ANUDIP FOUNDATION

ITPR Project

Project Title

“ Efarming System Using Django ”

By

Name	Enrollment No
1.Vaishnavi Mohite	AF0481879
2.Poonam Dabhade	AF0481824

Under Guidance of

Rajshri Thete

ABSTRACT

E-Farming is a project based on modern web technology that provides a user-friendly interface for farmers and buyers to interact, trade, and access agricultural resources online anytime from anywhere. Today, online farming platforms are the most efficient and cost-effective means to bridge the gap between rural producers and urban consumers. Farmers find it more convenient to promote and sell their crops directly through the platform, while buyers benefit from real-time updates, transparency, and ease of ordering.

This system not only saves time but also empowers farmers with digital tools like crop recommendations, weather-based tips, soil health checkers, and a personalized dashboard. Buyers can search for crops, view farmer profiles, place orders, and make secure payments seamlessly.

Through this project, we present a comprehensive solution to revolutionize agriculture through digital transformation. The farmer dashboard manages crop entries, order tracking, and farming insights, while the buyer dashboard offers a smooth browsing and ordering experience with real-time crop availability and feedback options.

This dynamic and interactive software is tailored for farmers and buyers alike, making agricultural trade simple, transparent, and efficient. The admin panel controls all backend functionalities like user management, crop listings, weather data integration, and order history. The system also generates reports like crop demand, user activity, and order trends to help improve service and transparency. The e-Farming platform supports rural empowerment and promotes a smart, connected agriculture ecosystem.

ACKNOWLEDGEMENT

The project “ **Efarming System Using Django** ” is the Project work carried out by

Name	Enrollment No
1.Vaishnavi Mohite	AF0481879
2.Poonam Dabhade	AF0481824

Under the Guidance:-
Rajshri Thete.

We are thankful to my project guide for guiding me to complete the Project.
Her suggestions and valuable information regarding the formation of the Project Report have provided me a lot of help in completing the Project and its related topics.

We are also thankful to my family member and friends who were always there to provide support and moral boost up.

INDEX

E-Farming System Using Django

INTRODUCTION

The **e-Farming System** is a Django-based web application designed to digitize and streamline agricultural commerce. The project aims to bridge the gap between farmers and consumers by providing an intuitive and efficient online marketplace for farm products. With the rise of internet accessibility, the system allows users to browse, search, and order agricultural products directly from farmers, reducing dependency on intermediaries and enabling better price realization.

The system incorporates modern e-commerce features such as product categorization, user profiles, cart management, and secure payment gateways using Instamojo. It also includes administrative controls for managing product listings, user activities, and sales analytics. This project not only promotes agricultural sales but also empowers farmers by extending their reach beyond local markets.

Objectives

The main objectives of the e-Farming System are:

✓ To provide a digital platform for farmers to sell their products directly to customers.

✓ To simplify product browsing, searching, and ordering for consumers.

✓ To reduce dependency on manual selling and middlemen, improving farmer profits.

✓ To implement secure online transactions via integrated payment gateways.

✓ To maintain user and admin dashboards for smooth profile and product management.

✓ To support product categorization and better visibility across regions

SYSTEM ANALYSIS

In the absence of adequate security in the e-Farming platform:

- **User privacy is compromised**: Farmers' and buyers’ personal data (e.g., contact, payment details) could be exposed.
- **Misuse of data**: Personal or product-related information could be misused—such as unauthorized sales, false listings, or publishing sensitive details online.
- **Concurrent access issues**: Multiple users accessing the platform simultaneously can cause data inconsistency, delayed transactions, or system crashes if not securely managed.
- **Loss of trust**: Users may stop using the platform if their data or payments are not protected, causing a collapse in digital adoption among farmers and rural users.

PRELIMINARY INVESTIGATION:

Purpose –

To create a digital platform that allows farmers to list and sell products and customers to buy fresh produce directly online—**anytime, anywhere**—reducing middlemen and improving reach..

Benefits-

- Eliminates the need for physical marketplaces or long travel.
- Easy access to current listings, seasonal produce, and pricing.
- Encourages repeated usage through convenience.
- Enables digital communication via email and mobile notifications.
- Reduces overhead for both farmers and customers.

Proposed System -

The proposed **e-Farming system** offers:

- A reliable, scalable platform for agricultural commerce.
- Convenient order and payment options through online methods.
- Instant confirmation via mobile messages after a successful order.
- Secure order storage and tracking in a central database.
- Eliminates manual handling of stock and improves transaction management.

Technical Specifications

- **Frontend:** HTML, CSS, Bootstrap **Database:** MySQL
- **Payment Integration:** Instamojo API
- **Backend:** Django (Python framework.

□ REQUIREMENT SPECIFICATIONS

- **Reliability**: System handles high user load without failure.
- **Organization**: Clear separation of user and admin workflows.
- **Correctness**: Every transaction is processed and recorded accurately.
- **Usability**: User interface is intuitive and mobile-friendly.
- **Maintainability**: Easy to update crops, categories, and fix issues.
- **Expandability**: New features or crop types can be added easily.

- **Portability**: Can be hosted on various cloud/server platforms.
- **Accurateness**: No duplication, wrong data, or failed transactions.
- **Error-handling**: Prevents and logs errors clearly.
- **Communication**: Confirms all actions through alerts or messages.

□ FEASIBILITY STUDY

A feasibility study is an evaluation of a proposal designed to determine the difficulty in carrying out a designated task. Generally, a feasibility study precedes technical development and project implementation. In other words, a feasibility study is an evaluation or analysis of the potential impact of a proposed project

We can also say that a feasibility study is a detailed analysis of a company and its operations that is conducted in order to predict the results of a specific future course of action. Small business owners may find it helpful to conduct a feasibility study whenever they anticipate making an important strategic decision. For example, a company might perform a feasibility study to evaluate a proposed change in location, the acquisition of another company, a purchase of major equipment or a new computer system, the introduction of a new product or service, or the hiring of additional employees. In such situations, a feasibility study can help a small business's managers understand the impact of any major changes they might contemplate.

Steps in Conducting a Feasibility Study:

- The main objective of a feasibility study is to determine whether a certain plan of action is feasible—that is, whether or not it will work, and whether or not it is worth doing economically. Although the core of the study is dedicated to showing the outcomes of specific actions, it should begin with an evaluation of the entire operation. For example, a good feasibility study would review a company's strengths and weaknesses, its position in the marketplace, and its financial situation. It would also include information on a company's major competitors, primary customers, and any relevant industry trends. This sort of overview provides small business owners and managers with an objective view of the company's current situation and opportunities. By providing information on consumer needs and how best to meet them, a feasibility study can also lead to new ideas for strategic changes.
- The second part of a good feasibility study should focus on the proposed plan of action and provide a detailed estimate of its costs and benefits. In some cases, a feasibility study may lead management to determine that the company could achieve the same benefits through easier or cheaper means. For example, it may be possible to improve a manual filing system rather than purchase an expensive new computerized database. If the proposed project is determined to be both feasible and desirable, the information provided in the feasibility study can prove valuable in implementation. It can be used to develop a strategic plan for the project, translating general ideas into measurable goals. The goals can then be broken down further to create a series of concrete steps and outline how the steps can be implemented. Throughout the process, the feasibility study will show the various consequences and impacts associated with the plan of action.
- In most cases, a feasibility study should be performed by a qualified consultant in order to ensure its accuracy and objectivity. To be able to provide a meaningful analysis of the data, the consultant should have expertise in the industry. It is also important for small businesses to assign an internal person to help gather information for the feasibility study. The small business owner must be sure that those conducting the study have full access to the company and the specific information they need.

Advantages of making Feasibility study:

- ✓ There are many advantages of making feasibility study some of which are summarized below:
- ✓ This study being made as the initial step of software development life cycle has all the analysis part in it which helps in analyzing the system requirements completely. .
- ✓ Helps in identifying the risk factors involved in developing and deploying the systemsaaAXAaxsaS..
- ✓ The feasibility study helps in planning for risk analysis.

  .
- ✓ Feasibility study helps in making cost/benefit analysis which helps the organization and system to run efficiently.

  .
- ✓ Feasibility study helps in making plans for training developers for implementing the system.

  .
- ✓ So a feasibility study is a report which could be used by the senior or top persons in the organization. This is because based on the report the organization decides about cost estimation, funding and other important decisions which is very essential for an organization to run profitably and for the system to run stable.

Different Types of Feasibility:

In the conduct of feasibility study, the analyst will usually consider seven distinct but inter-related types of feasibility.

The different types of feasibility studies are as follows:

- ***Technical feasibility study:*** It is used to determine the requirements of technologies for the current system.
- ***Schedule feasibility study:*** It is used to determine the time factor related to the current system.
- ***Social feasibility study:*** It is used to determine whether a proposed project will be acceptable to the people or not. This determination typically examines the probability of the project being accepted by the group directly affected by the proposed system change.

- ***Legal feasibility study:*** It is used to determine the legal scrutiny of the current system.
- ***Marketing feasibility study:*** It is used to determine the single and multidimensional market forces that affect the current system. There are four types of marketing feasibility study, which are as follows:
    1) **Economic feasibility**
    2) **Legal feasibility**
    3) **Operational feasibility**
    4) **Schedule feasibility**
    5) **Is it technically feasible or not?**

The project is made with the help of a PC is easily available.

Secondly, the software used for encryption and decryption purpose is alo easily available.

Finally, the other software used to develop this project can easily interface with each other and also compatible with Windows 2000, Windows XP, Windows Server 2003, or Windows NT 4.0 operating system.

6) **Is it technically feasible or not?**

This website is very user –friendly. Any user who knows simple English language can use this website. The user does not need to be familiar with .net framework or SQL server to use this website. So we can conclude that the website is operationally feasible as doing any operation in this website is very simple and it does not involve any complicacy.

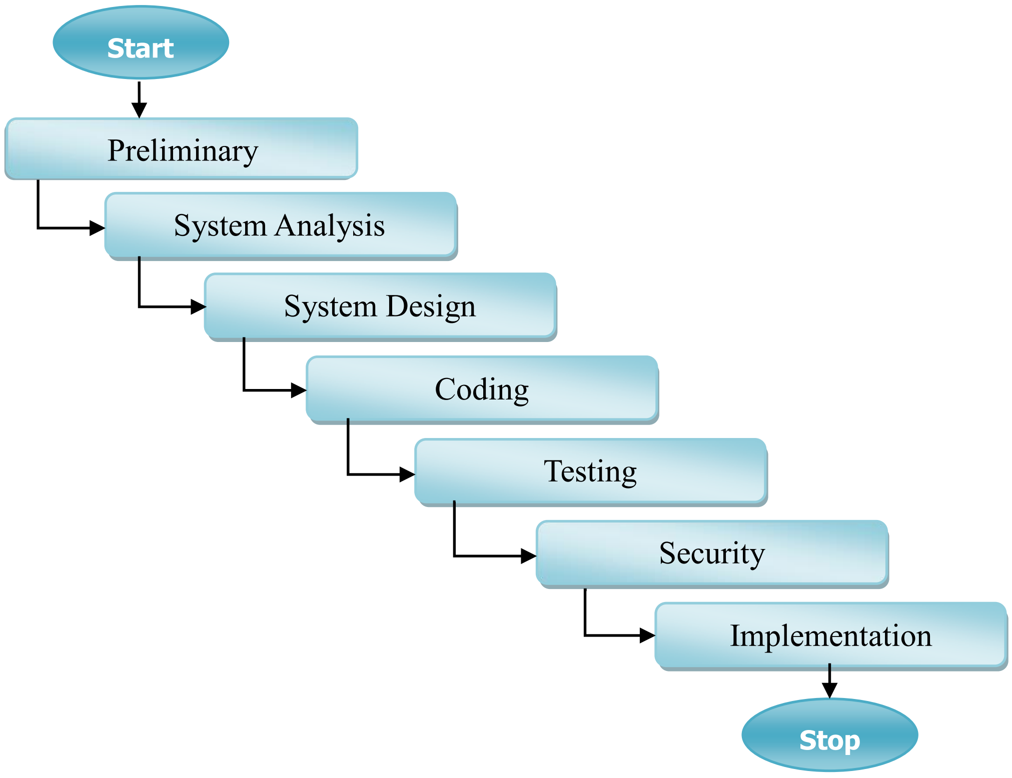
7) **Is it economically feasible or not?**

The project is economically feasible as to install this software we need the minimum hardware configuration that we require installing Windows 2000, Windows XP, Windows Server 2003, or Windows NT 4.0whose cost is minimal. Moreover to make this software run we do not need to install any additional software except Dot Net Framework and Windows Installer 3.1. These two software's will be installed automatically while installing our software. So, now we can easily conclude that this software is economically feasible.

PROJECT PLANNING

Phase To Cover

1. Preliminary Investigation
2. System Analysis
3. System Design
4. Coding
5. Security
6. Testing
7. Implementation



□ SOFTWARE REQUIREMENT SPECIFICATION (SRS)

Requirements analysis is done in order to understand the problem, which is to be solved. That is very important activity for the development of database system. The requirements and the collection analysis phase produce both data requirements and functional requirements. The data requirements are used as a source of database design. The data requirements should be specified in as detailed and complete form as possible. In parallel with specifying the data requirements, it is useful to specify the known functional requirements of the application. These consist of user defined operations that will be applied to the database (retrievals and updates). The functional requirements are used as a source of application software design.

Online Efarming.

The Data-requirements are given as follows :-

User Module :

1. User Registration and Authentication Module

Purpose: Allows farmers and customers to create accounts and securely log in.

Functions:

- ➢ Register with personal details (name, email, phone number, password).
- ➢ Login and session management.
- ➢ Admin approval process for farmer accounts.
- ➢ Password recovery (optional).

2. Product Management Module

Purpose: Enables farmers to manage their product listings.

Functions:

- ➢ Add new products with name, category, price, image, and description.
- ➢ delete existing product entries.
- ➢ Assign products to predefined categories (e.g., Vegetables, Fruits).

3. Category Management Module (Admin Only)

Purpose: Maintains the list of product categories available in the system.

Functions:

- ➢ Add, edit, or delete categories.
- ➢ Display categories for customers and farmers during product listing or browsing.

4. Cart and Booking Module

Purpose: Allows customers to select items, add them to a cart, and place orders.

Functions:

- Add items to cart with quantity selection.
- Remove items from cart or update quantity.
- Place bookings (orders) for cart contents.
- Booking linked to specific farmer and product.

Functional Requirements

Customer Operations

- Register using name, email, phone number, and password.
- Log in securely using credentials.
- Browse products by category or keyword.
- Add items to a personal cart.
- Confirm bookings and proceed to checkout.
- Make payments and view booking confirmations.

Farmer Operations

- Register and submit profile for admin approval.
- Log in and manage own profile.
- Upload new product listings with name, price, image, and category.
- View bookings received from customers.

Admin Operations

- Admin login for secure access.
- Approve or reject farmer registrations.
- Add or update product categories.
- Remove inappropriate or inactive product listings.
- Manage and update status of bookings.
- View and download transaction reports.

SOFTWARE AND HARDWARE REQUIREMENTS

User Interface: There should be user friendly web pages [Graphical User Interface (GUI)] which will be easily understandable and manageable

Hardware Requirements:

Following Hardware will be used for Development testing of the proposed project

Processor: Pentium 4, 2.0 GHz or equivalent AMD processor

RAM: 128 MB DDR RAM

Hard Disk Drive (HDD): 40 GB

Video Card: 32 MB

Networking Components: LAN card, 8-Port HUB

Software Requirements:

- ✓ ***Following Software will be used for Development testing of the proposed project***

- Windows 98SE,Windows 2000, Windows XP, Windows 10 or 11.

-Microsoft internet explorer or Netscape web browser.

- MySQL Server

-Microsoft Office 2013

-Chrome

FUNCTIONAL REQUIREMENTS

a) User Registration
Customers and farmers can register on the platform by providing details such as name, email, mobile number, password, and address.

- Farmers require admin approval before gaining full access.
- Registered users can log in and manage their profiles.
- One-time registration ensures users don’t have to re-enter their details for future use.

b) Category Management
This section provides a list of predefined product categories (e.g., Vegetables, Fruits, Grains).

- Categories help organize products for easier browsing.
- Admin has control to add, update, or delete categories.

c) Product Listing
Farmers can list their products by providing:

- Product name, price, image, description, and associated category.
- Listed products are viewable by customers and available for ordering.
- Customers can filter products by category and search keywords.

d) Cart Management
Customers can add products to their shopping cart for review before checkout.

- Allows modification of quantity and removal of items.
- Stores items temporarily until the user confirms the booking.

