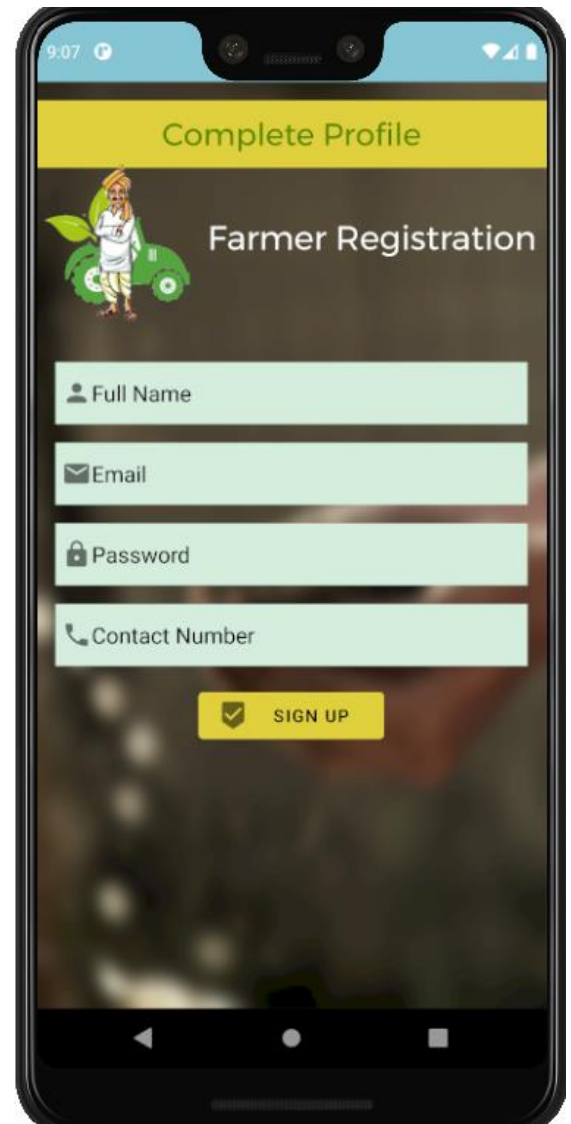


```
private void loginUser() {
    User user = new DAOClass().setData();
    stringRequest = new StringRequest(Request.Method.POST, Constants.LOGIN_URL,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                Log.i( tag: "INFO",response);
                if(user!=null)
                {
                    if(response.equals("success")){
                        Toast.makeText(getApplicationContext(), text: "Login Successful",Toast.LENGTH_SHORT).show();
                        SharedPreferences.Editor editor = preferences.edit();
                        editor.putString("email", user.getEmail());
                        editor.putString("password", user.getPasswd());
                        editor.apply();
                        editor.commit();
                        startActivity(new Intent(getApplicationContext(), DashBoard.class));
                        finish();
                    }else {
                        Toast.makeText(getApplicationContext(),response,Toast.LENGTH_SHORT).show();
                    }
                }
            }
        })
}
```

## 4. Farmer Registration

Farmer Registration is the registration page for farmers where user will be entering his details and will get their account ready.

Here also, we had created Database for storing the data and accessing it through PHP connection than we had used volleySingleton and put validation over the code while taking the input from the user.



```
private void registerUser()
{
    User user = new DAOClass().setData();
    StringRequest = new StringRequest(Request.Method.POST, Constants.REG_URL,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                Log.i("tag: \"INFO\", response);
                if(user!=null)
                {
                    if(response.equals("success")){
                        Toast.makeText(getApplicationContext(), "User Registered Successfully", Toast.LENGTH_SHORT).show();
                        SharedPreferences.Editor editor = preferences.edit();
                        editor.putString("username", user.getUserName());
                        editor.apply();
                        Intent intent = new Intent(getApplicationContext(), LoginAct.class);
                        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_CLEAR_TASK);
                        startActivity(intent);
                        finish();
                    }else{
                        Toast.makeText(getApplicationContext(), response, Toast.LENGTH_SHORT).show();
                    }
                }
            }
        }, new Response.ErrorListener() {
```

## 5. Farmer Dashboard

Farmer Dashboard is the page where user gets access to all the features like card view and also temperature information through weather API.

