Software Engineering Project

Farmer’s Era - Farmera

Sri Guru Gobind Singh College of Commerce
University of Delhi

Project: Online Farmer’s Market
Guidance: Ms. Ushveen Kaur
Subject: Software Engineering, SEM - IV

Team Members:
- Aditi Juneja(21078570002)
- Sonali (21078570052)

Index

Project : Online farmer’s market

Problem statement :

In today's era of urbanization , we often neglect the role of farmers and their contribution in our living world. We are familiar with the injustice that farmers face. They are often not paid enough for their produce due to middlemen’s commissions. There are rooftop farming start-ups too but they are often unable to sell their produce because of improper marketing.

On the other hand, the consumers are shifting towards a healthy lifestyle. We prefer fresh fruits and vegetables which generally we don’t get living in the cities.

So this delivery app tries to help the farmers in selling their produce to the consumers living in the remote location and consumers to buy and get fresh products living in the cities. It will be a three user app for the producers(farmers) , consumers and the porters.

CERTIFICATE

This is to certify that Aditi Juneja (21078570002) and Sonali (21078570052) students of B.Sc. Honors Computer Science, Semester IV have submitted the project entitled “Online Farmer’s Market” for the partial fulfillment of the requirements of Software Engineering project. It embodies the work done by them during semester IV of their course under the due supervision of Ms. Ushveen Kaur.

Acknowledgement

We would like to express our sincere thanks to our Software Engineering teacher “Ms.Ushveen Kaur” for giving us the opportunity to work under her in this project.

We truly appreciate and value her esteemed guidance and encouragement from the beginning to end of this project . We are extremely grateful to her . We want to thank all our teachers for providing a solid background for our studies. They have been a great source of inspiration to us and we thank them from the bottom of our heart .

Last but not the least , we would like to thank our Computer Science department for providing us with all the facilities that were required.

Initial requirements:

- **PRODUCER PORTAL**
    - Registration for verified sellers
    - Listing options to list their products and prices
    - Option for landless farmers to get employment
    - Help desk to get the various farming techniques and suggestions for the farmers
    - Record of past sales

- **CONSUMER PORTAL**
    - Interface to find out the nearby farmers
    - Option for purchasing item in cart
    - Option for giving feedback
    - History of previous sales
    - Option for online payment

- **DELIVERY EXECUTIVE PORTAL**
    - Option for delivery partnership services
    - Option for optimally assigned deliveries
    - Option for passing over a delivery

Requirement Analysis:

- Producers

1. Mr. Rohit(owns 2 acre farm in Najafgarh; mostly grows sugarcanes; dairy products) :

After communicating from Mr. Rohit over multiple calls we have confirmed the following requirements from him :

1. Option for selling secondary goods too like gur(jaggery), ghee etc.
2. Option for hiding goods : since the crops are grown seasonally, he didn't want to show some goods “out of stock” for too long nor did he want to add and delete the goods every season.

2. Mr. Rahul(owns a farm in Najafgarh; mostly grows radish, carrot and salad vegetables) :

After communicating from Mr. Rahul over multiple calls we have confirmed the following requirements from him :

1. Wanted option for self delivery : He wanted to have this option because he had people for delivery and just wanted to increase his reach.
2. Simple UI/UX for updating the quantity of goods, so that his parents could update them, as he is mostly not around, because of his job in the city.

- Consumers

1. Ms. Kritika

After communicating from Ms. Kritika over multiple calls we have confirmed the following requirements from her :

1. Clearly mentioned return/refund policies and helpline number available in case of any queries/complaints
2. Clearly mentioned quality standards and food quality certificates

2. Mrs. Kavita

After communicating from Mrs. Kavita over multiple calls we have confirmed the following requirements from her :

1. Subscription option : she wanted to order some salad items on a daily basis.
2. Information page of the farm.

Final Requirements :

- **Producer**
    - Functionalities :
        1. Login
        2. Listing products : add, delete, hide, pricing, tags, quantity, return policy etc.
        3. Farm detail page : having location, contact number, latest pictures of the farm, reviews of the farm etc.
        4. Past sales record : consumer details, sales graphs and variation over a given time period
    - View :
        1. Consumer details : quantity purchased, approximate location
        2. Delivery executive's details : name, contact number
        3. Product page : reviews, price, pictures of product etc.

- **Consumer**
    - Functionalities :
        1. Login
        2. Searching products : based on location, tags, reviews and price
        3. Payment and checkout options : adding items in cart, online payment, saving address etc.
        4. Writing reviews after purchase

    - View :
        1. Farmer's detail page
        2. Past purchases
        3. Delivery executive's details : name, contact number

- **Delivery executive**
    - Functionalities :
        1. Login
        2. Pick-up and drop location : based on delivery executive’s preferred locations
        3. Package details : weight, delivery time
        4. Map API for guiding from farm to the destination
        5. Online vault for delivery payment

    - View :
        1. Farm detail page : contact
        2. Record of past deliveries

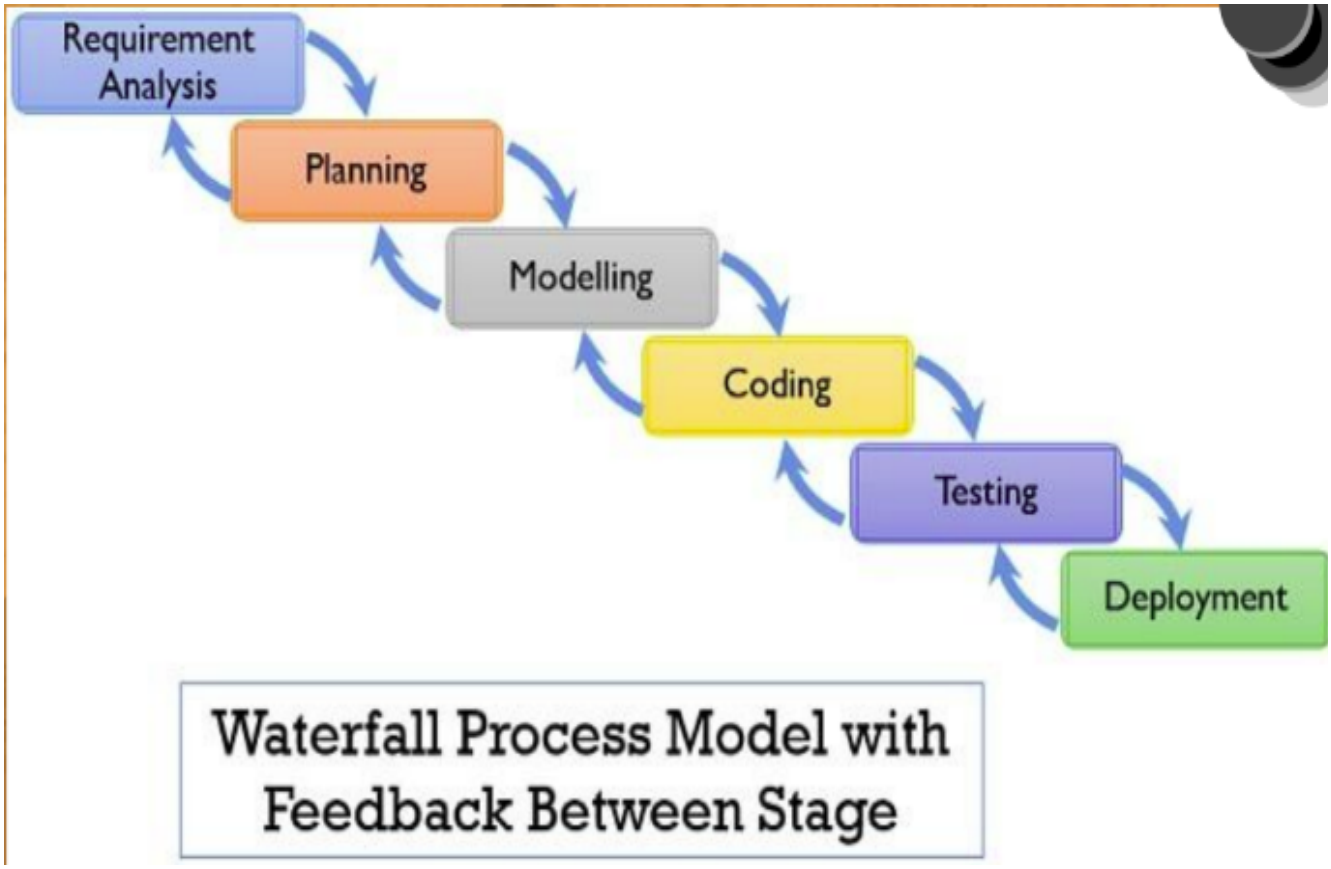
Process Model:

We have chosen Waterfall Model as our process model because of the following reasons:

- All the requirements are known, clear, and fixed .
- The project is short and simple.
- The development environment is stable .
- The necessary tools and techniques used are stable, and not dynamic.
- The project is time bound .

What Is Waterfall Model?

The waterfall model is a straightforward project management system that follows a linear stream. It entails breaking down one massive assignment that your clients have handed you in your briefing stage. By sectioning one large project into several scheduled phases with associated tasks, our project will be much more achievable and easier to complete on time. The waterfall model is organized and efficient, so all members of our team will thrive while following it.



