Project Report

Title: Agventure

Team Members

1) Bittu Kumar Ray – S20170010027 (bittukumar.r17@iiits.in)
2) Aaquib Niaz – S20170010002 (aaquib.n17@iiits.in)
3) Wilson Patro – S20170010184(wilson.p17@iiits.in)
4) Rohit Kumar Singh – S20170010127(rohitkumar.s17@iiits.in)

Group No: 123

Abstract

The main goal of our project is to provide a platform for the farmers where they can connect with other farmers and the buyers to sell their crops at an effective price. We will be providing all the useful information for the farmers like details about the crops, fertilizers etc. along with a detailed weather report for the farmers to be prepared for any kind of natural calamities. There will be a pooling system for the farmers where they can connect with the other farmers to share their crops. There will be a delivery system also where the farmers can directly sell and deliver their crops to the potential buyers. It is an attempt to make the lives of the farmers a little bit easier and give them the profit they truly deserve.

Modules

1) Authentication and Authorization

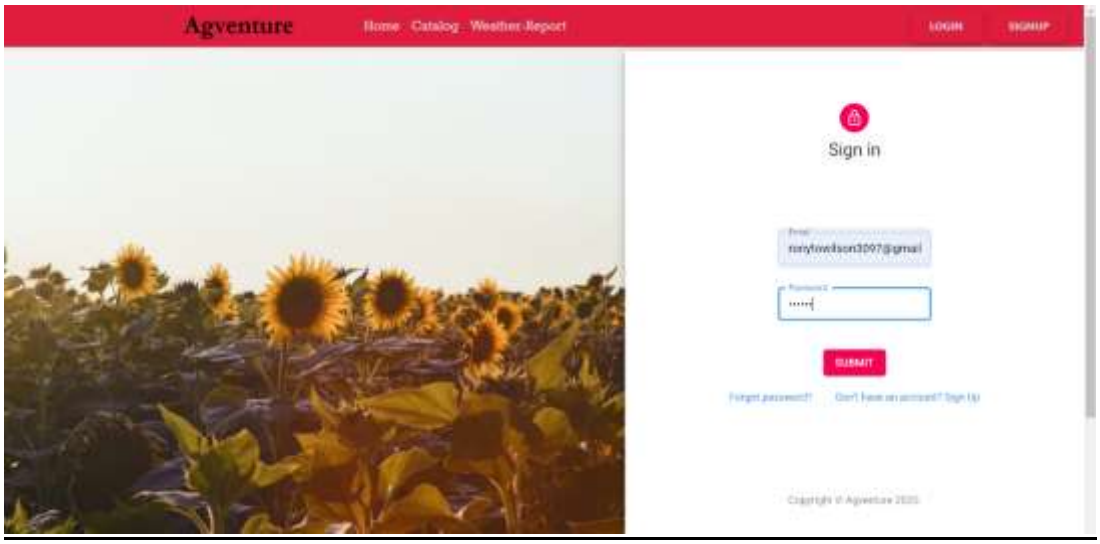
a. Description:

This module deals with the registering and logging of the users. We have two types of users namely – Farmer and General User. All the features and pages will come according to the role of the user. We have handled all the edge cases.

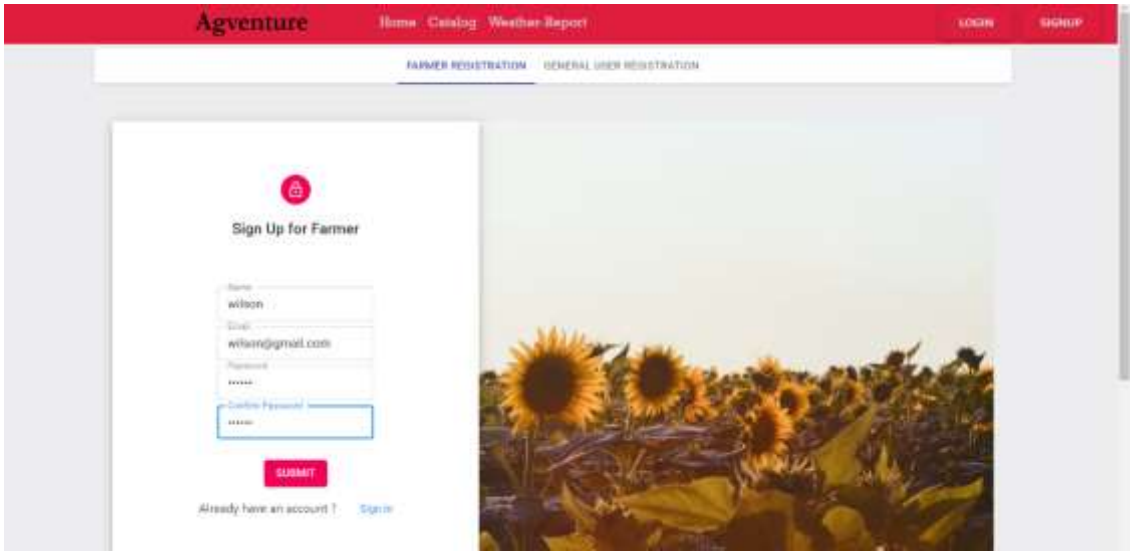
b. APIs:

Sl No	Routes	Method	Access	Description
1.	api/farmer/login	POST	Public	Login for farmers
2.	api/farmer/register	POST	Public	Signup for farmers
3.	api/user/register	POST	Public	Signup for general users
4.	api/user/login	POST	Public	Login for general users

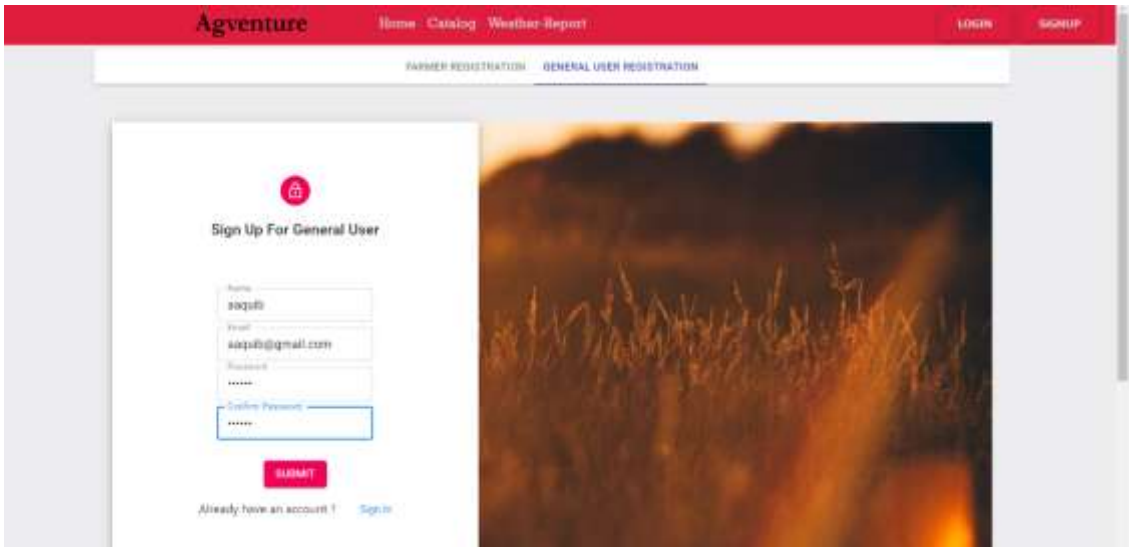
c. Use Cases:



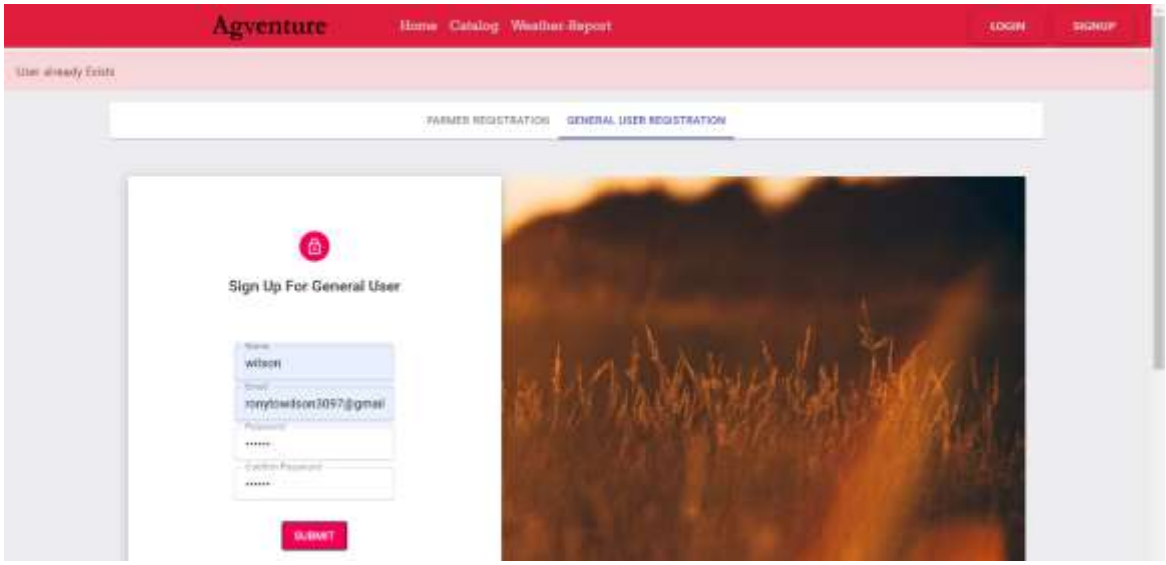
A Common login for all types of users



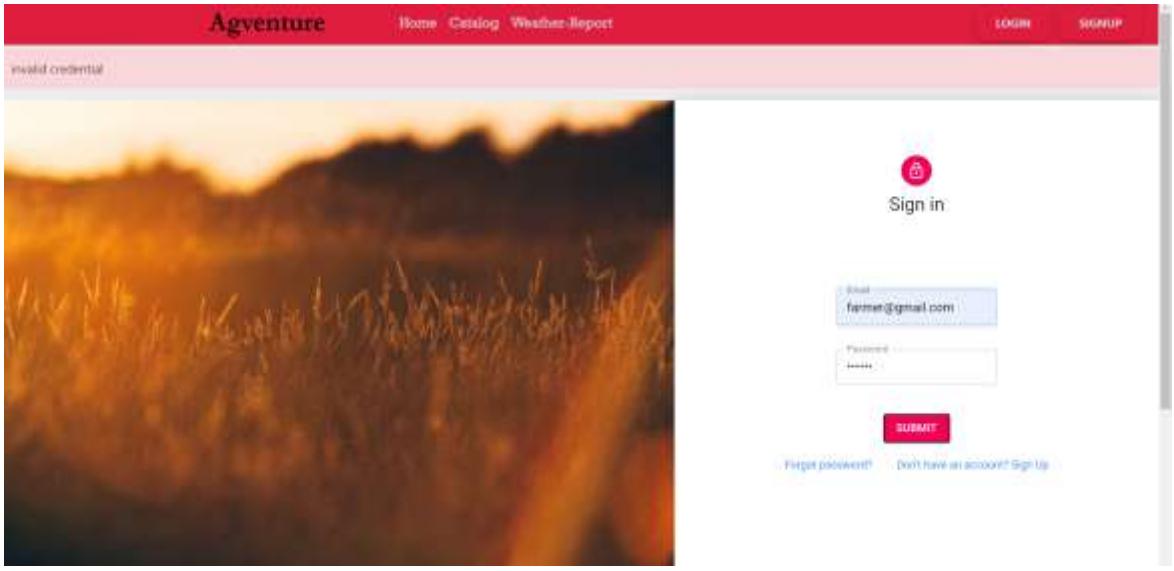
Signup for farmers



Signup for General Users



An error test case for registration



An Error Test Case for Login

d. Workflow

2) Dashboard for farmers

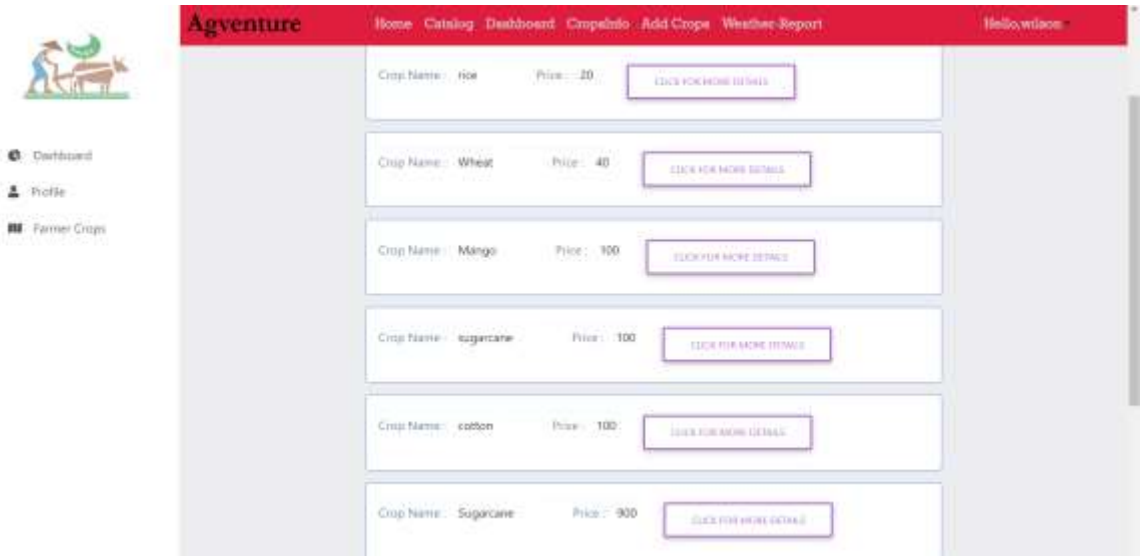
a. Description:

This module consists of the dashboard, profile and other details for the farmers. Farmers can edit their profile keep track of their earnings, orders and the crops. The best thing is they don't need to do anything, we will be doing everything for them in the back-end.

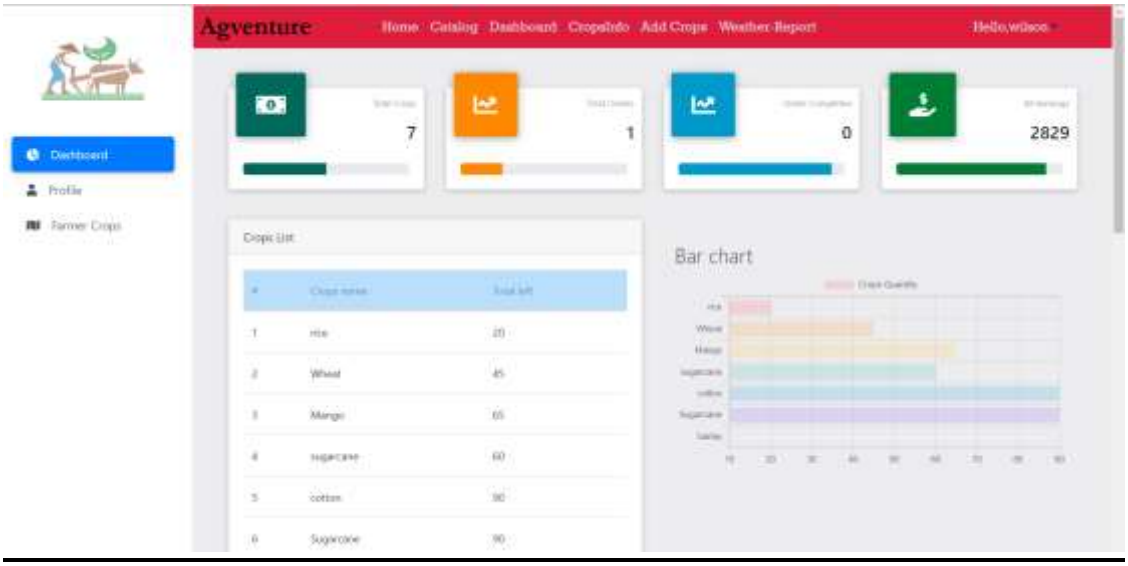
b. APIs

Sl No	Routes	Method	Access	Description
1.	api/farmer/get-farmer-crops	GET	Private	Get the specific crops for a farmer
2.	api/farmer/pie-chart	GET	Private	Get the sales and crops data for the charts
3.	api/farmer/get-farmer-details	GET	Private	Get the farmer details
4.	api/farmer/update-farmer	POST	Private	Edit profile
5.	api/farmer/farmer-crops/:id	GET	Private	Get the details for a single crop for the farmer.

