CHAPTER 1

INTRODUCTION

1.1 PERSCPECTIVE

The Farm Management System represents a transformative approach to agricultural management, leveraging technology to optimize farming operations and enhance the agricultural marketplace. By integrating crop and livestock management, inventory tracking, user account management, and seamless shopping experiences, the farm management system streamlines the entire process from farm to consumer. It empowers administrator to efficiently manage their resources while providing consumers with convenient access to a wide range of agricultural products. With its user-friendly interface, personalized experiences, and secure payment integration.

1.2 OBJECTIVE

The objective of the Farm Management System is to revolutionize agricultural management by providing farmers with a comprehensive platform to efficiently manage crops and livestock, track inventory, and optimize farm operations. Additionally, the FMS aims to enhance the consumer experience by offering a user-friendly interface for browsing, shopping, and purchasing agricultural products.

1.3 SCOPE

The Farm Management System facilitates seamless shopping experiences for users, allowing them to browse and purchase crops while accessing detailed benefits information. User accounts are required for viewing details, and upon completing shopping, users proceed directly to the payment page. The system enables administrators to manage cattle quantities and oversee inventory, providing users with visibility into available items while granting admins control over quantities and product listings.

CHAPTER 2

REQUIREMENT DESCRIPTION

2.1 FUNCTIONAL REQUIREMENTS

The functional requirement in Farm Management System is the collective information about what are the operations available in the system.

- Users must be able to create accounts or log in using existing credentials to access the system.
- Access to detailed crop information and shopping functionalities is restricted to authenticated users only.
- The system must provide functionality for users to create accounts, update their profiles, and manage their personal information.
- Only authenticated users should have access to detailed crop information and be able to proceed to the payment page.
- Users should be able to browse available crops, including detailed descriptions and benefits.
- Search functionality should allow users to easily find specific crops based on criteria such as type, benefits, or price.

2.2 NON-FUNCTIONAL REQUIREMENTS

The non-functional requirement describes about platform and physical resource required for building the farm management system.

- The system should be responsive and able to handle concurrent user requests without significant latency, even during peak usage periods.
- The system must ensure secure user authentication and data transmission to protect user information and payment details.
- The user interface must be intuitive and easy to navigate, catering to users with varying levels of technical expertise.
- The system should be scalable to accommodate increasing numbers of users, crops, and inventory items as the platform grows.

CHAPTER 3

SYSTEM DESIGN

3.1 ARCHITECTURE DESIGN

This diagram represents the flow of “farm management system”. The user and admin can login, if they already have an account or else they have to sign up and then login into this farm website. The admin can add, delete and update farm items. They can view the user’s feedback. The users can view the crops, cattle's and their benefits.The user can also buy the fresh farm products. And then they can share their feedback about the farm.

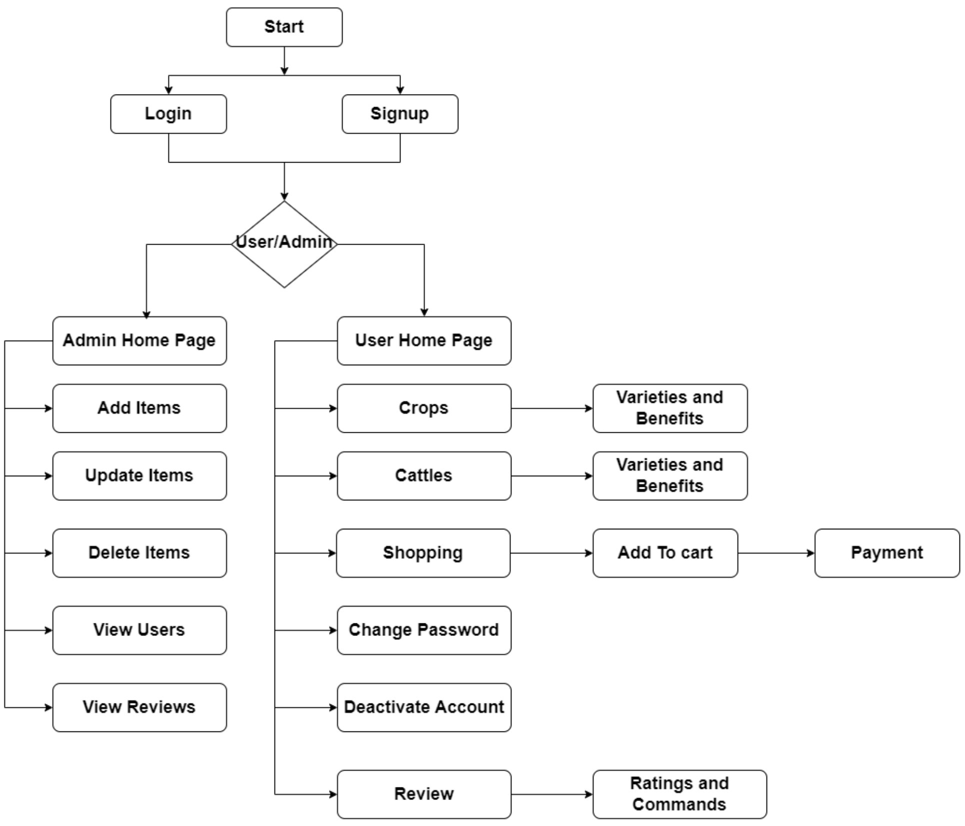


Figure 3.1: Architecture Diagram of Farm Management System

3.2 DESIGN COMPONENTS

3.2.1 Front End:

The Farm Management System uses Angular for developing interactive pages.

3.2.2 Back End:

Uses Mongo DB for back end to store data.

3.3 DATABASE DESCRIPTION

Listed below gives a description of database document schemas used for Farm Management System.

3.3.1 User sign up Structure

As shown in table 3.3.1, User sign up structure contains the details of the users. The user signup has the description required for the user and has the attributes such as the full name, user name, email, phone number, and password, confirm password and gender for the user.

Table 3.3.1: User sign up Description

Attribute Name	Type	Constraint(s)	Description
Full Name	String	NOT NULL	Full Name of the user
Username	String	NOT NULL	User Name of the user
Email	String	Must be unique	Email id of the user
Phone Number	String	Must be unique	Phone number of the user
Password	String	NOT NULL	Password of the user
Confirm Password	String	NOT NULL	Password of the user
Gender	String	NOT NULL	Gender of the user

3.3.2 User Login Structure

As shown in table 3.3.2, User Login structure contains the details of the users. The login description has the attributes such as the email id and password of the user.

Table 3.3.2: User Login Description

Attribute Name	Type	Constraint(s)	Description
Email	String	Must be unique	Email id of the user
Password	String	NOT NULL	Password of the user

3.3.3 Change password Structure

As shown in table 3.3.3, User Login structure contains the details of the users. The Change password description has the attributes such as the email id ,old password and new password of the User.

Table 3.3.3: Change Password Description

Attribute Name	Type	Constraint(s)	Description
Email	String	Must be unique	Email id of the user
Old Password	String	NOT NULL	Password of the user
New Password	String	NOT NULL	Password of the user

3.3.4 Deactivate Account Structure

As shown in table 3.3.4, User Deactivate structure contains the details of the users. The login description has the attributes such as the email id and password of the user.

Table 3.3.4: Deactivate Account Description

Attribute Name	Type	Constraint(s)	Description
Email	String	Must be unique	Email id of the user
Password	String	NOT NULL	Password of the user

3.3.5 Admin Login Structure

As shown in table 3.3.5, Admin Login structure contains the details of the admin. The login description has the attributes such as the email id and password of the admin.

Table 3.3.5: Admin Login Description

Attribute Name	Type	Constraint(s)	Description
Email	String	Must be unique	Email id of the user
Password	String	NOT NULL	Password of the user

3.3.6 Add Items Structure

As shown in table 3.3.6, Add Items structure contains the details of the items. The Add Items description has the attributes such as the name and quantity of the Item.

Table 3.3.6: Add Items Description

Attribute Name	Type	Constraint(s)	Description
Name	String	NOT NULL	Name of the Item
Quantity	String	NOT NULL	Quantity of the Item.

3.3.7 Update Items Structure

As shown in table 3.3.7, Items Update structure contains the details of the itemd. The Update Items description has the attributes such as the name, old quantity and new quantity of the Item.

Table 3.3.7: Update Items Description

Attribute Name	Type	Constraint(s)	Description
Name	String	NOT NULL	Name of the Item
Old Quantity	String	NOT NULL	Old Quantity of the Item.
New Quantity	String	NOT NULL	New Quantity of the Item.

3.3.8 Delete Items Structure

As shown in table 3.3.8, Delete Items structure contains the details of the items. The Delete Items description has the attributes such as the name and quantity of the Item.

Table 3.3.8: Delete Items Description

Attribute Name	Type	Constraint(s)	Description
Name	String	NOT NULL	Name of the Item
Quantity	String	NOT NULL	Quantity of the Item.

3.3.9 Review Structure

As shown in table 3.3.9, Review structure contains the details of the items. The Review description has the attributes such as the email of the user and review of the item.

Table 3.3.9: Review Description

Attribute Name	Type	Constraint(s)	Description
Email	String	Must be unique	Email id of the user
Review	String	NOT NULL	Review of the Item gives by the user.

3.4 LOW LEVEL DESIGN

The following section illustrates the functionalities of the system. This includes admin and user login to the application, User Signup,

3.4.1 Home

Table 3.4.1 shows the Home details of the application.

Table 3.4.1 Home Details

Files used	home.ts, home.html, home.css
Short Description	Allows the user wants to know about our service
Arguments	Home, about us and sign In
Return	Null
Pre-Condition	The user must have an internet.
Post-Condition	Access the site
Exception	Site can’t be reached
Actor	User

3.4.2 Admin Login

Table 3.4.2 shows the Admin login details of the application.

Table 3.4.2 Admin Login Details

Files used	adminlogin.ts, adminlogin.html, adminlogin.css
Short Description	Allows the admin to login to the application
Arguments	Email, Password
Return	Success/Failure in login
Pre-Condition	The admin must have an account
Post-Condition	The admin home page will be displayed
Exception	Invalid Email and password
Actor	Admin

3.4.3 User Login

Table 3.4.3 shows the User login details of the application.

Table 3.4.3 User Login Details

Files used	userlogin.ts, userlogin.html, userlogin.css
Short Description	Allows the user to login to the application
Arguments	Email, Password
Return	Success/Failure in login
Pre-Condition	The user must have an account
Post-Condition	The user home page will be displayed
Exception	Invalid Email and password
Actor	User

3.4.4 Add Item

Table 3.4.4 shows the add items details of the application.

Table 3.4.4 Add item Details

Files used	Additem.ts, Additem.html, Additem.css
Short Description	Allows the admin to Adds items to the application
Arguments	Item name, Quantity

Return	Success/Failure in is insertion of the item
Pre-Condition	The item must be unavailable.
Post-Condition	New item added successfully
Exception	Item already available
Actor	Admin

3.4.5 User Signup

Table 3.4.5 shows the User signup details of the application.

Table 3.4.5 User Signup Details

Files used	signup.ts, signup.html, signup.css
Short Description	Allows the user to login to the application
Arguments	Full name, User Name, Email, Phone number, Password , confirm password and gender
Return	Success/Failure in creating an account
Pre-Condition	The user must have an valid email and password
Post-Condition	The user home page will be displayed
Exception	Please enter the required fields
Actor	User

3.4.6 Update Item

Table 3.4.6 shows the Update item details of the application.

Tables 3.4.6 Update item Details

Files used	Updateitem.ts, Updateitem.html, Updateitem.css
Short Description	Allows the admin to update items to the application
Arguments	Item name ,old quantity and new quantity of the item
Return	Success/Failure in is updating of the item
Pre-Condition	The item must be available.
Post-Condition	Item Updated successfully
Exception	Item is unavailable
Actor	Admin

3.4.7 Delete Item

Table 3.4.7 shows the delete item details of the application.

Tables 3.4.7 Delete item Details

Files used	Deleteitem.ts, Deleteitem.html, Deleteitem.css
Short Description	Allows the admin to delete items to the application
Arguments	Item name and quantity
Return	Success/Failure in is deletion of the item
Pre-Condition	The item must be available.
Post-Condition	Item deleted successfully
Exception	Item is unavailable
Actor	Admin

3.4.8 Change password

Table 3.4.8 shows the change password details of the application.

Table 3.4.8 Change password Details

Files used	changepwd.ts, changepwd.html, changepwd.css
Short Description	Allows the user to change password to their login
Arguments	Email id, Old password and New password
Return	Success/Failure in is change of their passwords
Pre-Condition	The user must have an account.
Post-Condition	Password updated successfully
Exception	Old password and new password are same
Actor	User

3.4.9 Deactivate account

Table 3.4.9 shows the Deactivate account details of the application.

Table 3.4.9 Deactivate account Details

Files used	delete.ts, delete.html, delete.css
Short Description	Allows the user wants to deactivate for their login
Arguments	Email id and password

Return	Success/ Failure in their deactivation
Pre-Condition	The user must have an account.
Post-Condition	The user account will be deactivated
Exception	No data found
Actor	User

3.4.10 Shopping cart

Table 3.4.10 shows the Shopping cart details of the application.

Table 3.4.10 Shopping-cart Details

Files used	Shopping-cart.ts, shopping-cart.html, shopping-cart.css
Short Description	Allows the user shopping
Arguments	Items, quantity and amount
Return	Success/ Failure in their shopping
Pre-Condition	The user must have an account
Post-Condition	Successful shopping
Exception	Item unavailable
Actor	User

3.4.11 Payment

Table 3.4.11 shows the Payment details of the application.

Table 3.4.11 Payment Details

Files used	pay.ts, pay.html, pay.css
Short Description	Allows the user once they complete their shopping they directly go to payment page
Arguments	Email id and amount
Return	Success/ Failure in their payment
Pre-Condition	The user must shopped the products.
Post-Condition	Payment successful
Exception	Fill all the fields
Actor	User

3.4.12 Review

Table 3.4.12 shows the Review details of the application.

Table 3.4.12 Review Details

Files used	review.ts, review.html, review.css
Short Description	Allows the user gives their review about our farm
Arguments	Rating, review and email id
Return	Success in their review
Pre-Condition	The user must have an account.
Post-Condition	Review send successful
Exception	Fill all required fields
Actor	User

3.5 USER INTERFACE DESIGN

3.5.1 Main Home

Figure 3.5.1 Provide the interface for main activity.

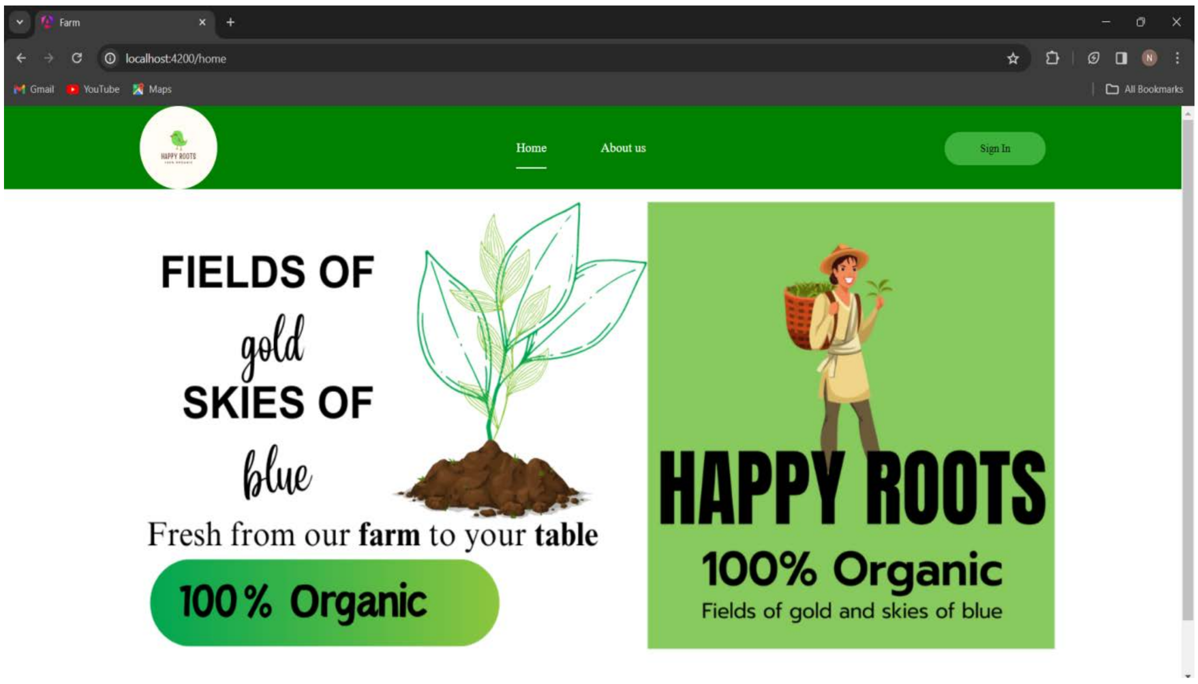
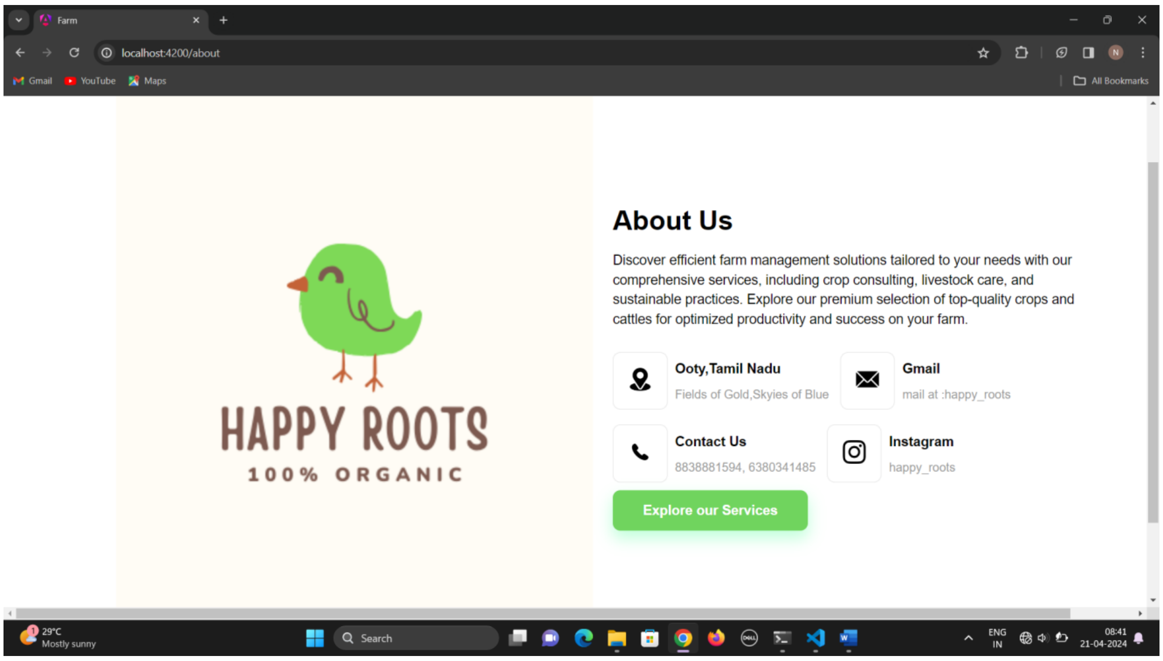


Figure 3.5.1: Main Home layout of Farm Management System

3.5.2 About Us

Figure 3.5.2 Provide the interface for About Us.