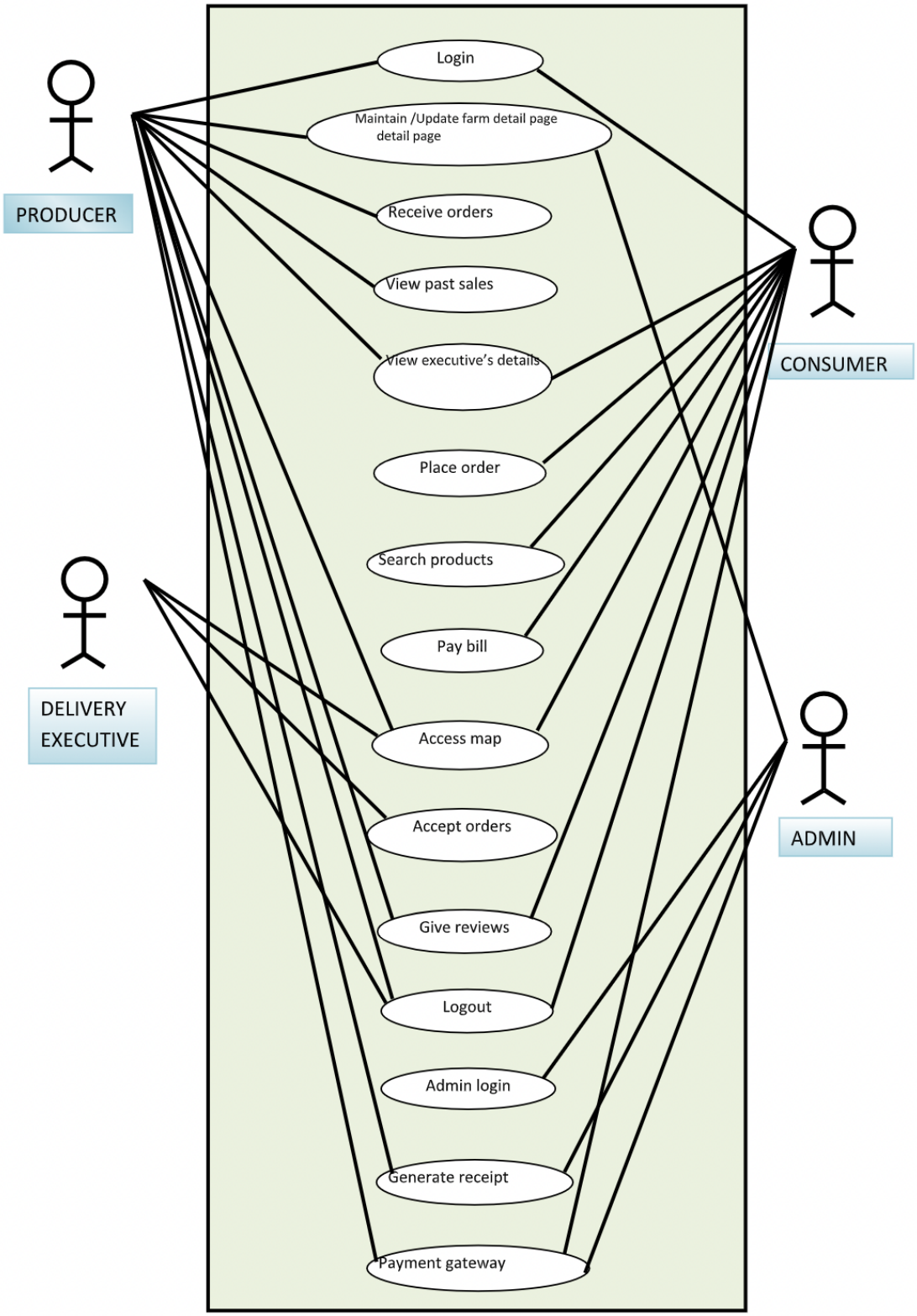
Use case diagram

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

Purpose of Use Case diagram

1. It gathers the system's needs.
2. It depicts the external view of the system.
3. It recognizes the internal as well as external factors that influence the system.
4. It represents the interaction between the actors.

Use case diagram for this project



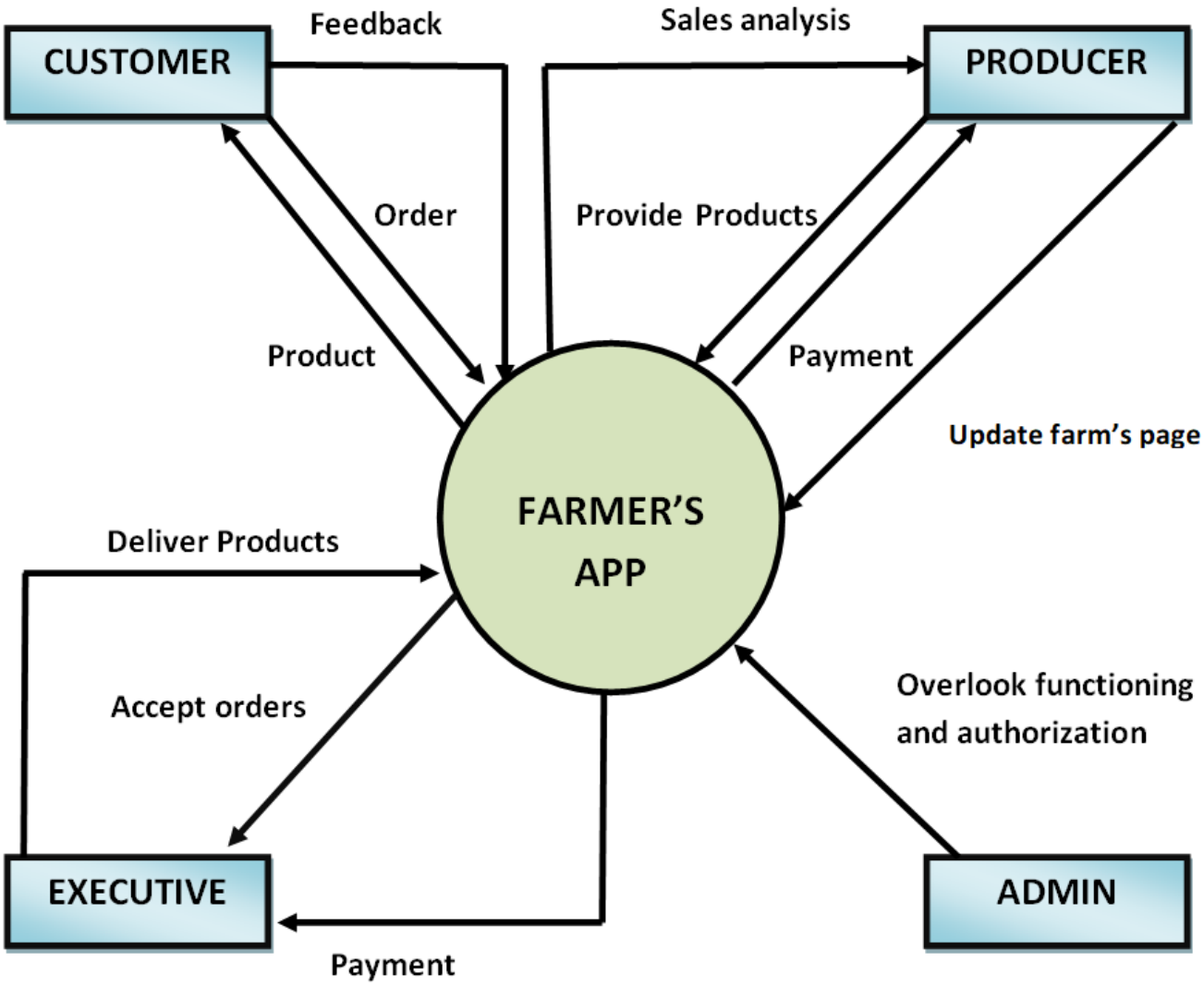
Advantages(USP):

1. Focuses on small scale producers
2. Live updates and reviews of farms
3. Wide variety of goods(seasonality taken into consideration)
4. Verification of farmers and their products : Quality certificates, identity proof etc.
5. Sales analysis and sales data management for farmers
6. Fresh products delivered because of fast delivery from nearby farms

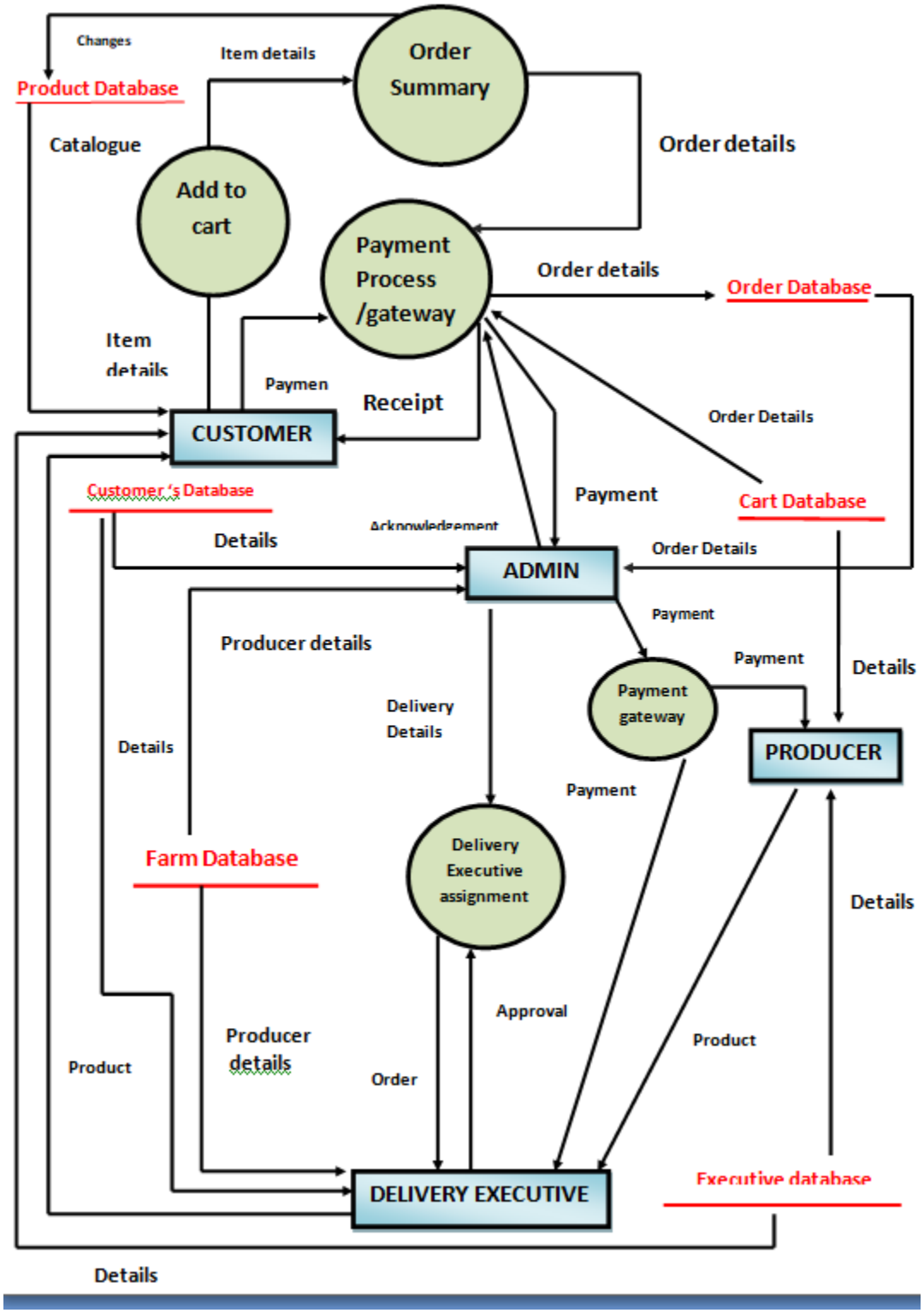
Disadvantages :