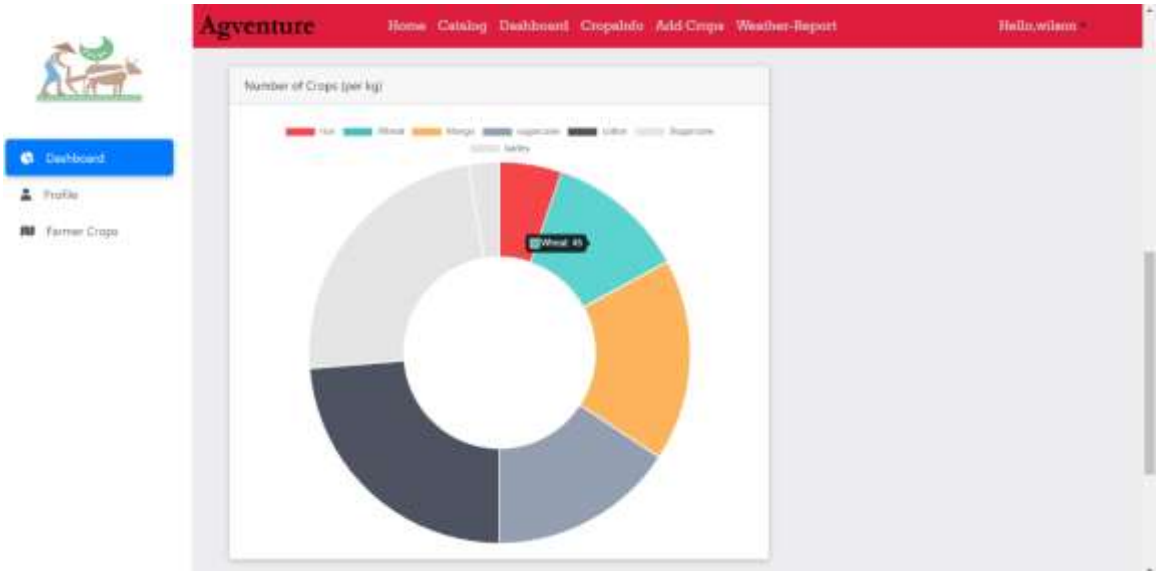
c. Use Cases:



Crop Details for farmers

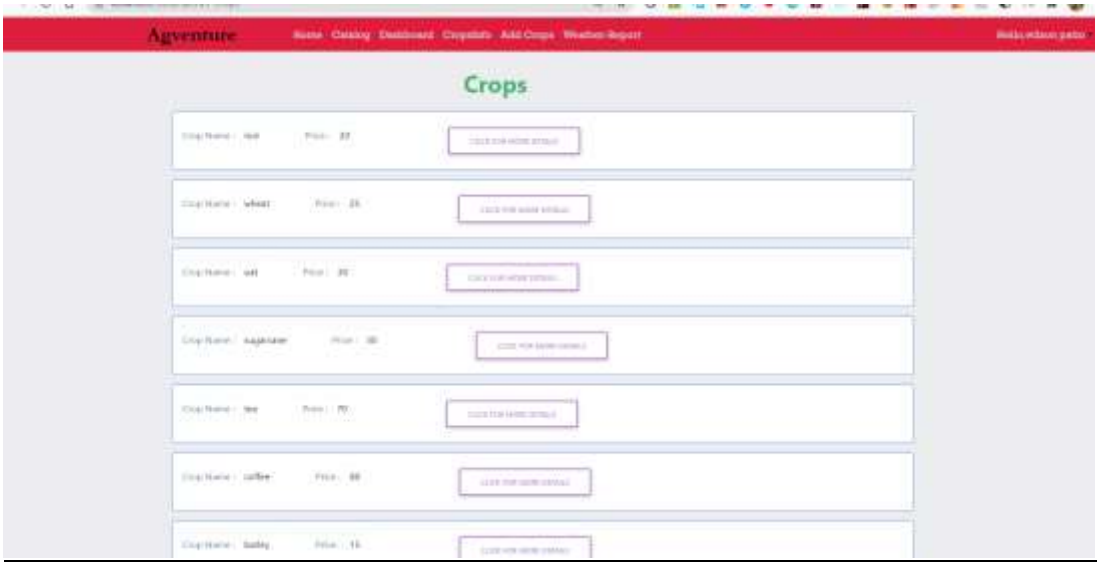


Dashboard

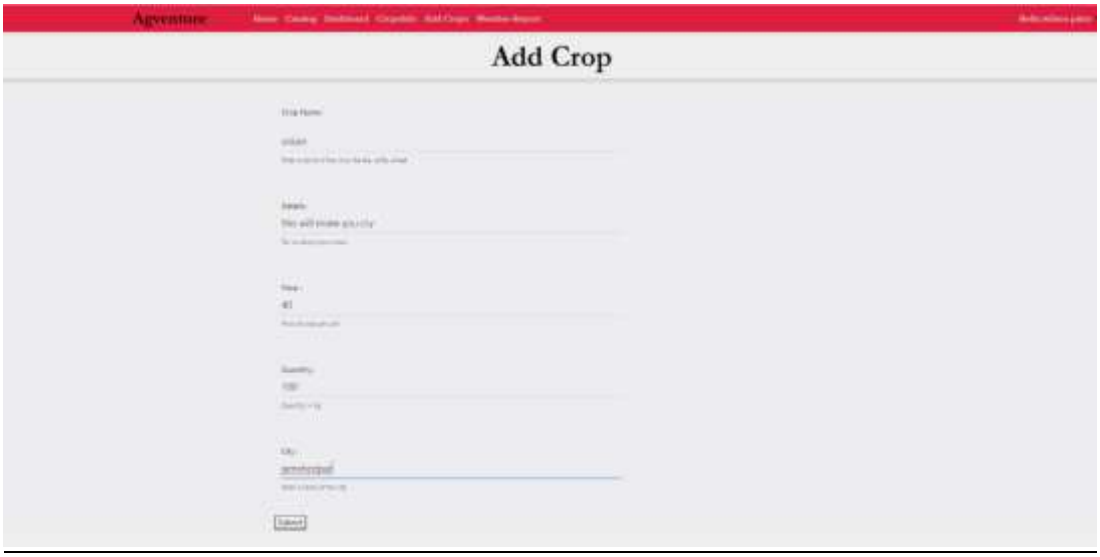


Pie chart for the crops

Crop detail page for farmers

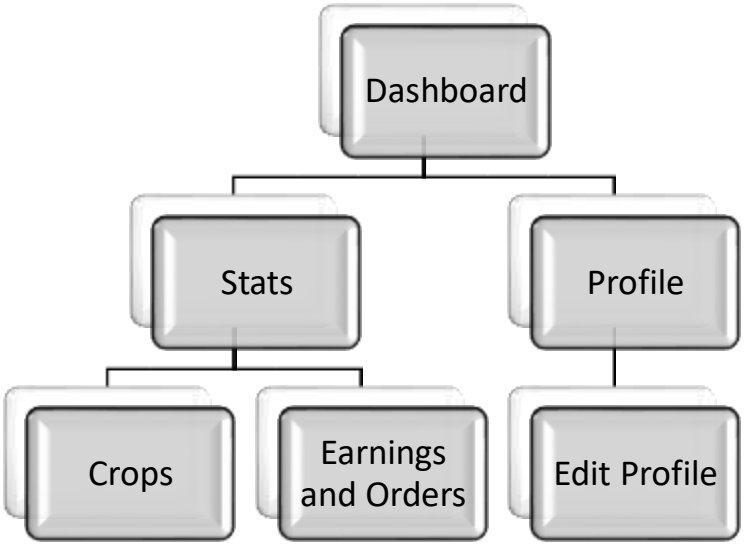


Crops List for farmers



Add crop form for farmers

d. Workflow



3) Dashboard for General Users

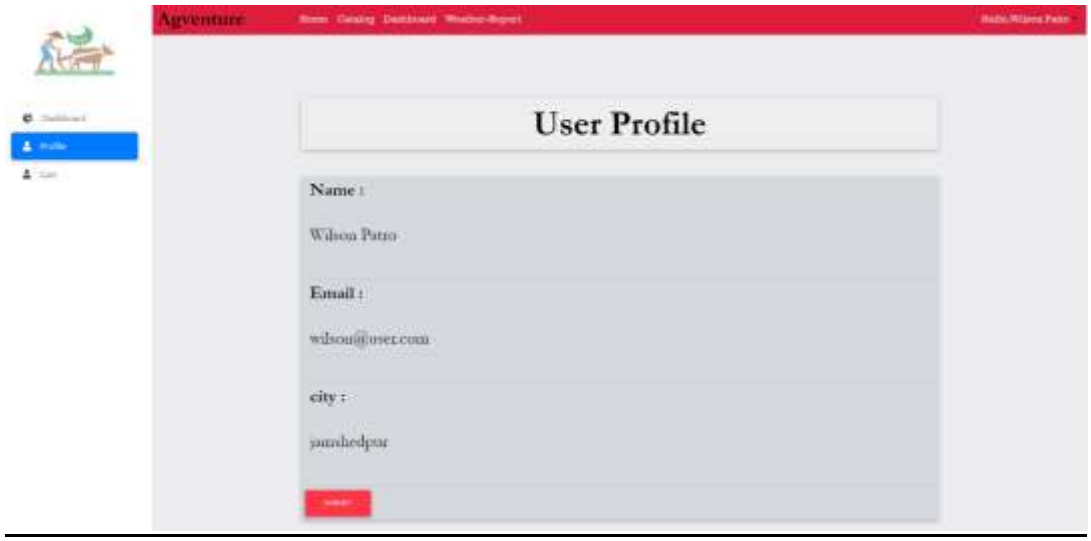
a. Description:

This module consists of the user dashboard with the users details such as profile, orders, cart and other stats.