About Us

Discover efficient farm management solutions tailored to your needs with our comprehensive services, including crop consulting, livestock care, and sustainable practices. Explore our premium selection of top-quality crops and cattles for optimized productivity and success on your farm.

Ooty,Tamil Nadu
Fields of Gold,Skyies of Blue

Gmail
mail at :happy_roots

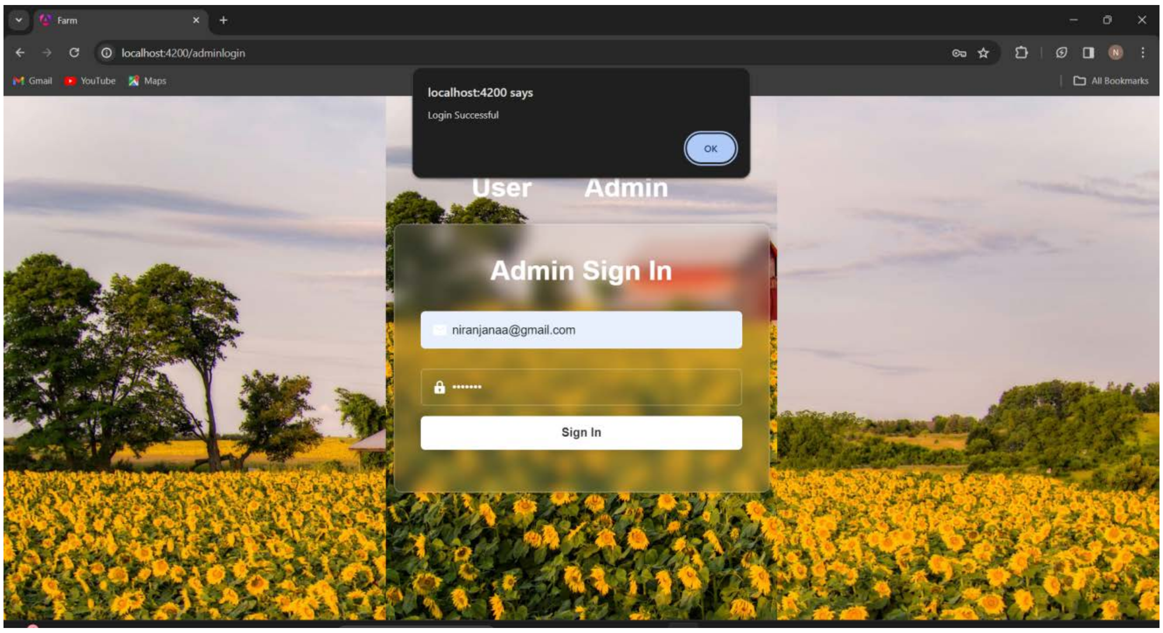
Contact Us
8838881594, 6380341485

Instagram
happy_roots

Figure 3.5.2: about us layout of Farm Management System

3.5.3 Admin Login page

Figure 3.5.3 provide the interface for Admin Login page



localhost:4200 says

Login Successful

niranjanaa@gmail.com

•••••••

Figure 3.5.3: Admin Login Page

3.5.4 Admin Home page

Figure 3.5.4 provide the interface for Admin Home page

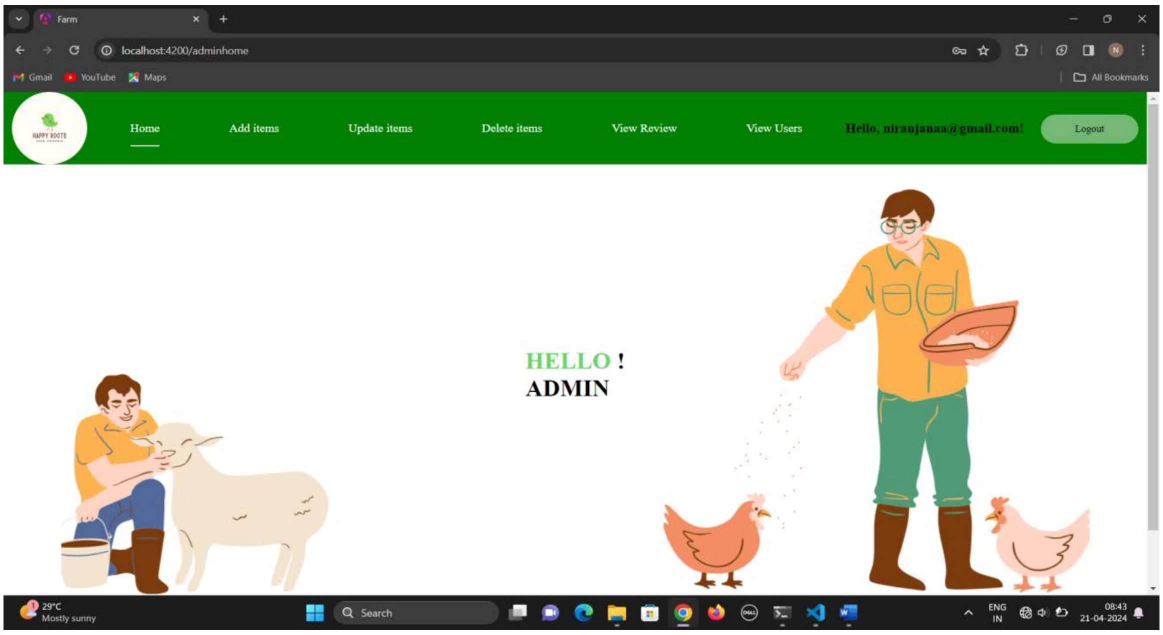


Figure 3.5.4: Admin Home layout of Farm Management System

3.5.5 Add Item page

Figure 3.5.5 provide the interface for Add Item page

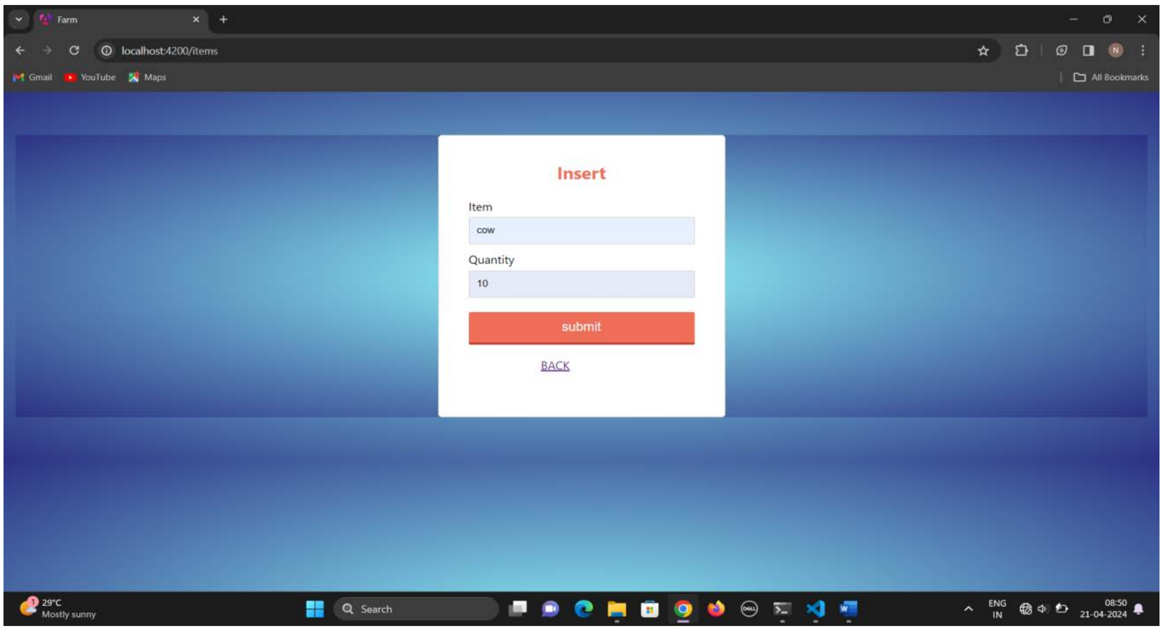
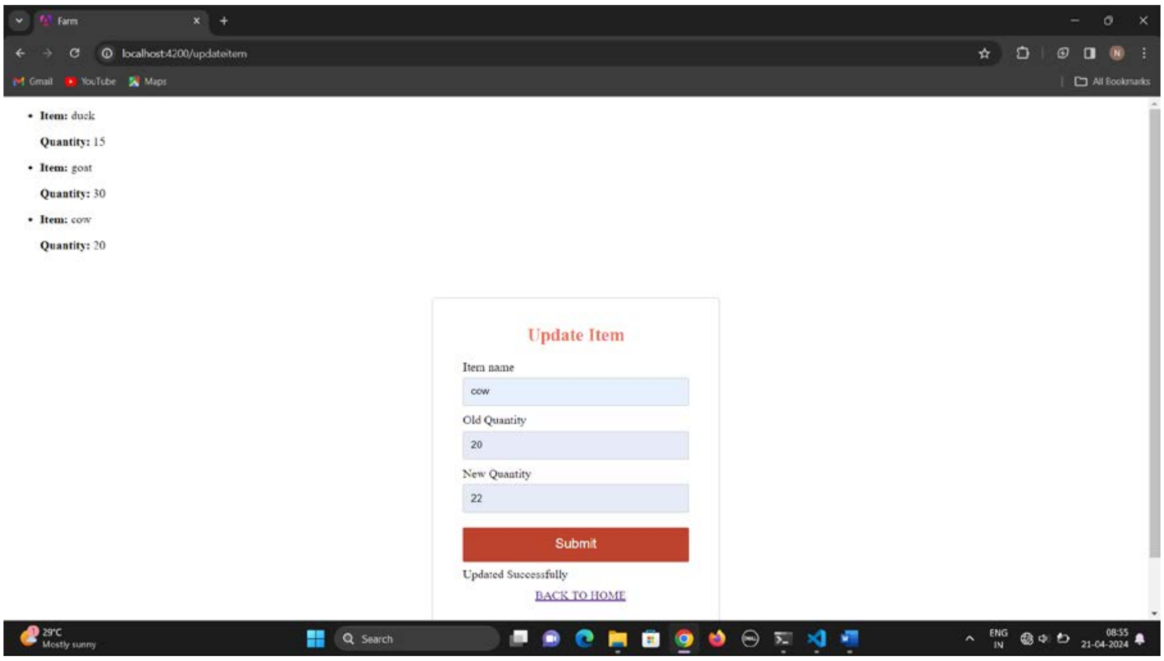


Figure 3.5.5: Add item layout of Farm Management System

3.5.6 Update Item page

Figures 3.5.6 provide the interface for Update Item page.



- **Item:** duck

  Quantity: 15
- **Item:** goat

  Quantity: 30
- **Item:** cow

  Quantity: 20

Update Item

Item name: cow

Old Quantity: 20

New Quantity: 22

Submit

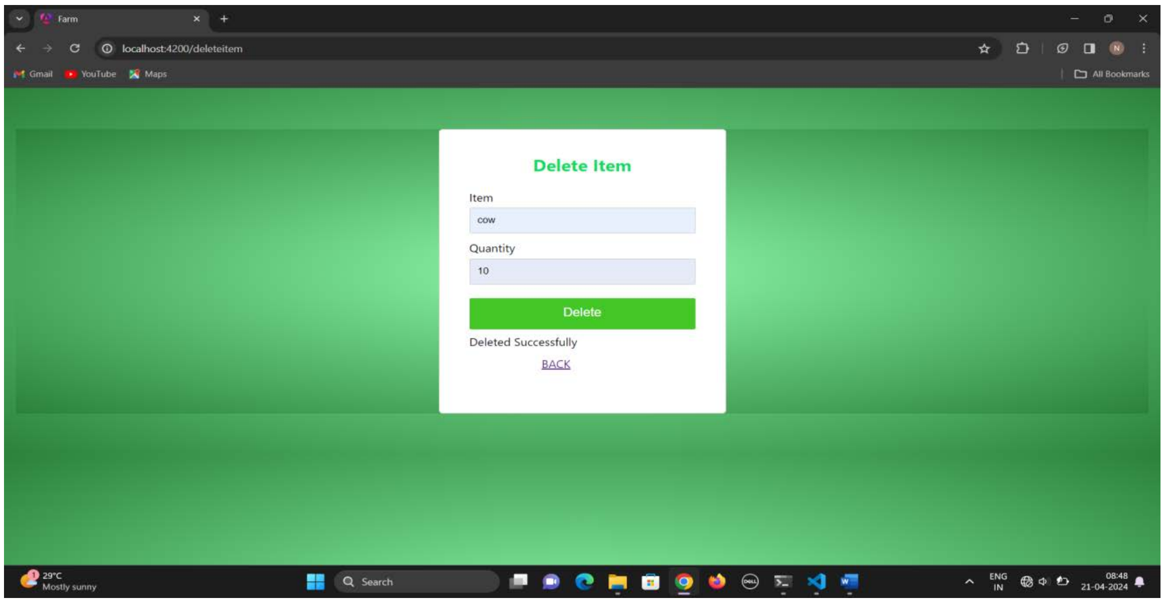
Updated Successfully

BACK TO HOME

Figure 3.5.6: Update Item layout of Farm Management System

3.5.7 Delete Item page

Figure 3.5.7 provide the interface for Delete Item page



Delete Item

Item: cow

Quantity: 10

Delete

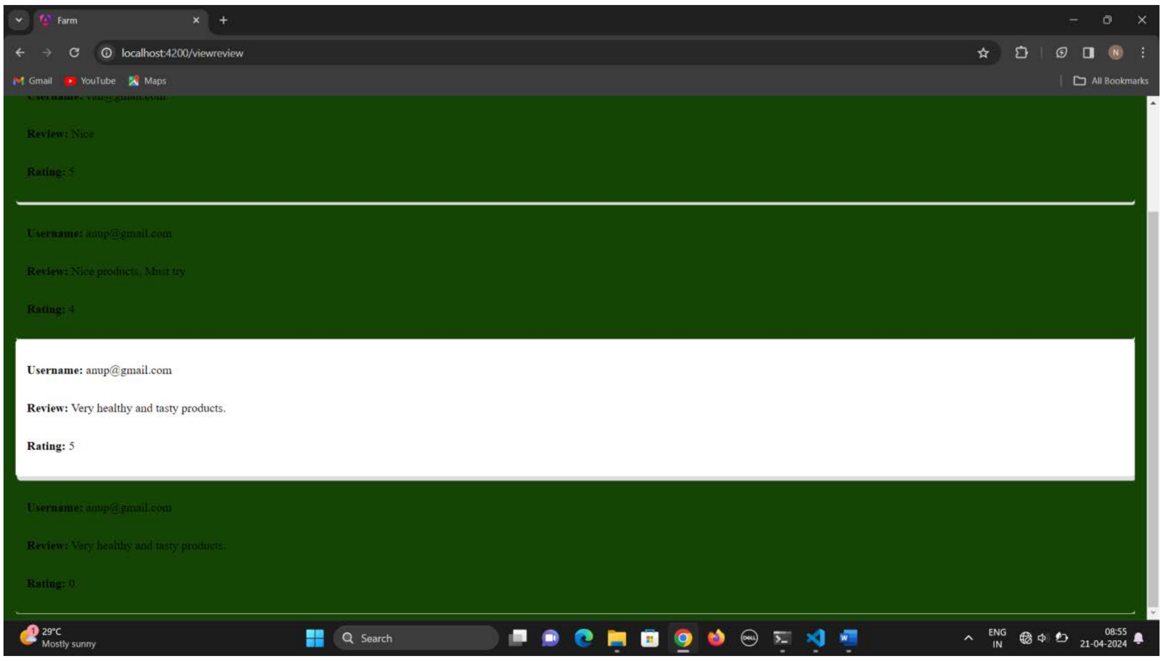
Deleted Successfully

BACK

Figure 3.5.7: Delete Item layout of Farm Management System

3.5.8 Overall Reviews page

Figure 3.5.8 provide the interface for Overall Reviews page



Review: Nice

Rating: 5

Username: anup@gmail.com

Review: Nice products, Must try

Rating: 4

Username: anup@gmail.com

Review: Very healthy and tasty products.