1. Delivery limitations : Due to economic constraints we cannot guarantee long and large scale delivery facilities
2. Storage limitations : Due to lack of storage facilities we cannot ensure the regularity in the stocks
3. Immediate commits cannot be made in the software, since we are using the waterfall model.

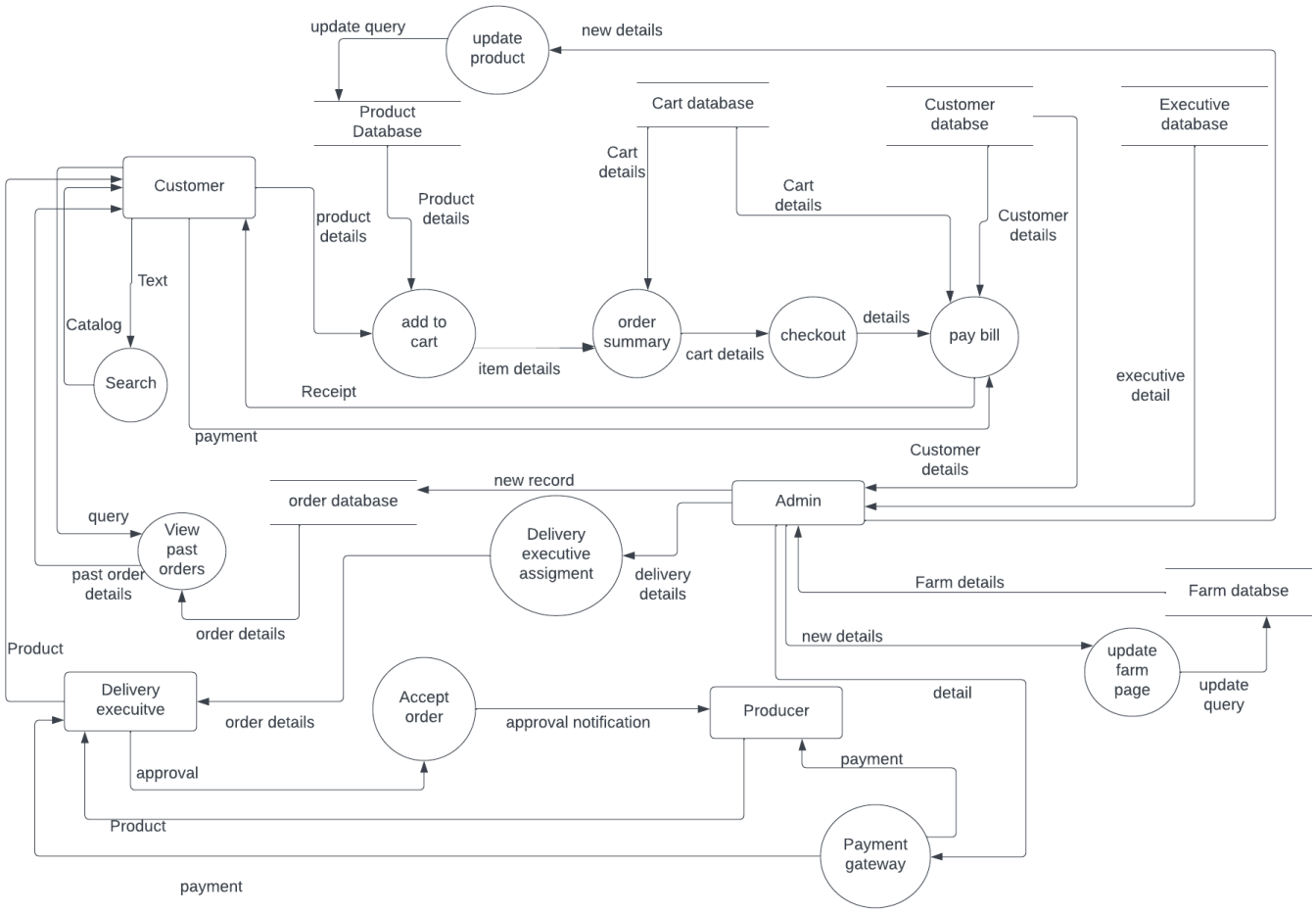
Level - 0 DFD (context diagram)



Level - 1 DFD

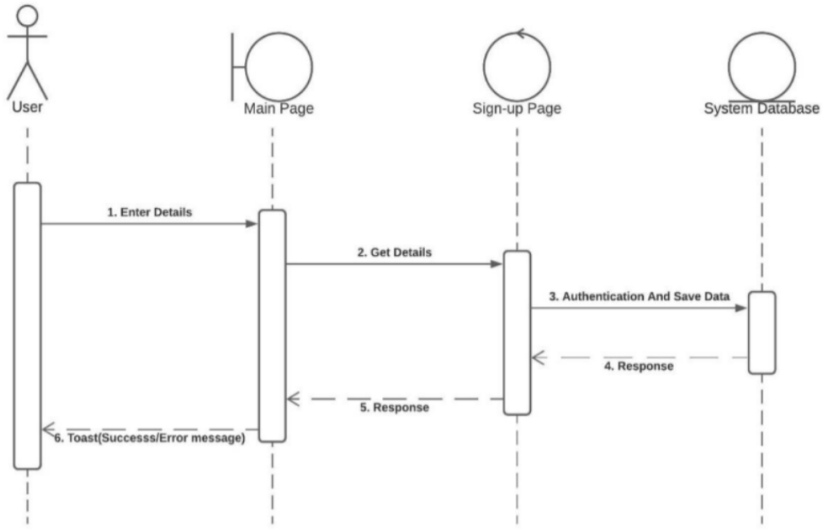

Changes
Item details
Order Summary
Product Database
Catalogue
Order details
Add to cart
Payment Process /gateway
Order details
Order Database
Item details
Paymen
Receipt
CUSTOMER
Order Details
Customer 's Database
Payment
Cart Database
Details
Acknowledgement
ADMIN
Order Details
Payment
Producer details
Payment
Details
Payment gateway
Delivery Details
Details
PRODUCER
Payment
Farm Database
Delivery Executive assignment
Details
Approval
Producer details
Product
Order
Product
Executive database
DELIVERY EXECUTIVE
Details

Level - 2 DFD

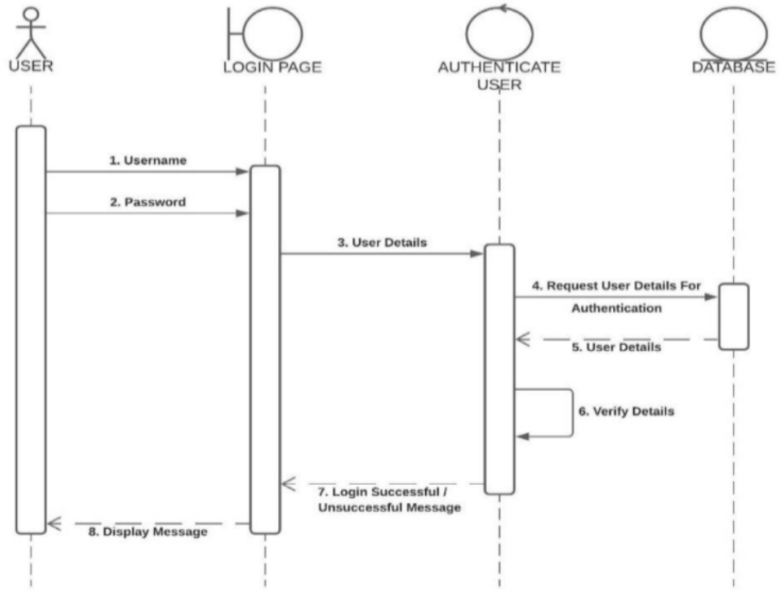

update query
update product
new details
Product Database
Cart database
Customer databse
Executive database
Customer
product details
Product details
Cart details
Cart details
Customer details
Text
Catalog
Search
add to cart
order summary
checkout
details
pay bill
item details
cart details
Receipt
payment
executive detail
Customer details
new record
order database
Admin
query
View past orders
past order details
order details
Delivery executive assigment
delivery details
Farm details
Farm databse
new details
update farm page
update query
Product
Delivery executive
order details
Accept order
approval notification
Producer
detail
payment
approval
Product
Payment gateway
payment