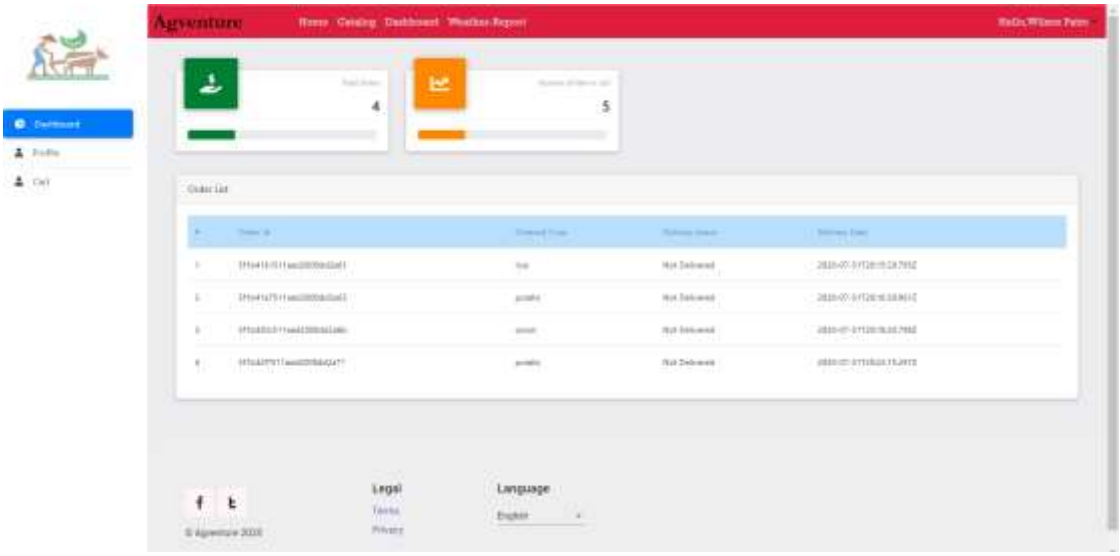
b. APIs

Sl No	Routes	Method	Access	Description
1.	api/user/get-orders	GET	Private	Get the orders made by the user.
2.	api/user/get-user-details	GET	Private	Get the user details
3.	api/user/update-user	POST	Private	Edit profile of the user
4.	api/user/get-cart	GET	Private	Get the items in the cart for the user

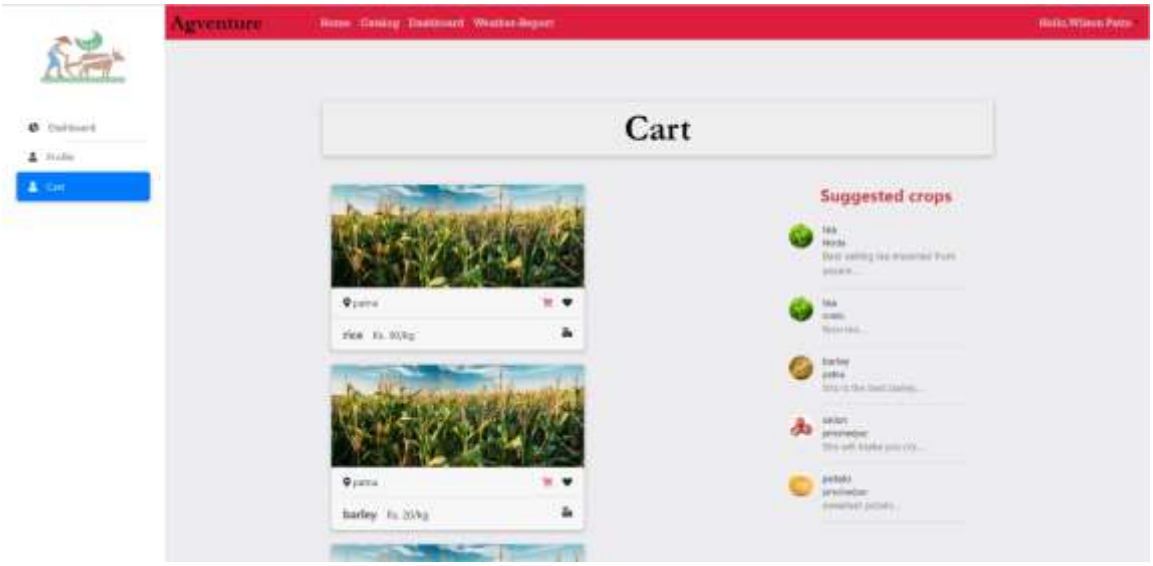
c. Use Cases:



Editing the profile



Dashboard where he can see orders



Cart section for the User

d. Workflow:

4) Information related to agriculture

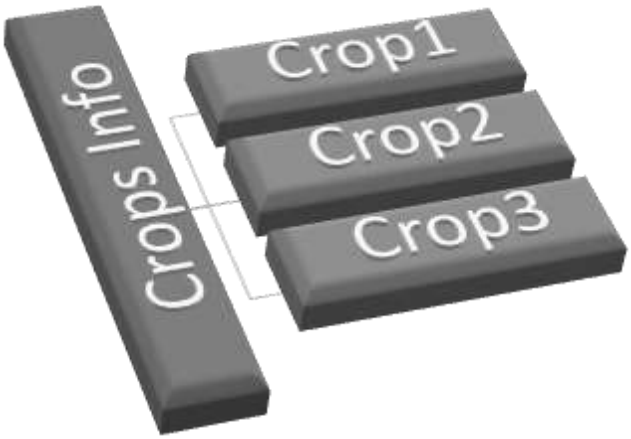
a. Description: This module deals with the information about the crops, fertilizers and pesticides which we are providing to the users.

b. APIs: This module is completely on the front end as we are web crawling and extracting data from various websites and not storing them in the backend.

c. Use Cases:

Getting the information about various crops

d. Workflow:



5) Weather Report

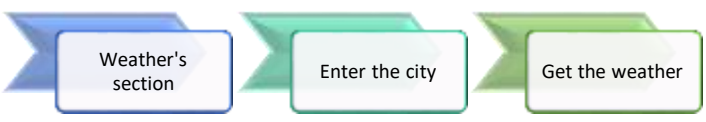
a. Description: This module handles the weather report for any location. The farmers can check about their location and get the details about the weather.

b. APIs: This also takes the information through external APIs so no api for it.

c. Use Cases:

Weather information for a city

d. Workflow:



6) Catalogue, Cart and Orders

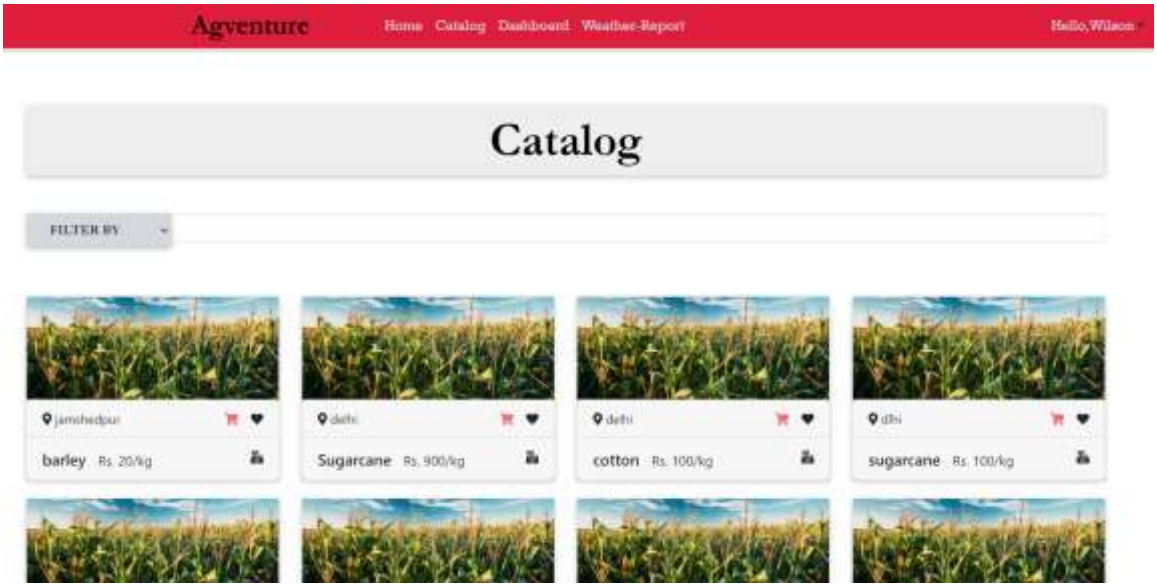
a. Description:

This section deals with the catalogue of crops which the user will see. The user can choose any crop according to his convenience and either add to cart or directly order it.