And through Selecting card user can register their crops. Also navigate through different pages by bottom navigation bar.

```
private void loadData() {
    StringRequest stringRequest = new StringRequest(Request.Method.GET,
        Constants.WAR_DATA_URL, new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            Log.i( tag: "INFO", response);
            try {
                JSONObject jsonObject = new JSONObject(response);
                JSONArray jsonArray = jsonObject.getJSONArray( name: "data");
                for(int i=0;i<jsonArray.length();i++ )
                {
                    JSONObject jOBJ = jsonArray.getJSONObject(i);
                    Warehouse mWarehouse = new Warehouse(jOBJ.getString( name: "warehouse_name"),
                        jOBJ.getString( name: "raddress"));
                    warehouse.add(mWarehouse);
                }
                adapter = new Warehouse_Adapter( context: DashBoard.this, warehouse);
                recyclerView.setAdapter(adapter);
                adapter.setOnItemClickListener(DashBoard.this);
            }catch (JSONException ex){
                Log.e( tag: "JSON",ex.getMessage());
            }
        }
    });
}
```



```
public void get(View v){
    String city = et.getText().toString();
    String url = "https://api.openweathermap.org/data/2.5/weather?q="+city+"&appid=6d6143a489ce617afd8c3def27b4d08f";
    RequestQueue queue = Volley.newRequestQueue(getApplicationContext());
    JsonObjectRequest request = new JsonObjectRequest(Request.Method.GET, url, jsonRequest: null, new Response.Listener<JS

    @Override
    public void onResponse(JSONObject response) {

        try {
            JSONObject array= (JSONObject) response.get("main");
            String temprature= array.getString( name: "temp");
            Double temp = Double.parseDouble(temprature)-273.15;
            tv.setText(temp.toString().substring(0,4)+" °C");
            //Toast.makeText(getApplicationContext(),s,Toast.LENGTH_LONG).show();
        } catch (JSONException e) {
            tv.setText(e.getMessage());
            e.printStackTrace();
        }
    }
}
```



## 6. Live Rates

Live Rates is the page where we get market rates of all the crop at which it can be sell.

Here we are fetching JSON data through website and display it on app by recycle view.



```
@Override
public void onResponse(String s) {
    try {
        JSONObject jsonObject = new JSONObject(s);
        JSONArray array = jsonObject.getJSONArray( "live");

        for(int i = 0; i< array.length(); i++){
            JSONObject o = array.getJSONObject(i);
            Rate item = new Rate(
                o.getString( "cropName"),
                o.getString( "cropRate")
            );
            listItems.add(item);
        }

        adapter = new RateAdapter(listItems, getApplicationContext());
        recyclerView.setAdapter(adapter);
    }
    catch (JSONException e)
    {
        e.printStackTrace();
    }
}
```



## 7. User Profile

User Profile is the page where it will be displayed like which user has logged in through which email and also the button to logout.

All the data that are been displayed has been fetched through Shared Preferences and displayed on a layout.

And in Logout ending the session through clearing the editor.