Sequence Diagrams

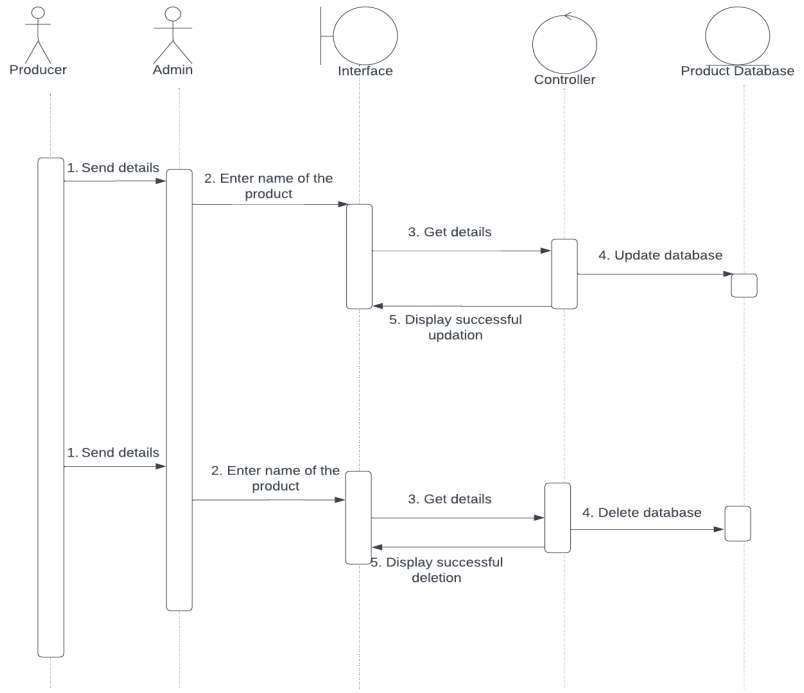
Sequence diagram for Sign up



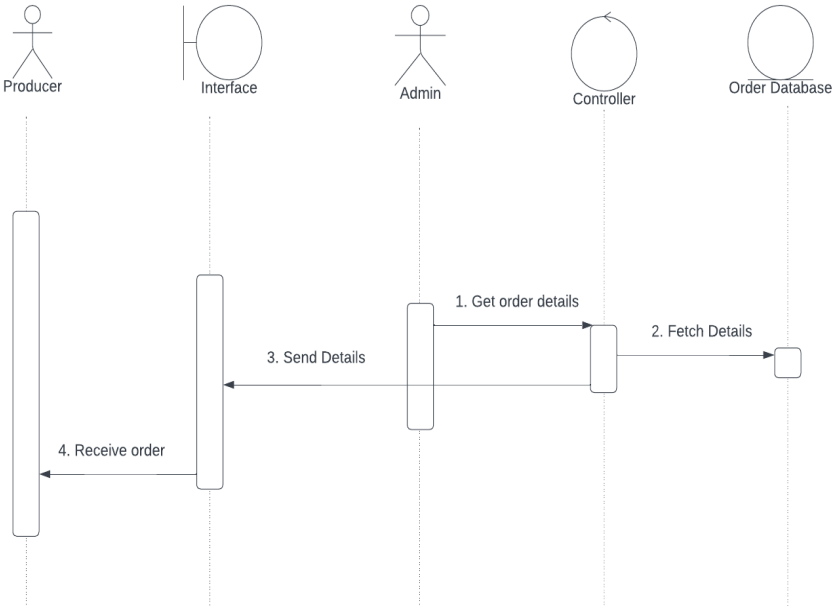
Sequence diagram for Login



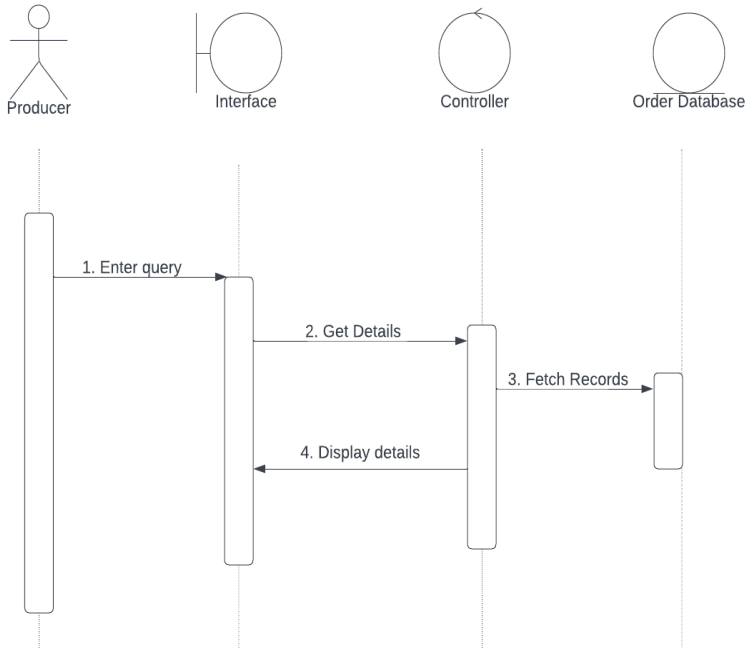
Sequence diagram for update farm detail page