Rating: 5

Username: anup@gmail.com

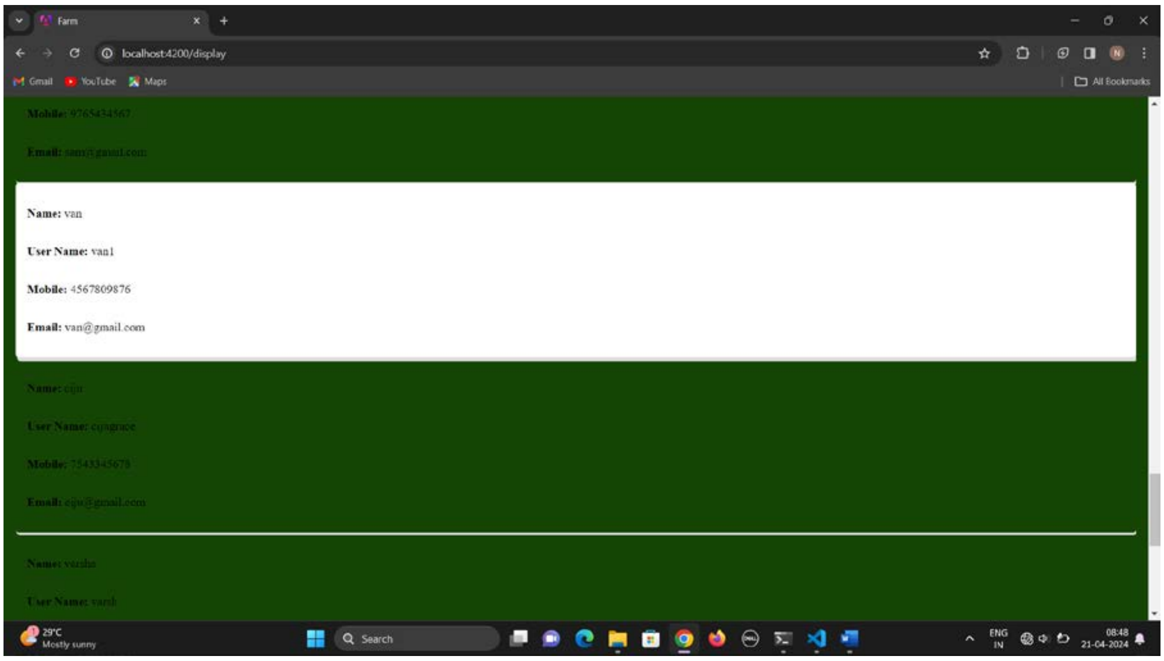
Review: Very healthy and tasty products.

Rating: 0

Figure 3.5.8 Overall Reviews layout of Farm Management System

3.5.9 Overall Customers page

Figure 3.5.9 provide the interface for Overall Customers page



Mobile: 9765434567

Email: sam@gmail.com

Name: van

User Name: van1

Mobile: 4567809876

Email: van@gmail.com

Name: ciju

User Name: cijugrace

Mobile: 7543345678

Email: ciju@gmail.com

Name: varsha

User Name: varsh

Figure 3.5.9: Overall Customers layout of Farm Management System

3.5.10 User Registration page

Figure 3.5.10 provide the interface for User Registration page

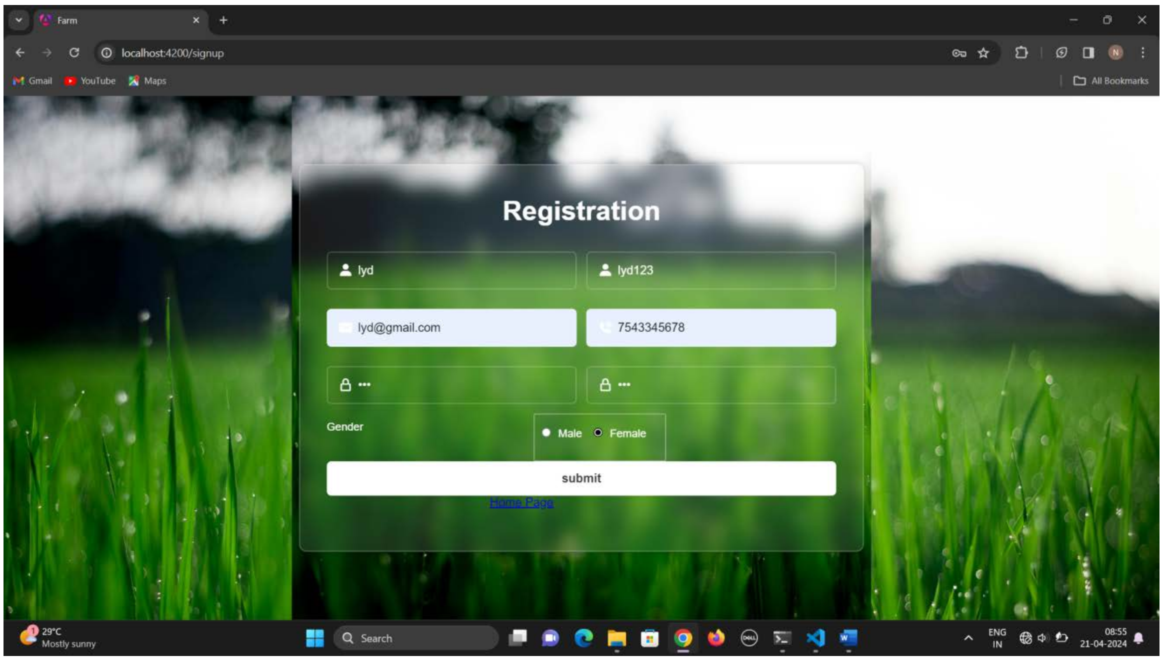


Figure 3.5.10: User Registration layout of Farm Management System

3.5.11 User Login page

Figure 3.5.11 provide the interface for User Login page

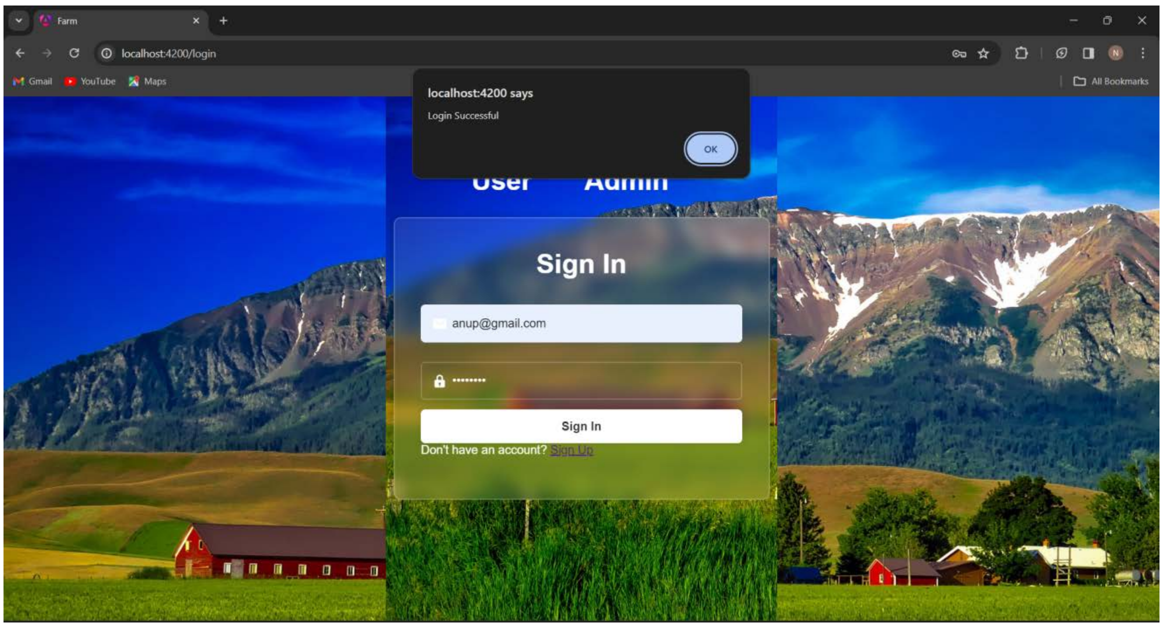


Figure 3.5.11: User Login Page

3.5.12 User Home page

Figure 3.5.12 provide the interface for User Home page

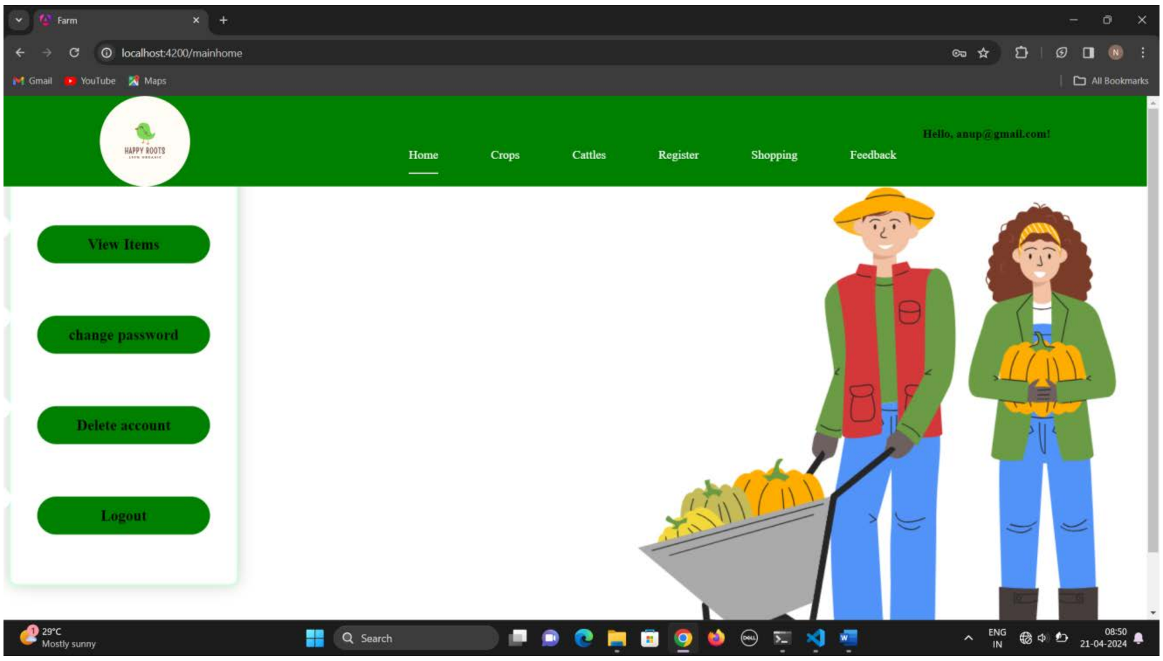


Figure 3.5.12: User Home layout of Farm Management System

3.5.13 Crops page

Figure 3.5.13 provide the interface for Crops page

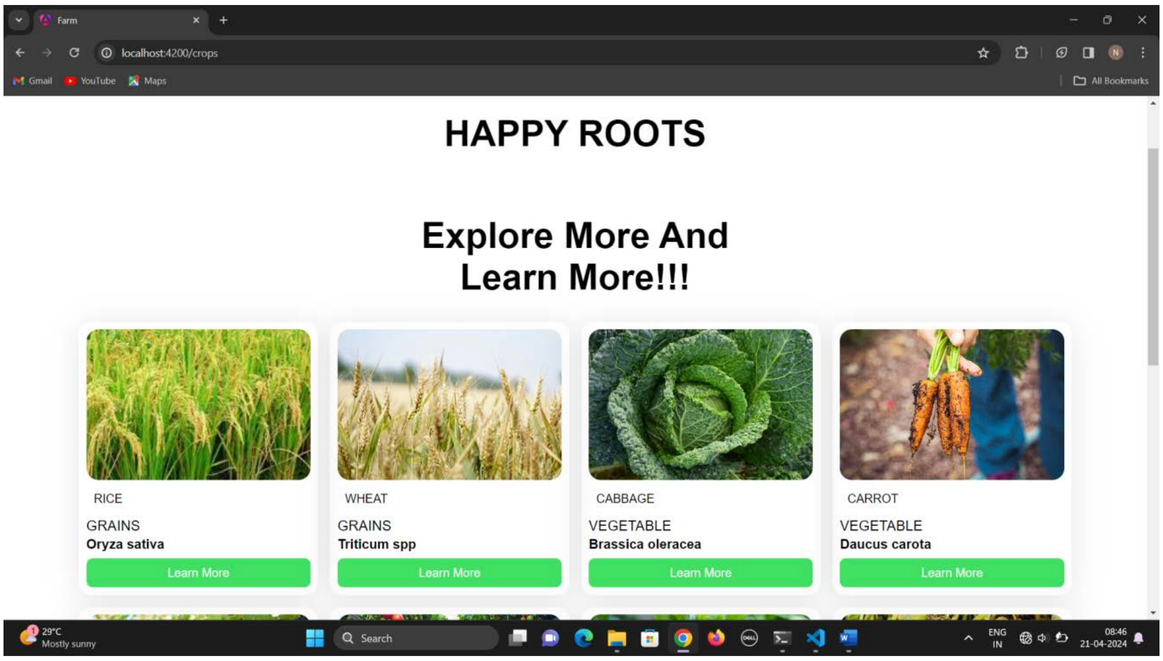
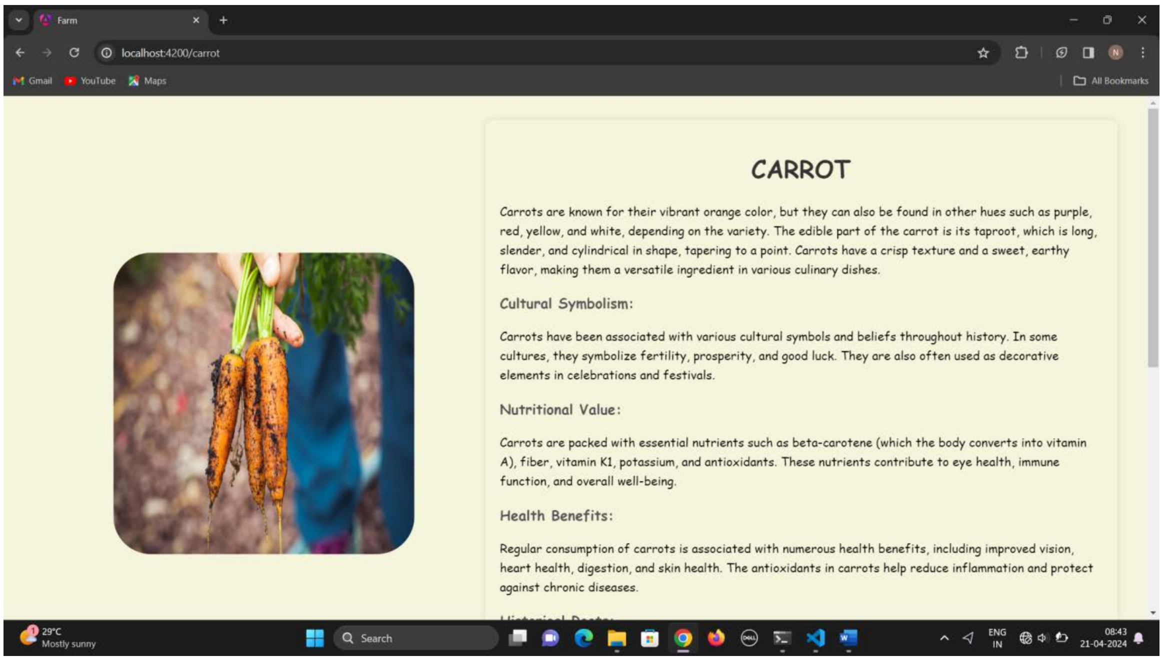


Figure 3.5.13: Crops layout of Farm Management System

3.5.14 Crop Description page

Figure 3.5.14 provide the interface for Crop Description page



CARROT

Carrots are known for their vibrant orange color, but they can also be found in other hues such as purple, red, yellow, and white, depending on the variety. The edible part of the carrot is its taproot, which is long, slender, and cylindrical in shape, tapering to a point. Carrots have a crisp texture and a sweet, earthy flavor, making them a versatile ingredient in various culinary dishes.

Cultural Symbolism:

Carrots have been associated with various cultural symbols and beliefs throughout history. In some cultures, they symbolize fertility, prosperity, and good luck. They are also often used as decorative elements in celebrations and festivals.

Nutritional Value:

Carrots are packed with essential nutrients such as beta-carotene (which the body converts into vitamin A), fiber, vitamin K1, potassium, and antioxidants. These nutrients contribute to eye health, immune function, and overall well-being.

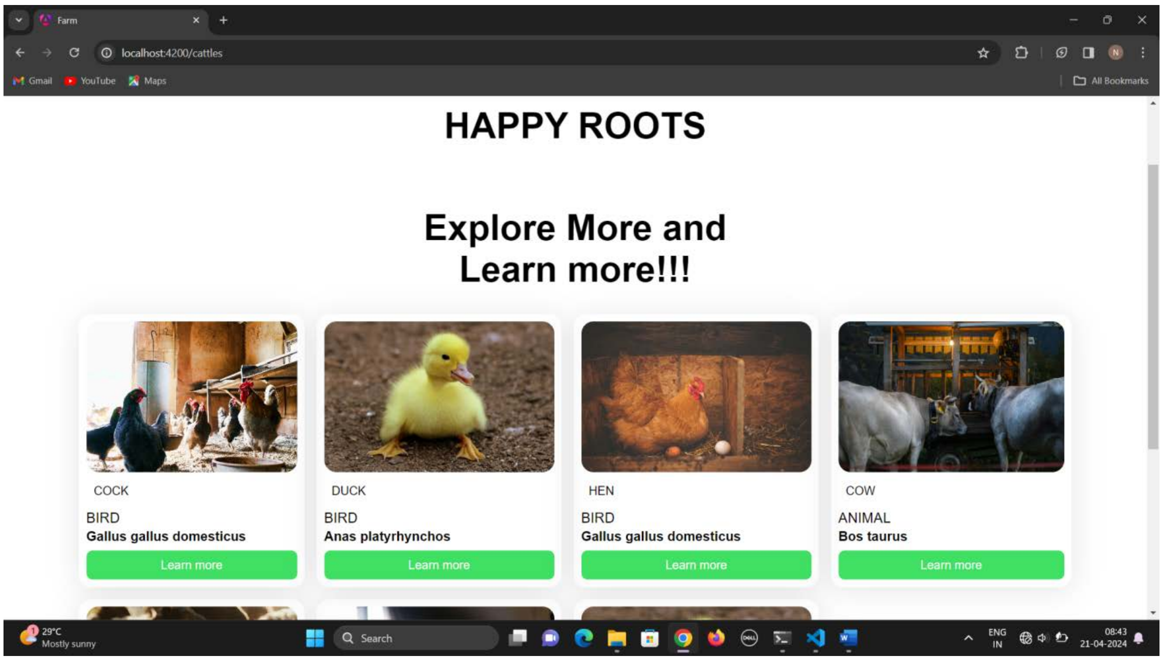
Health Benefits:

Regular consumption of carrots is associated with numerous health benefits, including improved vision, heart health, digestion, and skin health. The antioxidants in carrots help reduce inflammation and protect against chronic diseases.