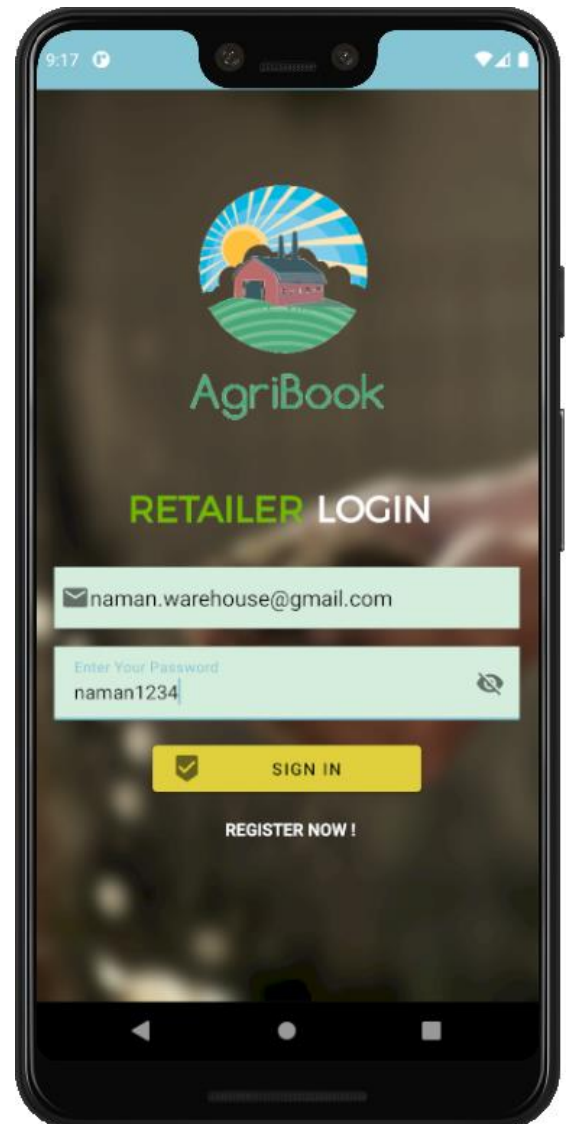
```
@Override
public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {

    switch (menuItem.getItemId()) {
        case R.id.dashboard:
            startActivity(new Intent(getApplicationContext(),
                LiveRates.class));
            overridePendingTransition(enterAnim: 0, exitAnim: 0);
            return true;
        case R.id.homoe:
            startActivity(new Intent(getApplicationContext(),
                DashBoard.class));
            overridePendingTransition(enterAnim: 0, exitAnim: 0);
            return true;
        case R.id.about:
            overridePendingTransition(enterAnim: 0, exitAnim: 0);
            return true;
    }
    return false;
}
```

## 8. Retail Login

Retailer Login is the login page where user will be put his credentials and will get access by registering.

Here also, we had created Database for storing the data and accessing it through PHP connection than we had used volleySingleton and put validation over the code.

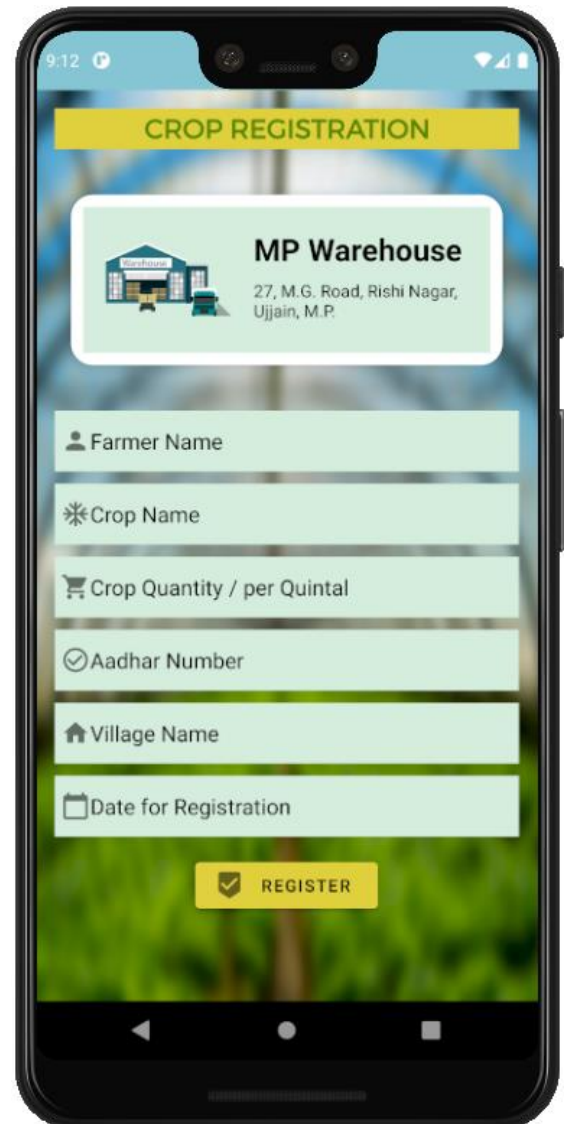


```
@SuppressWarnings("CommitPrefEdits")
private void retloginUser() {
    Retailer retailer = new DAOClass().setData();
    StringRequest = new StringRequest(Request.Method.POST, Constants.RET_LOGIN_URL,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                Log.i("tag: INFO", response);
                if (retailer != null) {
                    if (response.equals("success")) {
                        Toast.makeText(getApplicationContext(), "Login Successful", Toast.LENGTH_SHORT).show();
                        SharedPreferences.Editor editor = preferences.edit();
                        editor.putString("remail", retailer.getRemail());
                        editor.putString("rpassword", retailer.getRpassword());
                        editor.apply();
                        editor.commit();
                        startActivity(new Intent(getApplicationContext(), Retailer_DashBoard.class));
                        finish();
                    } else {
                        Toast.makeText(getApplicationContext(), response, Toast.LENGTH_SHORT).show();
                    }
                }
            }
        })
}
```