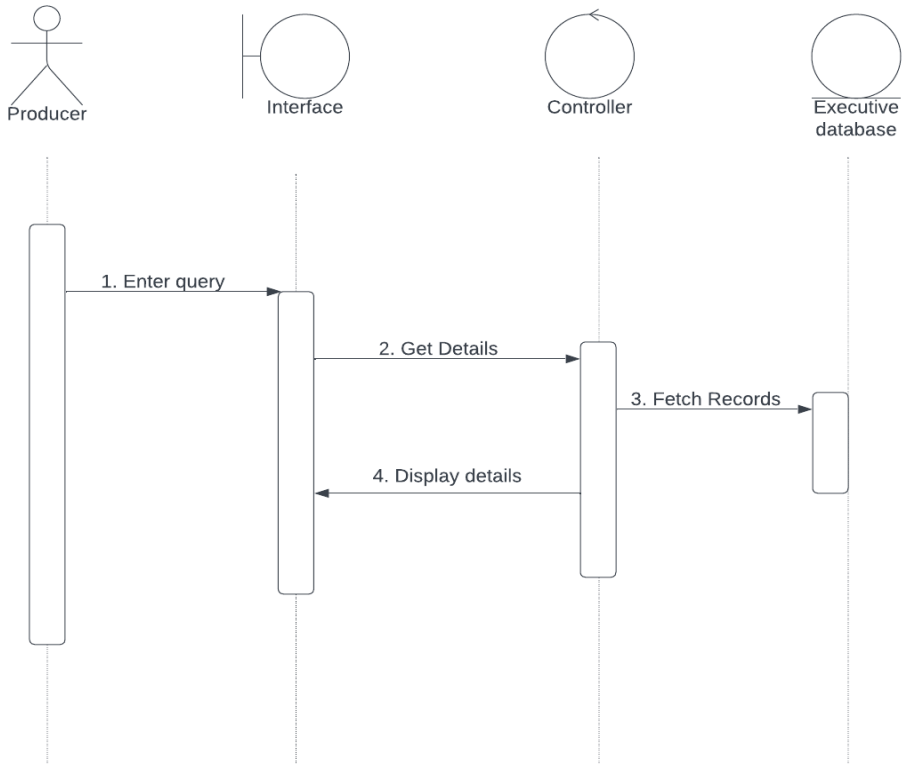
Sequence diagram for receive orders



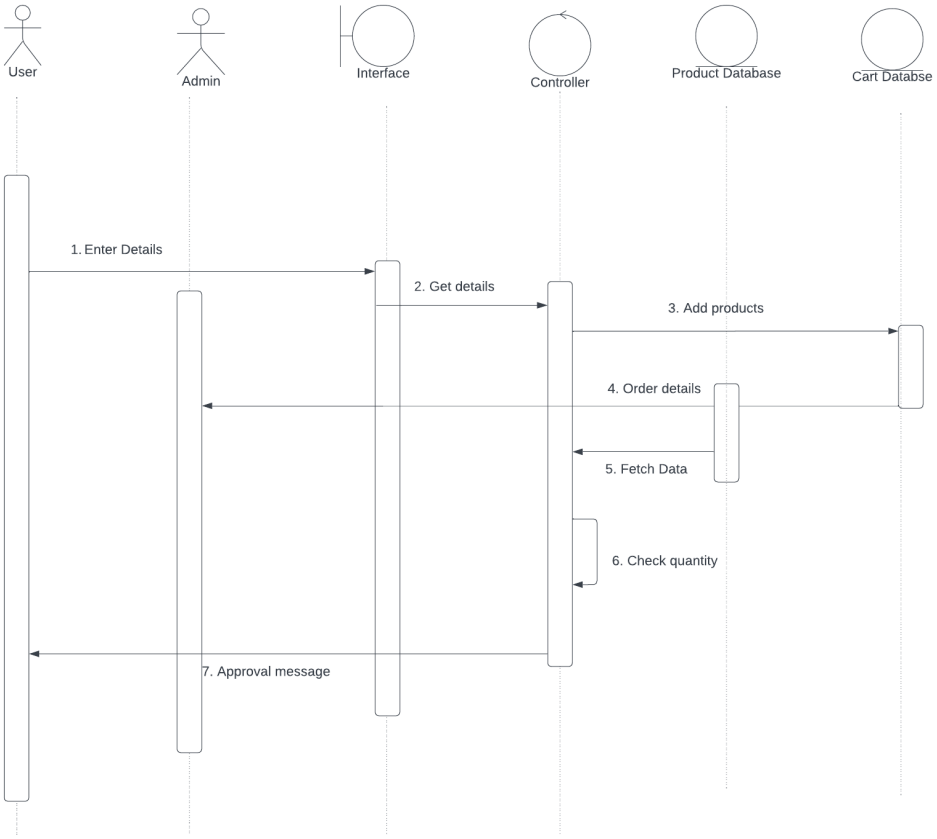
Sequence diagram for view past sales



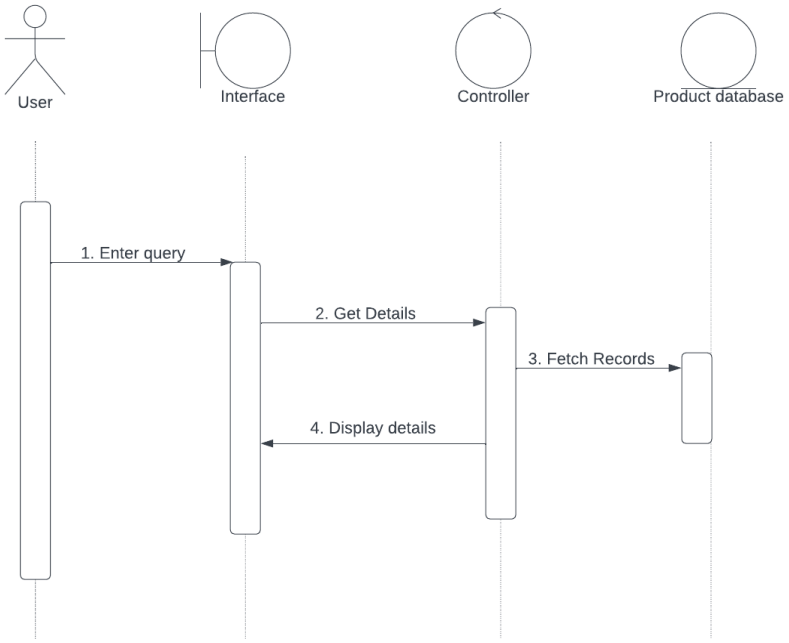
Sequence diagram for view executive detail