Figure 3.5.14: Crop Description layout of Farm Management System

3.5.15 Cattles page

Figure 3.5.15 provide the interface for Cattles page



HAPPY ROOTS

Explore More and Learn more!!!

COCK
BIRD
Gallus gallus domesticus
Learn more

DUCK
BIRD
Anas platyrhynchos
Learn more

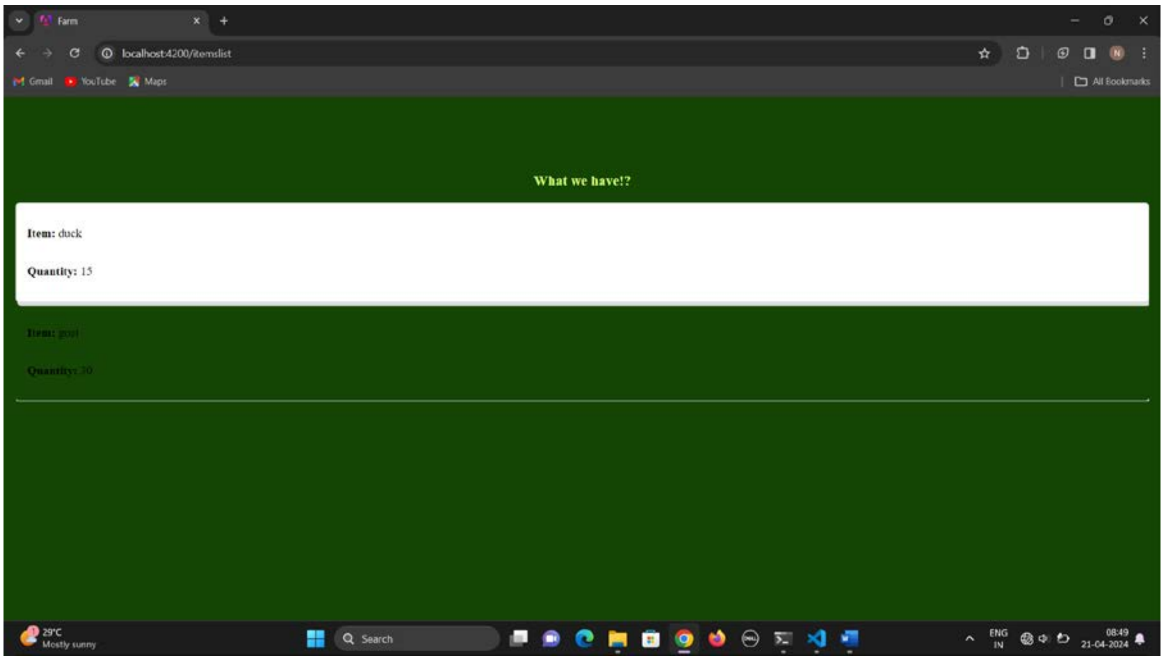
HEN
BIRD
Gallus gallus domesticus
Learn more

COW
ANIMAL
Bos taurus
Learn more

Figure 3.5.15: Cattles layout of Farm Management System

3.5.16 Cattles Quantity page

Figure 3.5.16 provide the interface for Cattles Quantity page



What we have!?

Item: duck

Quantity: 15

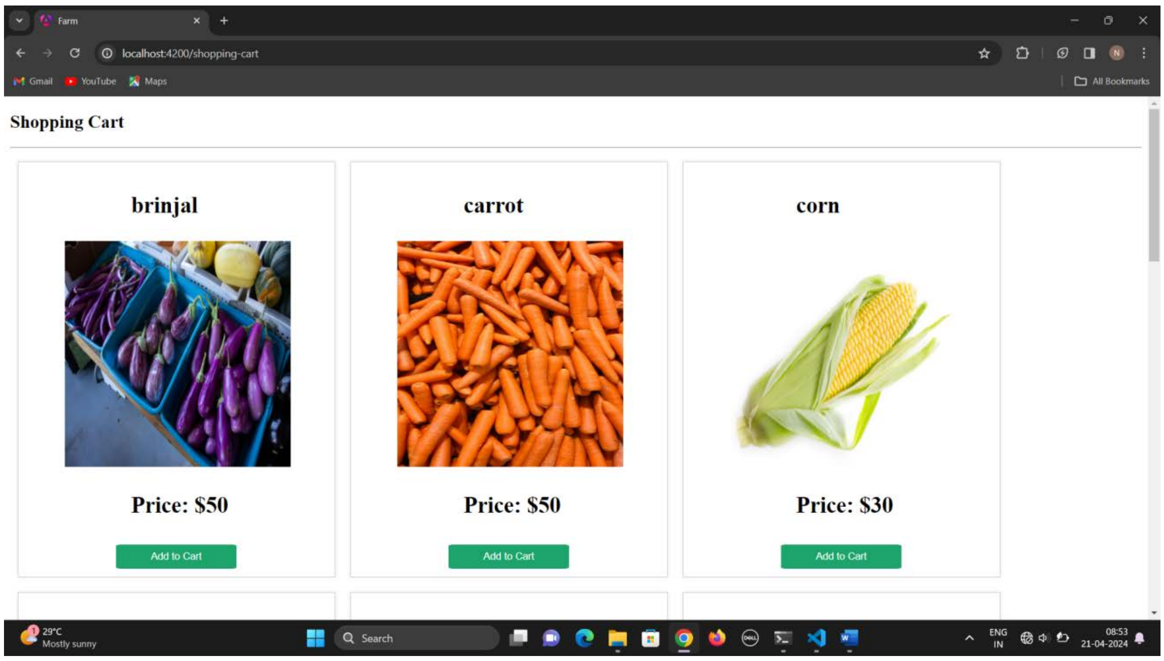
Item: goat

Quantity: 30

Figure 3.5.16: Cattles Quantity layout of Farm Management System

3.5.17 Shopping Cart page

Figure 3.5.17 provide the interface for Shopping Cart page



Shopping Cart

brinjal

Price: $50

carrot

Price: $50

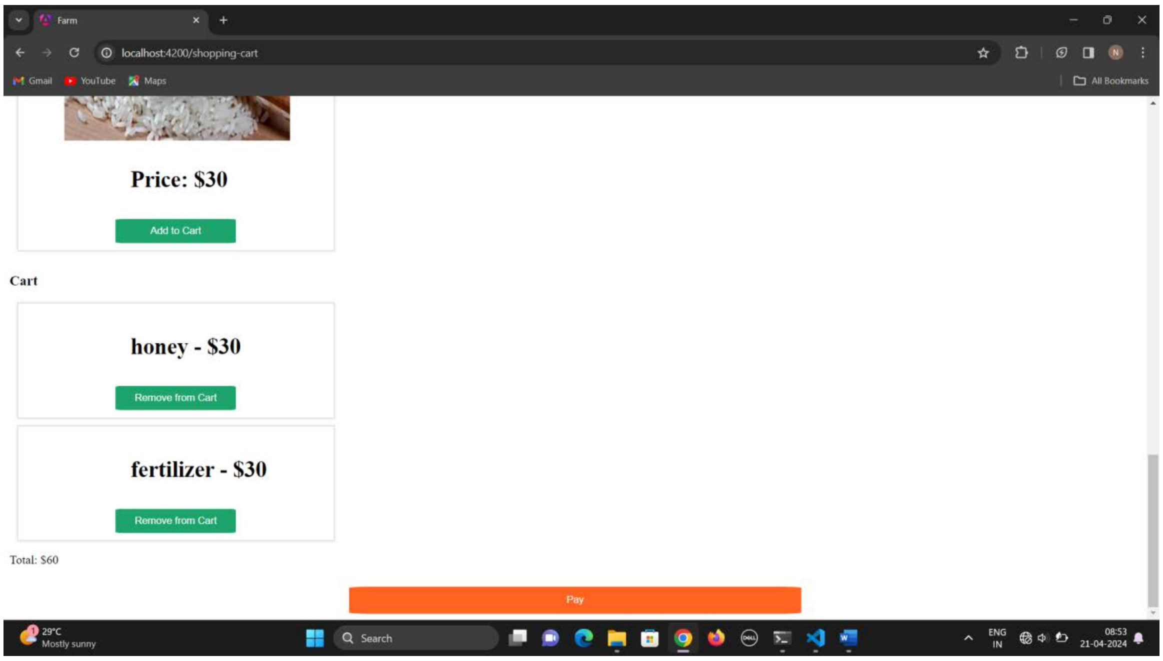
corn

Price: $30

Figure 3.5.17: Shopping Cart layout of Farm Management System

3.5.18 Total Amount page

Figure 3.5.18 provide the interface for Total Amount page



Price: $30

Cart

honey - $30

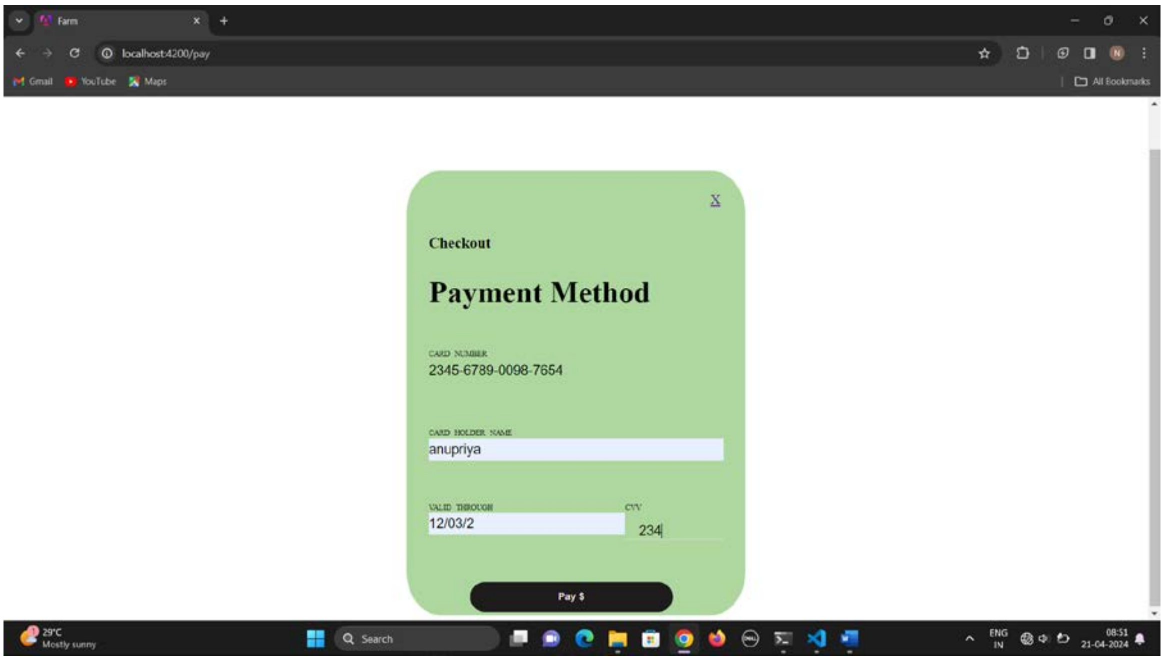
fertilizer - $30

Total: $60

Figure 3.5.18 Total Amount layout of Farm Management System

3.5.19 Payment Details Home page

Figure 3.5.19 provide the interface for Payment Details page



Checkout

Payment Method

CARD NUMBER
2345-6789-0098-7654

CARD HOLDER NAME
anupriya

VALID THROUGH
12/03/2

CVV
234

Figure 3.5.19: Payment Details layout of Farm Management System

3.5.20 Payment status page

Figure 3.5.20 provide the interface for Payment status page

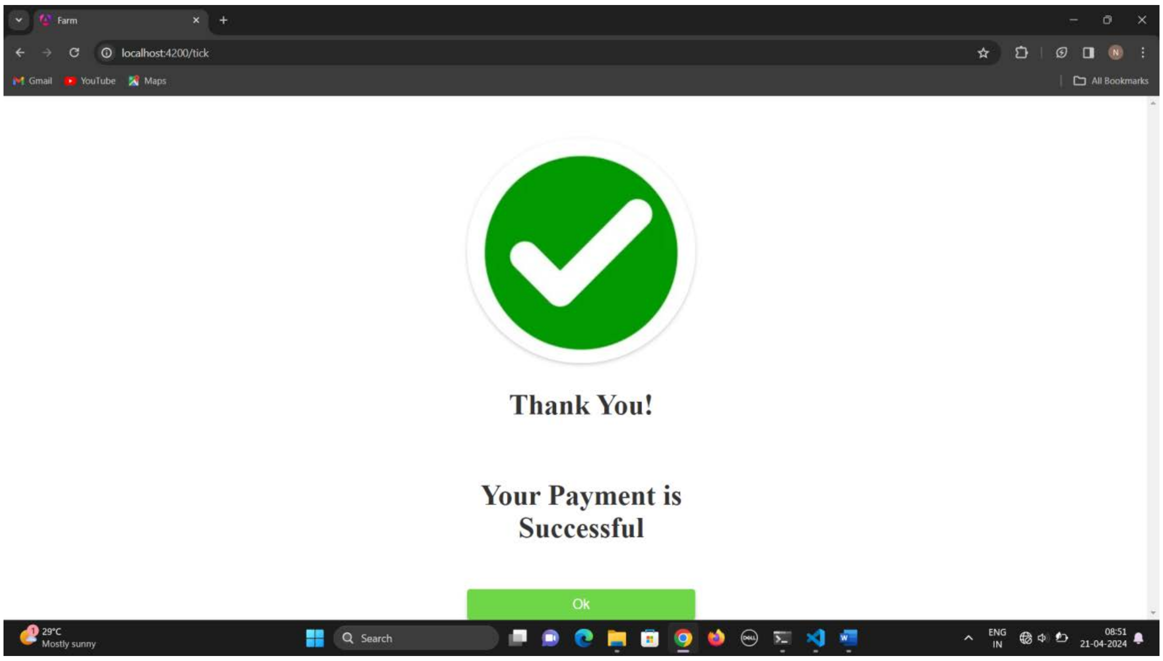
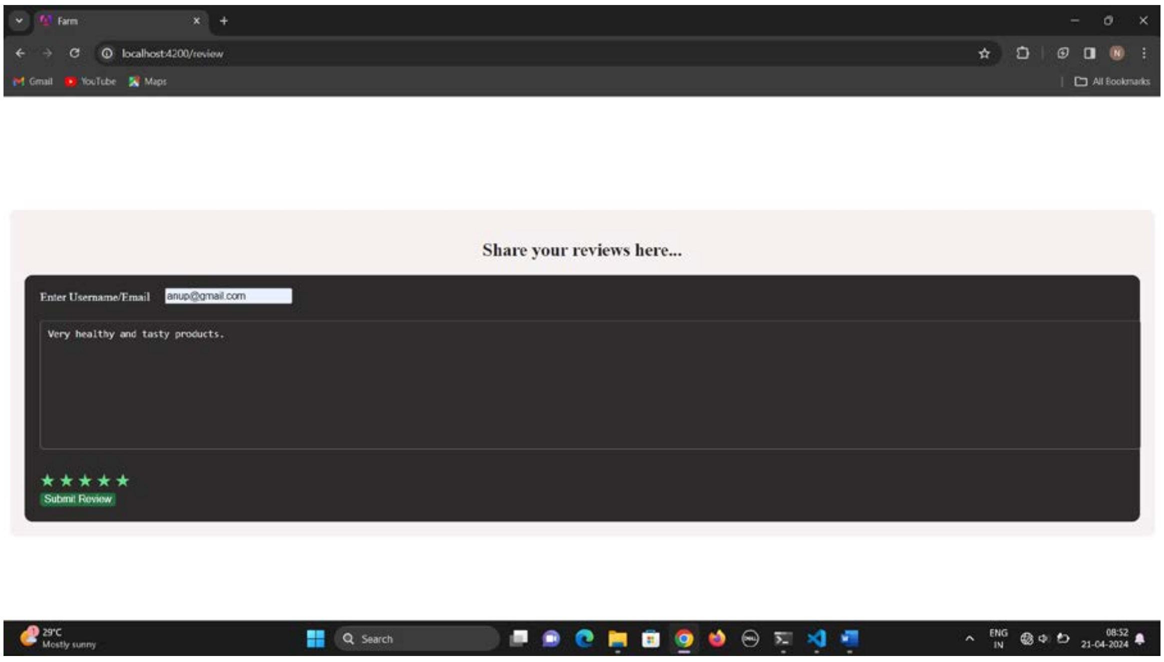


Figure 3.5.20: Payment status layout of Farm Management System

3.5.21 Review page

Figure 3.5.21 provide the interface for Review page



anup@gmail.com

Very healthy and tasty products.