## 9. Crop Booking

Crop Booking is the very main feature of our application where user is registering his crop to sell at his preferred warehouse. By filling the general registration form.

Data is been stored at backend at displayed over retailer dashboard.

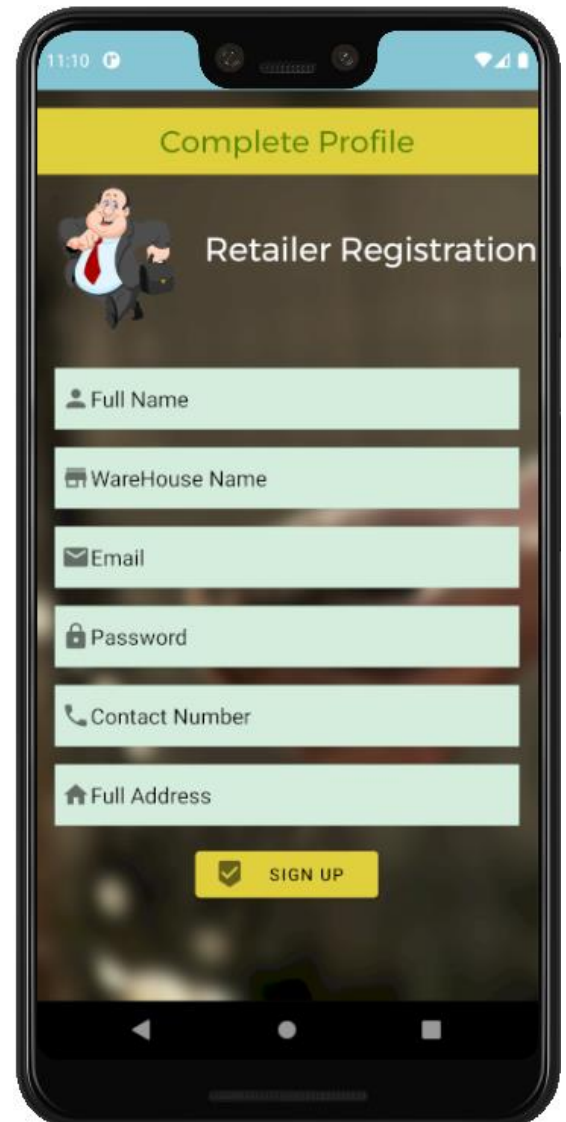


```
if(TextUtils.isEmpty(rname))
    rnameEdit.setError("Please Enter the name");
else if(TextUtils.isEmpty(warehouse_name))
    warehousenameEdit.setError("Please Enter Warehouse Name");
else if(TextUtils.isEmpty(remail))
    remailEdit.setError("Please Enter Email correctly");
else if (TextUtils.isEmpty(rpassword))
    rpasswordEdit.setError("Please Enter the password");
else if (rpassword.length()<8)
    rpasswordEdit.setError("Password must be 8 characters");
else if(TextUtils.isEmpty(rcontact))
    rcontactEdit.setError("Please Enter the contact");
else if (rcontact.length()!=10)
    rcontactEdit.setError("Contact Number must be 10 characters");
else if(TextUtils.isEmpty(raddress))
    raddressEdit.setError("Please Enter the address");
else{
    return new Retailer(rname,warehouse_name,remail,rpassword,rcontact,raddress);
}
```

## 10. Retail Registration

Retailer Registration is the registration page for retailer where user will be entering his details and will get their account ready.