SOFTWARE ENGINEERING PARADIGM

The conventional and mostly used software engineering paradigm till now – the Waterfall Model is applied on this project. But according to project complexity and nature slight deviation from the proposed original waterfall model is taken into consideration. The three characteristics of Waterfall Model as linear, rigid, and monolithic are diverted here.

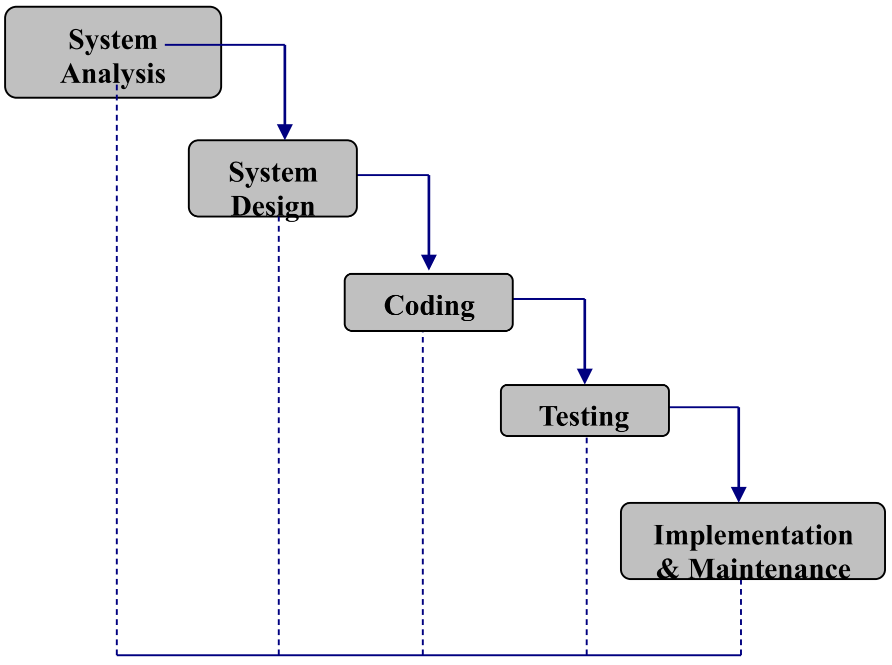
The reasons and assumptions are explained below:

The linearity of Waterfall Model is based on the assumption that software development may proceed linearly from analysis down to coding. But, in this project feedback loops are applied to the immediately preceding stages.

Another underlying assumption of the Waterfall Model is phase rigidity – i.e., that the results of each phase are frozen before proceeding to the next phase. But, this characteristic is not strictly maintained throughout the project. Therefore overlapping of two or more phases is allowed according to needs and judgments.

Finally, the Waterfall Model is monolithic in the sense that all planning is oriented to single delivery date. But since this project has deadline imposed by the organization, planning is done into successive phases. This project is the term-end project and obviously need to submit as soon as possible calendar date. With such considerations is selected the Waterfall model as the project development paradigm.

Water–fall Model with feedback mechanism for the proposed application as Software Engineering paradigm:



ANALYSIS DIAGRAMS

DFD (Data Flow Diagram):

In the late 1970s *data-flow diagrams (DFDs)* were introduced and popularized for structured analysis and design (Game and Sarson 1979). DFDs show the flow of data from external entities into the system, showed how the data moved from one process to another, as well as its logical storage.

A **data flow diagram** (DFD) is a significant modeling technique for analyzing and constructing information processes. DFD literally means an illustration that explains the course or movement of information in a process. DFD illustrates this flow of information in a process based on the inputs and outputs. A DFD can be referred to as a Process Model.

Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out to precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

A designer usually draws a context-level DFD showing the relationship between the entities inside and outside of a system as one single step. This basic DFD can be then disintegrated to a lower level diagram demonstrating smaller steps exhibiting details of the system that is being modeled. Numerous levels may be required to explain a complicated system.

Process

The process shape represents a task that handles data within the application. The task may process the data or perform an action based on the data.

Multiple Processes

The multiple process shape is used to present a collection of sub processes. The multiple processes can be broken down into its sub processes in another DFD.

External Entity

The external entity shape is used to represent any entity outside the application that interacts with the application via an entry point.

Data Flow

The data flow shape represents data movement within the application. The direction of the data movement is represented by the arrow.

Data Store

The data store shape is used to represent locations where data is stored. Data stores do not modify the data, they only store data.

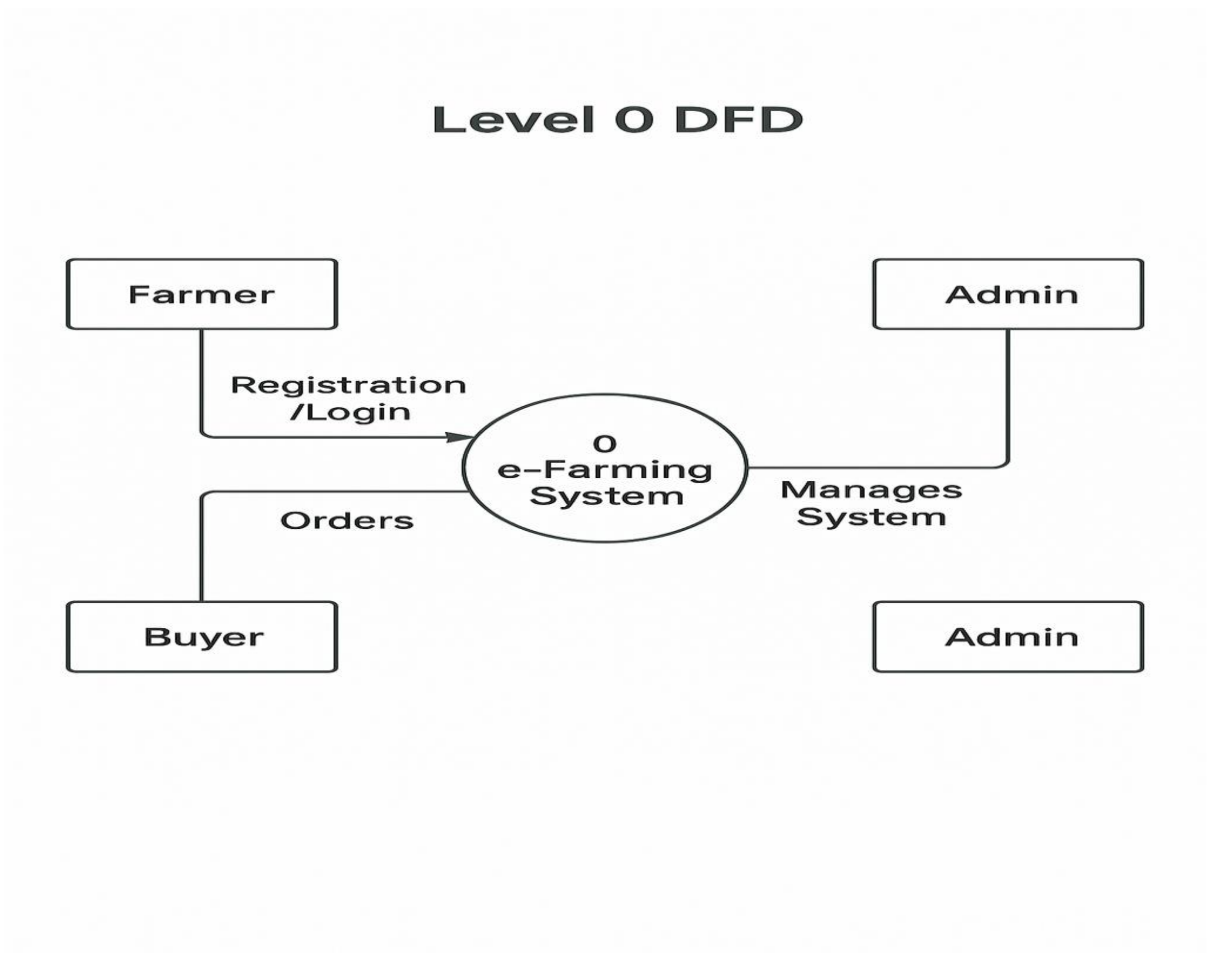
Privilege Boundary

The privilege boundary shape is used to represent the change of privilege levels as the data flows through the application.

Examples of Data Flow Diagrams - These examples demonstrate how to draw data flow diagram.

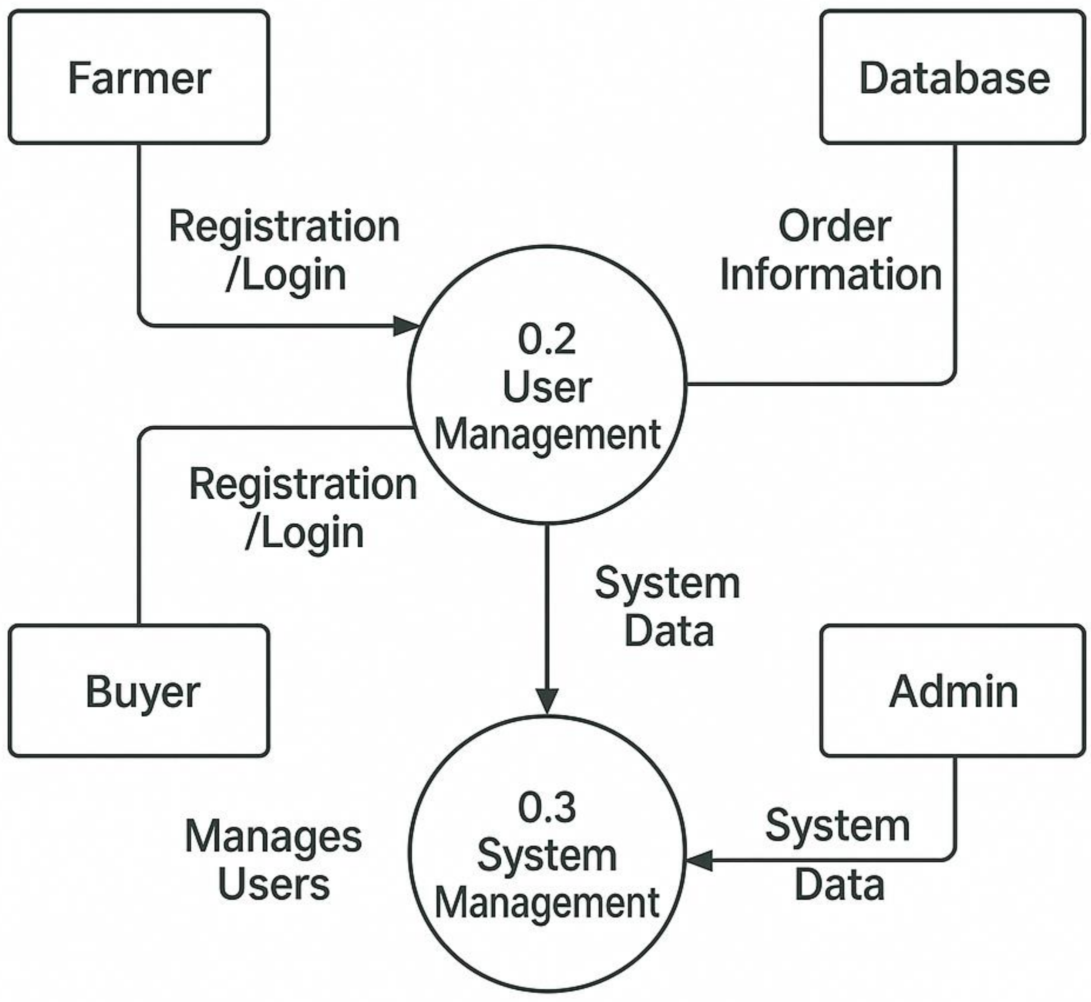
Before it was eventually replaced, a copy machine suffered frequent paper jams and became a notorious troublemaker. Often, a problem could be cleared by simply opening and closing the access panel. Someone observed the situation and flowcharted the troubleshooting procedure used by most people.

CONTEXT LEVEL DIAGRAM

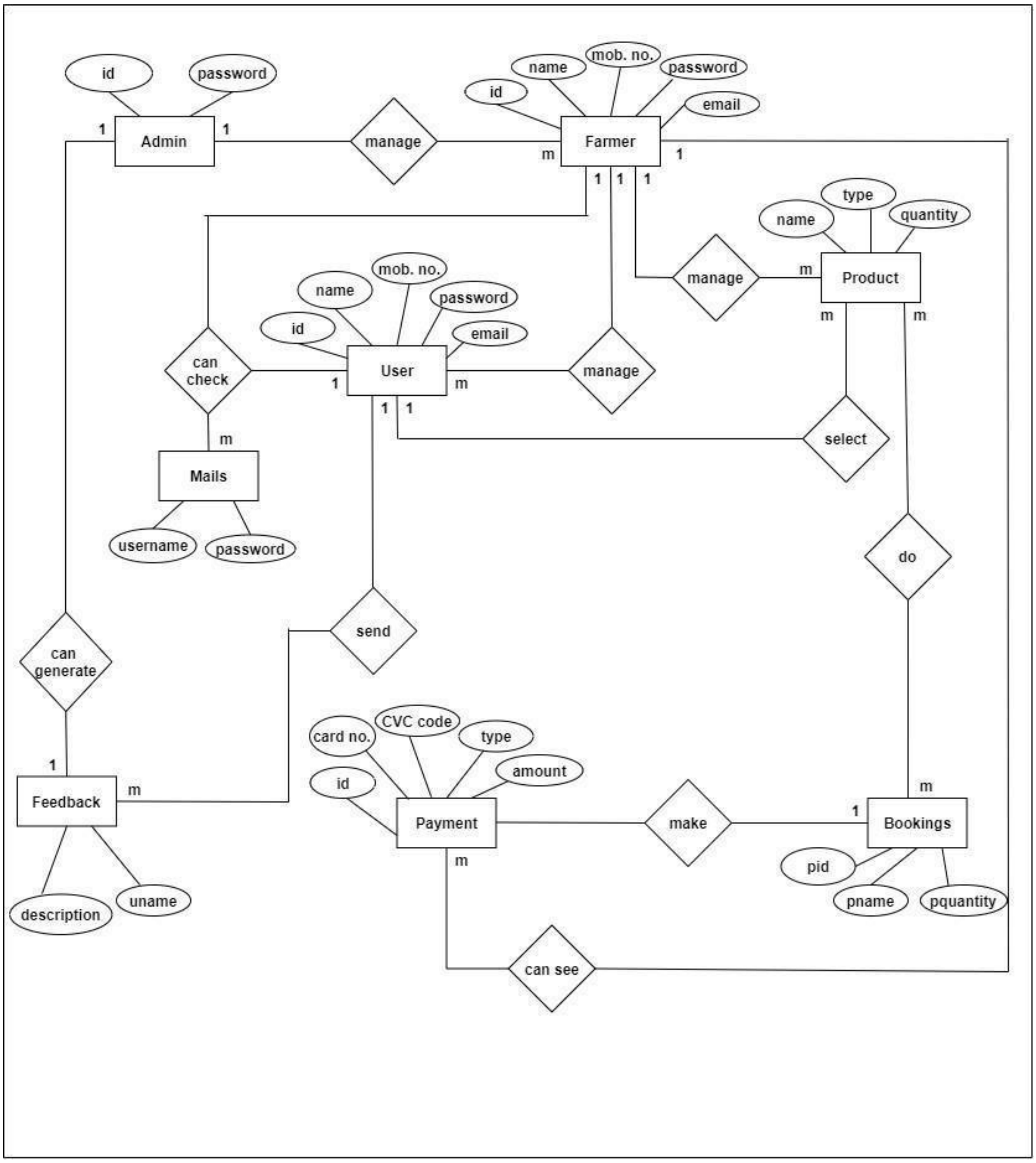


Level 1 DFD

eFarming Project



ER Diagam :-



id
password
Admin
1
1
manage
m
name
mob. no.
password
id
email
Farmer
1
1
1
1
type
name
quantity
manage
m
Product
m
m
name
mob. no.
password
id
email
can check
User
1
m
manage
1
1
select
m
Mails
username
password
do
send
can generate
1
Feedback
m
card no.
CVC code
type
amount
id
Payment
m
make
1
m
Bookings
pid
pname
pquantity
description
uname
can see

SYSTEM DESIGN

MODULES AND THEIR DESCRIPTION 1

The various modules used in the project are as follows:

MODULES AND THEIR DESCRIPTION

a) **User Registration** – This module allows **customers** and **farmers** to register on the EFarming platform.
   - ➢ Users provide information such as name, username, password, phone number, and email.
   - ➢ Farmers provide additional details like address, city, and profile image.
   - ➢ Once registered, users can log in without re-entering details using stored credentials.