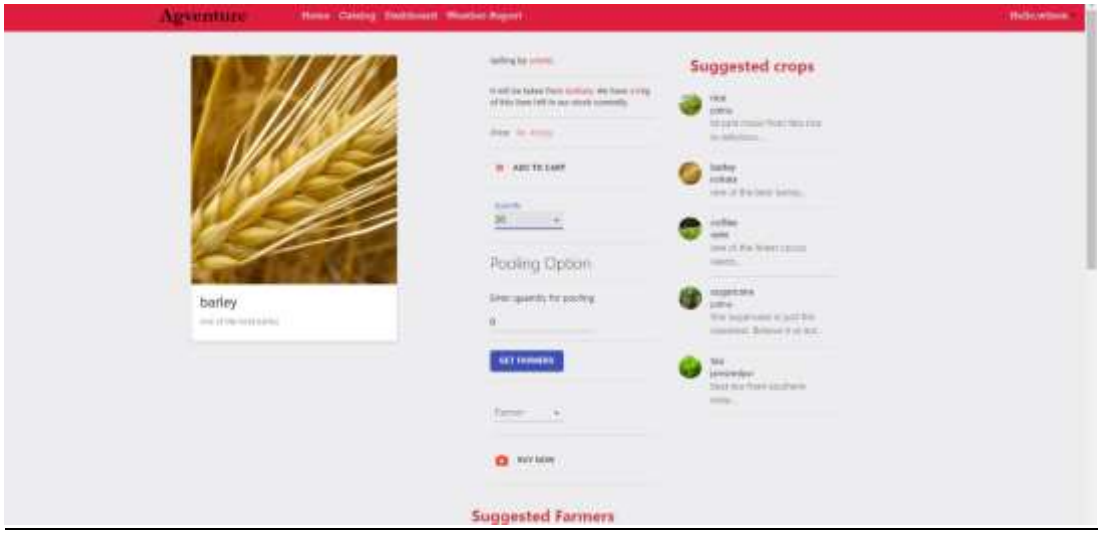
b. APIs:

Sl No	Routes	Method	Access	Description
1.	api/filter/all	GET	Public	Get all the crops to show
2.	api/get-crop/:id	GET	Public	Get the crop details
3.	api/user/add-cart	POST	Private	Add the crop to the cart
4.	api/user/order	POST	Private	Order any crop

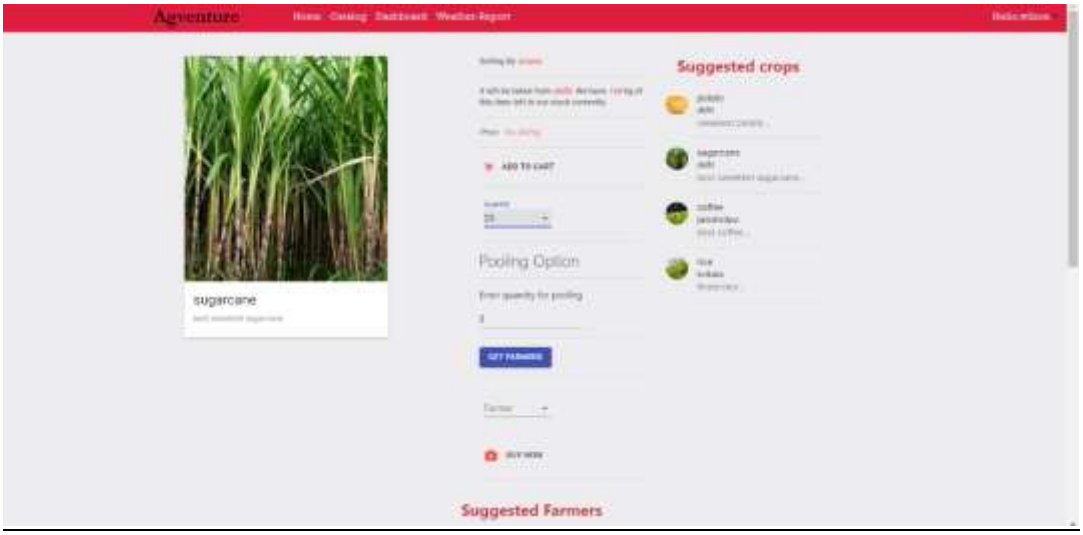
c. Use Cases:



The catalog for the users with all the crops



Add to cart and order page with product details



Suggested crops are according to the crop selected

d. Workflow

7) Elastic Search

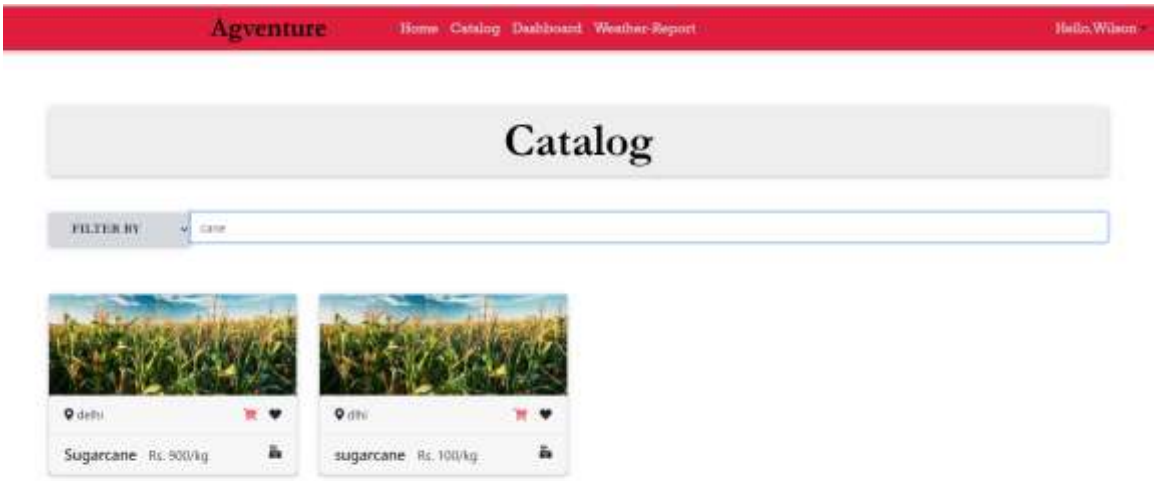
a. Description:

Users can search for crops, by different filters like city, username, crop's name, etc. For a faster search experience, we are using elastic search server which is running in Docker container and we have made api which are querying from elastic search index. It is 10 times faster than normal search. You don't even need to write the whole word. Just a part of the word also works and it gives the search result in real time.

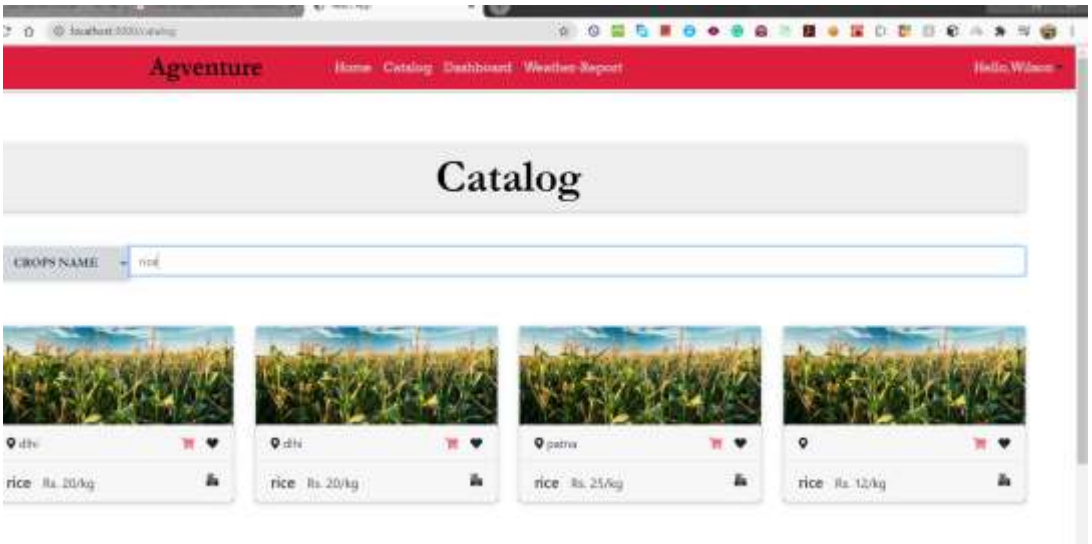
b. APIs:

Sl No	Routes	Method	Access	Description
1.	/es-search/city/:crop	GET	Public	Filter by city
3.	/es-search/name/:crop	GET	Public	Filter by crop name
4.	es-search/name/filter	GET	Public	Filter all

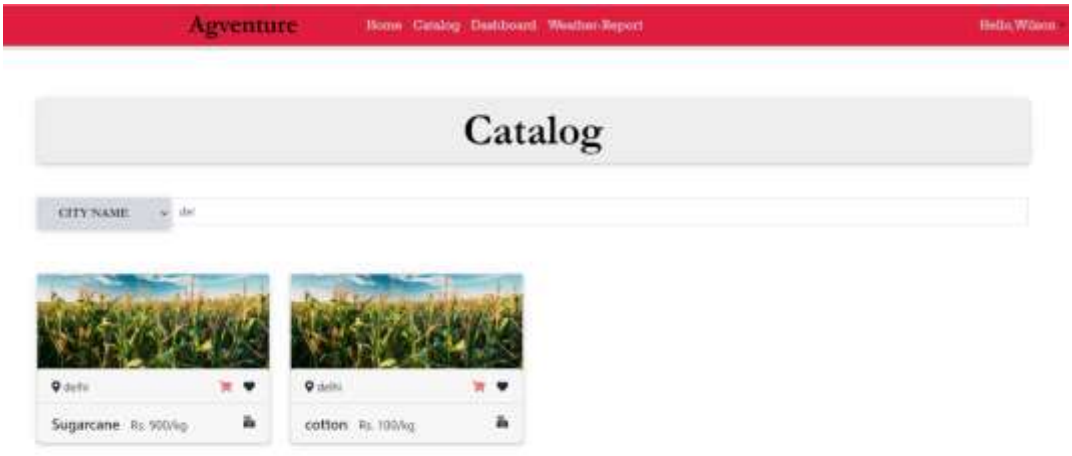
c. Use Case:



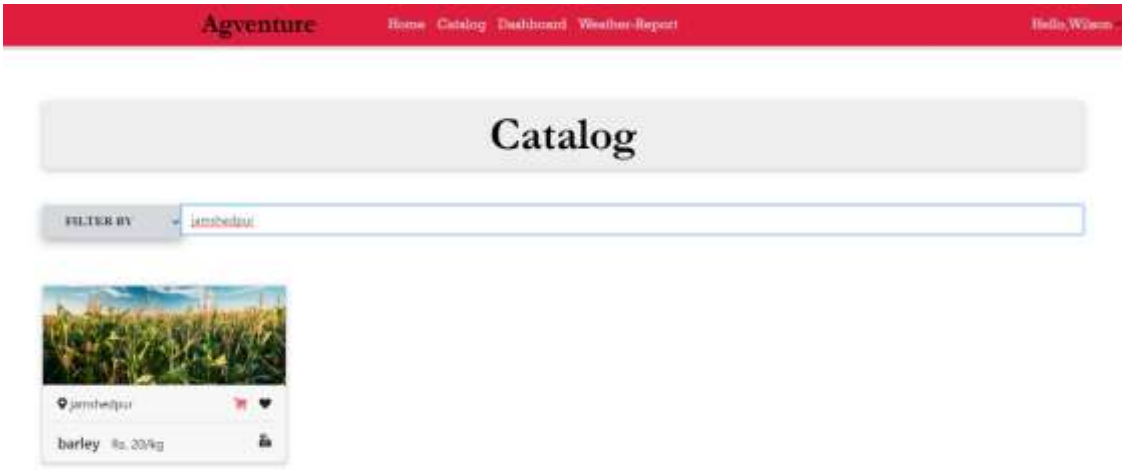
Filtered by just part of a word



Filtered by the name of crop.

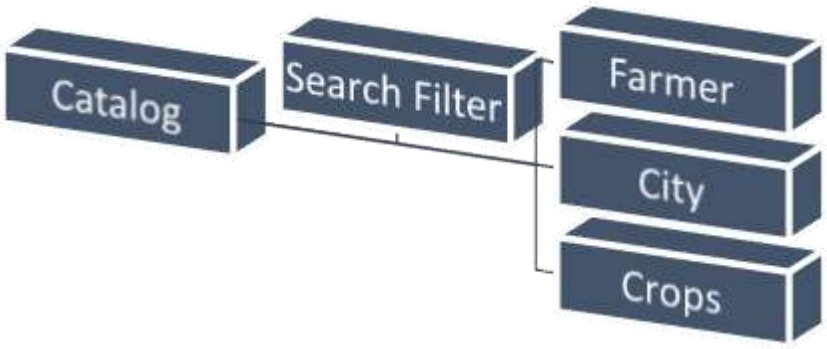


Filtered by just a part of city



Filtered by the name of city

d. Workflow



8) Payment

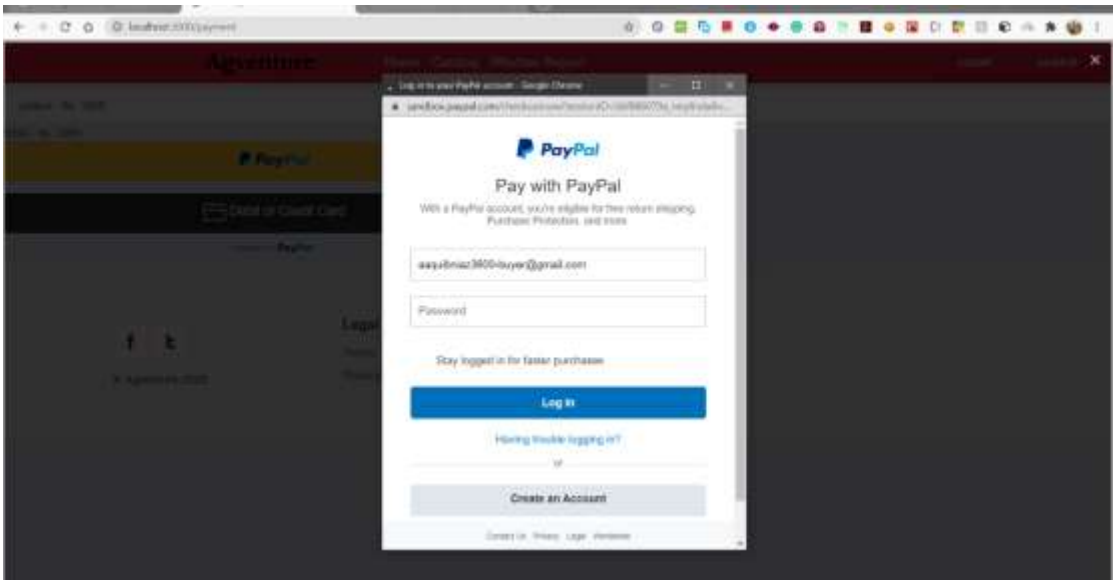
a. Description:

This module deals with the payment of the orders. We have integrated the PayPal payment service for this task. User can either directly add their details or add a card save it then pay for the orders.

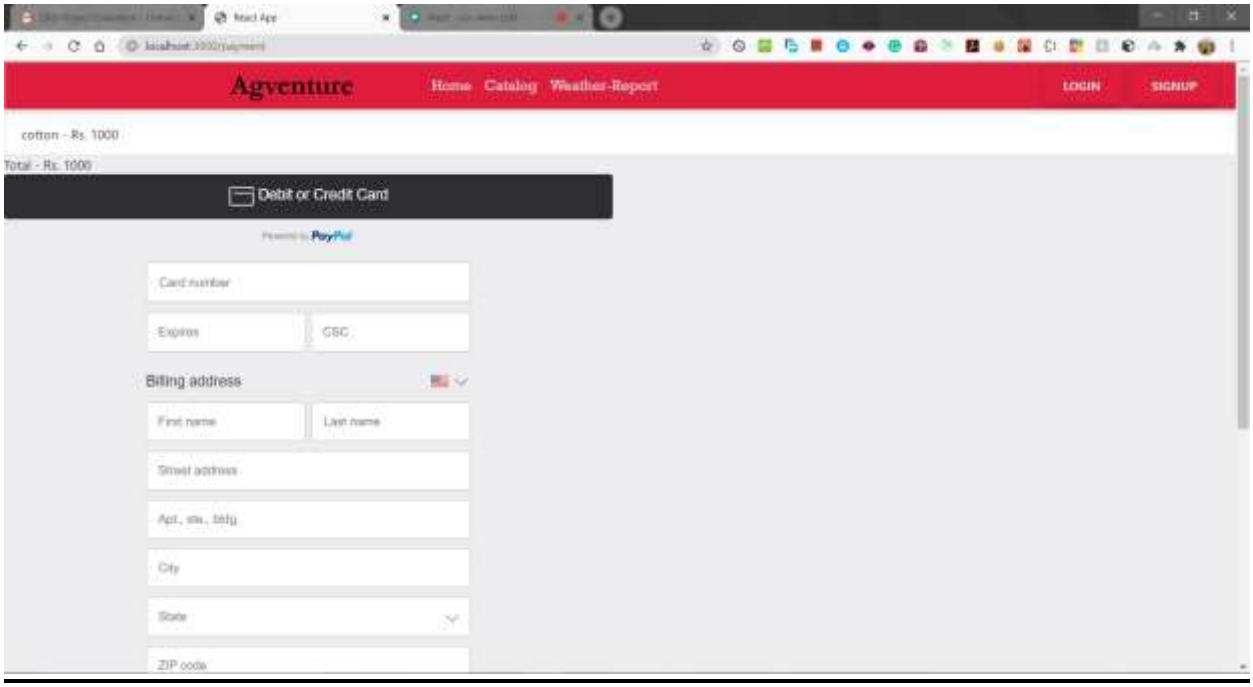
b. APIs:

Sl No	Routes	Method	Access	Description
1.	/order-successful	POST	Private	Payment

c. Use Cases



Paypal window



Card Details

d. Workflow

9) Pooling

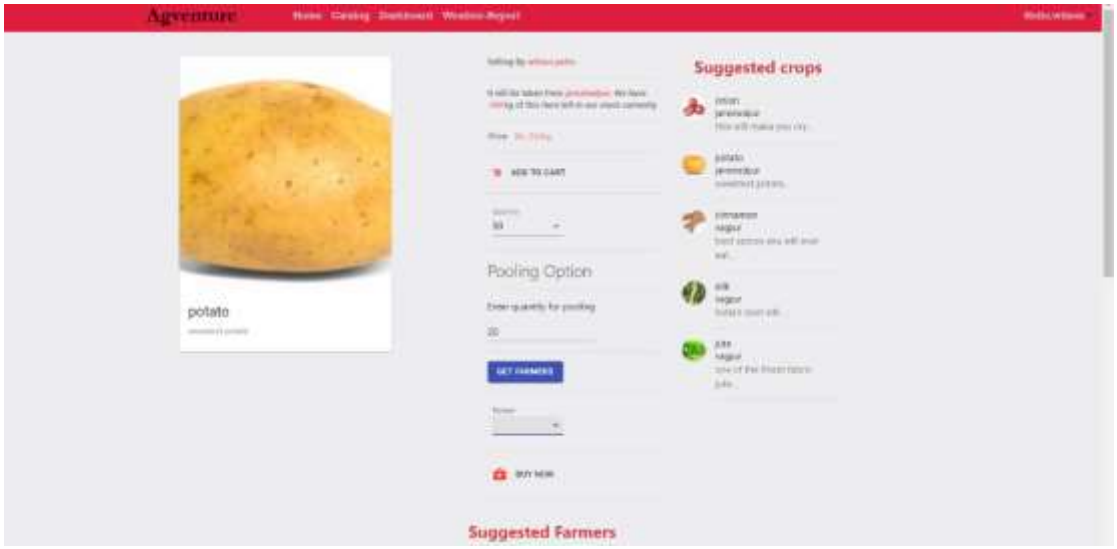
a. Description:

This is a special feature offered by us. If a user wants more quantity than it is available with a farmer. He can select another farmer from the list which we give according to his entered quantity. Rest everything is handled in the back-end and the order will be placed with the amount shared between the two farmers.

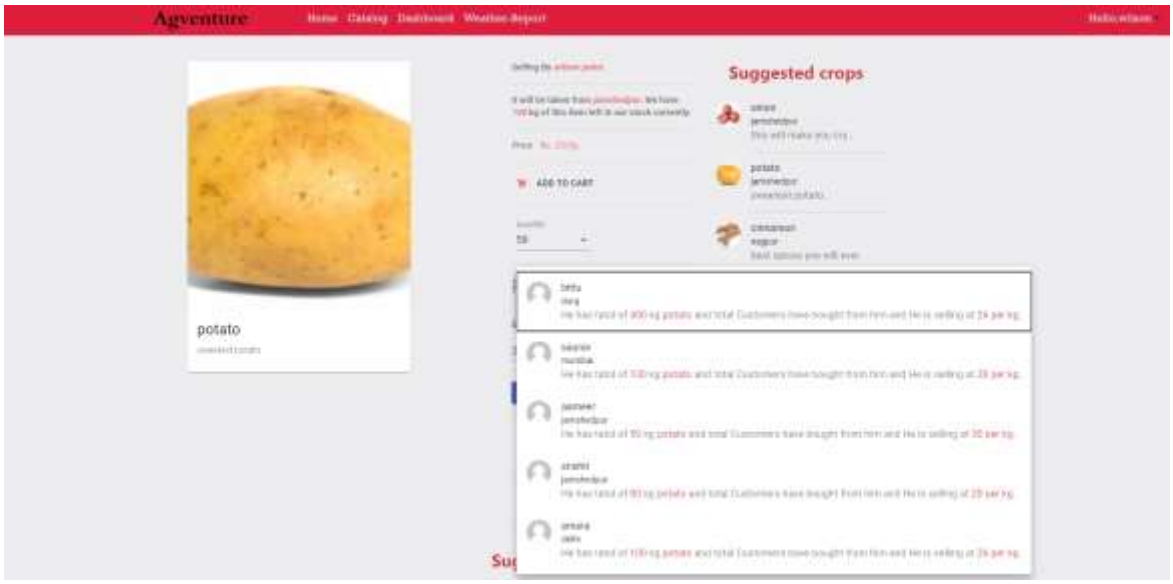
b. APIs:

Sl No	Routes	Method	Access	Description
1.	suggested-crops-pooling/:id1/:id2/:id3	GET	Private	Pooling