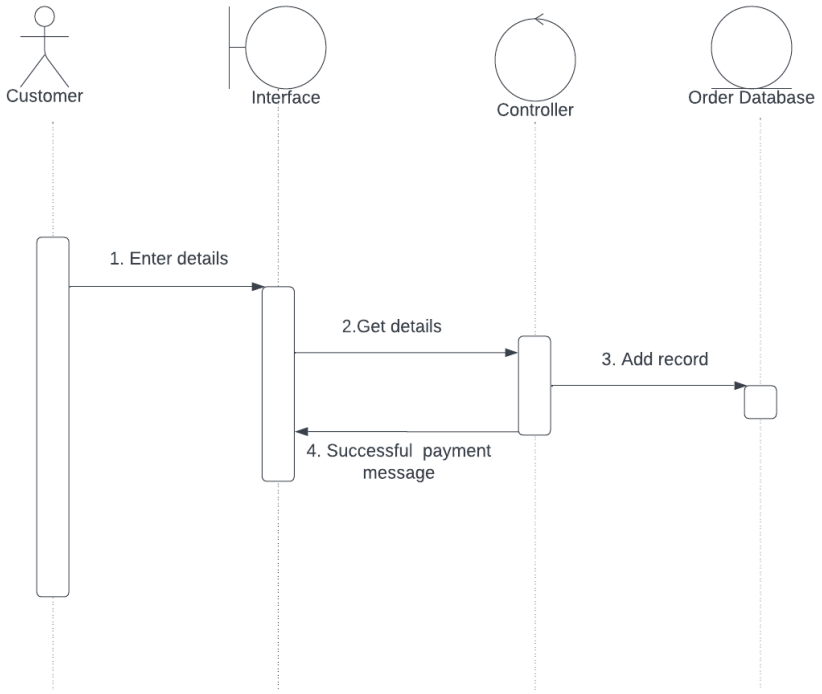
Sequence diagram for place order



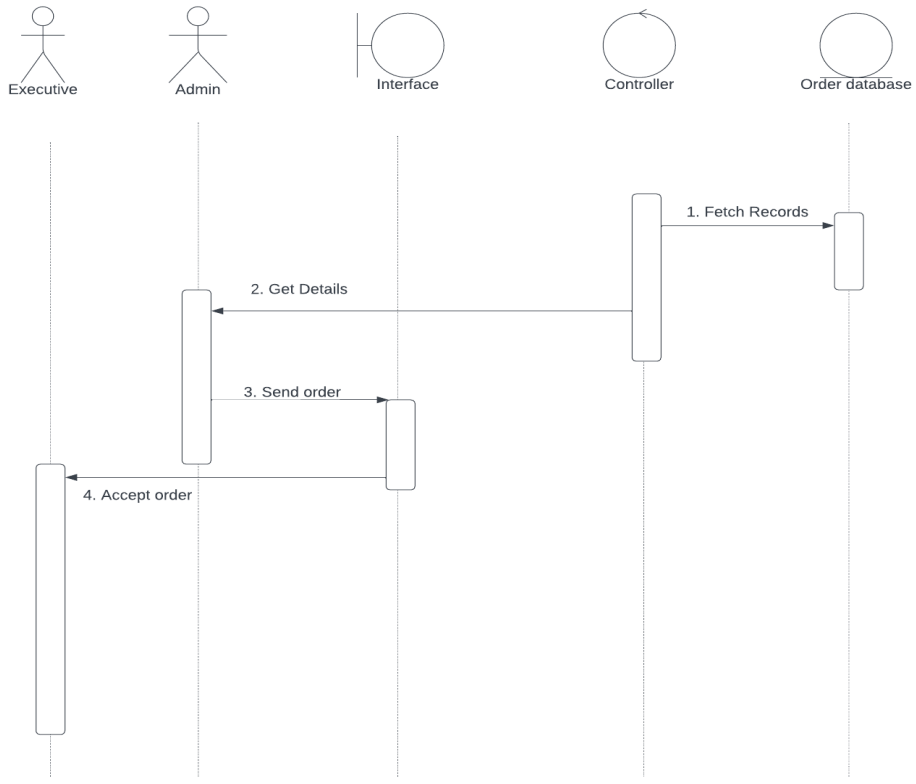
Sequence diagram for search product



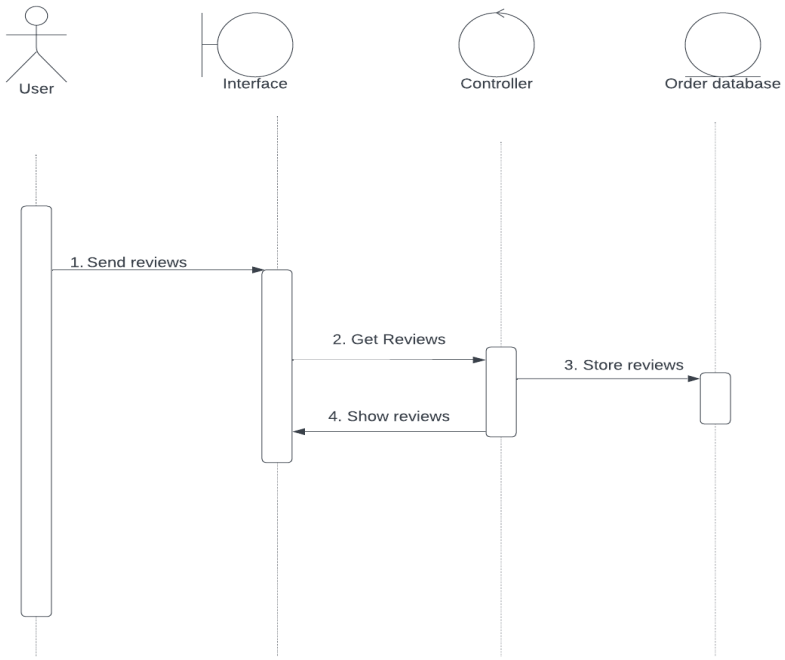
Sequence diagram for pay bill



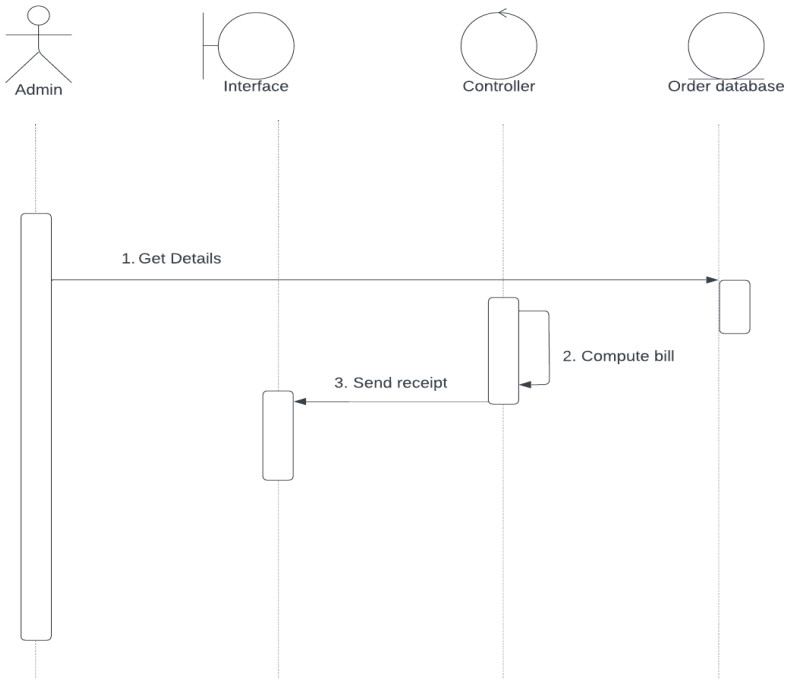
Sequence diagram for accept order



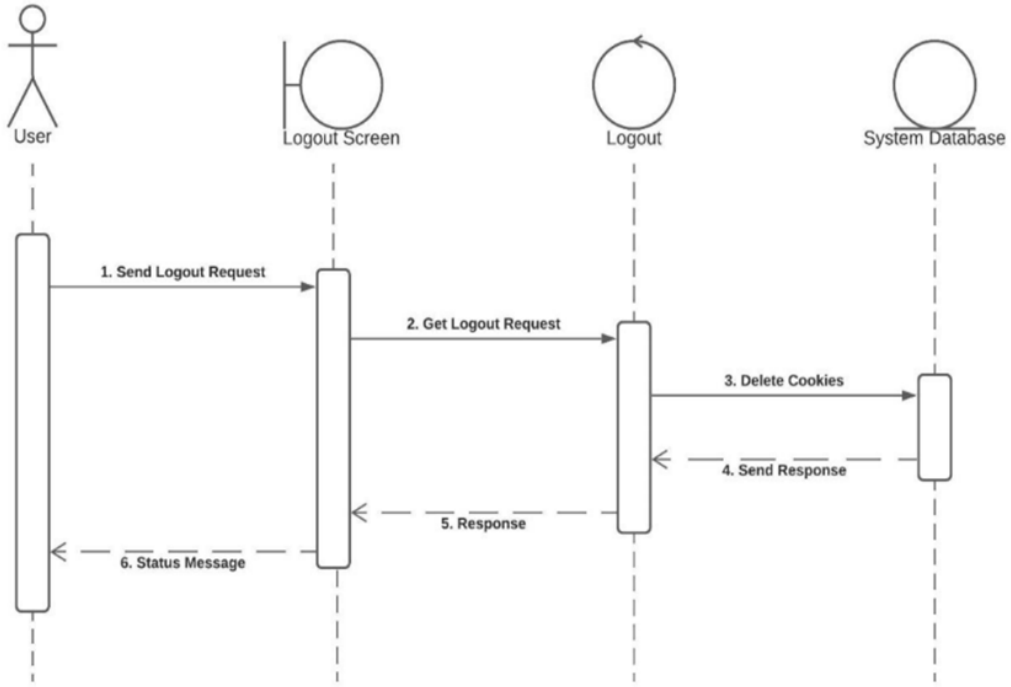