b) **Category Management-** This module handles the display and organization of product categories such as Vegetables, Fruits, Grains, and Dairy.
   - ➢ Admin can add, edit, or delete categories.
   - ➢ Customers browse products using categories.
   - ➢ Farmers assign products to categories while listing

c) **Product Information**– This section provides thumbnail view of currently running movies along with necessary information in brief.
   - ➢ Includes product name, image, description, category, and price.
   - ➢ Customers can view all available products and use search or filters.
   - ➢ Each product is linked to the farmer who listed it.

d) **Cart & Booking**– s module allows customers to **add products to a cart** and **place orders (bookings)**.
   - ➢ Cart allows users to review and update selected items.
   - ➢ Upon confirming the cart, a booking is created.
   - ➢ Bookings are linked to the customer and the respective farmers

e) **Admin Control**– The admin panel provides comprehensive **system oversight and configuration tools**.
   - ➢ Approves or blocks users (especially farmers).
   - ➢ Manages categories, monitors bookings, and maintains database health

DATA STRUCTURE OF ALL MODULES:

Table Name: farmerapp_Profile

FIELD NAME	DATA TYPE	DESCRIPTION	REMARKS
ID	Number	User id	Primary Key
DOB	Date	Birth date of user	
City	Varchar	City of user	
Address	Varchar	Address of user	
Contact	Varchar	Contact no. of user	
Image	Varchar	Image of user	
Status	Varchar	Status of User(Approved or Pending)	

Table Name:Farmerapp_Product

FIELD NAME	DATA TYPE	DESCRIPTION	REMARKS
Id	Int		Primary Key
Image	Varchar		
Name	Varchar		
Prise	Int		
Desc	Longtext		
Category_Id	Bigint		Foreign Keys
User_Id	Int		Foreign Keys

Table Name:Farmerapp_Category

FIELD NAME	DATA TYPE	DESCRIPTION	REMARKS
Id	Bigint	Category id	Primary Key
Name	Varchar	Category name	

Table Name:Farmerapp_Sent_Feedback

FIELD NAME	DATA TYPE	DESCRIPTION	REMARKS
ID	Int	Feedback Id	Primary key
Message1	Longtext	Feedback message	
Date	Varchar	Date of feedback	
Profile_Id	Bigint	Prifile id	Foreign Key

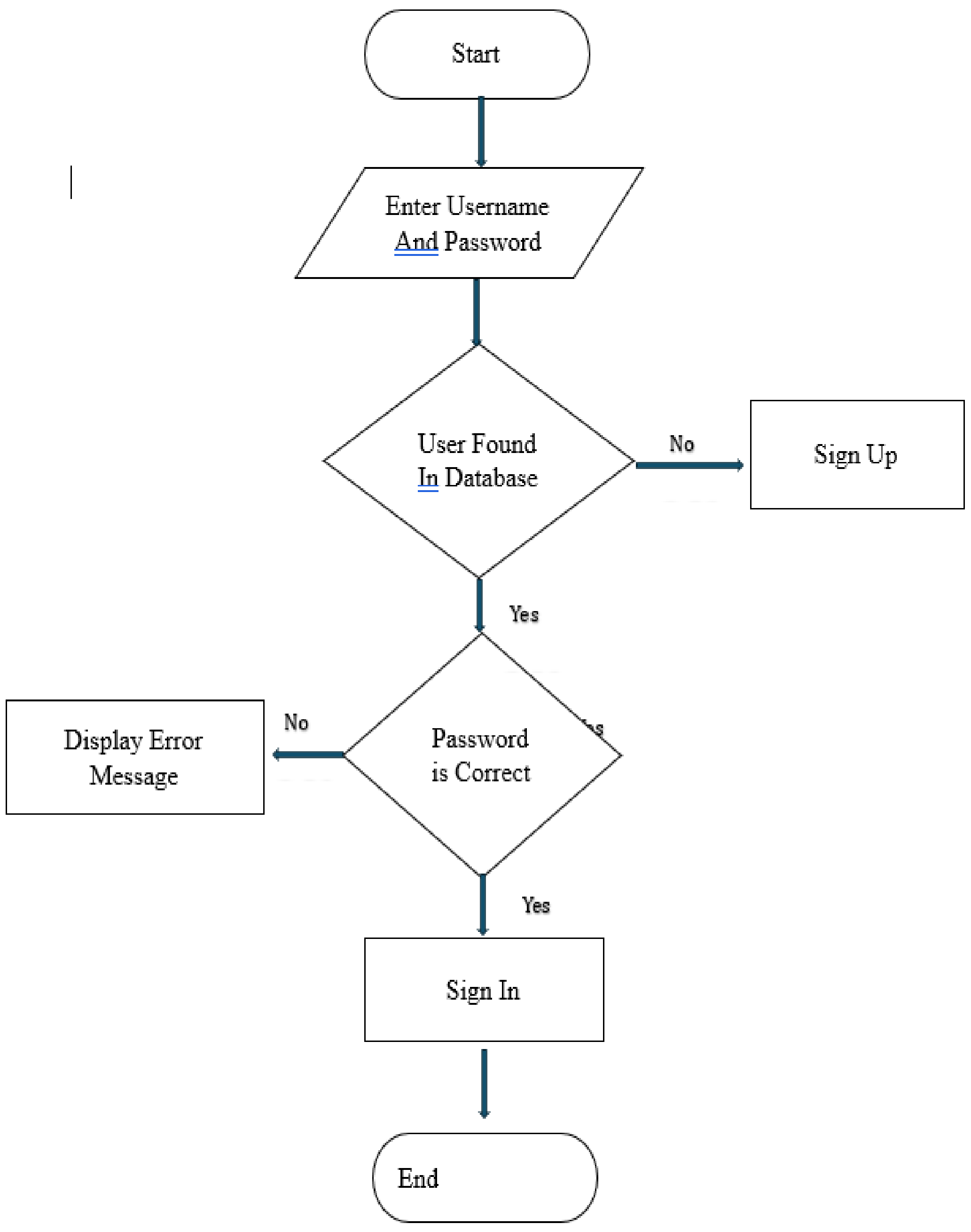
Table Name:Farmerapp_Status

FIELD NAME	DATA TYPE	DESCRIPTION	REMARKS
Id	Int	Id	Primary Key
Name	Varchar	Status of farmer(Approved/Pending)	

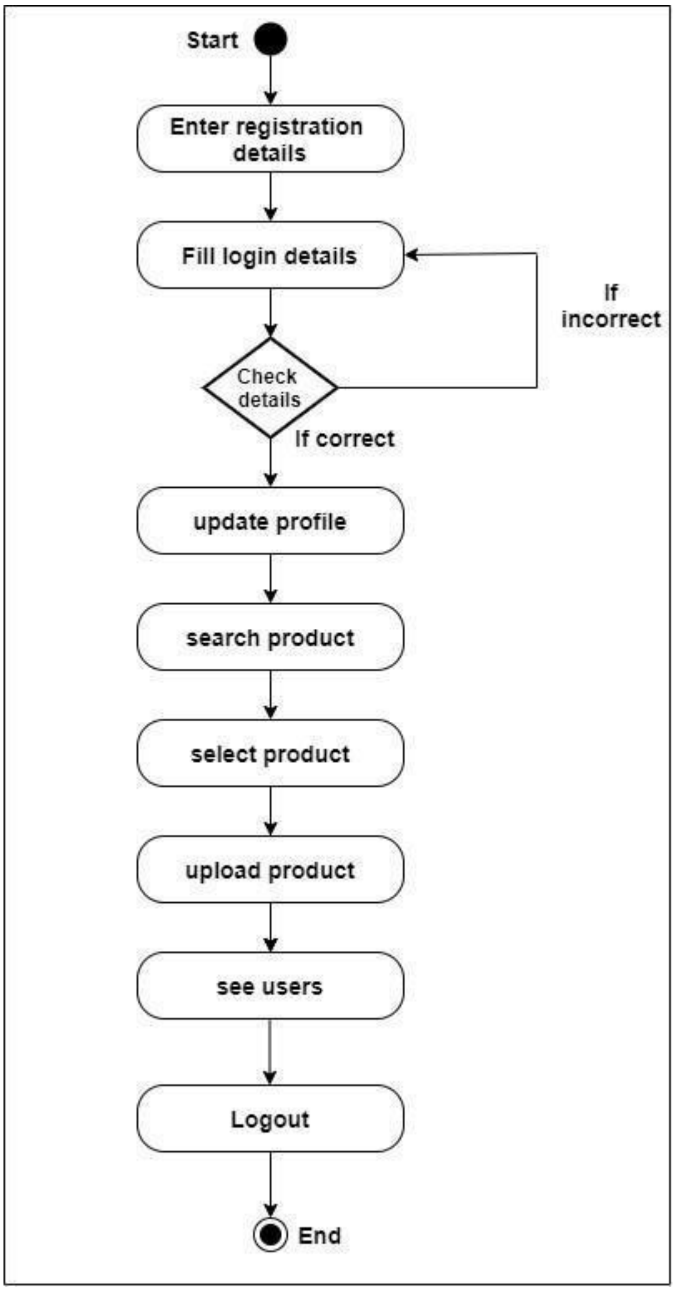
Table Name:Farmerapp_Booking

FIELD NAME	DATA TYPE	DESCRIPTION	REMARKS
Id	Bigint	User Id	Primary Key
Booking_id	Varchar	Booking Id of book	
Quantity	Varchar	Quantity of booking	
Book_Date	Varchar	Booking date	
Total	Int	Toatl Amount	
Profile_Id	Bigint	Profile id	Foreign Key
Status_Id	Bigint	Status id	Foreign Key
Payment_Id	Varchar	Payment id	
Payment_Status	Varchar	Payment Status	
Farmer	Longtext		

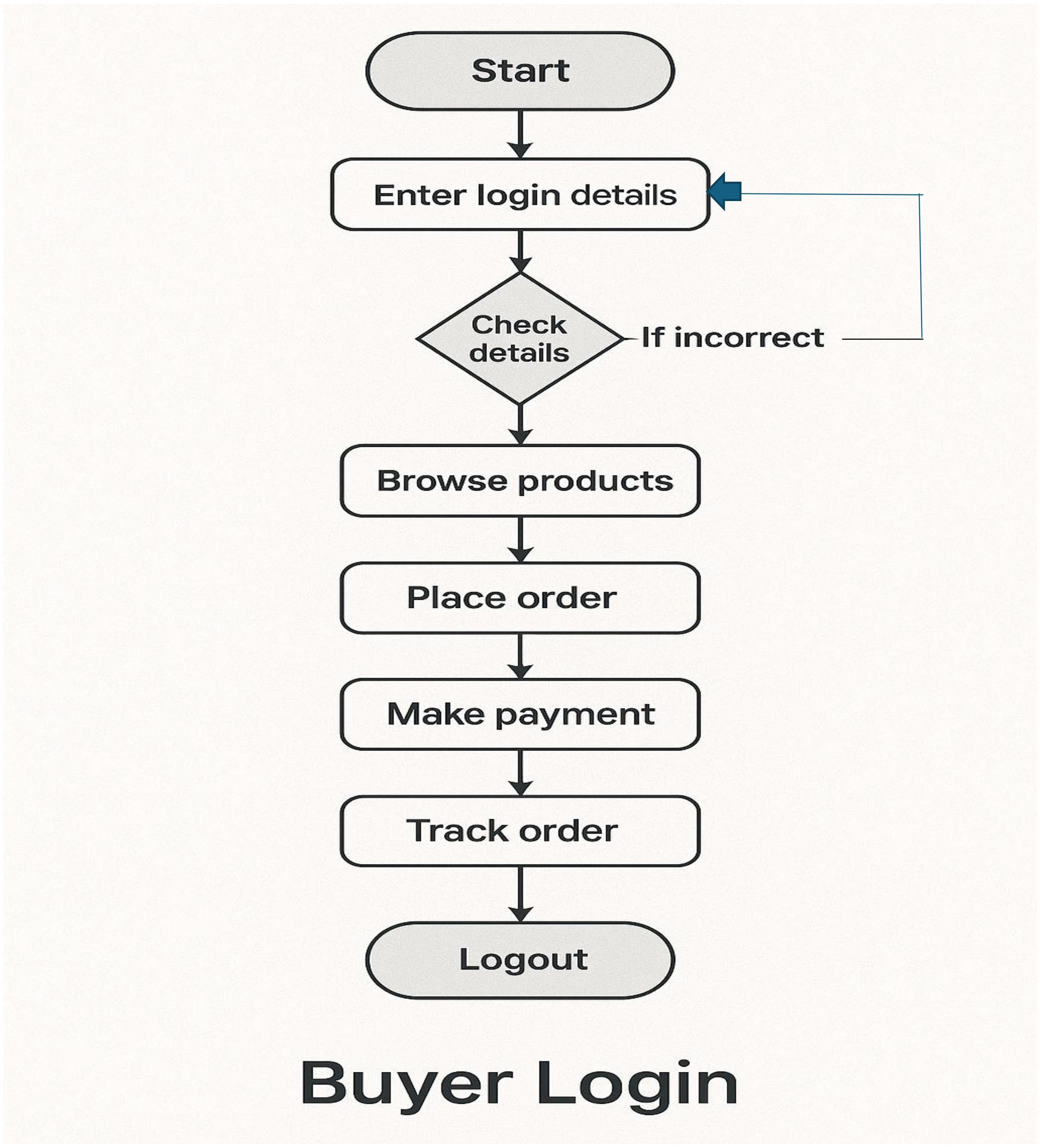
Flow chart for “ User Login ”



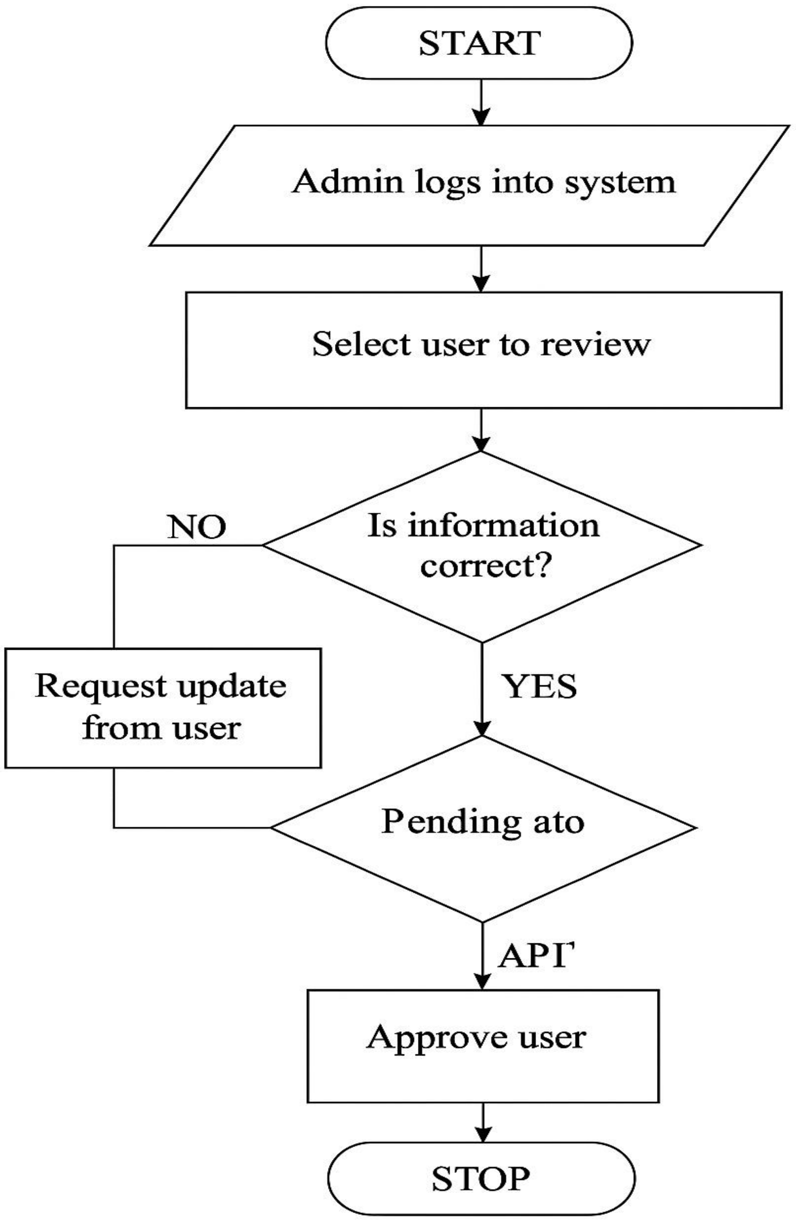
1) Flow chart for "Farmer Login And Add Product"