Here also, we had created Database for storing the data and accessing it through PHP connection than we had used volleySingleton and put validation over the code while taking the input from the user.



```
if(TextUtils.isEmpty(rname))
    rnameEdit.setError("Please Enter the name");
else if(TextUtils.isEmpty(warehouse_name))
    warehousenameEdit.setError("Please Enter Warehouse Name");
else if(TextUtils.isEmpty(remail))
    remailEdit.setError("Please Enter Email correctly");
else if (TextUtils.isEmpty(rpassword))
    rpasswordEdit.setError("Please Enter the password");
else if (rpassword.length()<8)
    rpasswordEdit.setError("Password must be 8 characters");
else if(TextUtils.isEmpty(rcontact))
    rcontactEdit.setError("Please Enter the contact");
else if (rcontact.length()!=10)
    rcontactEdit.setError("Contact Number must be 10 characters");
else if(TextUtils.isEmpty(raddress))
    raddressEdit.setError("Please Enter the address");
else{
    return new Retailer(rname,warehouse_name,remail,rpassword,rcontact,raddress);
}
```

## 11. Retail Dashboard

Retailer Dashboard is the page where all the booking done by the farmer will get displayed and the retailer can pay for the crop after the successful delivery by the farmer and on the booked date.

Also, the user which has been logged in will be displayed and also the user can logout from this page.



```
@Override
public void onItemClick(int position) {
    Bundle bundle = new Bundle();
    Customer mCustomer = customer.get(position);
    Intent intent = new Intent( packageContext: Retailer_DashBoard.this, UserSel.class);
    bundle.putString("farmer_name",mCustomer.getFarmer_name());
    bundle.putString("village_name",mCustomer.getVillage_name());
    bundle.putString("crop_name",mCustomer.getCrop_name());
    bundle.putString("crop_quantity",mCustomer.getFarmer_name());
    intent.putExtras(bundle);
    startActivity(intent);
}
```

### Testing and Deployment

---

Testing Phase is an integral part of the mobile application development lifecycle. To ensure the successful development of any app, testing phase must be involved in all stages of development, from creating the concept to analyzing

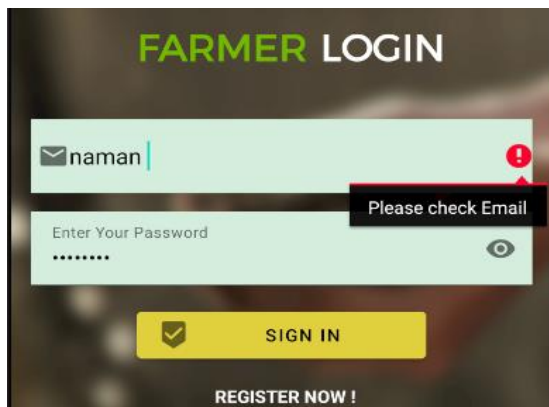
#### Functionality Test

In the Functionality test we are going to test all the major functionalities implemented in this app and generated a table of testing based on the outcome results.

Major Functionalities	Status	Session Handlings	Exception Handling
Login and Registration of Farmer & Retailer.	Working	Session Handling is done	All exceptions handled
Warehouse List on farmer Dashboard	Working	Session Handling is done	Not Required
Crop Registration and date scheduling	Working	Session Handling is done	All exceptions handled
Weather information through API	Working	Not Required	Present
Live Market rates through websites	Working	Not Required	Not Required
Farmers Booking on retailer dashboard	Working	Session Handling is done	All exceptions handled
Payment Gateway Integration	Working	Session Handling is done	All exceptions handled

All the above functionalities and also Black-Box testing for inputs given have been tested successfully and can be verified through Snapshots given below.