Sequence diagram for give reviews



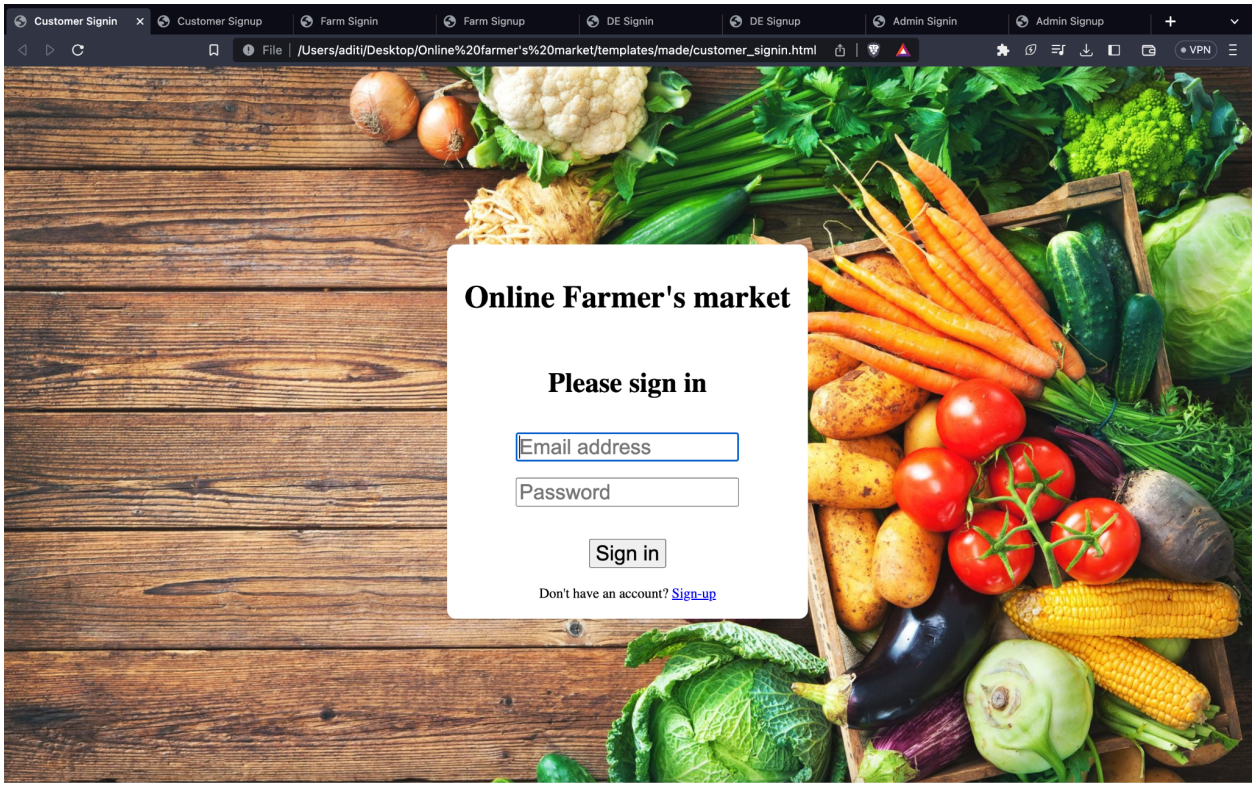
Sequence diagram for generate receipt

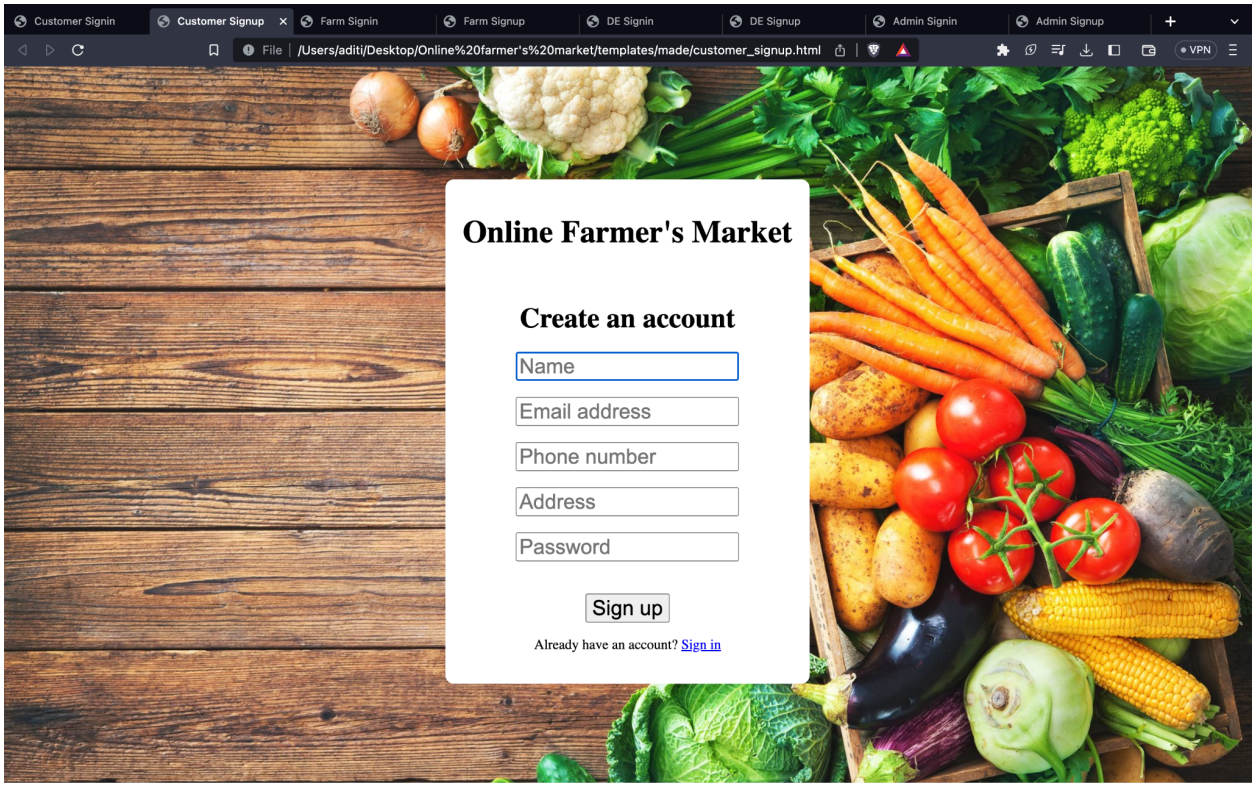


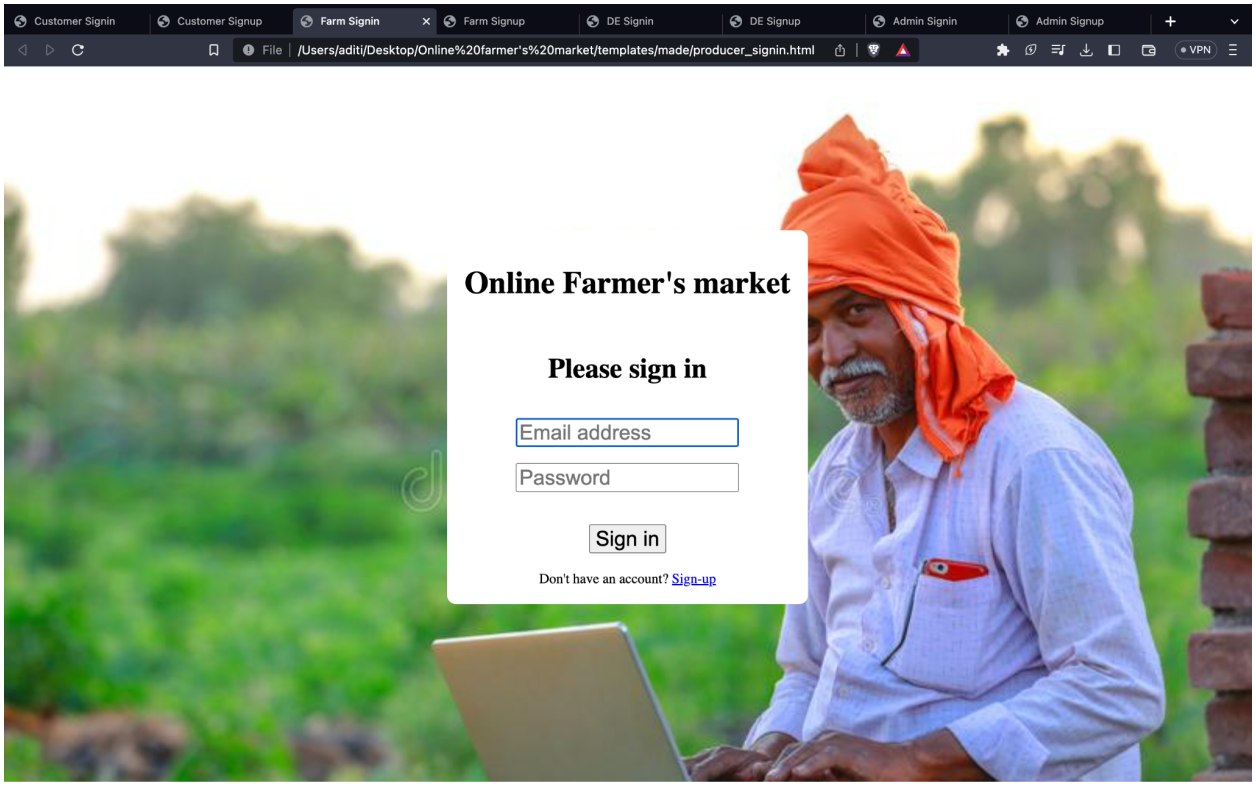
Sequence diagram for logout

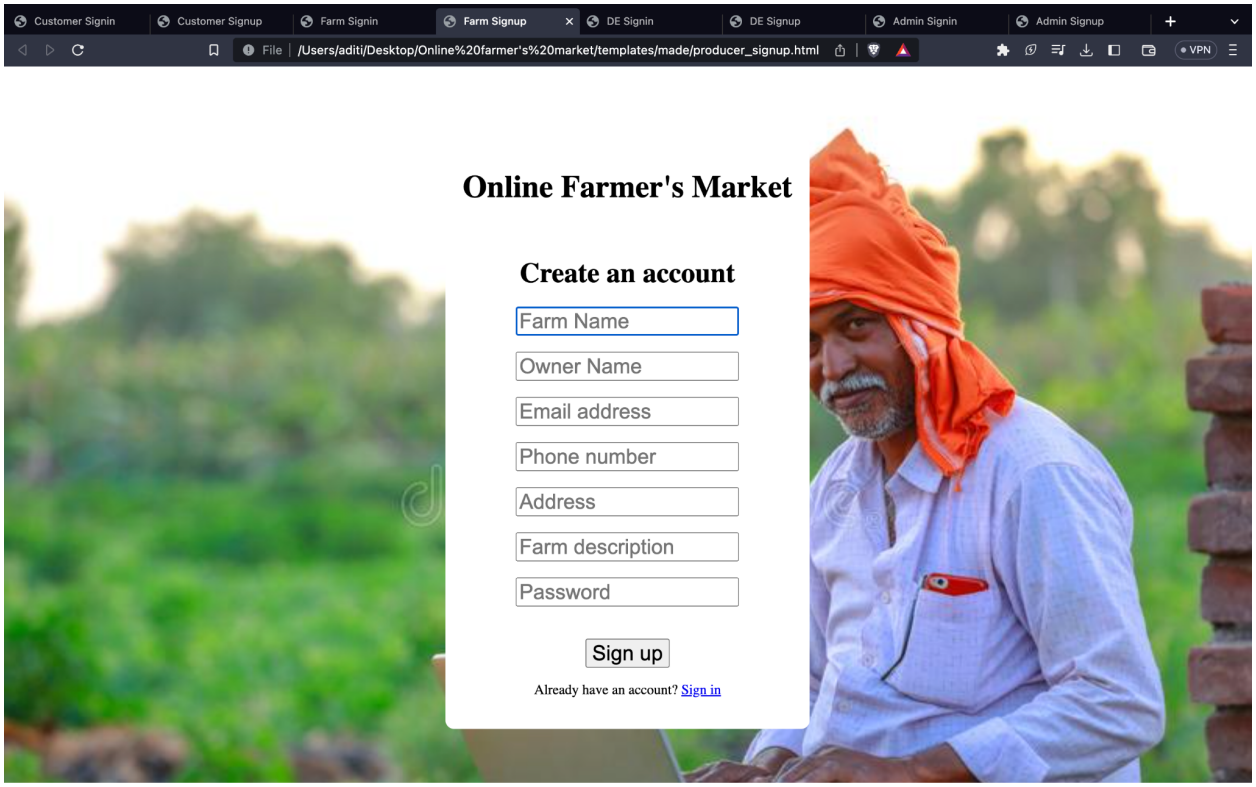


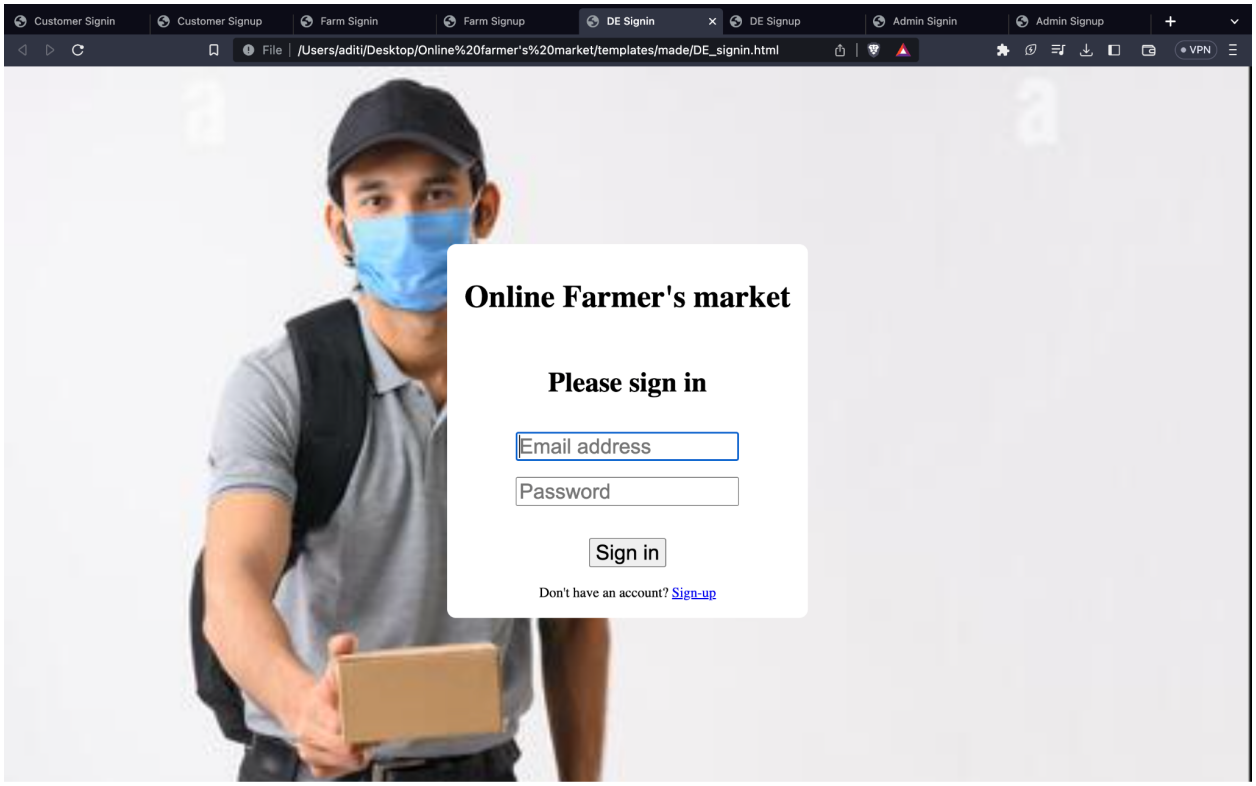
Screenshots(Sign in/ Sign up)