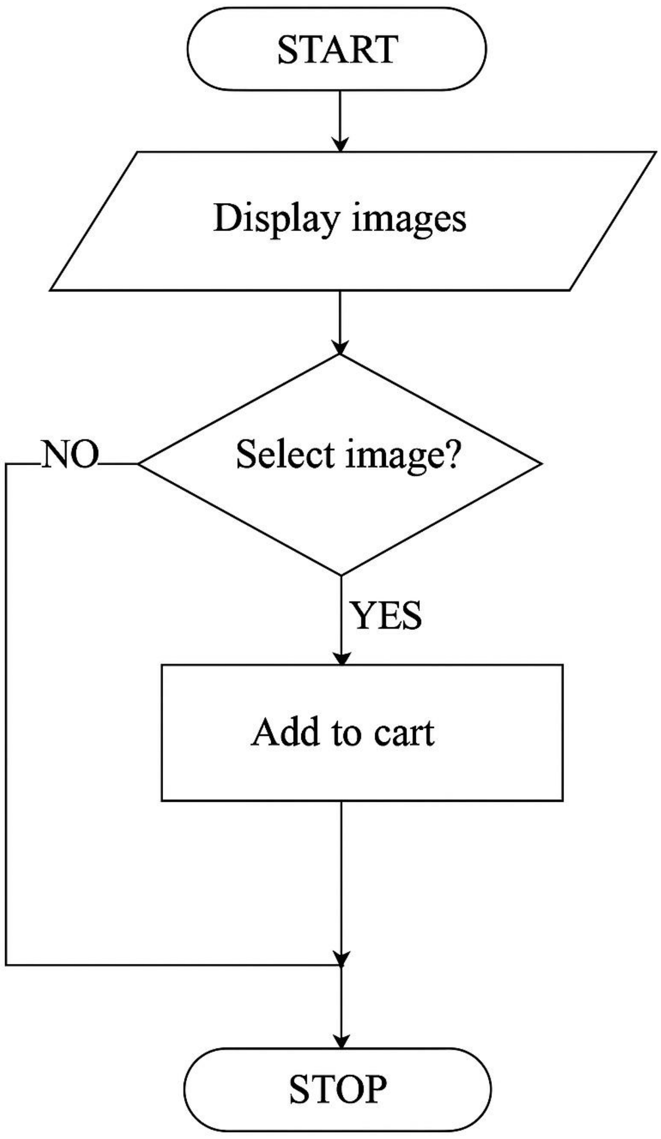
Flowchart for Buyer :-



Flow Chart For “Approvd User”

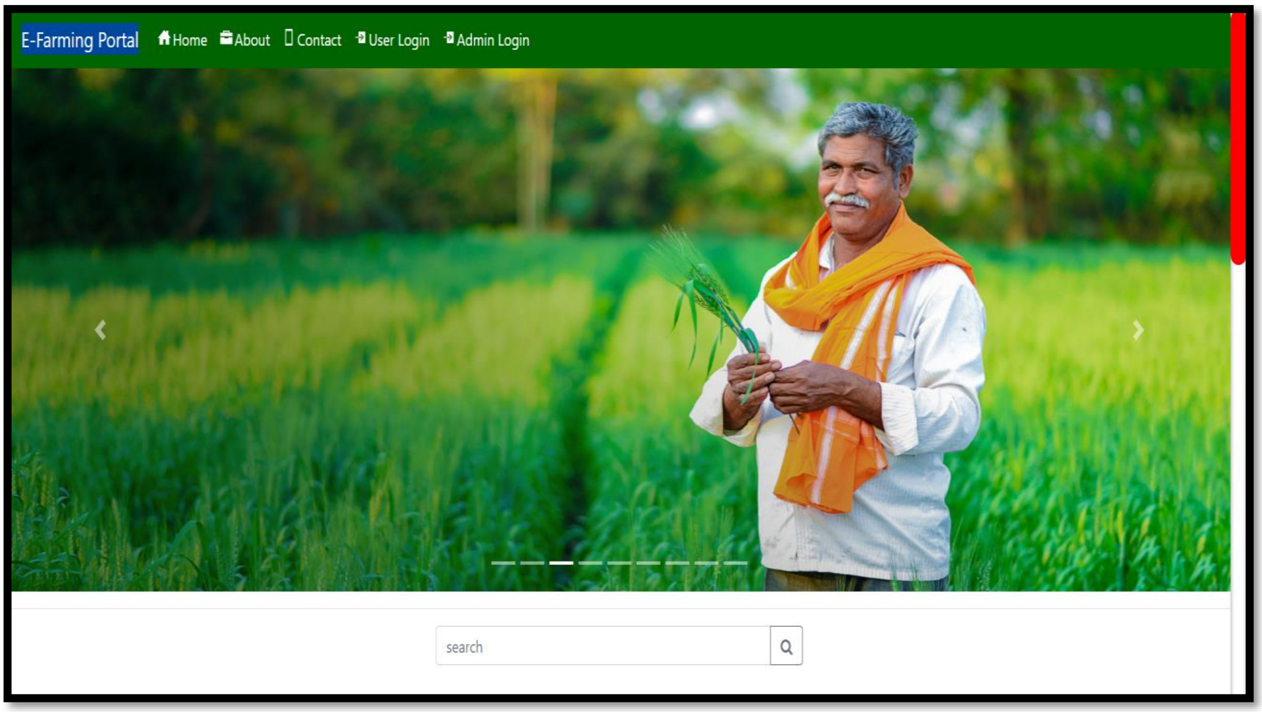


Flow chart for "Add To Cart"



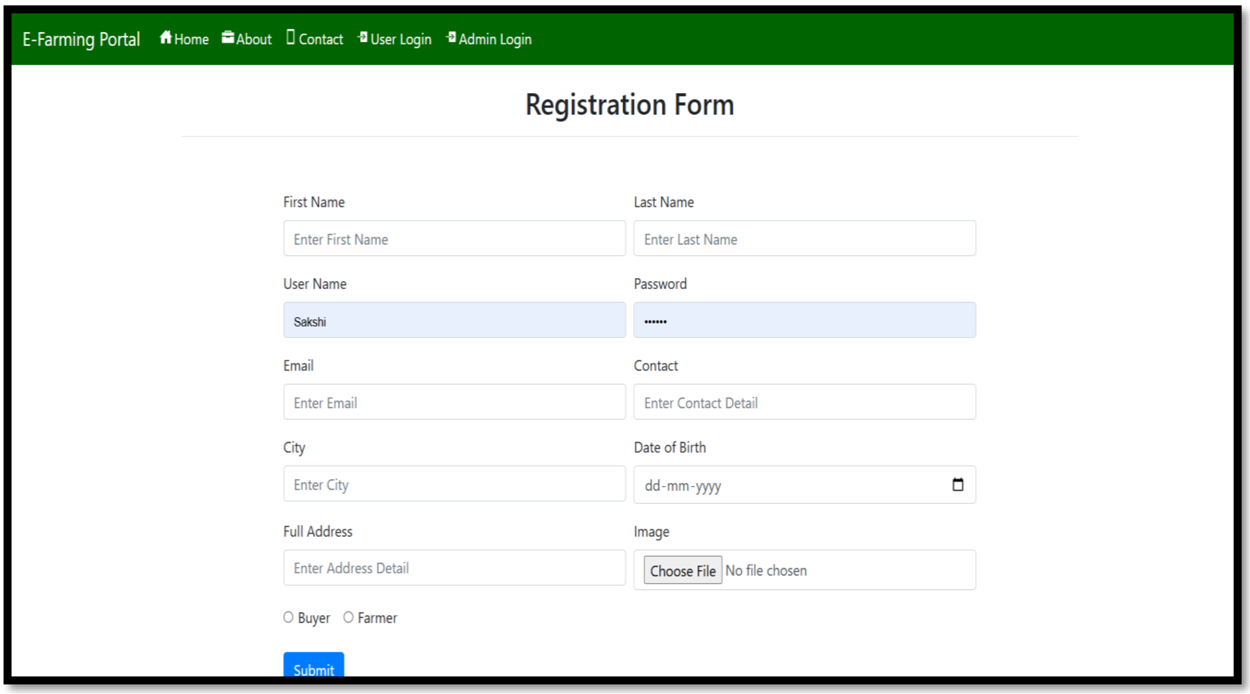
ScreenShot Of Project :-

1.Home Page of E-Farming Project



User Login page

Registration Form



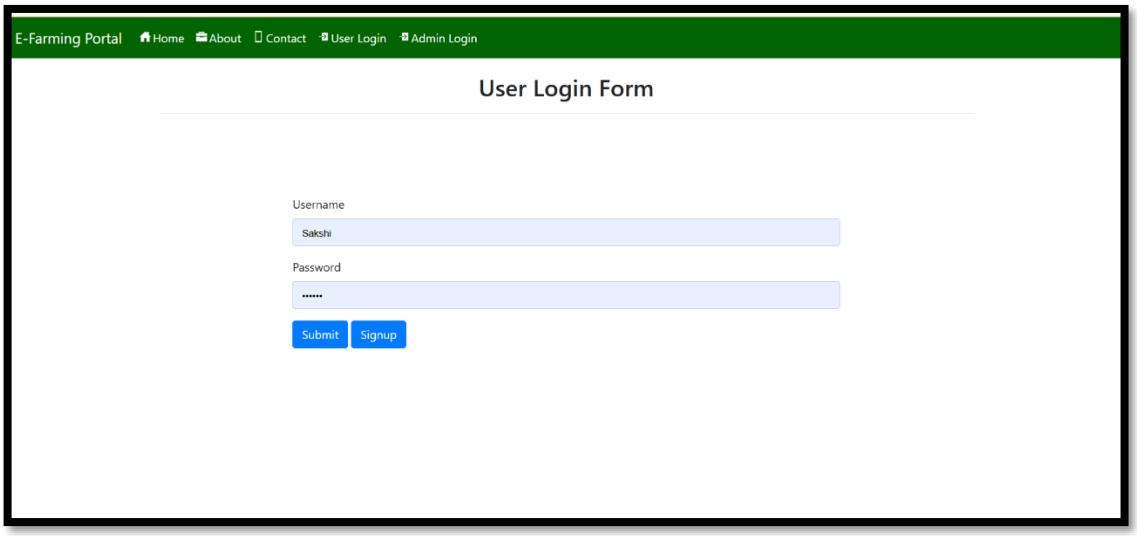
Registration Form

First Name	Last Name
Enter First Name	Enter Last Name
User Name	Password
Sakshi	••••••
Email	Contact
Enter Email	Enter Contact Detail
City	Date of Birth
Enter City	dd-mm-yyyy
Full Address	Image
Enter Address Detail	Choose File No file chosen

○ Buyer ○ Farmer

Submit

User SignIn Page



User Login Form

Username

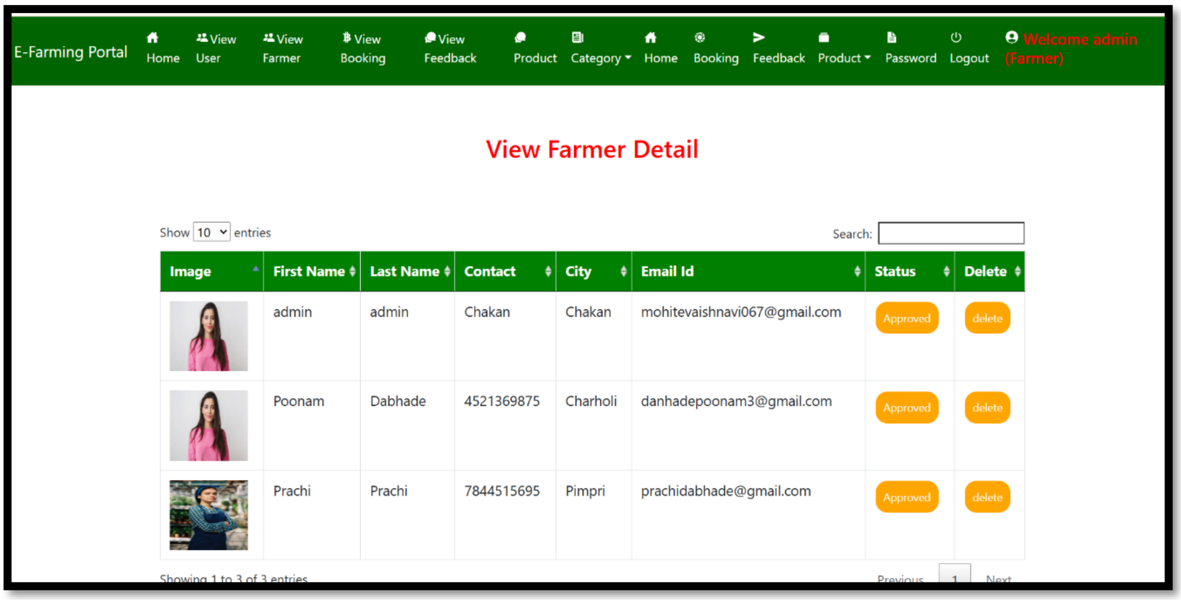
Sakshi

Password

••••••

Admin Page

View farmer



View Farmer Detail

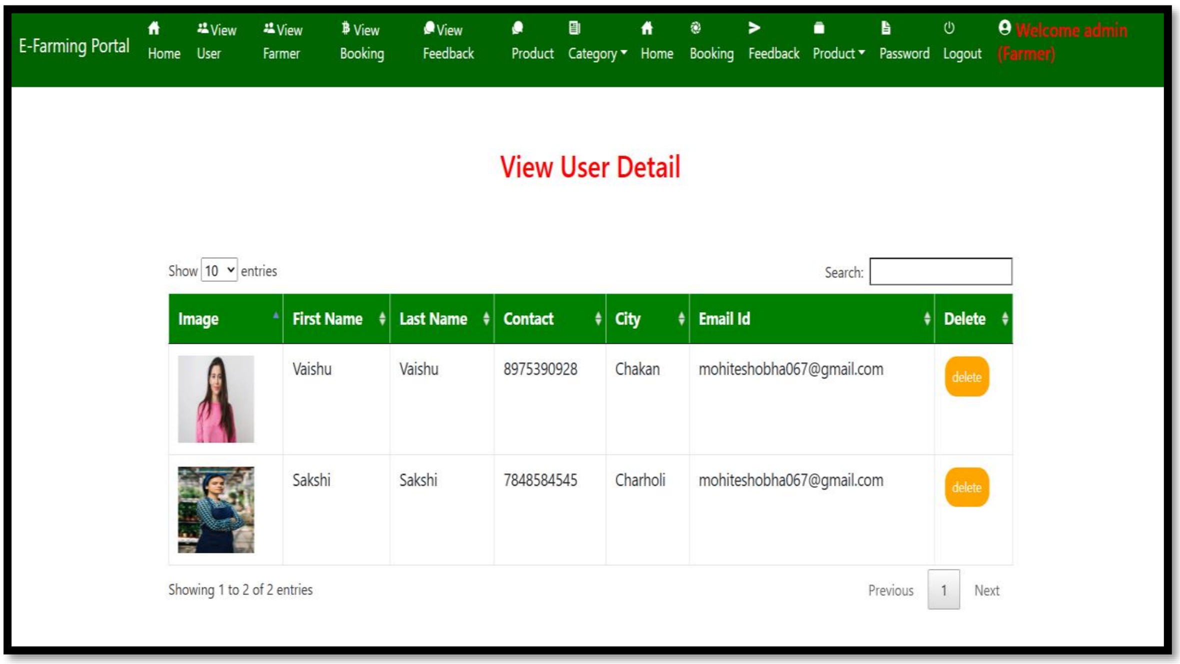
Show 10 entries

Search:

Image	First Name	Last Name	Contact	City	Email Id	Status	Delete
	admin	admin	Chakan	Chakan	mohitevaishnavi067@gmail.com	Approved	delete
	Poonam	Dabhade	4521369875	Charholi	danhadepoonam3@gmail.com	Approved	delete
	Prachi	Prachi	7844515695	Pimpri	prachidabhade@gmail.com	Approved	delete