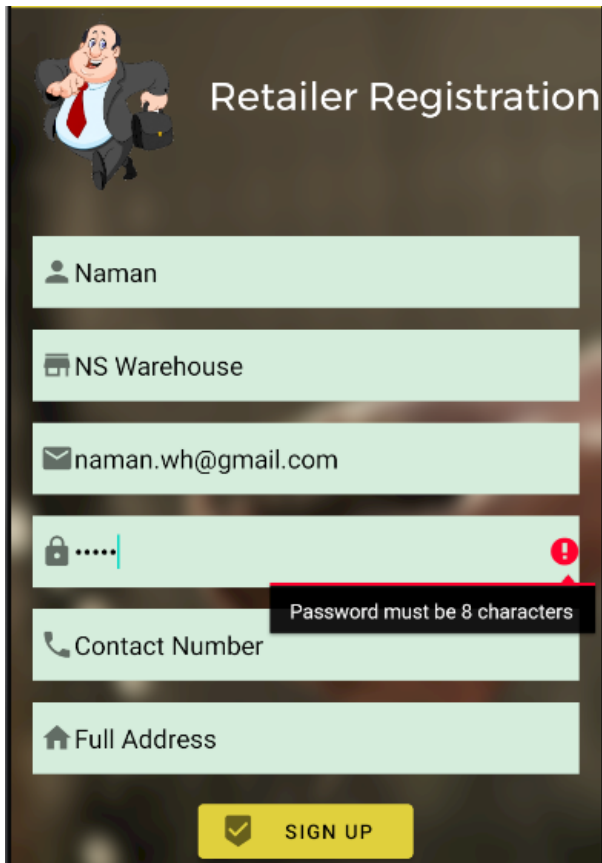


Here in giving wrong input to the email it puts a warning like its not in correct order.





Giving wrong input to weather API. Through exception handling it will throw a warning toast to the user for checking the city name. And also the session has been handled on this page.









**Retailer Registration**


 Naman


 NS Warehouse

 naman.wh@gmail.com

 ..... 

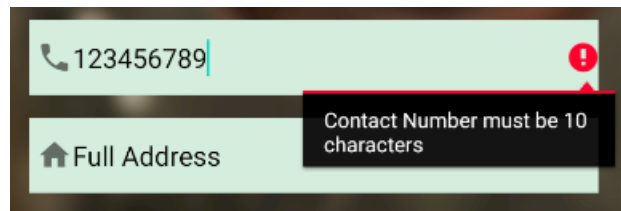
 Contact Number



 Full Address


 **SIGN UP**

**Validation Message:** Password must be 8 characters

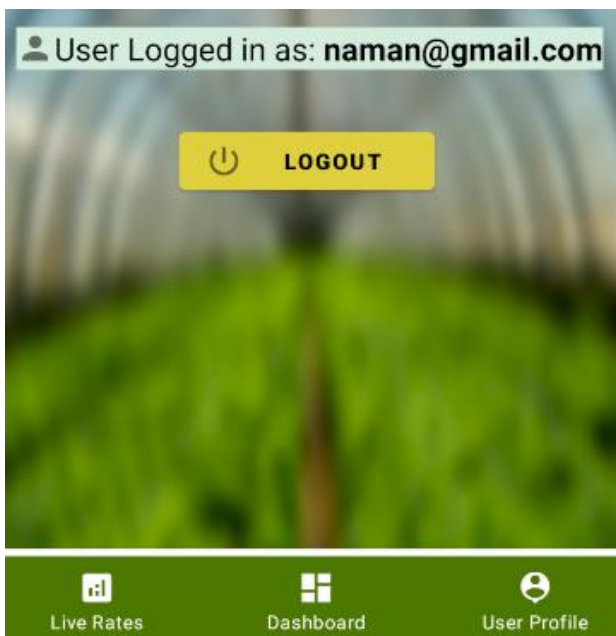
In registering the retailer all the fields have been validated with required validations. Like some fields are pre -defined size as we can see in the password and contact number and many more.





 123456789 


 Full Address


**Validation Message:** Contact Number must be 10 characters




 User Logged in as: naman@gmail.com

 **LOGOUT**

 Live Rates

 Dashboard

 User Profile

In the user profile section, the session is getting handled and can be observed that which account is been logged in into.

CROP REGISTRATION



MP Warehouse

27, M.G. Road, Rishi Nagar,  
Ujjain, M.P.

1234

!