c. Use Cases

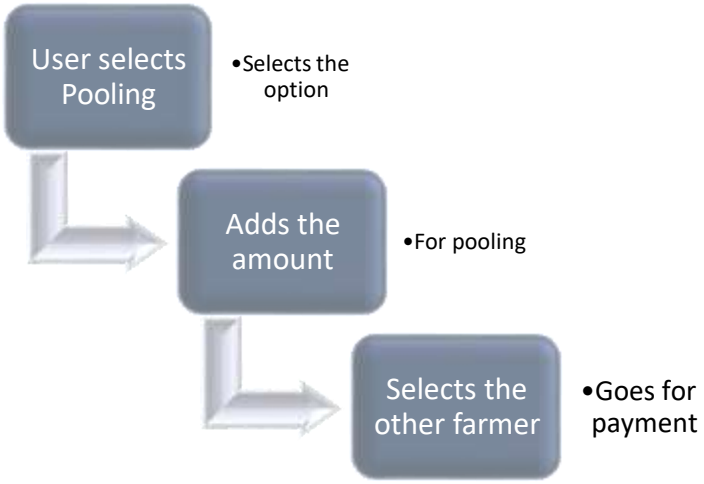


User has selected pulling and entered amount

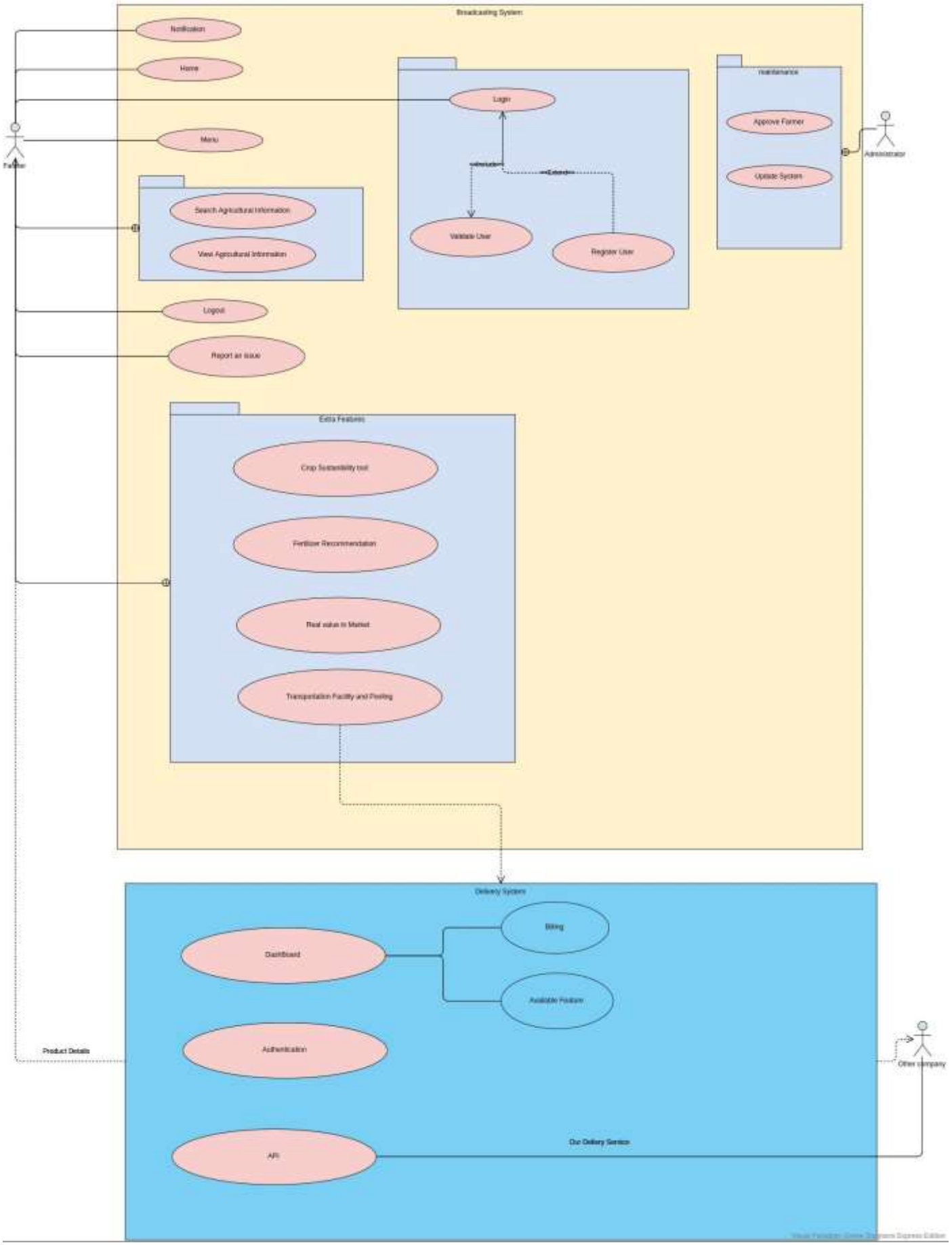


This gives the list of farmers with that quantity of the crop

d. Workflow


User selects Pooling
•Selects the option
Adds the amount
•For pooling
Selects the other farmer
•Goes for payment

Use Case Diagram:



Database Schema:

1) We are using a NoSQL (MongoDB) database for our project.

2) There are several reasons as of why we are using a NoSQL database and some of which are listed below:

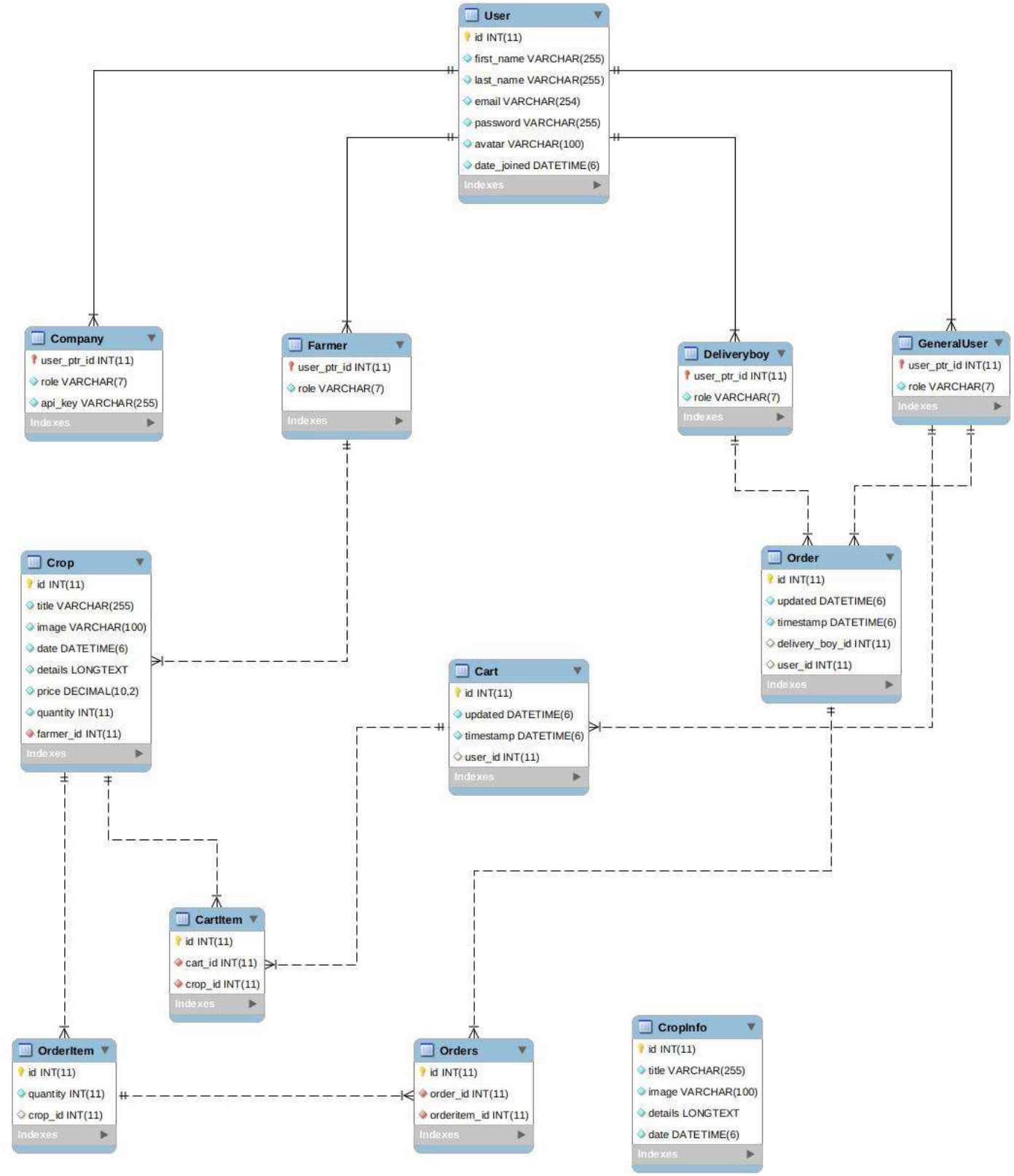
a. NoSQL databases are more scalable and provide superior performance, and their data model addresses several shortcomings of the relational databases.

b. It can handle large volume of structured, semi-structured and unstructured data and we have mostly unstructured data.

c. Because of the above mentioned property it is flexible, the exact kind of database we need for this project.

d. We are using mongo DB as it offers the advantages of the relational database along with the innovations of NoSQL.

5) Here is our database schema. The major models of our database are **User (Farmer, Customer, Company), Crops, Orders, Delivery and Cart.**


User
id INT(11)
first_name VARCHAR(255)
last_name VARCHAR(255)
email VARCHAR(254)
password VARCHAR(255)
avatar VARCHAR(100)
date_joined DATETIME(6)
Indexes
Company
user_ptr_id INT(11)
role VARCHAR(7)
api_key VARCHAR(255)
Indexes
Farmer
user_ptr_id INT(11)
role VARCHAR(7)
Indexes
Deliveryboy
user_ptr_id INT(11)
role VARCHAR(7)
Indexes
GeneralUser
user_ptr_id INT(11)
role VARCHAR(7)
Indexes
Crop
id INT(11)
title VARCHAR(255)
image VARCHAR(100)
date DATETIME(6)
details LONGTEXT
price DECIMAL(10,2)
quantity INT(11)
farmer_id INT(11)
Indexes
Order
id INT(11)
updated DATETIME(6)
timestamp DATETIME(6)
delivery_boy_id INT(11)
user_id INT(11)
Indexes
Cart
id INT(11)
updated DATETIME(6)
timestamp DATETIME(6)
user_id INT(11)
Indexes
CartItem
id INT(11)
cart_id INT(11)
crop_id INT(11)
Indexes
OrderItem
id INT(11)
quantity INT(11)
crop_id INT(11)
Indexes
Orders
id INT(11)
order_id INT(11)
orderitem_id INT(11)
Indexes
CropInfo
id INT(11)
title VARCHAR(255)
image VARCHAR(100)
details LONGTEXT
date DATETIME(6)
Indexes

Technology Stack

1) We will be using MERN stack for our project.
2) The front end will be in ReactJs and we will use MDbootstrap for styling and some components.
3) We will be using NodeJs for creating API's and connecting them with front end.
4) We will be using mongo DB for our database and better scalability.
5) We have used Docker also to containerize the elastic search with elastic index which made our search real time and extremely fast and flexible. 10 times faster than the normal search.
6) We have also built the Jenkins CI/CD pipeline which will help us easily deploy whenever we want to deploy it.

Thank You