Showing 1 to 3 of 3 entries

View User



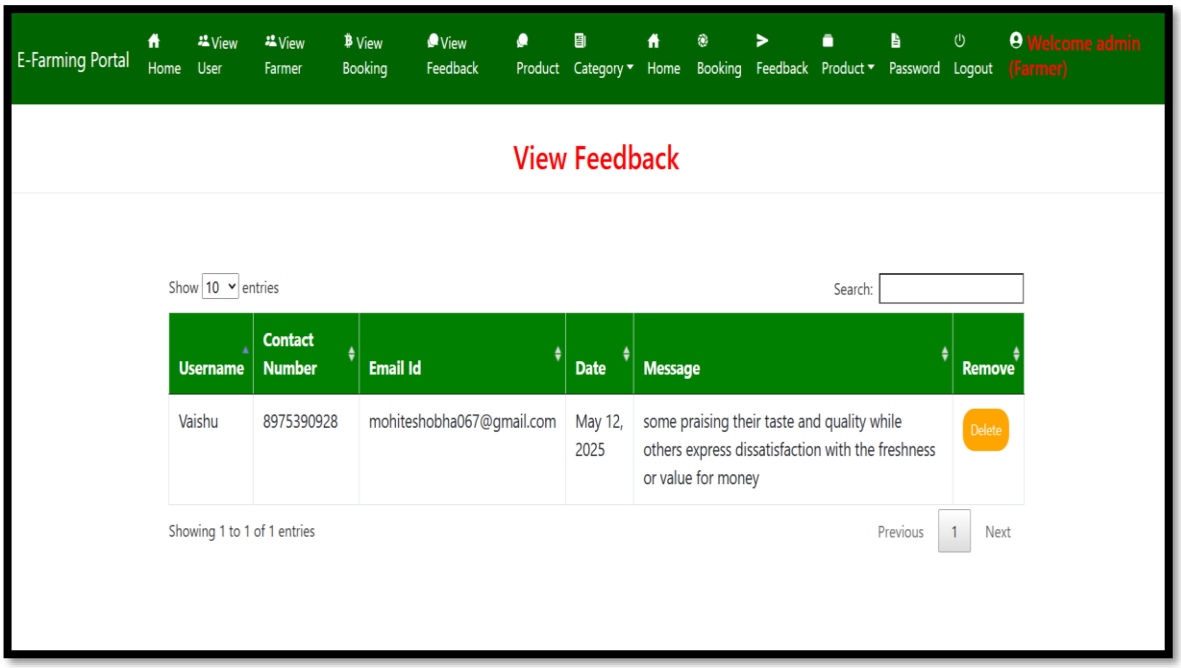
View User Detail

Show 10 entries Search:

Image	First Name	Last Name	Contact	City	Email Id	Delete
	Vaishu	Vaishu	8975390928	Chakan	mohiteshobha067@gmail.com	delete
	Sakshi	Sakshi	7848584545	Charholi	mohiteshobha067@gmail.com	delete

Showing 1 to 2 of 2 entries Previous 1 Next

View Feedback



View Feedback

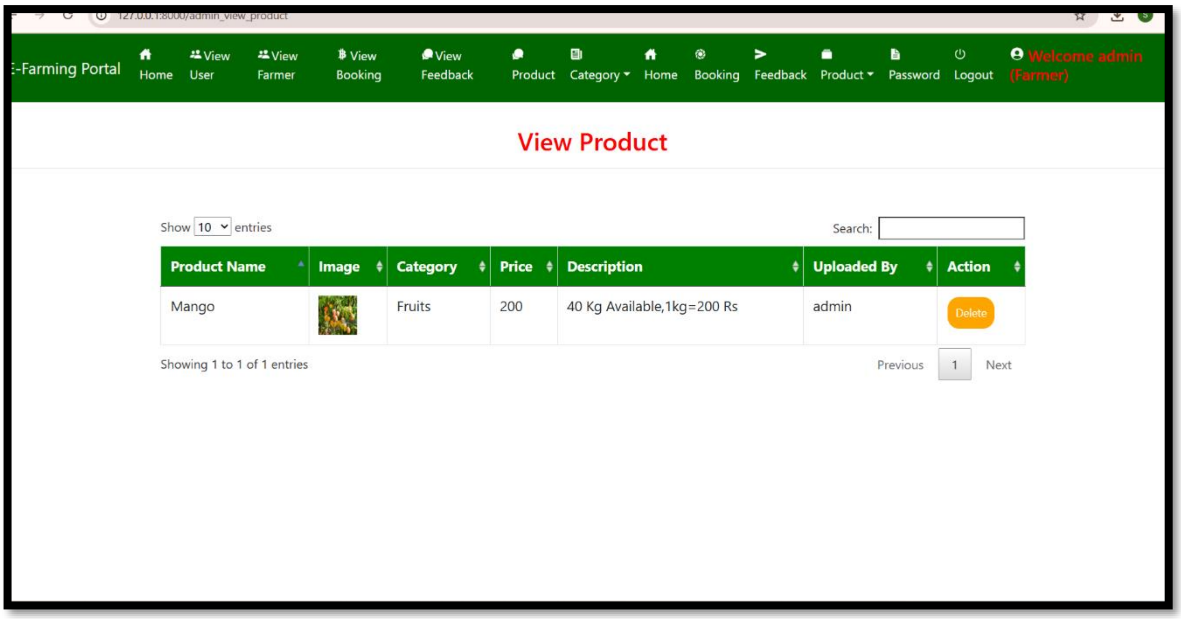
Show 10 entries Search:

Username	Contact Number	Email Id	Date	Message	Remove
Vaishu	8975390928	mohiteshobha067@gmail.com	May 12, 2025	some praising their taste and quality while others express dissatisfaction with the freshness or value for money	Delete

Showing 1 to 1 of 1 entries Previous 1 Next

User Login

Products


127.0.0.1:8000/admin_view_product

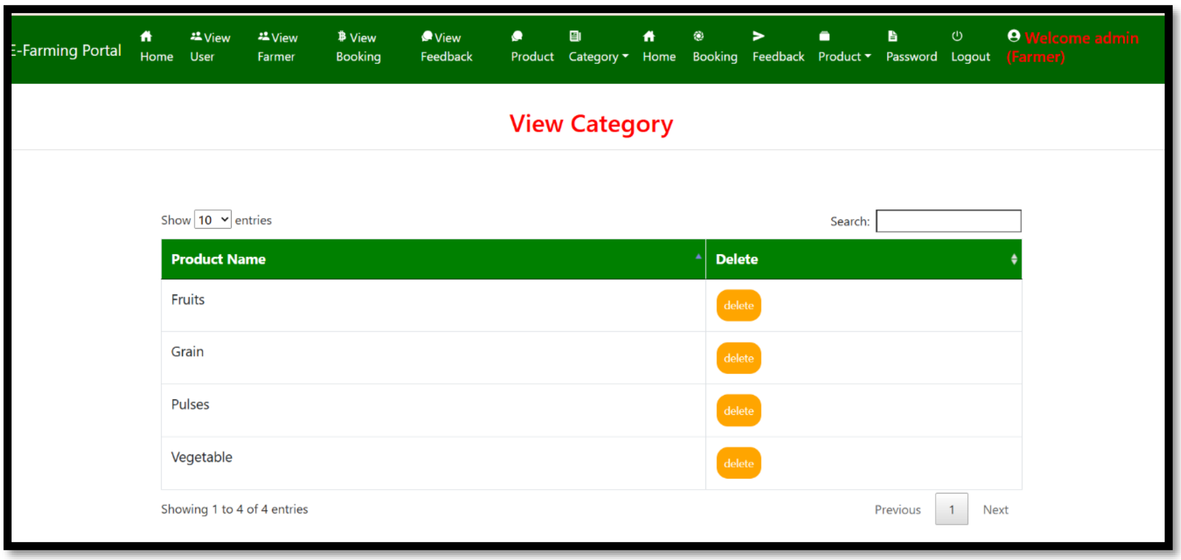
View Product

Show 10 entries Search:

Product Name	Image	Category	Price	Description	Uploaded By	Action
Mango		Fruits	200	40 Kg Available,1kg=200 Rs	admin	Delete

Showing 1 to 1 of 1 entries Previous 1 Next

Category



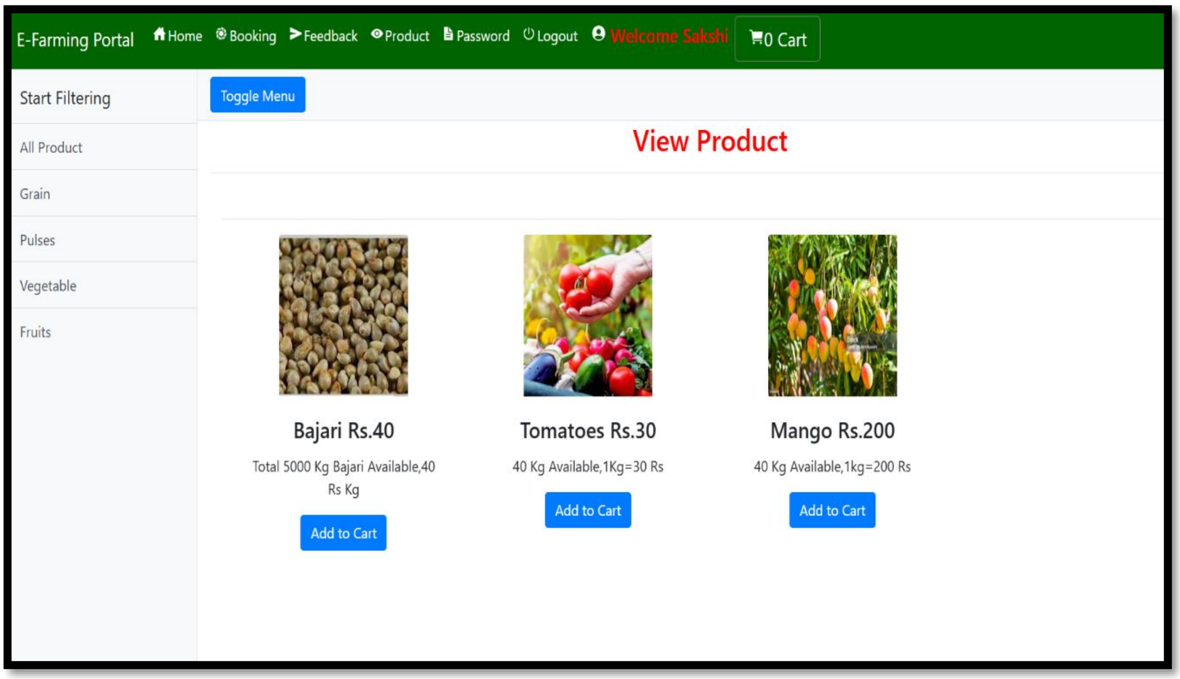
View Category

Show 10 entries Search:

Product Name	Delete
Fruits	delete
Grain	delete
Pulses	delete
Vegetable	delete

Showing 1 to 4 of 4 entries Previous 1 Next

User Login


E-Farming Portal Home Booking Feedback Product Password Logout Welcome Sakshi 0 Cart

Start Filtering

All Product

Grain

Pulses

Vegetable

Fruits

Toggle Menu

View Product

Bajari Rs.40

Total 5000 Kg Bajari Available,40 Rs Kg

Add to Cart

Tomatoes Rs.30

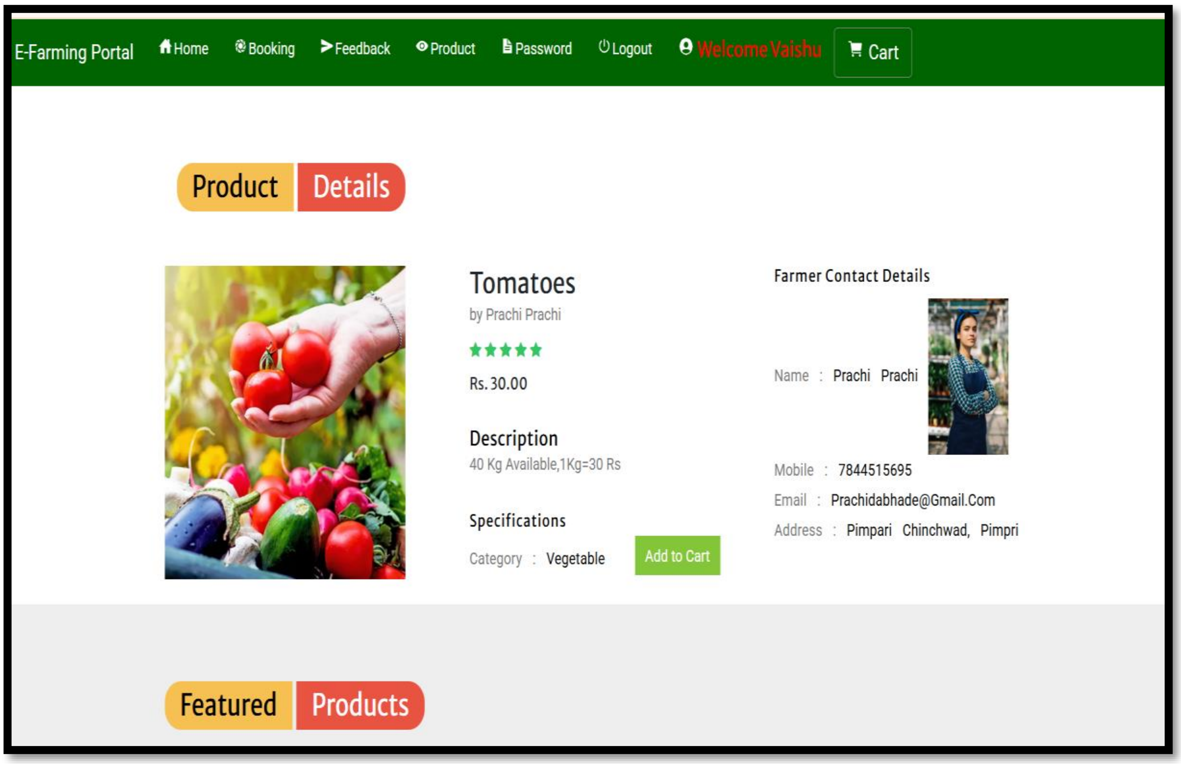
40 Kg Available,1Kg=30 Rs

Add to Cart

Mango Rs.200

40 Kg Available,1kg=200 Rs

Add to Cart


E-Farming Portal Home Booking Feedback Product Password Logout Welcome Vaishu Cart

Product Details

Tomatoes

by Prachi Prachi

★★★★★

Rs. 30.00

Description

40 Kg Available,1Kg=30 Rs

Specifications

Category : Vegetable

Add to Cart

Farmer Contact Details

Name : Prachi Prachi

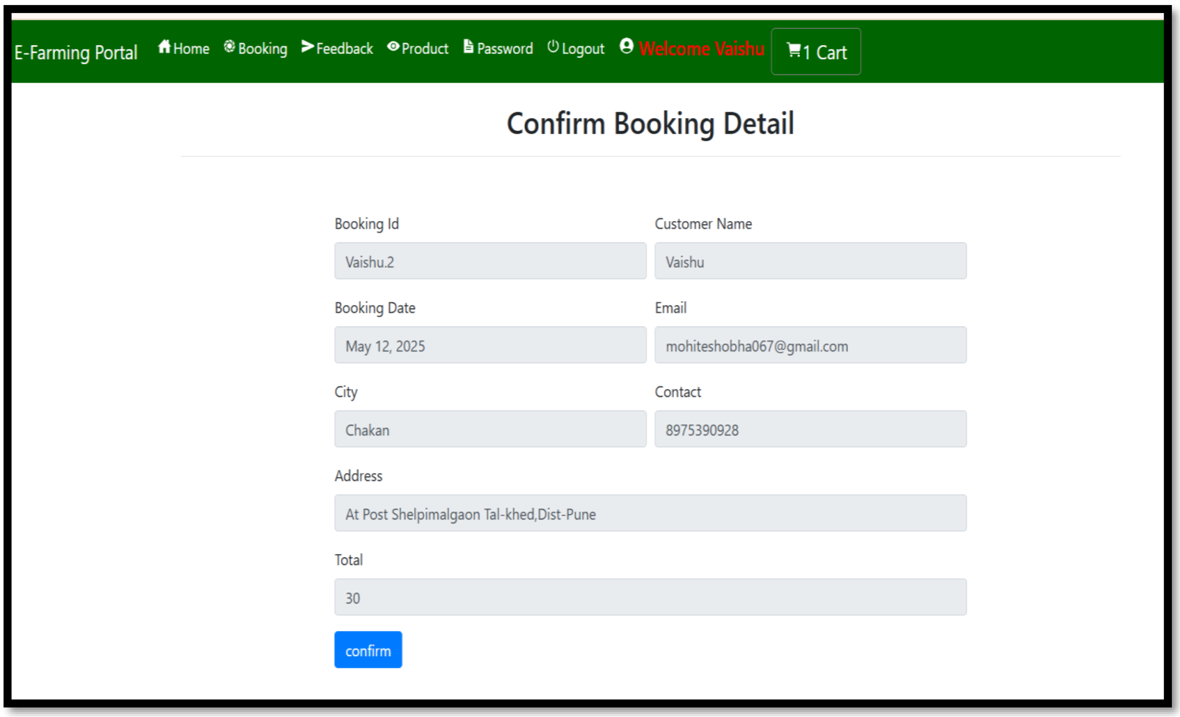
Mobile : 7844515695

Email : Prachidabhade@Gmail.Com

Address : Pimpari Chinchwad, Pimpri

Featured Products

Booking confim



E-Farming Portal Home Booking Feedback Product Password Logout Welcome Vaishu 1 Cart