\* Crop Name

Enter Alphabetical Characters Only

🛒 Crop Quantity / per Quintal

✓ Aadhar Number

🏠 Village Name

📅 Date for Registration

✓ REGISTER

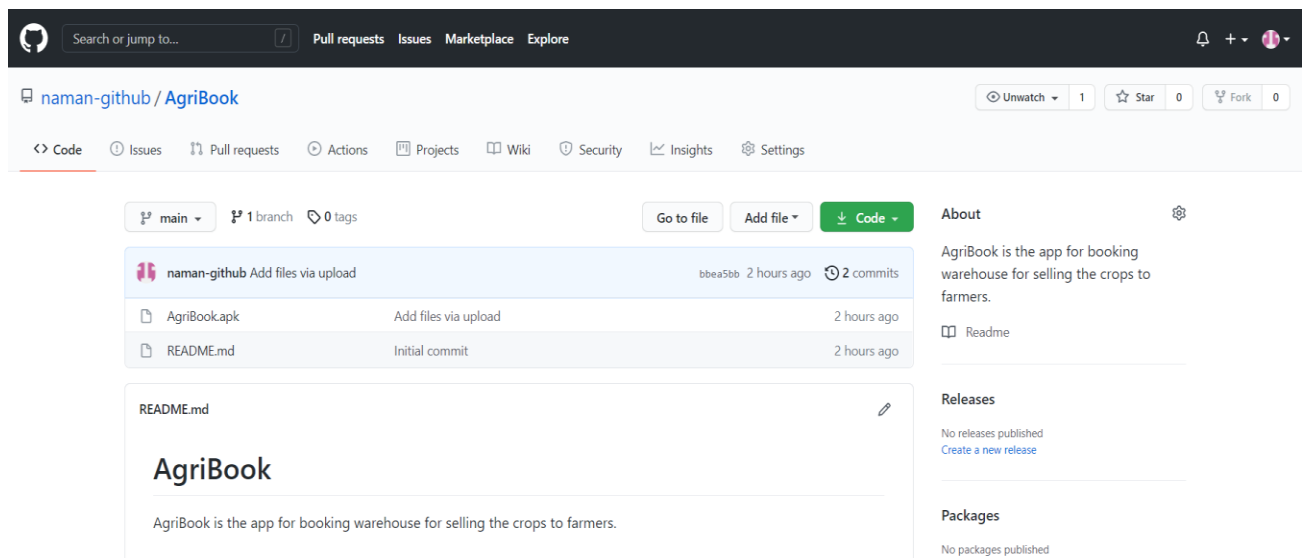
All the required field with all the possible validation has been implemented and all the exceptions getting handled in this activity.

# Deployment

---

The deployment phase is the final phase of the software development life cycle and puts the product into production. After the product passes each testing phase, the product is ready to go live. This means that the product is ready to be used in a real environment by all end users of the product

The environment which we have chosen is GitHub for deploying our project. Link for the application and snapshot of the same is given below <https://github.com/naman-github/AgriBook>



# Results & Conclusion

---

## Results

While creating this whole project many challenges have been faced by me as we had faced some constraints like debugging the error has taken much of the time and also, we are in our learning phase so learning the syntax and implement it has consumed lot of time.

Even though as an achievement we had got many new learnings about Java for functionalities and XML for creating layout design, etc. And all of this learning ends up with our dynamic application which is of good aesthetics, better interface and quality transactions and also, we had completed this application to end to end transactions.

## Conclusion

Initially, we had started to create the application which is a dynamic, functional and user-friendly for most of the crop selling solutions farmers nowadays facing. We wanted to give users a one stop solution kind of system to their problems regarding this problem.

At this point we have achieved more than 85% of what we had planned at the initial level of time and also there are some things which we had not planned but still we had implemented.

Our future plans are taking this website to the upmost level like implementing the chat system, better transitioning and more features to retailers.

### References

---

1. <https://developer.android.com/>
2. <https://www.tutorialspoint.com/android/index.htm>
3. [000webhost.com](http://000webhost.com)
4. <https://www.javatpoint.com/android-tutorial>
5. <https://www.androidauthority.com/android-app-development-1128595/>