★★★★★

Figure 3.5.21: Review layout of Farm Management System

3.5.22 Change Password page

Figure 3.5.22 provide the interface for Change Password page

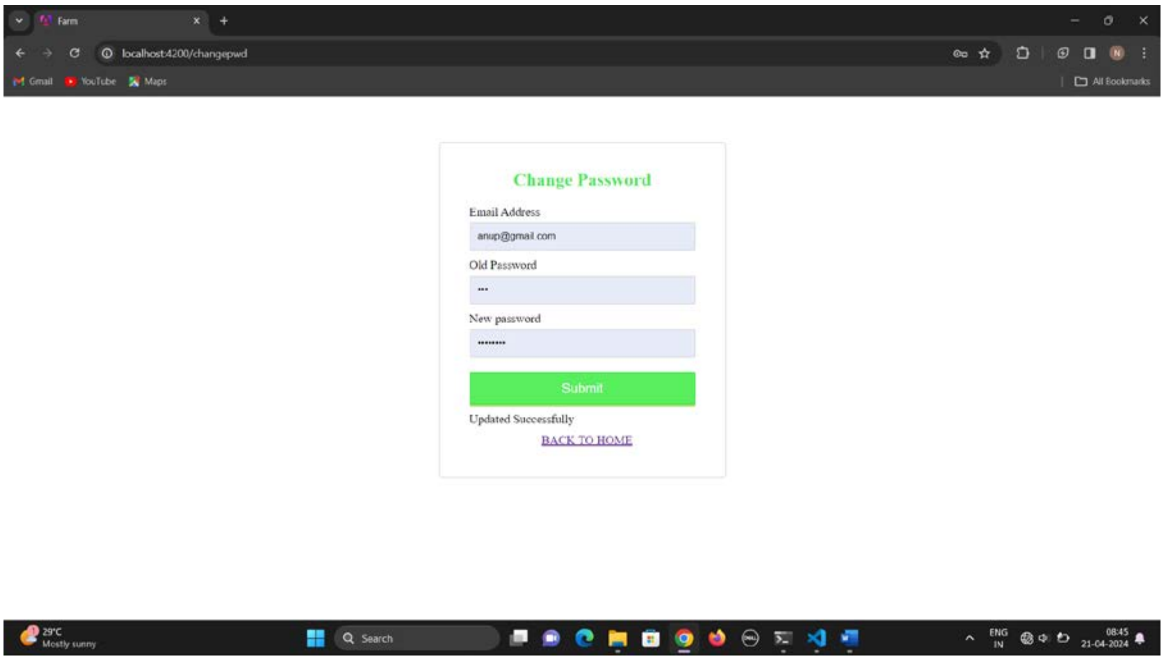


Figure 3.5.22: Change Password layout of Farm Management System

3.5.23 Deactivate Account page

Figure 3.5.23 provide the interface for Deactivate Account page

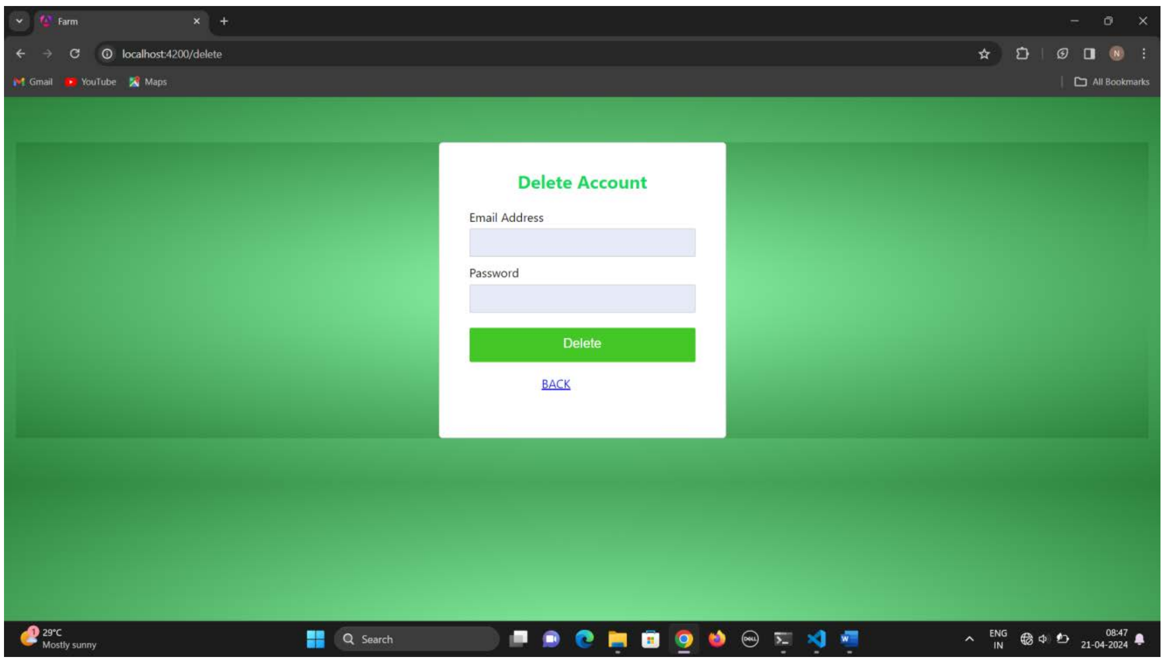


Figure 3.5.23: Deactivate Account layout of Farm Management System

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 ADMIN LOGIN IMPLEMENTATION

The login credentials are obtained. If the credentials are OK, then the user is redirected to the homepage

```
GET email id, password
   IF email id, password valid
          RETURN admin homepage
   ELSE
          TOAST Invalid Credential
```

4.1 USER LOGIN IMPLEMENTATION

The login credentials are obtained. If the credentials are OK, then the user is redirected to the homepage

```
GET email id, password
   IF email id, password valid
          RETURN user homepage
   ELSE
          TOAST Invalid Credential
```

4.2 USER SIGNUP IMPLEMENTATION

The form fields are obtained. If they are valid, then the user is added to the database.

```
GET requestFields
   IF requestFields valid
          RETURN added to db
   ELSE
          TOAST enter valid details
```

CHAPTER 5

RESULTS AND DISCUSSION

5.1 TEST CASES AND RESULTS

5.1.1 Test Cases and Results for Admin Login function:

The Table 5.1.1.1, Table 5.1.1.1 shows that the possible test data for the both positive and negative test case given below, if the admin is already having account then the output is true otherwise false.

Table 5.1.1.1: Positive Test Case and result for Admin Login

Test Case ID	TC1
Test Case Description	It tests whether the given login details are valid or not
Test Data	niranjanaa@gmail.com,niranju1
Expected Output	TRUE
Result	PASS

Table 5.1.1.2: Negative Test Case and result for Admin Login

Test Case ID	TC2
Test Case Description	It tests whether the given login details are valid or not
Test Data	niranju1
Expected Output	FALSE
Result	PASS

5.1.2 Test Cases and Results for User Login function:

The Table 5.1.2.1, Table 5.1.2.1 shows that the possible test data for the both positive and negative test case given below, if the user is already having account then the output is true otherwise false.

Table 5.1.2.1: Positive Test Case and result for User Login

Test Case ID	TC3
Test Case Description	It tests whether the given login details are valid or not
Test Data	anup@gmail.com,anupriya1
Expected Output	TRUE
Result	PASS

Table 5.1.2.2: Negative Test Case and result for User Login

Test Case ID	TC4
Test Case Description	It tests whether the given login details are valid or not
Test Data	anupriya1
Expected Output	FALSE
Result	PASS

5.1.3 Test Cases and Results for User Signup function:

The Table 5.1.3.1, Table 5.1.3.1 shows that the possible test data for the both positive and negative test case given below, if the user have no already having account then the output is true otherwise false.

Table 5.1.3.1: Positive Test Case and result for User Signup

Test Case ID	TC5
Test Case Description	It tests whether the given user signup details are valid or not
Test Data	Lyd,lyd23,lyd123@gmail.com,9876543210,lyd1,lyd1,female
Expected Output	TRUE
Result	PASS

Table 5.1.3.2: Negative Test Case and result for User Signup

Test Case ID	TC6
Test Case Description	It tests whether the given user signup details are valid or not
Test Data	Lyd,lyd23,lyd123@gmail.com,9876543210,lyd1,lyd1
Expected Output	FALSE
Result	PASS

5.1.4 Test Cases and Results for Add Item function:

The Table 5.1.4.1, Table 5.1.4.1 shows that the possible test data for the both positive and negative test case given below, if the item is unavailable then the output is true otherwise false.

Table 5.1.4.1: Positive Test Case and result for Add Item

Test Case ID	TC7
Test Case Description	It tests whether the given Add item details are valid or not
Test Data	Duck,10
Expected Output	TRUE
Result	PASS

Table 5.1.4.2: Negative Test Case and result for Add Item

Test Case ID	TC8
Test Case Description	It tests whether the given add item details are valid or not
Test Data	Duck
Expected Output	FALSE
Result	PASS

5.1.5 Test Cases and Results for Update Item function:

The Table 5.1.5.1, Table 5.1.5.1 shows that the possible test data for the both positive and negative test case given below, if the item is available then the output is true otherwise false.

Table 5.1.5.1: Positive Test Case and result for Update Item

Test Case ID	TC9
Test Case Description	It tests whether the given update item details are valid or not
Test Data	Duck,10,20
Expected Output	TRUE
Result	PASS

Table 5.1.5.2: Negative Test Case and result for Update Item

Test Case ID	TC10
Test Case Description	It tests whether the given update item details are valid or not
Test Data	Duck,20
Expected Output	FALSE
Result	PASS

5.1.6 Test Cases and Results for Delete Item function:

The Table 5.1.6.1, Table 5.1.6.1 shows that the possible test data for the both positive and negative test case given below, if the item is available then the output is true otherwise false.

Table 5.1.6.1: Positive Test Case and result for Delete Item

Test Case ID	TC11
Test Case Description	It tests whether the given delete item details are valid or not
Test Data	Duck,10
Expected Output	TRUE
Result	PASS

Table 5.1.6.2: Negative Test Case and result for Delete Item

Test Case ID	TC12
Test Case Description	It tests whether the given delete item details are valid or not
Test Data	Duck
Expected Output	FALSE
Result	PASS

5.1.7 Test Cases and Results for Change password function:

The Table 5.1.7.1, Table 5.1.7.1 shows that the possible test data for the both positive and negative test case given below, if the user already have an account then the output is true otherwise false.

Table 5.1.7.1: Positive Test Case and result for Change password

Test Case ID	TC13
Test Case Description	It tests whether the given user details are valid or not
Test Data	anup@gmail.com,anupriya1,anu12
Expected Output	TRUE
Result	PASS

Table 5.1.7.2: Negative Test Case and result for Change password

Test Case ID	TC14
Test Case Description	It tests whether the given user details are valid or not
Test Data	anup@gmail.com,anupriya12
Expected Output	FALSE
Result	PASS

5.1.6 Test Cases and Results for Deactivate Account function:

The Table 5.1.8.1, Table 5.1.8.1 shows that the possible test data for the both positive and negative test case given below, if the user already have an account then the output is true otherwise false.

Table 5.1.8.1: Positive Test Case and result for Deactivate Account

Test Case ID	TC15
Test Case Description	It tests whether the given user details are valid or not
Test Data	anup@gmail.com, anu12
Expected Output	TRUE
Result	PASS

Table 5.1.8.2: Negative Test Case and result for Deactivate Account

Test Case ID	TC16
Test Case Description	It tests whether the given user details are valid or not
Test Data	anup@gmail.com
Expected Output	FALSE
Result	PASS

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT(S)

The Farm Management System revolutionizes agricultural management by providing users with a seamless shopping experience for crops while empowering administrators with efficient inventory control. By enabling users to view crop benefits and complete purchases securely, Farm Management System the promotes transparency and convenience in agricultural transactions. With the ability for administrators to maintain cattle quantities and oversee inventory, the system enhances operational efficiency and facilitates informed decision-making. Incorporating features to promote sustainable farming practices, such as highlighting eco-friendly farming methods or offering organic product options, can attract environmentally-conscious consumers.

APPENDIX – A

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENT:

Server :	Any Server with 10TB storage and 128 GB of memory
Processor:	Intel Core i3 10th Gen or above
Memory :	Minimum 8 GB
Storage :	Minimum 256 GB of Disk Storage
Network :	Minimum 10MBPS Bandwidth

SOFTWARE REQUIREMENT:

Operating System :	Any
DBMS :	MongoDB
IDE used :	Angular ,Visual Studio
Angular Version :	17

APPENDIX – B

SOURCE CODE

About.html

```
<body>
<div class="wrapper">
    <nav class="nav">
        <div class="nav-logo">
            <img src='../../assets/Happy Roots (1).png' alt="HAPPY ROOTS">
        </div>
        <div class="nav-menu">
            <ul>
                <li><a routerLink="/home" class="link active">Home</a></li>
            </ul>
        </div>
        <div class="nav-menu">
            <ul>
                <li><a routerLink="/items" class="link">Add items</a></li>
            </ul>
        </div>
        <div class="nav-menu">
            <ul>
                <li><a routerLink="/updateitem" class="link">Update items</a></li>
            </ul>
        </div>
        <div class="nav-menu">
            <ul>
                <li><a routerLink="/deleteitem" class="link">Delete items</a></li>
            </ul>
        </div>
        <div class="nav-button">
            <button class="btn signup-btn" id="logoutbtn"
onclick="window.open('adminlogin','mywindow')" >Logout</button>
        </div>
        <div class="nav-menu-btn">
            <i class="bx bx-menu"></i>
        </div>
    </nav>
    </div>
</body>
```

About.css

```
*{
    margin:0;
    padding:0;
    box-sizing:border-box;
    font-family: 'Times New Roman';}
body{
    background-size: auto;
    background-repeat:no-repeat;
    background-attachment:fixed;
    background-position: bottom;}
.wrapper{
    display:flex;
    justify-content:center;
    align-items: center;
    min-height: 110vh;}
.nav{
    position: fixed;
    top: 0;
    display: flex;
    justify-content: space-around;
    width: 100%;
    height: 100px;
    line-height: 100px;
    z-index: 100;
    background: green;}
 .nav-logo img {
    color: white;
    font-size: 25px;
    font-weight: 600;
    width: 100px;
    border-radius: 70px;
    justify-content:end;}
 .nav-menu ul{
    display: flex;}
.nav-menu ul li{
    list-style-type: none;}
.nav-menu ul li .link{
    text-decoration: none;
    font-weight: 100;
    font-size: small;
    color:white;
    padding-bottom: 15px;
    margin: 0 35px;}
.link:hover, .active{
    border-bottom: 2px solid #fff;}
 .nav-button .btn{
    width: 130px;
    height: 40px;
    font-weight: 500;
    background: rgba(39, 39, 39, 0.485);
    border: none;
    border-radius: 30px;
    cursor: pointer;
    transition: 3s ease;}
#signupbtn{
    margin-left: 15px;}
 .btn:hover{
    border: rgba(255, 255 ,255, 0.7);}
.btn.white-btn{
    background: rgba(255, 255 ,255, 0.7);
    font-weight: 700;}
 .btn.btn.white-btn:hover{
    background: rgba(255, 255 ,255, 0.5);}
.btn.signup-btn{
    background: rgba(255, 255 ,255, 0.7);
    font-weight: 700;}
 .btn.btn.signup-btn:hover{
    background: rgba(255, 255 ,255, 0.5);}
.nav-menu-btn{
    display: none;}
.footer {
    background-color: #333;
    text-align: center;
    padding: 10px 0;  }
```

About.ts

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-adminhome',
  templateUrl: './adminhome.component.html',
  styleUrl: './adminhome.component.css'})
export class AdminhomeComponent {}
```

Adminlogin.html

```
<link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css' rel='stylesheet'>
<body>
        <div class="wrapper">
        <nav class="nav">
        <div class="nav-menu">
            <ul>
                <li><a routerLink="/login" class="link active"  >User</a></li>
                <li><a routerLink="/adminlogin" class="link"  >Admin</a></li>
            </ul>
        </div>
        </nav>
        <form #form="ngForm" (ngSubmit)="onSubmit(form)" >
            <h1>Admin Sign In</h1>
            <div class="input-box">
            <div class="input-field">
               <input type="email" placeholder="Email" name="email"required ngModel
#email="ngModel">
               <i class='bx bxs-envelope'></i>
            </div>
         </div>
         <div class="input-box">
            <div class="input-field">
                  <input type="password" placeholder="Password" name="pw" required ngModel
#pw="ngModel">
                  <i class='bx bxs-lock'></i>
            </div>
         </div>
         <button type="submit"  class="btn">Sign In</button>
      </form>
   </div>
</body>
```

<u>Adminlogin.css</u>

```
@import
url("https://fonts.googleapis.com/css2?family=Popins:wght@300;400;500;600;700;800;900&disp
lay=swap");
* {
margin: 0;
padding: 0;
box-sizing: border-box;
font-family: "Poppins", sans-serif;}
body {
   display: flex;
   justify-content: center;
   align-items: center;
   min-height: 100vh;
   background-image:url('../../assets/img.jpg');
   background-size: cover;}
.wrapper {
   width: 500px;
   background: rgba(255, 255, 255, .1);
   border: 2px solid rgba(255, 255, 255, .2);
   box-shadow: 0 0 10px rgba(0, 0, 0, .2);
   backdrop-filter: blur(10px);
   border-radius: 10px;
   color: #fff;
   padding: 40px 35px 55px;
   margin: 0 10px;}
.wrapper h1{
   font-size: 36px;
   text-align: center;
   margin-bottom: 20px;}
```

```
.wrapper .input-box {
    display: flex;
    justify-content: center;
    flex-wrap: wrap;
    background: transparent;
    width: 100%;}
.input-box .input-field {
    position: relative;
    justify-content: center;
    width: 100%;
    height: 50px;
    margin: 13px 0;}
.input-box .input-field input{
    width: 100%;
    height: 100%;
    background: transparent;
    border: 2px solid rgba(255, 255, 255, .2);
    outline: none;
    font-size: 16px;
    color: #fff;
    border-radius: 6px;
    padding: 15px 15px 15px 40px;}
.input-box .input-field input::placeholder {
    color: #fff;}
.input-box .input-field i {
    position: absolute;
    left: 15px;
    top: 25px;
    transform: translateY(-50%);
    font-size: 20px;}
.wrapper label {
    display: inline-block;
     margin: 10px 0 23px;}
.wrapper label input {
    accent-color: #000;
    margin-right: 5px;}
.wrapper .btn {
    width: 100%;
    height: 45px;
    background: #fff;
    border: none;
    outline: none;
    border-radius: 6px;
    box-shadow: 0 0 10px rgba(0, 0, 0, .1);
    cursor: pointer;
    font-size: 16px;
    color: #333;
    font-weight: 600;}
.nav{
    position: fixed;
    top: 0;
    display: flex;
    justify-content: center;
    width: 100%;
    height: 100px;
    line-height: 100px;
    z-index: 100;
    transform: translate(-50px, -100px);}
.nav-menu ul{
    display: flex;
    font-size: 36px;}
.nav-menu ul li{
    list-style-type: none;
    font-weight: bold;}
.nav-menu ul li .link{
    text-decoration: none;
    font-weight: bold;
    font-size: 36px;
    color:white;
    padding-bottom: 15px;
    margin: 0 35px;}
```

Adminlogin.ts