Confirm Booking Detail

Booking Id	Customer Name
Vaishu.2	Vaishu
Booking Date	Email
May 12, 2025	mohiteshobha067@gmail.com
City	Contact
Chakan	8975390928

Address

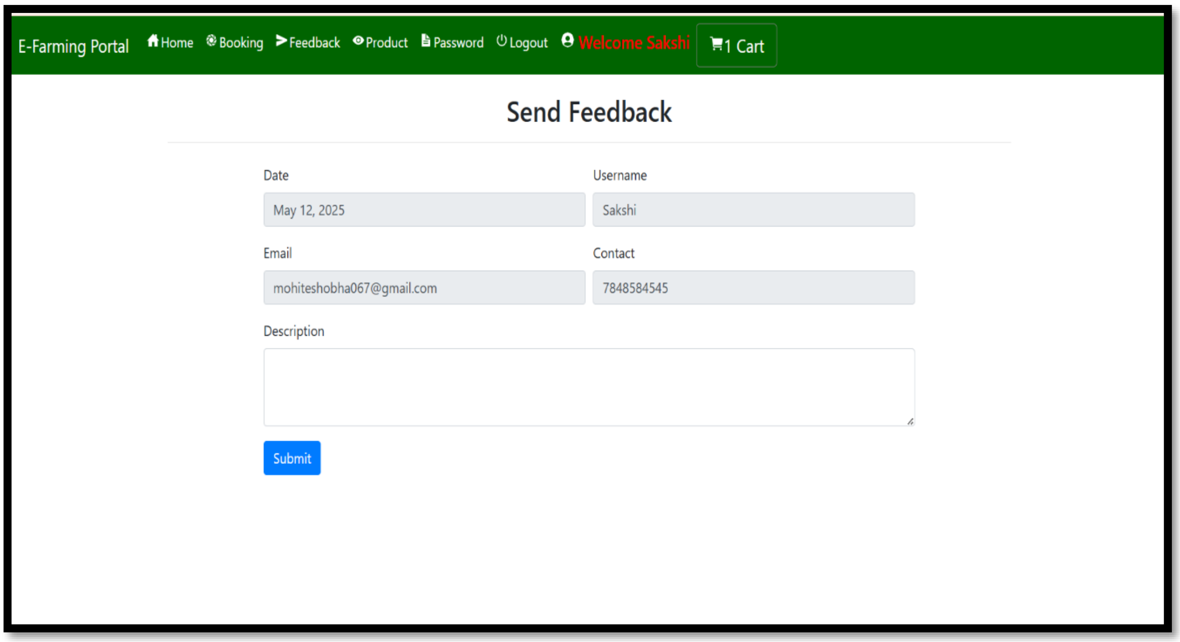
At Post Shelpimalgaon Tal-khed,Dist-Pune

Total

30

confirm

Feedback



E-Farming Portal Home Booking Feedback Product Password Logout Welcome Sakshi 1 Cart

Send Feedback

Date	Username
May 12, 2025	Sakshi
Email	Contact
mohiteshobha067@gmail.com	7848584545

Description

Submit

CODING

Templates

1)About.html

```
 {% extends 'navigation.html' %}

{% load static %}
{% block body %}
<div class="container" style = "margin-top : 150px">
         <h2 class = text-center style = "font-family : 'Monotype Corsiva' ; color : #E6120E">About Us</h2>
     <div class="row">
         <div class="col-sm-5">
               <img src="{% static 'images/natural farming hero desktop.jpg' %}" width=400 height = 400 class="img-responsive">
         </div>
         <div class="col-sm-7" style = "font-weight : bold ; margin-top : 30px">
           <h3>The e-Agriculture Community</h3>
           <p>e-Agriculture is a global Community of Practice, where people from all over the world exchange information, ideas, and resources related to the use of information and communication technologies (ICT) for sustainable agriculture and rural development.</p>

                 <p>e-Agriculture  Community is made up of over 12,000 members from 170 countries and territories, members are information and communication specialists, researchers, farmers, students, policy makers, business people, development practitioners, and others.</p>
         </div>
     </div>
  </div>


{% endblock %}
```

2)Add_Category.html

```
{% extends 'navigation.html' %}
{% block body %}
{% load static %}

{% if error %}
<script>
   alert('One New Category Added Successfully');
</script>
{% endif %}

<div class="container">
   <h2 style="margin-top:20px" align="center">Add Category</h2><hr>

<div class="container-fluid" style="width:70%;margin-top:10%">
<form method="post" action="">
   {% csrf_token %}
  <div class="form-group">
   <label for="exampleInputEmail1">Category Name</label>
        <input type="text" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" name="cat" placeholder="Enter Category">
  </div>

   <button type="submit" class="btn btn-primary">Submit</button>
</form>
</div>
</div>

{% endblock %}
```

3)Add_Product.html

```
{% extends 'navigation.html' %}

{% block body %}

{% load static %}

<div class="container">

   <h2 style="margin-top:20px" align="center">Add Product</h2><hr>



<div class="container-fluid" style="width:80%;margin-top:2%">

<form method="post" action="" enctype="multipart/form-data">

   {% csrf_token %}
```

```
  <div class="form-row">

  <div class="form-group col-md-6">

    <label for="inputEmail4">Product Name</label>

        <input type="text" class="form-control" placeholder="Enter Product Name" name="pname" id="inputEmail4">

  </div>

  <div class="form-group col-md-6">

  <label for="exampleInputPassword1">Product Image</label>

      <input type="file" class="form-control" id="exampleInputPassword1" name="img" value="choose file">

</div></div>

  <div class="form-row">

      <div class="form-group col-md-6">

    <label for="inputState">Category</label>

    <select id="inputState" class="form-control" name="cat" required>

      <option>----Select Category----</option>

        {% for i in cat %}

      <option value="{{i.name}}">{{i.name}}</option>

        {% endfor %}

    </select>

  </div>



  <div class="form-group col-md-6">

    <label for="inputEmail4">Product Price</label>

      <input type="number" class="form-control" name="price" placeholder="Enter Price" id="inputEmail4">

  </div></div>

  <div class="form-group">
```

```
    <label for="exampleFormControlTextarea1">Description</label>

        <textarea class="form-control" id="exampleFormControlTextarea1" name="desc" rows="3"></textarea>

  </div>

  <button type="submit" class="btn btn-primary">Submit</button>

</form>

</div>

</div>


{% if error %}

<script>

    alert('1 Product Added Successfully');

</script>

{% endif %}

{% endblock %}
```

4)Admin_Home.html

```
{% extends 'navigation.html' %}

{% load static %}

{% block body %}

<style>

.glow {

  font-size: 80px;

  color: #fff;

  text-align: center;

  -webkit-animation: glow 1s ease-in-out infinite alternate;

  -moz-animation: glow 1s ease-in-out infinite alternate;

  animation: glow 1s ease-in-out infinite alternate;
}

@-webkit-keyframes glow {

  from {

    text-shadow: 0 0 10px #fff, 0 0 20px #fff, 0 0 30px #e60073, 0 0 40px #e60073, 0 0 50px
#e60073, 0 0 60px #e60073, 0 0 70px #e60073;

  }

   to {

    text-shadow: 0 0 20px #fff, 0 0 30px #ff4da6, 0 0 40px #ff4da6, 0 0 50px #ff4da6, 0 0 60px
#ff4da6, 0 0 70px #ff4da6, 0 0 80px #ff4da6;

  }

}



/* Rounded border */

hr.rounded {

  border-top: 8px solid lightblue;

  border-radius: 5px;

}

</style>

</style>

<link       rel="stylesheet"       href="https://use.fontawesome.com/releases/v5.7.0/css/all.css"
integrity="sha384-
lZN37f5QGtY3VHgisS14W3ExzMWZxybE1SJSEsQp9S+oqd12jhcu+A56Ebc1zFSJ"
crossorigin="anonymous">

<br>

<center>

<div class="container" style="margin-top:10%;height:150px;width:80%">

    <div class="row" style="height:100%">

        <div class="col-md-3">

            <a href="{% url 'view_user' %}"><h1>Total User</h1>
            <h2 style="color:blue">{{total_customer}}</h2>
        <hr class="rounded"></a>
        </div>


        <div class="col-md-3">
            <a href="{% url 'view_farmer' %}"><h1>Total Farmer</h1>
            <h2 style="color:blue">{{seller}}</h2>
    <hr class="rounded"></a>
        </div>


        <div class="col-md-3">
           <a href="{% url 'admin_viewBooking' %}"><h1>Total Booking</h1>
            <h2 style="color:blue">{{total_book}}</h2>
        <hr class="rounded"></a>
        </div>
    <div class="col-md-3">
    <a href="{% url 'admin_view_product' %}">
            <h1>Total Product</h1>
            <h2 style="color:blue">{{total_pro}}</h2>
        <hr class="rounded"></a>
        </div>
    </div>
</div>
</center>
{% endblock %}
```

5)Admin_View_Booking_Detail.html

```
{% extends 'navigation.html' %}

{% block body %}

<center><h2 style="color:red;margin-top:2%">View Booking Detail</h2></center><hr>

{% for i in product %}

{% if i.id in book %}

<center>

 <style>

        .container{

            border:1px solid black;

            border-radius:6px;

            margin-bottom:5%;

        }.container1 ,.contain{

            border:1px solid black;

            border-radius:6px;

            width:70%;

            border:1px solid black;



        }

    </style>

<div class="container" style="margin-top:2%;height:150px;width:80%">

    <div class="row" style="height:90%">

        <div class="col-md-4" style="height:120px">

              <img  src="{{i.image.url}}"  style="width:30%;height:100px;border:1px  solid
darkgray;margin-top:5%">

        </div>
        <div class="col-md-4" style="height:100px">

           <h5 style="margin-top:5%">{{i.name}}</h5><hr>

        <h6>Price : Rs.{{i.price}}</h6>

        <p>{{i.desc}}</p>

        </div>

           <div class="col-md-4" style="height:100px;">

               <h4 style="margin-top:15%;color:blue">Booked Successfully</h4>

        </div>

    </div>

</div>

    </center>

{% endif %}

{% endfor %}

<div class="container" style="border:1px solid white">

<div class="row">


    <div class="col-md-6" style="height:100px;">

                <center><a style="margin-top:1%;width:100%;border:1px solid black" class="btn
btn-default btn-lg text-dark text-capitalize">Total : {{total.total}}</a></center>

           </div>

</div>

<center><h2 style="color:red;margin-top:2%">Customer Detail</h2></center><hr>

<div class="row">

 <div class="col-md-6" style="height:100px;"><hr>

        <h4>Name :  {{profile.user.first_name}} {{profile.user.last_name}}</h4><hr>

        <h4>Email :  {{profile.user.email}}</h4><hr>

        <h4>Address : {{profile.address}}</h4><hr>
        <h4>Contact : {{profile.contact}}</h4><hr>
    </div>
    <div class="col-md-6" style="height:100px;"><hr>
        <h4>  <img src="{{profile.image.url}}" style="width:150px;height:140px"></h4>
    </div>
</div>
</div>
{% endblock %}
```

6)Admin_View_Product.html

```
{% extends 'navigation.html' %}
{% block body %}
<center><h2 style="color:red;margin-top:2%">View Product</h2></center><hr>
 <div class="container" style="margin-top:4%">
<table class="table table-bordered" id="myTable">
  <thead>
  <tr>
    <th>Product Name</th>
    <th>Image</th>
      <th> Category</th>
    <th>Price</th>
    <th> Description</th>
    {% if request.user.is_staff %}<th> Uploaded By</th>{% endif %}
    <th>Action</th>
  </tr>

  </thead>

   <tbody>



    {% for i in pro %}

  <tr>

    <td>{{i.name}}</td>

    <td><img src="{{i.image.url}}" style="width:50px;height:50px"></td>

      <td>{{i.category.name}}</td>

      <td>{{i.price}}</td>

      <td>{{i.desc}}</td>

      {% if request.user.is_staff %}<td>{{i.user.username}}</td>{% endif %}

       <td><a href="{% url 'delete_product' i.id %}" onclick="return confirm('Are you
sure?')"><button class="button button1">Delete</button></a></td>

  </tr>

    {% endfor %}

    </tbody>

{{count}}

</table>

 </div>

<script>

  {% for i in message %}

  alert("{{i}}")

  {% endfor %}

</script>{% endblock %}
```

EFarming

Settings.py

```
from pathlib import Path

import os

BASE_DIR = Path(__file__).resolve().parent.parent

SECRET_KEY = 'django-insecure-3^=)p!zmnl!e1svd_$91_^hb+yt_wjkc(9@u*080x$j*2vu_3f'

DEBUG = True

ALLOWED_HOSTS = []

INSTALLED_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'farmerapp', ]

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

]
```

```
ROOT_URLCONF = 'Efarming.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'Efarming.wsgi.application'

# MySQL Database Configuration
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'Efarm',
        'USER': 'vaishu',
        'PASSWORD': '123',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_TZ = True
STATIC_URL = 'static/'
```

```
MEDIA_URL = '/media/'

MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

2) Urls.py

```
from django.contrib import admin

from django.urls import path

from farmerapp.payment import original_payment, payment_status

from farmerapp.views import *

from django.conf.urls.static import static

urlpatterns = [

    path('admin/', admin.site.urls),

    path('',Home,name="home"),

    path('signup',Signup,name="signup"),

    path('about/',About,name='about'),

    path('contact/',Contact,name='contact'),

    path('login/',Login,name="login"),

    path('logoutuser',Logout,name="logout"),

    path('view_user',View_user,name="view_user"),

    path('add_product',Add_Product,name="add_product"),

    path('view_feedback', View_feedback, name='view_feedback'),

    path('view_product/<int:pid>/', View_prodcut, name='view_product'),

    path('admin_view_product', Admin_View_product, name='admin_view_product'),

    path('login_admin',Admin_Login,name="login_admin"),

    path('admin_viewBooking', Admin_View_Booking, name='admin_viewBooking'),

    path('view_categary/', View_Categary, name='view_categary'),
    path('add_categary', Add_Categary, name='add_categary'),

    path('add_cart(?P<int:pid>)', Add_Cart, name='add_cart'),

    path('delete_product(?P<int:pid>)', delete_product, name='delete_product'),

    path('delete_user(?P<int:pid>)', delete_user, name='delete_user'),

    path('delete_feedback(?P<int:pid>)', delete_feedback, name='delete_feedback'),

    path('cart', view_cart, name='cart'),

    path('payment(?P<book>[0-9]+)', payment, name='payment'),

  path('delete_booking/(?P<str:pid>)/(?P<bid>[0-9]+)',delete_booking,
name='delete_booking'),

     path('delete_admin_booking/(?P<str:pid>)/(?P<bid>[0-9]+)',    delete_admin_booking,
name='delete_admin_booking'),

            path('booking_detail/(?P<str:pid>)/(?P<bid>[0-9]+)',           booking_detail,
name='booking_detail'),

           path('admin_booking_detail/(?P<str:pid>)/(?P<bid>[0-9]+)/(?P<uid>[0-9]+)',
admin_booking_detail, name='admin_booking_detail'),

    path('Edit_status/(?P<str:pid>)/(?P<bid>[0-9]+)', Edit_status, name='Edit_status'),

    path('remove_cart(?P<int:pid>)', remove_cart, name='remove_cart'),

    path('booking/<str:pid>/', Booking_order, name="booking"),

    path('view_booking', View_Booking, name='view_booking'),

    path('profile/<int:pid>', profile, name='profile'),

    path('edit_profile', Edit_profile, name='edit_profile'),

    path('delete_category(?P<int:pid>)', delete_category, name='delete_category'),

    path('admin_home', Admin_Home, name='admin_home'),

    path('change_password', Change_Password, name="change_password"),

    path('payment_status/', payment_status, name="payment_status"),

    path('original_payment/', original_payment, name="original_payment"),

    path('view_farmer/', View_farmer, name="view_farmer"),

    path('change_user_status/<int:pid>', change_user_status, name="change_user_status"),
    path('send_feedback/(?P<pid>[0-9]+)', Feedback, name='send_feedback'),

    path('product_detail/<int:pid>', product_detail, name='product_detail'),

    path('farmer_detail/<int:pid>', farmer_detail, name='farmer_detail'),

]+static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

3) Setting.py

```
from pathlib import Path

import os

BASE_DIR = Path(__file__).resolve().parent.parent

SECRET_KEY                          =                          'django-insecure-
3^=)p!zmnl!e1svd_$91_^hb+yt_wjkc(9@u*080x$j*2vu_3f'

DEBUG = True

ALLOWED_HOSTS = []

INSTALLED_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'farmerapp', ]

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
ROOT_URLCONF = 'Efarming.urls'
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages', ],   }, },]
WSGI_APPLICATION = 'Efarming.wsgi.application'
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'Efarm',
        'USER': 'vaishu',
        'PASSWORD': '123',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
```

```
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_TZ = True
STATIC_URL = 'static/'
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

Farmerapp

1) Admin.py