```
 import { Component } from '@angular/core';
import { NodeutilityService } from '../nodeutility.service';
import { Router } from '@angular/router';
@Component({
  selector: 'app-adminlogin',
  templateUrl: './adminlogin.component.html',
  styleUrl: './adminlogin.component.css'})
export class AdminloginComponent {
  msg:string="";
   user1:string | null="";
  onSubmit(form: any) {
    this.util.insert2(form.value.email, form.value.pw).subscribe((data) => {
        if (data.status){
          localStorage.setItem("user1",form.value.email);
          this.msg = data.message;
          alert(this.msg);        }
        else{
          this.msg = data.message;        }      });  }}
```

Changepassword.html

```
<form #form="ngForm" (ngSubmit)="onSubmit(form)">
```

```
    <h2>Forgot Password</h2>
    <div class="box">
        <label>Email Address</label>
        <input type="text" name="email" id="email" ngModel #email="ngModel">
        <div class="error" id="emailErr"></div>
    </div>
    <div class="box">
        <label>Old Password</label>
        <input type="password" name="oldpw"  id="oldpw" ngModel #oldpw="ngModel">
        <div class="error" id="mobileErr"></div>
    </div>
    <div class="box">
        <label>New password</label>
        <input type="password" name="newpw" id="newpw" ngModel #newpw="ngModel">
    </div>
    <div class="box">
        <input type="submit" value="Submit">
    </div>
    <div  class="box" >{{msg}}</div>
    <div>
        <a routerLink="/home" style="padding-left: 32%;">BACK TO HOME</a>
    </div>
</form>
```

Changepassword.css

```
body {
    font-size: 16px;
    font-family: system-ui;
    background:-webkit-radial-gradient(#90F6FB,#2C3283);}
.form{
    padding-top: 50%;
    width: 100%;}
h2 {
    text-align: center;
    color: #ef6e58;}
form {
    width: 300px;
    padding: 15px 40px 40px;
    border: 1px solid #ccc;
    border-radius: 5px;
    background-color: #fff;
    margin: 4% auto;}
label {
    display: block;
    margin-bottom: 5px;}
label i {
    color: #999;
```

```
   font-size: 80%;}
input, select {
   border: 1px solid #ccc;
   padding: 10px;
   display: block;
   width: 100%;
   box-sizing: border-box;
   border-radius: 2px;
   background-color: #a9bae64a;}
.box {
   padding-bottom: 10px;}
.form-inline {
   border: 1px solid #ccc;
   padding: 8px 10px 4px;
   border-radius: 2px;}
.form-inline label, .form-inline input {
   display: inline-block;
   width: auto;
   padding-right: 15px;}
.error {
   color: red;
   font-size: 90%;}
input[type=text]{
      outline: none;}
input[type="submit"] {
   font-size: 110%;
   font-weight: 100;
   background: #ef6e58;
   border-color: #ef6e58;
   box-shadow: 0 3px 0 #bd432e;
   color: #fff;
   margin-top: 10px;
   cursor: pointer;}
input[type="submit"]:hover {
   background: #bd432e;
   border-color: #bd432e;}
.input-1{
   background-repeat: no-repeat;
   background-position: right 10px center;
   border:1px solid #01cc40;}
.input-2{
   background-repeat: no-repeat;
   background-position: right 10px center;
   border:1px solid red;}
.input-3{
   border:1px solid #01cc40!important;}
```

```
select{
   outline: none!important;}
.input-4{
   border:1px solid red;}
```

<u>Changepassword.ts</u>

```
import { Component } from '@angular/core';
import { NodeutilityService } from '../nodeutility.service';
import { Router } from '@angular/router';
@Component({
  selector: 'app-changepwd',
  templateUrl: './changepwd.component.html',
  styleUrl: './changepwd.component.css'})
export class ChangepwdComponent {
  constructor(private util:NodeutilityService,private router:Router) { }
  msg:string='';
  onSubmit(form: any) {
    this.util.update(form.value.email, form.value.oldpw,form.value.newpw).subscribe((data) => {
        if (data.status) {
          this.msg = data.message;        }
        else{
          this.msg = data.message;        }      });}}
```

<u>Delete.html</u>

```
<body [style.background-image]="'url(' + imageUrl + ')'" >
   <form>
      <h2>Delete Account</h2>
      <div class="box">
         <label>Email Address</label>
         <input type="text" name="email" id="email" [(ngModel)]="email">
         <div class="error" id="emailErr"></div>
      </div>
      <div class="box">
         <label>Password</label>
         <input type="password" name="pw"  id="pw" [(ngModel)]="pw">
         <div class="error" id="mobileErr"></div>
      </div>
     <div class="box">
         <input type="submit" value="Delete" (click)="delete()">
      </div>
      <div class="box">{{msg}}</div>
      <div>
         <a routerLink="/admin" style="padding-left: 32%;">BACK</a>
         <br><br>
      </div>
   </form>
</body>
```

<u>Delete.css</u>

```
body {
   font-size: 16px;
   font-family: system-ui;
   background:-webkit-radial-gradient(#90F6FB,#2C3283);}
.form{
   padding-top: 50%;
   width: 100%;}
h2 {
   text-align: center;
   color: #ef6e58;}
form {
   width: 300px;
   padding: 15px 40px 40px;
   border: 1px solid #ccc;
   border-radius: 5px;
   background-color: #fff;
   margin: 4% auto;}
label {
   display: block;
   margin-bottom: 5px;}
label i {
   color: #999;
   font-size: 80%;}
input, select {
   border: 1px solid #ccc;
   padding: 10px;
   display: block;
   width: 100%;
   box-sizing: border-box;
   border-radius: 2px;
   background-color: #a9bae64a;}
.box {
   padding-bottom: 10px;}
.form-inline {
   border: 1px solid #ccc;
   padding: 8px 10px 4px;
   border-radius: 2px;}
.form-inline label, .form-inline input {
   display: inline-block;
   width: auto;
   padding-right: 15px;}
.error {
   color: red;
   font-size: 90%;}
input[type=text]{
      outline: none;}
```

```
input[type="submit"] {
    font-size: 110%;
    font-weight: 100;
    background: #ef6e58;
    border-color: #ef6e58;
    box-shadow: 0 3px 0 #bd432e;
    color: #fff;
    margin-top: 10px;
    cursor: pointer;}
input[type="submit"]:hover {
    background: #bd432e;
    border-color: #bd432e;}
.input-1{
    background-repeat: no-repeat;
    background-position: right 10px center;
    border:1px solid #01cc40;}
.input-2{
    background-repeat: no-repeat;
    background-position: right 10px center;
    border:1px solid red;}
.input-3{
    border:1px solid #01cc40!important;}
select{
    outline: none!important;}
.input-4{
    border:1px solid red;}
```

Delete.ts

```
import { Component } from '@angular/core';
import { NodeutilityService } from '../nodeutility.service';
import { Router } from '@angular/router';
@Component({
  selector: 'app-delete',
  templateUrl: './delete.component.html',
  styleUrl: './delete.component.css'})
export class DeleteComponent {
  constructor(private util:NodeutilityService,private router:Router) { }
  imageUrl='-webkit-radial-gradient(#90F6FB,#2C3283)';
  email:string='';
  pw:string="";
  msg:string="";
  delete(){
    this.util.delete(this.email,this.pw).subscribe((data)=>{
      if(data.status){
        this.msg=data.message;      }
      else{
        this.msg=data.message;     }     });  }}
```

Deleteitem.html

```
<body >
   <form>
      <h2>Delete Item</h2>
      <div class="box">
         <label>Item</label>
         <input type="text" name="item" id="item" [(ngModel)]="item">
      </div>
      <div class="box">
         <label>Quantity</label>
         <input type="text" name="quantity"  id="quantity" [(ngModel)]="quantity">
      </div>
      <div class="box">
         <input type="submit" value="Delete" (click)="delete()">
      </div>
      <div class="box">{{msg}}</div>
      <div>
         <a routerLink="/admin" style="padding-left: 32%;">BACK</a>
         <br><br>
      </div>
   </form></body>
```

Deleteitem.css

```
body {
   font-size: 16px;
   font-family: system-ui;
   background:-webkit-radial-gradient(#90F6FB,#2C3283); }
.form{
   padding-top: 50%;
   width: 100%;}
h2 {
   text-align: center;
   color: #ef6e58;}
form {
   width: 300px;
   padding: 15px 40px 40px;
   border: 1px solid #ccc;
   border-radius: 5px;
   background-color: #fff;
   margin: 4% auto;}
label {
   display: block;
   margin-bottom: 5px;}
label i {
   color: #999;
   font-size: 80%;}
input, select {
   border: 1px solid #ccc;
   display: block;
   width: 100%;
   box-sizing: border-box;
   border-radius: 2px;
   background-color: #a9bae64a;}
.box {
   padding-bottom: 10px;}
.form-inline {
   border: 1px solid #ccc;
   padding: 8px 10px 4px;
   border-radius: 2px;}
.form-inline label, .form-inline input {
   display: inline-block;
   width: auto;
   padding-right: 15px;}
.error {
   color: red;
   font-size: 90%;}
input[type=text]{
        outline: none;}
input[type="submit"] {
   font-size: 110%;
   font-weight: 100;
   background: #ef6e58;
   border-color: #ef6e58;
   box-shadow: 0 3px 0 #bd432e;
   color: #fff;
   margin-top: 10px;
   cursor: pointer;}
input[type="submit"]:hover {
   background: #bd432e;
   border-color: #bd432e;}
.input-1{
   background-repeat: no-repeat;
   background-position: right 10px center;
   border:1px solid #01cc40;}
.input-2{
   background-repeat: no-repeat;
   background-position: right 10px center;
   border:1px solid red;}
.input-3{
   border:1px solid #01cc40!important;}
select{
   outline: none!important;}
.input-4{
   border:1px solid red;}
```

Deleteitem.ts

```
import { Component } from '@angular/core';
import { NodeutilityService } from '../nodeutility.service';
import { Router } from '@angular/router';
@Component({
  selector: 'app-deleteitem',
  templateUrl: './deleteitem.component.html',
  styleUrl: './deleteitem.component.css'})
export class DeleteitemComponent {
  item:string='';
  quantity:string="";
  msg:string="";
  constructor(private util:NodeutilityService,private router:Router) { }
  delete(){
    this.util.delete1(this.item,this.quantity).subscribe((data)=>{
      if(data.status){
        this.msg=data.message;      }
      else{
        this.msg=data.message;      }    });
}}
```

Home.html

```
    <body>
        <div class="wrapper">
        <nav class="nav">
            <div class="nav-logo">
                <img src='../../assets/Happy Roots (1).png' alt="HAPPY ROOTS">
            </div>
            <div class="nav-menu">
                <ul>
 <li><a routerLink="/home" class="link active">Home</a></li>
 <li><a routerLink="/crops" class="link"
onclick="window.open('crops','mywindow')"Crops</a></li>
<li><a routerLink="/cattles" class="link"
onclick="window.open('cattles','mywindow')">Cattles</a></li>
   <li><a routerLink="/register"  class="link"
onclick="window.open('register','mywindow')">Register</a></li>
<li><a routerLink="/shopping-cart" class="link" onclick="window.open('shopping-
cart','mywindow')">Shopping</a></li>
<li><a routerLink="/Feedback" class="link"
onclick="window.open('review','mywindow')">Feedback</a></li>
 <li><a routerLink="/display" >Display</a></li>
            </div>
            <div class="nav-button">
                <button class="btn white-btn" id="loginbtn"
onclick="window.open('login','mywindow')" >Sign In</button>
```

```
            <button class="btn signup-btn" id="changebtn"
onclick="window.open('changepwd','mywindow')" >change password</button>
            <button class="btn signup-btn" id="deletebtn"
onclick="window.open('delete','mywindow')" >Delete account</button>
            <button class="btn signup-btn" id="logoutbtn"
onclick="window.open('home','mywindow')" >Logout</button>
         </div>
         <div class="nav-menu-btn">
            <i class="bx bx-menu"></i>
   </div></body>
```

Home.css

```
*{
   margin:0;
   padding:0;
   box-sizing:border-box;
   font-family: 'Times New Roman';}
body{
   background-size: auto;
   background-repeat:no-repeat;
   background-attachment:fixed;
   background-image: url('../../assets/homehome.png');
   background-position: bottom;}
.wrapper{
   display:flex;
   justify-content:center;
   align-items: center;
   min-height: 110vh;}
.nav{
   position: fixed;
   top: 0;
   display: flex;
   justify-content: space-around;
   width: 100%;
   height: 100px;
   line-height: 100px;
   z-index: 100;
   background: green;}
 .nav-logo img {
   color: white;
   font-size: 25px;
   font-weight: 600;
   width: 100px;
   border-radius: 70px;
   justify-content:end;}
 .nav-menu ul{
   display: flex;}
```

```
.nav-menu ul li{
    list-style-type: none;}
.nav-menu ul li .link{
    text-decoration: none;
    font-weight: 100;
    font-size: small;
    color:white;
    padding-bottom: 15px;
    margin: 0 35px;}
.link:hover, .active{
    border-bottom: 2px solid #fff;}
 .nav-button .btn{
    width: 130px;
    height: 40px;
    font-weight: 500;
    background: rgba(39, 39, 39, 0.485);
    border: none;
    border-radius: 30px;
    cursor: pointer;
    transition: 3s ease;}
#signupbtn{
    margin-left: 15px;}
 .btn:hover{
    border: rgba(255, 255 ,255, 0.7);}
.btn.white-btn{
    background: rgba(255, 255 ,255, 0.7);
    font-weight: 700;}
 .btn.btn.white-btn:hover{
    background: rgba(255, 255 ,255, 0.5);}
.btn.signup-btn{
    background: rgba(255, 255 ,255, 0.7);
    font-weight: 700;}
 .btn.btn.signup-btn:hover{
    background: rgba(255, 255 ,255, 0.5);}
  .nav-menu-btn{
    display: none;}
.footer {
    background-color: #333;
    color: #fff;
    text-align: center;
    padding: 10px 0;  }
```

Home.ts

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrl: './home.component.css'})
export class HomeComponent {}
```

Items.html

```
<body >
    <form #form="ngForm" (ngSubmit)="onSubmit(form)">
        <h2>Insert</h2>
        <div class="box">
            <label>Item</label>
            <input type="text" name="item" id="item" ngModel #item="ngModel">
        </div>
        <div class="box">
            <label>Quantity</label>
            <input type="text" name="quantity" id="quantity" ngModel #quantity="ngModel">
        </div>
        <div class="box">
            <input type="submit"  value="submit" class="btn">
        </div>
        <div class="box">{{msg}}</div>
        <div>
            <a routerLink="/admin" style="padding-left: 32%;">BACK</a>
            <br><br>
        </div>
    </form>
</body>
```

Items.css

```
body {
    font-size: 16px;
    font-family: system-ui;
    background:-webkit-radial-gradient(#90F6FB,#2C3283);}
.form{
     width: 100%;}
h2 {
    text-align: center;
    color: #ef6e58;}
form {
    width: 300px;
    padding: 15px 40px 40px;
    border: 1px solid #ccc;
    border-radius: 5px;
    background-color: #fff;
    margin: 4% auto;}
label {
    display: block;
    margin-bottom: 5px;}
label i {
    color: #999;
```

```
   font-size: 80%;}
input, select {
   border: 1px solid #ccc;
   padding: 10px;
   display: block;
   width: 100%;
   box-sizing: border-box;
   border-radius: 2px;
   background-color: #a9bae64a;}
.box {
   padding-bottom: 10px;}
.form-inline {
   border: 1px solid #ccc;
   padding: 8px 10px 4px;
   border-radius: 2px;}
.form-inline label, .form-inline input {
   display: inline-block;
   width: auto;
   padding-right: 15px;}
.error {
   color: red;
   font-size: 90%;}
input[type=text]{
      outline: none;}
input[type="submit"] {
   font-size: 110%;
   font-weight: 100;
   background: #ef6e58;
   border-color: #ef6e58;
   box-shadow: 0 3px 0 #bd432e;
   color: #fff;
   margin-top: 10px;
   cursor: pointer;}
input[type="submit"]:hover {
   background: #bd432e;
   border-color: #bd432e;}
.input-1{
   background-repeat: no-repeat;
   background-position: right 10px center;
   border:1px solid #01cc40;}
.input-2{
   background-repeat: no-repeat;
   background-position: right 10px center;
   border:1px solid red;}
.input-3{
   border:1px solid #01cc40!important;}
```

```
select{
   outline: none!important;}
.input-4{
   border:1px solid red;}
```

Items.ts

```
import { Component } from '@angular/core';
import { NodeutilityService } from '../nodeutility.service';
import { Router } from '@angular/router';
@Component({
  selector: 'app-items',
  templateUrl: './items.component.html',
  styleUrl: './items.component.css'})
export class ItemsComponent {
  constructor(private util:NodeutilityService,private router:Router) { }
  item:string='';
  quantity:string="";
  msg:string="";
  onSubmit(form: any) {
    this.util.insert3(form.value.item, form.value.quantity).subscribe((data) => {
        if (data.status)
          this.msg = data.message;
        else{
          this.msg = data.message;}
        if(data.status){
          this.router.navigate(['/adminhome']);}});}}
```

login.html

```
<link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css' rel='stylesheet'>
<body>
    <div class="wrapper">
        <nav class="nav">
        <div class="nav-menu">
            <ul>
                <li><a routerLink="/login" class="link active">User</a></li>
                <li><a routerLink="/adminlogin" class="link" >Admin</a></li>
            </ul>
        </div>
        </nav>
        <form #form="ngForm" (ngSubmit)="onSubmit(form)">
            <h1>Sign In</h1>
            <div class="input-box">
                <div class="input-field">
  <input type="email" placeholder="Email" name="email"required ngModel #email="ngModel">
                    <i class='bx bxs-envelope'></i>
                </div>
            </div>
            <div class="input-box">
```

```
                <div class="input-field">
<input type="password"placeholder="Password"name="pw"required ngModel #pw="ngModel">
                    <i class='bx bxs-lock'></i>
                </div>
            </div>
            <button type="submit"  class="btn">Sign In</button>
            <span>Don't have an account? <a routerLink="/signup">Sign Up</a></span>
        </form>
    </div>
</body>
```

Login.css

```
@import
url("https://fonts.googleapis.com/css2?family=Popins:wght@300;400;500;600;700;800;900&disp
lay=swap");
* {
margin: 0;
padding: 0;
box-sizing: border-box;
font-family: "Poppins", sans-serif;}
body {
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    background-image:url('../../assets/img.jpg');
    background-size: cover;}
.wrapper {
    width: 500px;
    background: rgba(255, 255, 255, .1);
    border: 2px solid rgba(255, 255, 255, .2);
    box-shadow: 0 0 10px rgba(0, 0, 0, .2);
    backdrop-filter: blur(10px);
    border-radius: 10px;
    color: #fff;
    padding: 40px 35px 55px;
    margin: 0 10px;}
.wrapper h1{
    font-size: 36px;
    text-align: center;
    margin-bottom: 20px;}
.wrapper .input-box {
    display: flex;
    justify-content: center;
    flex-wrap: wrap;
    background: transparent;
    width: 100%;}
```

```
.input-box .input-field {
    position: relative;
    justify-content: center;
    width: 100%;
    height: 50px;
    margin: 13px 0;}
.input-box .input-field input{
    width: 100%;
    height: 100%;
    background: transparent;
    border: 2px solid rgba(255, 255, 255, .2);
    outline: none;
    font-size: 16px;
    color: #fff;
    border-radius: 6px;
    padding: 15px 15px 15px 40px;}
.input-box .input-field input::placeholder {
    color: #fff;}
.input-box .input-field i {
    position: absolute;
    left: 15px;
    top: 25px;
    transform: translateY(-50%);
    font-size: 20px;}
.wrapper label {
    display: inline-block;
    font-size: 14.5px;
    margin: 10px 0 23px;}
.wrapper label input {
    accent-color: #000;
    margin-right: 5px;}
.wrapper .btn {
    width: 100%;
    height: 45px;
    background: #fff;
    border: none;
    outline: none;
    border-radius: 6px;
    box-shadow: 0 0 10px rgba(0, 0, 0, .1);
    cursor: pointer;
    font-size: 16px;
    color: #333;
    font-weight: 600;}
.nav{
    position: fixed;
    top: 0;
    display: flex;
    justify-content: center;
    width: 100%;
    height: 100px;
    line-height: 100px;
    z-index: 100;
    transform: translate(-50px, -100px);}
.nav-menu ul{
    display: flex;
    font-size: 36px;}
```

Login.ts

```
import { Component } from '@angular/core';
import { NodeutilityService } from '../nodeutility.service';
import { Router } from '@angular/router';
import { provideHttpClient } from '@angular/common/http';
@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrl: './login.component.css'})
export class LoginComponent {
  msg:string="";
  user:string | null="";
  constructor(private util:NodeutilityService,private router:Router) { }
  onSubmit(form: any) {
    this.util.insert(form.value.email, form.value.pw).subscribe((data) => {
        if (data.status){
          localStorage.setItem("user",form.value.username);
          this.msg = data.message;
          alert(this.msg);
          this.router.navigate(['\home']);        }
        else{
          this.msg = data.message;
          alert(this.msg);
          this.router.navigate(['\login']);        }  });  }}
```

Pay.html

```
<h4 class="udisplay">pay now</h4>
<div class="container" style="width:30%;height:590px;border-radius:10%;margin-
left:35%;margin-top:7%;">
    <br><br><br>
    <h6>Checkout</h6>
    <span><a href="shopping">X</a></span>
    <h1>Payment Method</h1>
    <form name="f1" action="tick">
        <br><label for="cardno">Card Number
            <input type="text" name="cardno" id="cardno" maxlength="19" (input)="cardSpace()" />
        </label>
        <br><br><br><label for="cn">Card Holder Name
            <input type="text" name="cn" id="cn" maxlength="19" />
        </label>
        <br><br><br><div class="float">
            <br><br><label for="validtill">Valid through
<input type="text" name="validtill" id="validtill" maxlength="7" (input)="addSlashes()" />
            </label>
            <br><br><label for="cvv">CVV
                <input type="text" name="cvv" id="cvv" maxlength="3" />
            </label>
        </div>
        <div class="but" style="padding-left:14%;">
            <button id="paymentButton" (click)="pay()">Pay ${{ total }}</button>
        </div>
    </form>
</div>
```

Pay.css

```
*{
    margin:0;
    padding:0;
    box-sizing: border-box;}
:root{
    --text-color:rgb(43, 40, 40);}
body{
    display: flex;
    align-items:center;
    background-color:rgb(15, 13, 13);}
.container{
    width:350px;
    background:#f3f0f0;
    padding: 30px;
    position: relative;}
h6{
    font-size:20px;
    color:var(--text-color);}
span{
    position:absolute;
    right:34px;
    top:28px;
    font-size:18px;
    color:var(--text-color);}
h1{
    padding: 30px 0px;
    font-size:40px;
    color:var(--text-color);}
form label{
    color:var(--text-color);
    text-transform:uppercase;
    font-size:10px;
    word-spacing:4px;}
form select{
    width:100%;
    outline:none;
    color:var(--text-color);
    padding-bottom: 3px;
    font-size:18px;
    margin-left: -3px;
    margin-bottom: 20px;
    background-color: transparent;}
form #cardno{
    width:100%;
    padding-top:5px;
    padding-bottom: 3px;
    padding-right:30px;
    font-size:18px;
    margin-bottom: 20px;
    background-color: transparent;}
.float{
    display:flex;}
form #cvv{
    border:none;
    border-bottom: 1px solid rgb(223,223,223);
      padding-top:14px;
    margin-bottom:56px;
    color:var(--text-color);
    background-color:transparent;
    font-size:18px;
    padding-left:18px;}
.udisplay {
    font-size: 1.5em;
    color: #fff;
    background-color: #4CAF50;
    text-align: center;}
.username {
    font-weight: bold;
    color: #FFD700;}
form #validtill{
    width:100%;
    margin-bottom: 20px;
    background-color: transparent;}
p{
    margin:-20px 0px 56px 25px;
    font-size:16px;
    text-transform:none;
    color:rgb(3, 2, 2);}
```

Pay.ts

```
import { Component } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { Router } from '@angular/router';
@Component({
  selector: 'app-pay',
  templateUrl: './pay.component.html',
  styleUrl: './pay.component.css'})
export class PayComponent {
  total: string='';
  username:string="";
  constructor(private route: ActivatedRoute,private router:Router) { }

  ngOnInit(): void {
    this.route.queryParams.subscribe(params => {
      this.total = params['total'];
      this.username=params['user'];    });  }
  pay(): void {
    const cno: string = (document.getElementById('cardno') as HTMLInputElement).value.trim();
    const exp: string = (document.getElementById('validtill') as HTMLInputElement).value.trim();
    const cname: string = (document.getElementById('cn') as HTMLInputElement).value.trim();
    const cvv: string = (document.getElementById('cvv') as HTMLInputElement).value.trim();
    if (cno.length === 0 || exp.length === 0 || cname.length === 0 || cvv.length === 0) {
        alert("Fill all details");    }
    else{
      this.router.navigate(['/tick']);    }  }
  cardSpace(): void {
    let cd: string = (document.getElementById('cardno') as HTMLInputElement).value.replace(/-/g,
''); // Remove existing dashes
    cd = cd.replace(/\s+/g, ''); // Remove existing spaces
    let formattedCd = '';
    for (let i = 0; i < cd.length; i++) {
      if (i > 0 && i % 4 === 0) {
        formattedCd += '-';      }
      formattedCd += cd[i];    }
    (document.getElementById('cardno') as HTMLInputElement).value = formattedCd;  }
  addSlashes(): void {
    const validTill: string = (document.getElementById('validtill') as
HTMLInputElement).value.trim();
    if (validTill.length === 2) {
      (document.getElementById('validtill') as HTMLInputElement).value = validTill + "/";    }  }}
```

Mainhome.html

```
<body>
   <div class="wrapper">
   <nav class="nav">
       <div class="nav-logo">
           <img src='../../assets/Happy Roots (1).png' alt="HAPPY ROOTS">
       </div>
       <div class="nav-menu">
           <ul>
               <li><a routerLink="/home" class="link active">Home</a></li>
<li><a routerLink="/crops"class="link"onclick="window.open('crops','mywindow')"
>Crops</a></li>
    <li><a routerLink="/cattles" class="link"
onclick="window.open('cattles','mywindow')">Cattles</a></li>
<li><a routerLink="/register"  class="link"
onclick="window.open('register','mywindow')">Register</a></li>
<li><a routerLink="/shopping-cart" class="link" onclick="window.open('shopping-
cart','mywindow')">Shopping</a></li>
  <li><a routerLink="/Feedback" class="link"
onclick="window.open('review','mywindow')">Feedback</a></li>
  </ul>
</div>
       <div class="nav-button">
<button class="btn white-btn"id="loginbtn"onclick="window.open('login','mywindow')">Sign
In</button>
<button class="btn signup-btn" id="changebtn"
onclick="window.open('changepwd','mywindow')" >change password</button>
<button class="btn signup-btn" id="deletebtn" onclick="window.open('delete','mywindow')"
>Delete account</button>
<button class="btn signup-btn" id="logoutbtn" onclick="window.open('home','mywindow')"
>Logout</button>
 </div>
       <div class="nav-menu-btn">
           <i class="bx bx-menu"></i>
       </div></body>
```

Mainhome.css

```
*{
   box-sizing:border-box;
   font-family: 'Times New Roman';}
body{
   background-size: auto;
   background-repeat:no-repeat;
   background-attachment:fixed;
   background-position: bottom;}
.wrapper{
   display:flex;
   justify-content:center;  }
.nav{
   position: fixed;
   top: 0;
   display: flex;
   justify-content: space-around;
   width: 100%;
   height: 100px;
   line-height: 100px;
   z-index: 100;
   background: green;}
.nav-menu ul{
   display: flex;}
.nav-menu ul li{
   list-style-type: none;}
.nav-menu ul li .link{
      color:white;
   padding-bottom: 15px;
   margin: 0 35px;}
.link:hover, .active{
   border-bottom: 2px solid #fff;}
.nav-button .btn{
   width: 130px;
   cursor: pointer;
   transition: 3s ease;}
.btn:hover{
   border: rgba(255, 255 ,255, 0.7);}
.btn.white-btn{
   background: rgba(255, 255 ,255, 0.7);
   font-weight: 700;}
.btn.btn.white-btn:hover{
   background: rgba(255, 255 ,255, 0.5);}
.nav-menu-btn{
   display: none;}
.footer {
   background-color: #333;
   color: #fff;
   text-align: center;
   padding: 10px 0;  }
```

Mainhome.ts

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-mainhome',
  templateUrl: './mainhome.component.html',
  styleUrl: './mainhome.component.css'})
export class MainhomeComponent {}
```

register.html

```
<link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css' rel='stylesheet'>
```

```
<body>
   <div class="wrapper">
      <form #form="ngForm" (ngSubmit)="onSubmit(form)" (click)="validateForm()">
         <h1>Registration</h1>
         <div class="input-box">
            <div class="input-field">
<input type="text" placeholder="Full Name" required id="fname" ngModel #fname="ngModel">
               <i class='bx bxs-user'></i>
            </div>
            <div class="input-field">
<input type="text" placeholder="No of Students" required id="std" ngModel #std="ngModel" >
               <i class='bx bxs-user'></i>
            </div>
         </div>
         <div class="input-box">
            <div class="input-field">
<input type="email" placeholder="Email" required id="email" ngModel #email="ngModel">
               <i class='bx bxs-envelope'></i>
            </div>
            <div class="input-field">
<input type="text" placeholder="Phone Number" required id="pno" ngModel #pno="ngModel">
 <i class='bx bxs-phone-call'></i>
            </div>
         </div>
         <div class="input-box">
            <div class="input-field">
<input type="date" name="" id="sDate"  ngModel #sDate="ngModel"></div>
<div class="input-field">
               <select id="Slot">
                  <option value="">Morning</option>
                  <option value="hin">Afternoon</option>
               </select>
            </div>
         </div>
         <div class="box">
            <input type="submit"  value="submit" class="btn">
         </div>
         <div class="box">{{msg}}</div>
         <div>
            <a routerLink="/index1" style="padding-left: 32%;">Home Page</a>
         </div>
      </form>
   </div>
</body>
```

Register.css

```
@import
url("https://fonts.googleapis.com/css2?family=Popins:wght@300;400;500;600;700;800;900&disp
lay=swap");
* {
margin: 0;
box-sizing: border-box;
font-family: "Poppins", sans-serif;}
.wrapper {
   width: 750px;
   background: rgba(255, 255, 255, .1);
   border: 2px solid rgba(255, 255, 255, .2);
   box-shadow: 0 0 10px rgba(0, 0, 0, .2);
   backdrop-filter: blur(10px);
   border-radius: 10px;
   color: #fff;
   padding: 40px 35px 55px;
   margin: 0 10px;}
.wrapper h1{
   font-size: 36px;
   text-align: center;
   margin-bottom: 20px;}
.input-box .input-field {
   position: relative;
   width: 49%;
   height: 50px;
   margin: 13px 0;}
.input-box .input-field input{
   width: 100%;
   height: 100%;
   background: transparent;
   border: 2px solid rgba(255, 255, 255, .2);
   border-radius: 6px;
   padding: 15px 15px 15px 40px;}
.input-box .input-field i {
   position: absolute;
   left: 15px;
   top: 25px;
   transform: translateY(-50%);
   font-size: 20px;}
.wrapper label {
   display: inline-block;
   font-size: 14.5px;
   margin: 10px 0 23px;}
.wrapper label input {
   accent-color: #000;
   margin-right: 5px;}
```

Register.ts

```
import { Component } from '@angular/core';
import { NodeutilityService } from '../nodeutility.service';
import { Router } from '@angular/router';
@Component({
  selector: 'app-register',
  templateUrl: './register.component.html',
  styleUrl: './register.component.css'})
export class RegisterComponent {
  constructor(private util:NodeutilityService,private router:Router){}
  imageUrl='../../assets/Reg.jpg';
  msg:string="";
  uname:string="";
  selectedGender:string="";
  onSubmit(form: any) {
    this.util.insert1(form.value.fname, form.value.uname, form.value.pno,form.value.email,
form.value.pwd).subscribe((data) => {
        if (data.status)
          this.msg = data.message;
        else{
          this.msg = data.message;        }
        if(data.status){
          this.router.navigate(['/login']);        }       });    }
    printError(Id: string, Msg: string) {
      const element = document.getElementById(Id);
      if (element) {
        element.innerHTML = Msg;      }    }
    validateForm() {
      const fnameElement = <HTMLInputElement>document.querySelector("#fname");
      const unameElement = <HTMLInputElement>document.querySelector("#std");
      const emailElement = <HTMLInputElement>document.querySelector("#email");
      const mobileElement = <HTMLInputElement>document.querySelector("#pno");
      const slotElement = <HTMLInputElement>document.querySelector("#slot");
      if (!fnameElement || !unameElement|| !emailElement || !mobileElement  || !slotElement) {
        return false; // Return false if any required element is missing    }
      const fname = fnameElement.value;
      const uname = unameElement.value;
      const email = emailElement.value;
      const mobile = mobileElement.value;
      const slot = slotElement.value;
      let fnameErr: boolean = true, unameErr: boolean = true, emailErr: boolean = true, mobileErr:
boolean = true , slotErr: boolean = true;
      if (fname == "") {
        this.printError("fnameErr", "Please enter your name");
        fnameElement.classList.add("input-2");
        fnameElement.classList.remove("input-1");      }
```

```
else {
        const regex = /^[a-zA-Z\s]+$/;
        if (regex.test(fname) === false) {
          this.printError("fnameErr", "Please enter a valid name");
          fnameElement.classList.add("input-2");
          fnameElement.classList.remove("input-1");
        } else {
          this.printError("fnameErr", "");
          fnameErr = false;
          fnameElement.classList.add("input-1");
          fnameElement.classList.remove("input-2");        }      }
      if (uname == "") {
        this.printError("unameErr", "Please enter number of people");
        unameElement.classList.add("input-2");
        unameElement.classList.remove("input-1");
      } else {
        const regex = /^[a-zA-Z0-9\s]+$/;
        if (regex.test(uname) === false) {
          this.printError("unameErr", "Please enter a valid number");
          unameElement.classList.add("input-2");
          unameElement.classList.remove("input-1");
        } else {
          this.printError("unameErr", "");
          unameErr = false;
          unameElement.classList.add("input-1");
          unameElement.classList.remove("input-2");        }      }
      if (email == "") {
        this.printError("emailErr", "Please enter your email address");
        emailElement.classList.add("input-2");
        emailElement.classList.remove("input-1");
      } else {
        var regex = /^\S+@\S+\.\S+$/;
        if (regex.test(email) === false) {
          this.printError("emailErr", "Please enter a valid email address");
          emailElement.classList.add("input-2");
          emailElement.classList.remove("input-1");
        } else {
          this.printError("emailErr", "");
          emailErr = false;
          emailElement.classList.add("input-1");
          emailElement.classList.remove("input-2");        }      }
      if (mobile == "") {
        this.printError("mobileErr", "Please enter your mobile number");
        mobileElement.classList.add("input-2");
        mobileElement.classList.remove("input-1");
      } else {
        var regex = /^[1-9]\d{9}$/;
        if (regex.test(mobile) === false) {
          this.printError("mobileErr", "Please enter a valid 10 digit mobile number");
          mobileElement.classList.add("input-2");
          mobileElement.classList.remove("input-1");
        } else {
          this.printError("mobileErr", "");
          mobileErr = false;
          mobileElement.classList.add("input-1");
          mobileElement.classList.remove("input-2");        }      }
      if (slot == "") {
        this.printError("slotErr", "Please select your slot");
        slotElement.classList.add("input-4");
        slotElement.classList.remove("input-3");
      } else {
        this.printError("slotErr", "");
        slotErr = false;
        slotElement.classList.add("input-3");
        slotElement.classList.remove("input-4");      }
      if (fnameErr ||unameErr || emailErr || mobileErr ) {
        return false;      }
      return true;    }}
```

Review.html