```
from django.contrib import admin
from .models import *

# Register your models here.

admin.site.register(Category)

admin.site.register(Product)

admin.site.register(Booking)

admin.site.register(Profile)

admin.site.register(Send_Feedback)

admin.site.register(Status)

admin.site.register(Cart)
```

2) apps.py

```
from django.apps import AppConfig

class FarmerappConfig(AppConfig):

    default_auto_field = 'django.db.models.BigAutoField'

    name = 'farmerapp'
```

3) models.py

```
from django.db import models
from django.contrib.auth.models import User
# Create your models here.
class Category(models.Model):
    name = models.CharField(max_length=30, null=True)

    def __str__(self):
        return self.name

class Product(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE, null=True)
    category = models.ForeignKey(Category, on_delete=models.CASCADE, null=True)
    image = models.FileField(null=True)
    name = models.CharField(max_length=30, null=True)
    price = models.IntegerField(null=True)
    desc = models.TextField(null=True)

    def __str__(self):
        return self.category.name+"--"+self.name

class Status(models.Model):
    name = models.CharField(max_length=20, null=True)

    def __str__(self):
        return self.name

class Profile(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE, null=True)
    dob = models.DateField(null=True)
    city = models.CharField(max_length=30, null=True)
    address = models.CharField(max_length=50, null=True)
    contact = models.CharField(max_length=10, null=True)
    user_type = models.CharField(max_length=10, null=True)
    status =  models.CharField(max_length=30, null=True, default="Pending")
    image = models.FileField(null=True)

    def __str__(self):
        return self.user.username

class Cart(models.Model):
    profile = models.ForeignKey(Profile,on_delete=models.CASCADE,null=True)
    product = models.ForeignKey(Product, on_delete=models.CASCADE, null=True)
    def __str__(self):
        return self.profile.user.username + " . " + self.product.name

class Booking(models.Model):
    status = models.ForeignKey(Status, on_delete=models.CASCADE, null=True)
    profile = models.ForeignKey(Profile, on_delete=models.CASCADE, null=True)
    farmer =  models.TextField(null=True)
    booking_id = models.CharField(max_length=200,null=True)
    quantity = models.CharField(max_length=100,null=True)
    # quantity_json = models.JSONField(null=True,default=dict)
    payment_id = models.CharField(max_length=200,null=True)
    payment_status = models.CharField(max_length=200,null=True, default="Pending")
    book_date = models.CharField(max_length=30, null=True)
    total = models.IntegerField(null=True)

    def __str__(self):
        return self.book_date+" "+self.profile.user.username


class Send_Feedback(models.Model):
    profile = models.ForeignKey(Profile, on_delete=models.CASCADE, null=True)
    message1 = models.TextField(null=True)
    date = models.CharField(max_length=30, null=True)

    def __str__(self):
        return self.profile.user.username
```

4) payment.py

```
from django.http import HttpResponseRedirect

from django.shortcuts import redirect

from instamojo_wrapper import Instamojo

from farmerapp.models import Booking

api = Instamojo(api_key="test_1bc15839a9776d7ec22a192d55c", auth_token="test_41c435fac3f8400c8aa675da674", endpoint='https://test.instamojo.com/api/1.1/');

# settings.SITE_DOMAIN + "/payment_status/?book_id=" + str(request.GET.get('book_id'))

# https://e0f737b7b539.ngrok.io/payment_status/?book_id=" + str(request.GET.get('book_id'))

# Create a new Payment Request

# response = api.payment_request_create(

#       amount='100',

#       purpose='Ecommerce Grocery Shop',

#       send_email=True,

#       email="bhuwanbhaskar761@gmail.com",

#       redirect_url= "https://linkedin.com/in/bhuwanbhaskar761"

#       )

# print(response['payment_request']['id'])

def original_payment(request):

    print(request.GET.get('total'),"AAAAAAAAAAAAAAA")

    book = Booking.objects.get(id=request.GET.get('book_id'))

    response = api.payment_request_create(

        amount=request.GET.get('total'),

        purpose='Ecommerce Grocery Shop',
        send_email=True,

        email="bhuwanbhaskar761@gmail.com",

            redirect_url="https://e0f737b7b539.ngrok.io/payment_status/?book_id="   +
str(request.GET.get('book_id')),

        )

    # print the long URL of the payment request.



    print(response)

    payment_id = response['payment_request']['id']

    book.payment_id = payment_id

    book.save()

    return HttpResponseRedirect(response['payment_request']['longurl'])

def payment_status(request):

    book = Booking.objects.get(id=request.GET.get('book_id'))

    response = api.payment_request_status(book.payment_id)

    status = response['payment_request']['status']

    book.payment_status = status

    print(status)

    return re

direct('view_booking')
```

5) Views.py

```
from random import shuffle, random

from django.contrib import messages

from django.http import HttpResponse

from instamojo_wrapper import Instamojo
from django.db.models import Q

from django.shortcuts import render,redirect

from django.contrib.auth.models import User

from django.contrib.auth import authenticate,logout,login

from .models import *

from django.conf import settings

from datetime import date

# Create your views here.

def Home(request):

    search = request.GET.get('search',0)

        search_pro    =    Product.objects.filter(Q(name__icontains    =    search)    |
Q(category__name__icontains = search))

    cat = ""

    pro = ""

    cat = ""

    num = 0

    num1 = 0

    cat = Category.objects.all()

    pro = Product.objects.all()

    num = []

    num1 = 0

    product = None

    try:

        user = User.objects.get(id=request.user.id)

        profile = Profile.objects.get(user=user)

        cart = Cart.objects.filter(profile=profile)

        product = recommended_product(request)
        for i in cart:
            num1 += 1
    except:
        pass
    a = 1
    li = []
    for j in pro:
        b = 1
        for i in cat:
            if i.name == j.category.name:
                if not j.category.name in li:
                    li.append(j.category.name)
                    if b == 1:
                        num.append(a)
                        b = 2
        a += 1
        d = {'pro': pro, 'cat': cat,'num':num,'num1':num1, 'product':product,
'search_pro':search_pro}
    return render(request, 'all_product.html', d)
    def About(request):
    return render(request, 'about.html')
def Contact(request):
    return render(request, 'contact.html')
def Signup(request):
    if request.method == 'POST':
        u = request.POST['uname']
        f = request.POST['fname']
        l = request.POST['lname']

        p = request.POST['pwd']

        d = request.POST['date']

        c = request.POST['city']

        ad = request.POST['add']

        e = request.POST['email']

        i = request.FILES['img']

        con = request.POST['contact']

        t = request.POST['type']

        user = User.objects.create_user(username=u, email=e, password=p,
first_name=f,last_name=l)

        Profile.objects.create(user=user, dob=d, city=c, address=ad, contact=con,image=i,
user_type=t)

        messages.success(request, "Registeration Successfully")

        return redirect('login')

    return render(request, 'signup.html')

def Login(request):

    if request.method == "POST":

        u = request.POST['uname']

        p = request.POST['pwd']

        user = authenticate(username=u, password=p)



        if user:

            if not user.is_staff:

                profile = Profile.objects.get(user=user)

                if profile.user_type == "Seller" and profile.status == "Approved":

                    login(request, user)
                    messages.success(request, "Logged in Successfully")
                    return redirect('home')
                elif profile.user_type == "Seller" and profile.status == "Pending":
                    messages.error(request, "Farmer verification pending. Try again later.")
                elif profile.user_type == "Buyer":
                    login(request, user)
                    messages.success(request, "Logged in Successfully")
                    return redirect('home')
            else:
                messages.error(request, "Invalid credentials for user.")
        else:
            messages.error(request, "Invalid credentials.")
    return render(request, 'login.html')


def Admin_Login(request):
    error = ""
    if request.method == "POST":
        u = request.POST['uname']
        p = request.POST['pwd']
        user = authenticate(username=u, password=p)
        try:
            if user.is_staff:
                login(request, user)
                error = "yes"
            else:
                error = "not"
        except:
            error="not"
    d = {'error': error}
    return render(request,'loginadmin.html',d)
def Logout(request):
    logout(request)
    messages.success(request, 'Logout Successfully')
    return redirect('home')
def View_user(request):
    if not request.user.is_authenticated:
        return redirect('login_admin')
    pro = Profile.objects.filter(user_type="Buyer")
    d = {'user':pro}
    return render(request,'view_user.html',d)
def View_farmer(request):
    if not request.user.is_authenticated:
        return redirect('login_admin')
    pro = Profile.objects.filter(user_type="Seller")
    d = {'user':pro, 'data':'farmer'}
    return render(request,'view_user.html',d)
def Add_Product(request):
    if not request.user.is_authenticated:
        return redirect('login_admin'
    cat = Category.objects.all()
    error = False;
    if request.method == "POST":
        try:
            c = request.POST['cat']
            p = request.POST['pname']
            pr = request.POST['price']
            i = request.FILES.get('img')
            d = request.POST['desc']
            if not i:
                return HttpResponse("Please upload an image.")
            ct = Category.objects.get(name=c)
            Product.objects.create(
                category=ct,
                name=p,
                price=pr,
                image=i,
                desc=d,
                user=request.user
            )
            error = True
        except Exception as e:
            return HttpResponse(f"Error: {e}")
    return render(request, 'add_product.html', {'cat': cat, 'error': error}
def All_product(request):
    if not request.user.is_authenticated:
        return redirect('login')
    user = User.objects.get(id=request.user.id)
    profile = Profile.objects.get(user=user)
    cart = Cart.objects.filter(profile=profile)
    num1=0
    for i in cart:
        num1 += 1
    cat = Category.objects.all()
    pro = Product.objects.all()
    d ={'pro':pro,'cat':cat,'num1':num1}
    return render(request,'all_product.html',d)
def Admin_View_Booking(request):
    if not request.user.is_authenticated:
        return redirect('login_admin')
    book = Booking.objects.all()
    d = {'book': book}
    return render(request, 'admin_viewBokking.html', d)
def View_feedback(request):
    if not request.user.is_authenticated:
        return redirect('login_admin')
    feed = Send_Feedback.objects.all()
    d = {'feed': feed}
    return render(request, 'view_feedback.html', d)
def View_prodcut(request,pid):
    if not request.user.is_authenticated:
        return redirect('login_admin')
    cat = ""
    cat1 = ""
    pro1 = ""
```

```
num1 = 0
user=""
profile=""
cart=""
pro=""
num=""
if not request.user.is_staff:
    user = User.objects.get(id=request.user.id)
    profile = Profile.objects.get(user=user)
    cart = Cart.objects.filter(profile=profile)
    for i in cart:
        num1 += 1
if pid == 0:
    cat = "All Product"
    pro1 = Product.objects.all()
else:
    cat1 = Category.objects.get(id=pid)
    pro1 = Product.objects.filter(category=cat1).all()
cat = Category.objects.all()
pro = Product.objects.all()
num = []
b = 1
for j in cat:
    a = 1
    for i in pro:
        if j.name == i.category.name:
            if a == 1:
                num.append(i.id)
                a = 2
    d = {'pro': pro, 'cat': cat,'cat1': cat1,'num':num,'pro1':pro1,'num1':num1}
    return render(request, 'view_product.html',d)
def Add_Categary(request):
    if not request.user.is_authenticated:
        return redirect('login_admin')
    error=False
    if request.method=="POST":
        n = request.POST['cat']
        Category.objects.create(name=n)
        error=True
    d = {'error':error}
    return render(request, 'add_category.html', d)
def View_Categary(request):
    if not request.user.is_authenticated:
        return redirect('login_admin')
    pro = Category.objects.all()
    d = {'pro': pro}
    return render(request,'view_category.html', d)
def View_Booking(request):
    user = User.objects.get(id=request.user.id)
    profile = Profile.objects.get(user=user)
    cart = Cart.objects.filter(profile=profile)
    book = Booking.objects.filter(profile=profile)
    if profile.user_type == "Seller":
        book = Booking.objects.filter(farmer__icontains=request.user.username)
    pro = recommended_product(request)
    num1=0
    for i in cart:
        num1 += 1
    d = {'book': book,'num1':num1, 'pro':pro}
    return render(request, 'view_booking.html', d)
def Feedback(request, pid):
    if not request.user.is_authenticated:
        return redirect('login')
    error = False
    user1 = User.objects.get(id=request.user.id)
    profile = Profile.objects.get(user=user1)
    cart = Cart.objects.filter(profile=profile)
    num1 =0
    for i in cart:
        num1 += 1
    date1 = date.today()
    user = User.objects.get(id=pid)
    pro = Profile.objects.filter(user=user).first()
    if request.method == "POST":
        d = request.POST['date']
        u = request.POST['uname']
        e = request.POST['email']
        con = request.POST['contact']
        m = request.POST['desc']
        user = User.objects.filter(username=u, email=e).first()
        pro = Profile.objects.filter(user=user, contact=con).first()
        Send_Feedback.objects.create(profile=pro, date=d, message1=m)
        error = True
    d = {'pro': pro, 'date1': date1,'num1':num1,'error':error}
    return render(request, 'feedback.html', d)
def Change_Password(request):
    if not request.user.is_authenticated:
        return redirect('login')
    error = ""
    num1=0
    user = User.objects.get(id=request.user.id)
    profile = Profile.objects.get(user=user)
    cart = Cart.objects.filter(profile=profile)
    for i in cart:
        num1 += 1
    if request.method=="POST":
        n = request.POST['pwd1']
        c = request.POST['pwd2']
        o = request.POST['pwd3']
        if c == n:
            u = User.objects.get(username__exact=request.user.username)
            u.set_password(n)
            u.save()
            error = "yes"
        else:
            error = "not"
    d = {'error':error,'num1':num1}
    return render(request,'change_password.html',d)
def Add_Cart(request,pid):
    if not request.user.is_authenticated:
        return redirect('login')
    if request.method=="POST":
        user = User.objects.get(id=request.user.id)
        profile = Profile.objects.get(user=user)
        product = Product.objects.get(id=pid)
        Cart.objects.create(profile=profile, product=product)
        return redirect('cart')
def recommended_product(request):
    user = User.objects.get(id=request.user.id)
    profile = Profile.objects.get(user=user)
    book = Booking.objects.filter(profile=profile).order_by('-id')[:2]
    recommend = []
    for i in book:
        recommend+=i.booking_id.split('.')[1:]
    pro1 = Product.objects.filter(id__in=recommend)
    cat = []
    for i in pro1:
        if not i.category.id in cat:
            cat.append(i.category.id)
    pro = Product.objects.filter(category__id__in=cat).order_by('?')
    return pro

def view_cart(request):

    if not request.user.is_authenticated:

        return redirect('login')

    user = User.objects.get(id=request.user.id)

    profile = Profile.objects.get(user=user)

    cart =  Cart.objects.filter(profile=profile).all()

    pro = recommended_product(request)

    total=0

    num1=0

    book_id=request.user.username

    message1="Here ! No Any Product"

    for i in cart:

        total+=i.product.price

        num1+=1

        book_id = book_id+"."+str(i.product.id)

                                                                                 =
{'profile':profile,'cart':cart,'total':total,'num1':num1,'book':book_id,'message':message
1,'pro':pro}

    return render(request,'cart.html',d)

def remove_cart(request,pid):

    if not request.user.is_authenticated:

        return redirect('login')

    cart = Cart.objects.get(id=pid)

    cart.delete()

    return redirect('cart')

from datetime import date
from django.shortcuts import render, redirect

from django.http import HttpResponse

from django.contrib.auth.models import User

from .models import Profile, Cart, Booking, Status

def Booking_order(request, pid):

    if not request.user.is_authenticated:

        return redirect('login')

    stage = request.GET.get('stage')

    total_price = request.GET.get('total_price')

    data1 = User.objects.get(id=request.user.id)

    data = Profile.objects.filter(user=data1).first()

    cart = Cart.objects.filter(profile=data).all()

    total = 0

    num1 = 0

    farmer = []

    for i in cart:

        total += i.product.price

        farmer.append(i.product.user.username)

    user1 = data1.username

    li = pid.split('.')