```
<div class="review-container">
   <h2>Write Your Review</h2>
   <div class="form-container">
     <label>username</label>
     <input type="text" name="username" placeholder="enter your email">
     <textarea class="review-textarea" [(ngModel)]="reviewText" placeholder="Write your review here..."></textarea>
     <div class="rating-container">
       <span class="star" *ngFor="let _ of stars; let i = index" (click)="rate(i + 1)">
         <ng-container *ngIf="i < rating; else emptyStar">&#9733;</ng-container>
         <ng-template #emptyStar>&#9734;</ng-template>
       </span>
     </div>
     <button class="submit-button" (click)="submitReview()">Submit Review</button>
   </div>
  </div>
```

Review.css

```
.review-container {
   background-color: #f0f0f0; /* Light gray background */
   padding: 20px;
   border-radius: 10px;}
h2 {
   color: #333; /* Dark gray text color */}
```

```
.form-container {
    background-color: #fff; /* White background for form */
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0010pxrgba(0, 0, 0, 0.1); /* Shadow effect */}
.review-textarea {
    width: 100%;
    resize: none;}
.star {
    display: inline-block;
    cursor: pointer;
    font-size: 24px;
    color: #FFD700; /* Gold color for stars */
    margin-right: 5px;}
```

Review.ts

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-review',
  templateUrl: './review.component.html',
  styleUrl: './review.component.css'})
export class ReviewComponent {
  rating:number=0;
  stars:number[] = [1, 2, 3, 4, 5];
  reviewText:string="; // Add a property to hold the review text
  rate(rating:number) {
    this.rating=rating;
    console.log('Rating:', this.rating);
  }
  submitReview() {
    console.log('Review Rating:', this.rating);
    console.log('Review Text:', this.reviewText);
    this.reviewText=";
    this.rating=0;}}
```

Shopping-cart.html

```
<h2>Shopping Cart</h2>
<hr>
    <div class="products-container">
        <div *ngFor="let product of products" class="product-card">
            <h3>{{ product.name }}</h3>
            <img [src]="product.imageUrl" alt="{{ product.name }} Image">
            <p>Price: ${{ product.price }}</p>
            <button (click)="addToCart(product)">Add to Cart</button>
        </div>
    </div>
    <div *ngIf="cart.length > 0">
        <h3>Cart</h3>
        <div *ngFor="let product of cart" class="product-card">
            <p>{{ product.name }} - ${{ product.price }}</p>
            <button (click)="removeFromCart(product)">Remove from Cart</button>
        </div>
        <p>Total: ${{ getTotal() }}</p>
        <button (click)="checkout()" class="checkout-button">Pay</button>
    </div>
```

Shopping-cart.css

```
.shopping-cart-container {
    width: 400px;
    padding: 20px;
    margin: 0 auto;}
.products-container {
    display: flex;
    flex-wrap: wrap;}
.product-card h3{
    padding-left:35%;
    font-weight:1000;
    font-size:30px;
    color:#000000;}
.product-card p{
    padding-left:35%;
    font-weight:1000;
    font-size:30px;
    color:#000000;}
.welcome-message {
    font-size: 1.5em;
    color: #fff;
    text-align: center;}
.username {
    font-weight: bold;
    color: #FFD700;}
.container{
    width:1000px;
    margin:auto;
    height: 200px;
    display: flex;}
.a1 img{
    width: 15%;
    margin-top:-4%;
    margin-left:2%;}
.a1 a{
    padding-top: 100px;
    font-size: 30px;
    text-decoration: none;
    color: white;
    padding-left: 150px;
    font-family: fantasy;}
.product-card {
    border: 1px solid #ccc;
    padding: 10px;
    margin: 10px;
    border-radius: 0%;}
img{
    width: 300px;
    height: 300px;
    margin-left:13%;}
button {
    background-color: #1da46c;
    padding: 8px 12px;
    width:40%;}
button:hover {
    background-color: #3ffa7d;}
.total-section {
    background-color: #f0f0f0;
    padding: 10px;
    margin-top: 20px;}
.checkout-button {
    background-color: #ff6320;
    color: #fff;
    border: none;
    padding: 10px 20px;
    margin-top: 10px;
    cursor: pointer;}
.checkout-button:hover {
    background-color: #f59e73;}
.cartitems h3{
    font-weight:bold;
    font-size:90px;
    padding-left:39%;}
.product-card1 {
    padding: 10px;
   width: 30%; /* Adjust width as needed */
    position: relative;}
.product-card1 p{
    font-weight:bold;
    font-size:30px;}
.rbut{
    margin-right:10px;
    width:10%;
    flex-shrink:0;}
```
Shopping-cart.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { ShoppingCartComponent } from './shopping-cart.component';
describe('ShoppingCartComponent', () => {
  let component: ShoppingCartComponent;
  let fixture: ComponentFixture<ShoppingCartComponent>;
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ShoppingCartComponent]    })
    .compileComponents();
    fixture = TestBed.createComponent(ShoppingCartComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();  });
  it('should create', () => {
    expect(component).toBeTruthy();  });});
```

Signup.html

```
<link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css' rel='stylesheet'>
<body [style.background-image]="'url('+imageUrl +')'">
    <div class="wrapper">
        <form #form="ngForm" (ngSubmit)="onSubmit(form)" (click)="validateForm()">
<h1>Registration</h1>
            <div class="input-box">
                <div class="input-field">
<input type="text"placeholder="Full Name"  name="fname"id="fname" ngModel
#fname="ngModel">
                    <i class='bx bxs-user'></i>
                    <div class="error" id="fnameErr"></div>
                </div>
                <div class="input-field">
<input type="text" placeholder="User Name" name="uname" id="uname" ngModel
#uname="ngModel">
                    <i class='bx bxs-user'></i>
                    <div class="error" id="unameErr"></div>
                </div>
            </div>
            <div class="input-box">
                <div class="input-field">
<input type="email" placeholder="Email" name="email" id="email" ngModel
#email="ngModel">
                    <i class='bx bxs-envelope'></i>
                    <div class="error" id="emailErr"></div>
                </div>
                <div class="input-field">
<input type="text" placeholder="Phone Number" name="pno" maxlength="10" id="pno"
ngModel #pno="ngModel">
                    <i class='bx bxs-phone-call'></i>
                    <div class="error" id="mobileErr"></div>
                </div>
```

```
          </div>
          <div class="input-box">
             <div class="input-field">
                <input type="password"  placeholder="Password" name="pwd" id="pwd"
maxlength="10" ngModel #pwd="ngModel">
                <i class="bx bx-lock-alt"></i>
                <div class="error" id="pwErr"></div>
             </div>
             <div class="input-field">
                <input type="password"  placeholder="Confrim Password" name="cpwd" id="cpwd"
ngModel #cpwd="ngModel">
                <i class="bx bx-lock-alt"></i>
                <div class="error" id="cpwErr"></div>
             </div>
          </div>
          <div class="input-box">
             <label>Gender</label>
             <div class="form-inline" id="gender" name="gender" >
                <label><input type="radio" name="gender" value="male"
[(ngModel)]="selectedGender"> Male</label>
                <label><input type="radio" name="gender" value="female"
[(ngModel)]="selectedGender"> Female</label>
             </div>
             <div class="error" id="genderErr"></div>
          </div>
          <div class="box">
             <input type="submit"  value="submit" class="btn">
          </div>
          <div class="box">{{msg}}</div>
                </form>
   </div>
</body>
```

<u>Signup.css</u>

```
@import
url("https://fonts.googleapis.com/css2?family=Popins:wght@300;400;500;600;700;800;900&disp
lay=swap");
* {
margin: 0;
padding: 0;
box-sizing: border-box;
font-family: "Poppins", sans-serif;}
body {
   display: flex;
   justify-content: center;
   align-items: center;
   min-height: 100vh;
   background-image:url('../../assets/Reg.jpg');
```

```
    background-size: cover;}
.wrapper {
    backdrop-filter: blur(10px);
    border-radius: 10px;
    color: #fff;
    padding: 40px 35px 55px;
    margin: 0 10px;}
.wrapper h1{
    font-size: 36px;
    text-align: center;
    margin-bottom: 20px;}
.wrapper .input-box {
    display: flex;
    justify-content: space-between;
    flex-wrap: wrap;
    background: transparent;}
.input-box .input-field {
    position: relative;
    height: 50px;
    margin: 13px 0;}
.input-box .input-field input{
    width: 100%;
    height: 100%;
    background: transparent;
    border: 2px solid rgba(255, 255, 255, .2);}
.input-box .input-field input::placeholder {
    color: #fff;}
.input-box .input-field i {
    position: absolute;
    left: 15px;
    top: 25px;
    transform: translateY(-50%);
    font-size: 20px;}
.form-inline {
    border: 1px solid #ccc;
    padding: 8px 10px 4px;
    border-radius: 2px;}
.form-inline label, .form-inline input {
    display: inline-block;
    width: auto;
    padding-right: 15px;}
```

Signup.ts

```
import { Component } from '@angular/core';
import { NodeutilityService } from '../nodeutility.service';
import { Router } from '@angular/router';
@Component({
  selector: 'app-signup',
  templateUrl: './signup.component.html',
  styleUrl: './signup.component.css'})
export class SignupComponent {
  constructor(private util:NodeutilityService,private router:Router){}
  imageUrl='../../assets/Reg.jpg';
  msg:string="";
  uname:string="";
  selectedGender:string="";
  onSubmit(form: any) {
    this.util.insert1(form.value.fname, form.value.uname, form.value.pno,form.value.email,
form.value.pwd).subscribe((data) => {
        if (data.status)
          this.msg = data.message;
        else{
          this.msg = data.message;        }
        if(data.status){
          this.router.navigate(['/login']);        }      });    }
    printError(Id: string, Msg: string) {
      const element = document.getElementById(Id);
      if (element) {
        element.innerHTML = Msg;      }    }
    validateForm() {
      const fnameElement = <HTMLInputElement>document.querySelector("#fname");
      const unameElement = <HTMLInputElement>document.querySelector("#uname");
      const emailElement = <HTMLInputElement>document.querySelector("#email");
      const mobileElement = <HTMLInputElement>document.querySelector("#pno");
      const pwElement = <HTMLInputElement>document.querySelector("#pwd");
      const cpwElement = <HTMLInputElement>document.querySelector("#cpwd");
      const genderElement = <HTMLInputElement>document.querySelector("#gender");
      if (!fnameElement || !unameElement|| !emailElement || !mobileElement ||!pwElement ||
!cpwElement || !genderElement) {
        return false; // Return false if any required element is missing      }
      const fname = fnameElement.value;
      const uname = unameElement.value;
      const email = emailElement.value;
      const mobile = mobileElement.value;
      const pw = pwElement.value;
      const cpw = cpwElement.value;
      const gender = genderElement.value;
      let fnameErr: boolean = true, unameErr: boolean = true, emailErr: boolean = true, mobileErr:
boolean = true ,pwErr:boolean=true, cpwErr: boolean = true, genderErr: boolean = true;
      if (fname == "") {
        this.printError("fnameErr", "Please enter your name");
        fnameElement.classList.add("input-2");
        fnameElement.classList.remove("input-1");
      } else {
```

```
  const regex = /^[a-zA-Z\s]+$/;
  if (regex.test(fname) === false) {
    this.printError("fnameErr", "Please enter a valid name");
    fnameElement.classList.add("input-2");
    fnameElement.classList.remove("input-1");
  } else {
    this.printError("fnameErr", "");
    fnameErr = false;
    fnameElement.classList.add("input-1");
    fnameElement.classList.remove("input-2");        }      }
if (uname == "") {
  this.printError("unameErr", "Please enter user name");
  unameElement.classList.add("input-2");
  unameElement.classList.remove("input-1");
} else {
  const regex = /^[a-zA-Z0-9\s]+$/;
  if (regex.test(uname) === false) {
    this.printError("unameErr", "Please enter a valid user name");
    unameElement.classList.add("input-2");
    unameElement.classList.remove("input-1");
  } else {
    this.printError("unameErr", "");
    unameElement.classList.add("input-1");
    unameElement.classList.remove("input-2");        }      }
if (email == "") {
  this.printError("emailErr", "Please enter your email address");
  emailElement.classList.add("input-2");
  emailElement.classList.remove("input-1");
} else {
  var regex = /^\S+@\S+\.\S+$/;
  if (regex.test(email) === false) {
    this.printError("emailErr", "Please enter a valid email address");
    emailElement.classList.add("input-2");
    emailElement.classList.remove("input-1");
  } else {
    this.printError("emailErr", "");
    emailErr = false;
    emailElement.classList.add("input-1");
    emailElement.classList.remove("input-2");        }      }
if (mobile == "") {
  this.printError("mobileErr", "Please enter your mobile number");
  mobileElement.classList.add("input-2");
  mobileElement.classList.remove("input-1");
} else {
  var regex = /^[1-9]\d{9}$/;
  if (regex.test(mobile) === false) {
        this.printError("mobileErr", "Please enter a valid 10 digit mobile number");
        mobileElement.classList.add("input-2");
        mobileElement.classList.remove("input-1");
       } else {
        this.printError("mobileErr", "");
        mobileErr = false;
        mobileElement.classList.add("input-1");
        mobileElement.classList.remove("input-2");        }      }
      if(pw == ""){
       this.printError("pwErr","Please enter your password");
       pwElement.classList.add("input-4");
       pwElement.classList.remove("input-3");
      } else {
       this.printError("pwErr", "");
       pwErr = false;
       pwElement.classList.add("input-3");
       pwElement.classList.remove("input-4");      }
      if(cpw == ""){
       this.printError("cpwErr","Please enter your password ");
       cpwElement.classList.add("input-4");
       cpwElement.classList.remove("input-3");
      } else {
       if(pw!=cpw){
        this.printError("cpwErr","password and confirm password does not match");
        emailElement.classList.add("input-2");
        emailElement.classList.remove("input-1");        }
       else{
        this.printError("cpwErr", "");
        cpwErr = false;
        cpwElement.classList.add("input-3");
        cpwElement.classList.remove("input-4");       }           }
      if (gender == "") {
       this.printError("genderErr", "Please select your gender");
       genderElement.classList.add("input-4");
       genderElement.classList.remove("input-3");
      } else {
       this.printError("genderErr", "");
       genderErr = false;
       genderElement.classList.add("input-3");
       genderElement.classList.remove("input-4");      }
if (fnameErr ||unameErr || emailErr || mobileErr || pwErr || cpwErr) {
       return false;      }
      return true;    }         }
```

Updateitem.html

```
<form #form="ngForm" (ngSubmit)="onSubmit(form)">
   <h2>Update Item</h2>
```

```
    <div class="box">
        <label>Item name</label>
        <input type="text" name="item" id="item" ngModel #item="ngModel">
    </div>
    <div class="box">
        <label>Old Quantity</label>
        <input type="text" name="oldquantity"  id="oldquantity" ngModel
#oldquantity="ngModel">
    </div>
    <div class="box">
        <label>New Quantity</label>
        <input type="text" name="newquantity" id="newquantity" ngModel
#newquantity="ngModel">
        <a routerLink="/adminhome" style="padding-left: 32%;">BACK TO HOME</a>
    </div>
</form>
```

Updateitem.css

```
body {
    font-size: 16px;
    font-family: system-ui;
    background:-webkit-radial-gradient(#90F6FB,#2C3283);}
.form{
    padding-top: 50%;
    width: 100%;}
h2 {
    text-align: center;
    color: #ef6e58;}
form {
    width: 300px;
    padding: 15px 40px 40px;
    border: 1px solid #ccc;
     margin: 4% auto;}
label {
    display: block;
    margin-bottom: 5px;}
label i {
    color: #999;
    font-size: 80%;}
input, select {
    border: 1px solid #ccc;
    padding: 10px;
    display: block;}
.box {
    padding-bottom: 10px;}
.form-inline {
    border: 1px solid #ccc;
    padding: 8px 10px 4px;
    border-radius: 2px;}
.form-inline label, .form-inline input {
    display: inline-block;
    width: auto;
    padding-right: 15px;}
.error {
    color: red;
    font-size: 90%;}
```

Updateitem.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { UpdateitemComponent } from './updateitem.component';
describe('UpdateitemComponent', () => {
  let component: UpdateitemComponent;
  let fixture: ComponentFixture<UpdateitemComponent>;
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [UpdateitemComponent]    })
    .compileComponents();
    fixture = TestBed.createComponent(UpdateitemComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();  });
  it('should create', () => {
    expect(component).toBeTruthy();  });});
```

REFERENCES

1. Angular Documentation - https://angular.io/docs
2. Angular Material Component Dev Kit - https://material.angular.io/cdk/categories
3. TypeScript by Infosys Springboard- https://infyspringboard.onwingspan.com/typescript
4. Angular by Infosys Springboard - https://infyspringboard.onwingspan.com/angular
5. Angular for Beginners by freeCodeCamp - https://youtu.be/3qBXWUpoPHo
6. Node.js and Express Tutorial by freeCodeCamp - https://youtu.be/Oe421EPjeBE
7. Complete MongoDB Tutorial - https://youtube.com/playlist?list=PL4cUxeGkcC9h77dJ-QJlwGlZlTd4ecZOA