    li2 = []

    for j in li:

        if user1 != j:

            li2.append(int(j))

            num1 += 1

    date1 = date.today()
```

```
if request.method == "POST":
    d = request.POST['date1']
    c = request.POST['name']
    c1 = request.POST['city']
    ad = request.POST['add']
    e = request.POST['email']
    con = request.POST['contact']
    b = request.POST['book_id']
    t = request.POST['total']
    try:
        user = User.objects.get(username=c)
    except User.DoesNotExist:
        return HttpResponse("User not found.")
    profile = Profile.objects.get(user=user)
    try:
        status = Status.objects.get(name="pending")
    except Status.DoesNotExist:
       return HttpResponse("Status 'pending' not found. Please add it in the database.")
    book1 = Booking.objects.create(
        profile=profile,
        book_date=d,
        booking_id=b,
        total=t,
        quantity=num1,
        status=status,
        farmer=farmer
```

```
            )
            cart2 = Cart.objects.filter(profile=profile).all()
            cart2.delete()
            return redirect('payment', book1.id)
        context = {
            'data': data,
            'data1': data1,
            'book_id': pid,
            'date1': date1,
            'total': total,
            'num1': num1
        }
        return render(request, 'booking.html', context)
    def payment(request,book):
        if not request.user.is_authenticated:
            return redirect('login')
        error = False
        book = Booking.objects.get(id=book)
        user = User.objects.get(id=request.user.id)
        profile= Profile.objects.get(user=user)
        cart = Cart.objects.filter(profile = profile).all()
        if request.method=="POST":
            error=True
        d ={'total':book.total,'error':error,'book':book}
        return render(request,'payment2.html',d)
```

```
def delete_admin_booking(request, pid,bid):
    if not request.user.is_authenticated:
        return redirect('login_admin')
    book = Booking.objects.get(booking_id=pid,id=bid)
    book.delete()
    return redirect('admin_viewBooking')
def delete_booking(request, pid,bid):
    if not request.user.is_authenticated:
        return redirect('login')
    book = Booking.objects.get(booking_id=pid,id=bid)
    book.delete()
    return redirect('view_booking')
def delete_user(request, pid):
    if not request.user.is_authenticated:
        return redirect('login_admin')
    user = User.objects.get(id=pid)
    user.delete()
    return redirect('view_user')
def delete_feedback(request, pid):
    if not request.user.is_authenticated:
        return redirect('login_admin')
    feed = Send_Feedback.objects.get(id=pid)
    feed.delete()
    return redirect('view_feedback')
def booking_detail(request,pid,bid):
    if not request.user.is_authenticated:
        return redirect('login')
    user = User.objects.get(id=request.user.id)
    profile = Profile.objects.get(user=user)
    cart =  Cart.objects.filter(profile=profile).all()
    product = Product.objects.all()
    book = Booking.objects.get(booking_id=pid, id=bid)
    total=0
    num1=0
    user1 = book.profile.user.username
    li = book.booking_id.split('.')
    li2=[]
    for j in li:
        if user1!= j :
            li2.append(int(j))
    for i in cart:
        total+=i.product.price
        num1+=1
                                        d                                          =
{'profile':profile,'cart':cart,'total':total,'num1':num1,'book':li2,'product':product,'total':b
ook}
    return render(request,'booking_detail.html',d)
def admin_booking_detail(request,pid,bid,uid):
    if not request.user.is_authenticated:
        return redirect('login_admin')
    user = User.objects.get(id=uid)
    profile = Profile.objects.get(user=user)
    cart =  Cart.objects.filter(profile=profile).all()
    product = Product.objects.all()

    book = Booking.objects.get(booking_id=pid, id=bid)

    total=0

    num1=0

    user1 = book.profile.user.username

    li = book.booking_id.split('.')

    li2=[]

    for j in li:

        if user1!= j :

            li2.append(int(j))

    for i in cart:

        total+=i.product.price

        num1+=1

                                                    d                                    =
{'profile':profile,'cart':cart,'total':total,'num1':num1,'book':li2,'product':product,'total':b
ook}

    return render(request,'admin_view_booking_detail.html',d)


def Edit_status(request,pid,bid):

    if not request.user.is_authenticated:

        return redirect('login_admin')

    book = Booking.objects.get(booking_id=pid,id=bid)

    stat = Status.objects.all()

    if request.method == "POST":

        n = request.POST['book']

        s = request.POST['status']

        book.booking_id = n
        sta = Status.objects.filter(name=s).first()
        book.status = sta
        book.save()
        return redirect('admin_viewBooking')
    d = {'book': book, 'stat': stat}
    return render(request, 'status.html', d)
def Admin_View_product(request):
    if not request.user.is_authenticated:
        return redirect('login_admin')
    pro = Product.objects.all()
    try:
        profile = Profile.objects.get(user=request.user)
        if profile.user_type == "Seller":
            pro = Product.objects.filter(user=request.user)
    except:
        pass
    d = {'pro':pro}
    return render(request,'admin_view_product.html',d)
def delete_product(request,pid):
    if not request.user.is_authenticated:
        return redirect('login_admin')
    pro = Product.objects.get(id=pid)
    pro.delete()
    return redirect('admin_view_product')
def profile(request, pid):
    if not request.user.is_authenticated:
        return redirect('login')
    user = User.objects.get(id=pid)
    pro = Profile.objects.get(user=user)
    cart = Cart.objects.filter(profile=pro)
    num1 = 0
    total = 0
    for i in cart:
        total += i.product.price
        num1 += 1
    d={'pro':pro,'user':user,'num1':num1,'total':total}
    return render(request,'profile.html',d)
def farmer_detail(request, pid):
    if not request.user.is_authenticated:
        return redirect('login')
    book = Booking.objects.get(id=pid)
    username = book.farmer[1:][:-1].replace(" ", "")
    if ',' in username:
        username = username.split(',')
    else:
        username = [username]
    print("All User before = ", username)
    all_user = [i[1:][:-1] for i in username]
    print("My Ysername", all_user)
    all_profile = Profile.objects.filter(user__username__in=all_user)
    d={ 'all_profile':all_profile}
    return render(request,'farmer_detail.html',d)
```

```
def Edit_profile(request):
    if not request.user.is_authenticated:
        return redirect('login')
    error = False
    user=User.objects.get(id=request.user.id)
    pro = Profile.objects.get(user=user)
    cart = ""
    try:
        cart = Cart.objects.get(profile=pro)
    except:
        pass
    num1=0
    total=0
    for i in cart:
        total+=i.product.price
        num1+=1
    if request.method == 'POST':
        f = request.POST['fname']
        l = request.POST['lname']
        u = request.POST['uname']
        c = request.POST['city']
        ad = request.POST['add']
        e = request.POST['email']
        con = request.POST['contact']
        d = request.POST['date']
    try:
        i = request.FILES['img']
        pro.image = i
        pro.save()

    except:
        pass
    if d:
        try:
            pro.dob = d
            pro.save()
        except:
            pass
    else:
        pass
    pro.user.username=u
    pro.user.first_name=f
    pro.user.last_name=l
    pro.user.email=e
    pro.contact=con
    pro.city=c
    pro.address=ad
    pro.save()
    error = True
d = {'error':error,'pro':pro,'num1':num1,'total':total}
```

```
    return render(request, 'edit_profile.html',d)

def Admin_Home(request):

    if not request.user.is_authenticated:

        return redirect('login_admin')

    book = Booking.objects.all()

    customer = Profile.objects.filter(user_type="Buyer")

    seller = Profile.objects.filter(user_type="Seller")

    pro = Product.objects.all()

    total_book = 0

    total_customer = 0

    total_pro = 0

    for i in book:

        total_book+=1

    for i in customer:

        total_customer+=1

    for i in pro:

        total_pro+=1

      d  =  {'total_pro':total_pro,'total_customer':total_customer,'total_book':total_book,
'seller':seller.count()}

    return render(request,'admin_home.html',d)

def delete_category(request,pid):

    if not request.user.is_authenticated:

        return redirect('login_admin')

    cat = Category.objects.get(id=pid)

    cat.delete()

    return redirect('view_categary')

def change_user_status(request, pid):
    if not request.user.is_authenticated:
        return redirect('login_admin')
    pro = Profile.objects.get(id=pid)
    if pro.status == "Pending":
        pro.status = "Approved"
    else:
        pro.status = "Pending"
    pro.save()
    messages.success(request, "Status Changed Successfully")
    return redirect('view_farmer')
def product_detail(request, pid):
    if not request.user.is_authenticated:
        return redirect('login')
    data = Product.objects.get(id=pid)
    data1 = Profile.objects.get(user=request.user)
    latest = Product.objects.filter(category=data.category).exclude(user=request.user).order_by('-id')[:4]
    return render(request, 'product_detail.html', {'data': data,'data1':data1, 'latest': latest})
```

Media

Manage.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys
def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'Efarming.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

ERROR HANDLING

Errors are what which proves the inaccuracy of the project. The project is complete only if it is capable of producing error free codes. Thus removing errors are an important factor while developing a project. Errors can be classified into the following sub categories:

Syntax Errors:

This kind of errors gets generated when the programmer is not following the grammatical rules of the language in which he/she is developing the code. These errors are identified in the code level itself. That is this error is identified by the compiler. Thus this form of error can be identified easily.

Logical Errors

This error is generated when the system is generated in correct output. This mans the user is expecting an output but the system fails to produce the output. This kind of error can be identified only while testing the software. Thus this form of error is identified by thorough execution of the system.

Runtime Errors

This kind of errors are most difficult to identified, moreover they are identified only when they are activated. This kind of error depends on both user input as well as the performance of the machine. The above two errors are identified or rectified in the coding level by implementing validations in the code.

This kind of errors gets generated only when the code is in the runtime. That is when the code is getting executed. These errors are returned by the system itself. Runtime Errors is handled by using the following code segment:

TESTING STRATEGY

Testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test—Software testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

Software testing can also be stated as the process of validating and verifying that a software program/application/product:

1. meets the business and technical requirements that guided its design and development;
2. works as expected; and
3. can be implemented with the same characteristics.

Software testing, depending on the testing method employed, can be implemented at any time in the development process. However, most of the test effort occurs after the requirements have been defined and the coding process has been completed. As such, the methodology of the test is governed by the software development methodology adopted.

Different software development models will focus the test effort at different points in the development process. Newer development models, such as Agile, often employ test driven development and place an increased portion of the testing in the hands of the developer, before it reaches a formal team of testers. In a more traditional model, most of the test execution occurs after the requirements have been defined and the coding process has been completed.

Testing can never completely identify all the defects within software. Instead, it furnishes a *criticism* or *comparison* that compares the state and behavior of the product against oracles—principles or mechanisms by which someone might recognize a problem. These oracles may include (but are not limited to) specifications, contracts, comparable products, past versions of the same product, inferences about intended or expected purpose, user or customer expectations, relevant standards, applicable laws, or other criteria.

Every software product has a target audience. For example, the audience for video game software is completely different from banking software. Therefore, when an organization develops or otherwise invests in a software product, it can assess whether the software product will be acceptable to its end users, its target audience, its purchasers, and other stakeholders. **Software testing** is the process of attempting to make this assessment.

Testing Methods

Software testing methods for the **e-Farming System** are categorized into **white-box** and **black-box testing**. These approaches help developers and testers ensure that the system performs as expected both internally (code logic) and externally (user experience).

Unit Testing

Unit testing for the **e-Farming System** focuses on validating the functionality of individual components such as **user registration**, **crop listing**, **AI crop recommendation**, **order placement**, and **weather-based suggestions**. These tests are written and executed by developers during the development stage.

Examples of unit tests include:

- Validating input fields for user registration, login, and crop submission forms.
- Verifying the correctness of cart operations and order calculations.
- Ensuring admin functionalities like adding or deleting crop records work correctly.

Each function or module (e.g., `User`, `Crop`, `Order`, `Dashboard`) is tested in **isolation** to confirm it performs its intended task. Unit testing is done using a **white-box testing** approach, where the tester has knowledge of the internal code.

Unit testing ensures that the **building blocks** of the system are functional and reliable before they are integrated.

Integration Testing

Integration testing ensures that the different modules of the **e-Farming System** (like user management, crop listings, cart system, payment gateway, AI crop recommendations, and weather tips) work together properly.

Examples:

- After a user logs in, they should be able to view and order crops seamlessly.
- Once an order is placed, it should correctly reflect in the buyer's dashboard and notify the corresponding farmer.
- Payment integration (e.g., Instamojo) should successfully record transactions and update order status.

Integration testing validates the **data flow** between modules and identifies defects in the interaction between front-end and back-end components, especially database interactions.

System Testing

System testing is conducted on the **fully integrated e-Farming System** to verify that it meets the **complete functional requirements** and behaves as expected in real-world scenarios.

Functionalities tested include:

- Complete user flow from signup to ordering crops.
- Search and filter functionality for different crops.
- Viewing live weather updates and crop price trends.
- Admin panel for managing users, crops, and dashboard analytics.
- Farmer dashboard functionalities like AI crop suggestion and soil health checks.

SYSTEM SECURITY

The **e-Farming System** integrates essential security measures to protect user data, ensure safe transactions, and maintain the integrity of the platform. As a web-based system handling sensitive information like user credentials, payment details, and crop-related records, maintaining system security is a critical objective.

Users cannot access system features like crop ordering, dashboard analytics, or profile management without proper registration and login. Every login requires valid credentials (username and password), which enhances security by preventing unauthorized access. Furthermore, all sensitive data is stored in an encrypted format, ensuring protection even in case of data breaches.

Security Issues :

- **Authentication:**
    - The system verifies the identity of users through secure login mechanisms before granting access to resources such as user dashboards, orders, or admin panels.
- **Access Control:**
    - Access control policies are implemented to ensure that only authorized users can perform specific tasks. For example, only farmers can upload crops, while buyers can only place orders.
- **Confidentiality:**
    - Sensitive information such as user profiles, payment details, and booking history is secured using encryption/decryption techniques to maintain confidentiality.
- **Data Integrity:**
    - The system ensures data consistency and reliability. Any changes in listings, orders, or payments are recorded securely to prevent unauthorized modification or loss.
- **Non-repudiation:**
    - Actions performed by users (like placing orders or updating crop listings) are logged with timestamps and user IDs. Only authenticated users can execute specific actions, and system logs ensure accountability.
    - A role-based security policy is implemented to ensure that different user roles (admin, farmer, buyer) have clear and restricted access to their respective features.

FUTURE SCOPE

The project named **“e-Farming System”** considers a web platform that has great potential in revolutionizing the agricultural market. Many farmers and buyers still rely on traditional farming and marketing methods. With the digital shift in every domain, this website can be customized to suit the needs of farmers, buyers, and agri-based organizations, making it a powerful tool for the future of smart agriculture.

The e-Farming System aims to digitize farming operations, provide market access, and promote better resource management. As technology becomes more affordable and accessible in rural areas, such platforms can empower farmers with real-time data, crop recommendations, weather forecasts, and e-commerce capabilities.

With continuous improvements, the system can expand to support:

- Smart Irrigation Systems
- IoT-based Crop Monitoring
- Government Subsidy Tracking
- AI-driven Market Predictions
- Multilingual Voice-based Assistance for Farmers

We believe this system will significantly reduce market dependency, increase transparency, and improve profits for farmers while making the agri-supply chain more efficient.

Further Enhancements

- **Mobile App Integration:** A mobile application can be developed to allow farmers and buyers to access services on-the-go, with offline features for remote areas.
- **Live Chat and Expert Support:** Farmers can connect with agricultural experts, get tips, and report issues for faster resolutions.
- **Blockchain for Supply Chain Transparency:** Enhance trust by implementing blockchain to trace product origins and transaction history.
- **Expanded Payment Options:** More robust online payment systems including UPI, credit/debit, wallet payments, and even cryptocurrency for future-readiness.
- **Community Forum:** A dedicated discussion space for farmers to share experiences, success stories, and farming tips.
- **Export and Logistics Module:** Help farmers connect with national/international buyers and manage transportation.

BIBLIOGRAPHY

Henceforth, we have reached the end of our e-Farming System project where we explored various sections including system design, implementation, user interaction, security, and future prospects. For compiling this project, several references and online resources were consulted. Valuable guidance and help from books and websites made it possible to bring this project to life.

Books:

1. **Fundamentals of Software Engineering** by Rajib Mall
2. **An Introduction to Database Systems**, 4th Edition, by C.J. Date, Publisher: Addison-Wesley
3. **Database Management Systems** by Bipin C. Desai
4. **MySQL: The Complete Reference** by Kevin Loney and George Koch
5. **Beginning Django** by Daniel Rubio
6. **Python for Data Analysis** by Wes McKinney
7. **Flask Web Development** by Miguel Grinberg

Websites:

- www.geeksforgeeks.org
- www.w3schools.com
- www.djangoproject.com
- www.mysql.com
- www.realpython.com
- www.weatherapi.com
- www.instamojo.com
- www.github.com