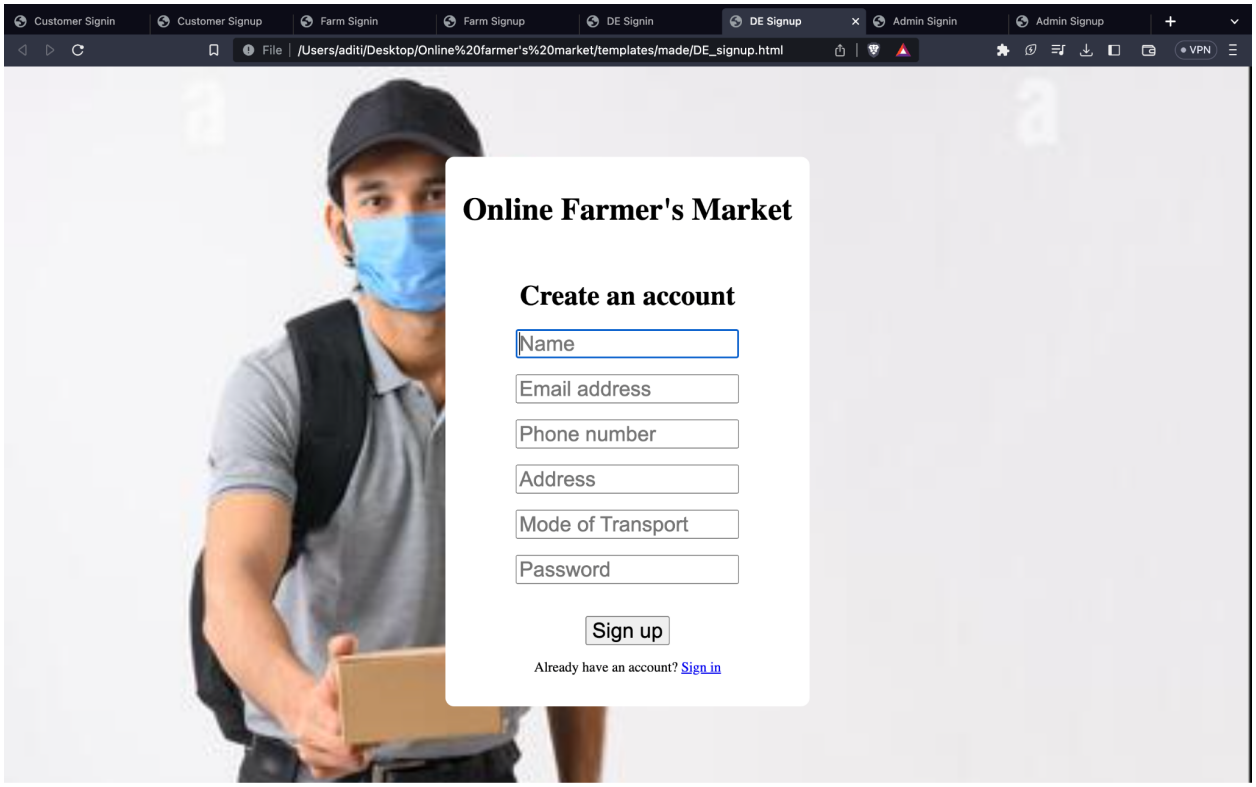


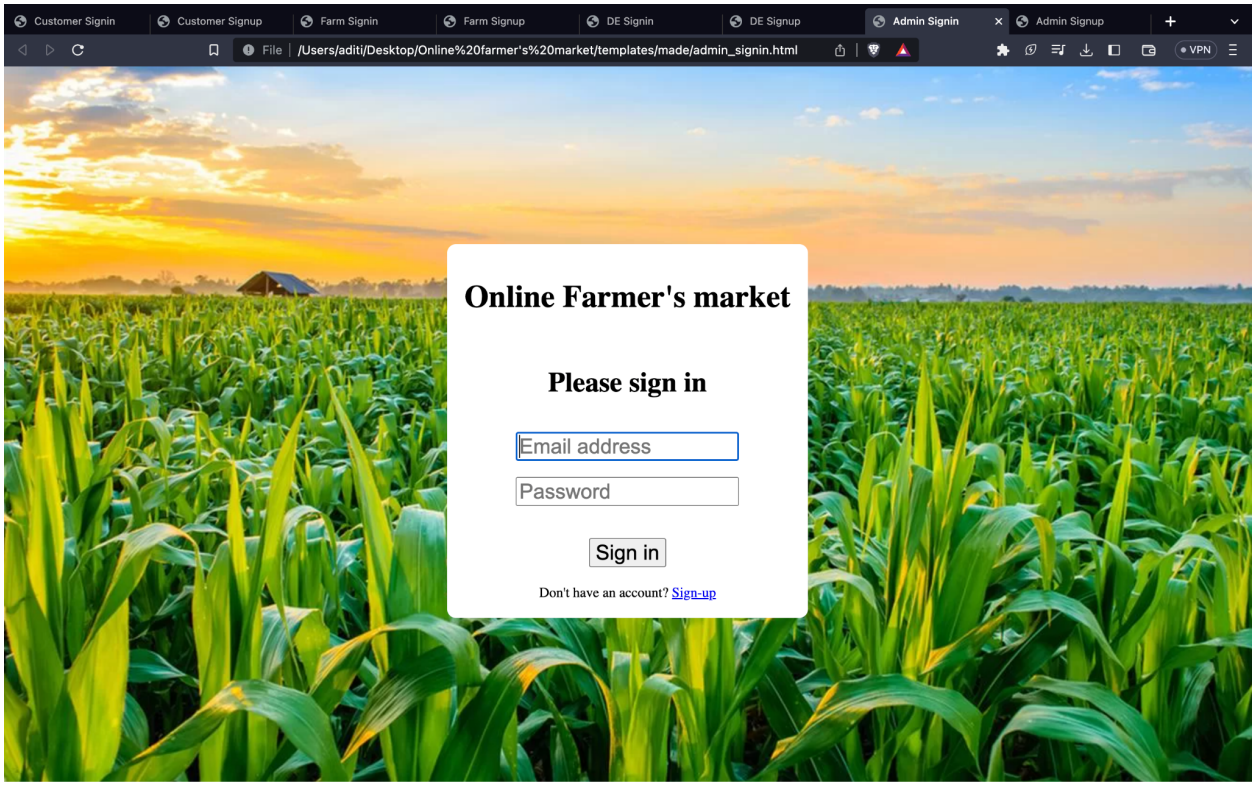


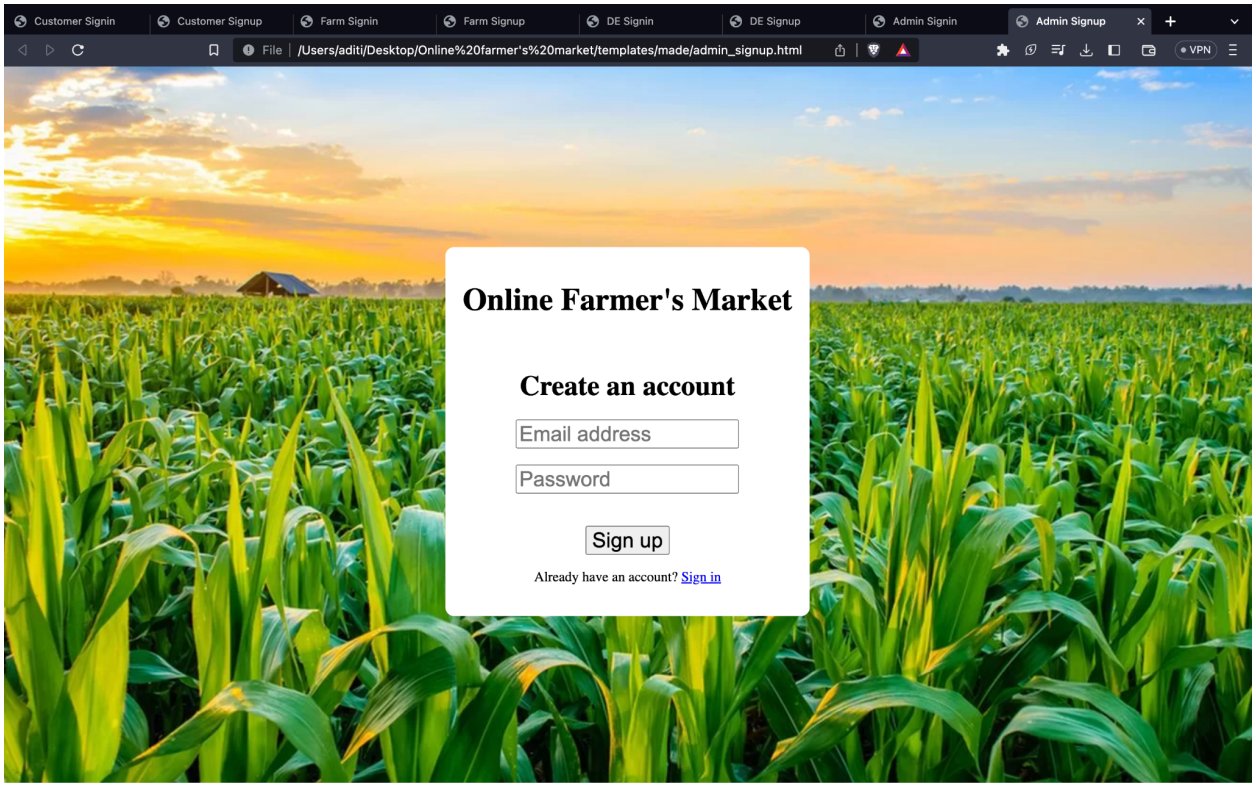

Customer Signin
Customer Signup
Farm Signin
Farm Signup
DE Signin
DE Signup
Admin Signin
Admin Signup
File /Users/aditi/Desktop/Online%20farmer's%20market/templates/made/producer_signin.html
VPN
Online Farmer's market
Please sign in
Email address
Password
Sign in
Don't have an account? Sign-up


Customer Signin
Customer Signup
Farm Signin
Farm Signup
DE Signin
DE Signup
Admin Signin
Admin Signup
File /Users/aditi/Desktop/Online%20farmer's%20market/templates/made/producer_signup.html
VPN
Online Farmer's Market
Create an account
Farm Name
Owner Name
Email address
Phone number
Address
Farm description
Password
Sign up
Already have an account? Sign in

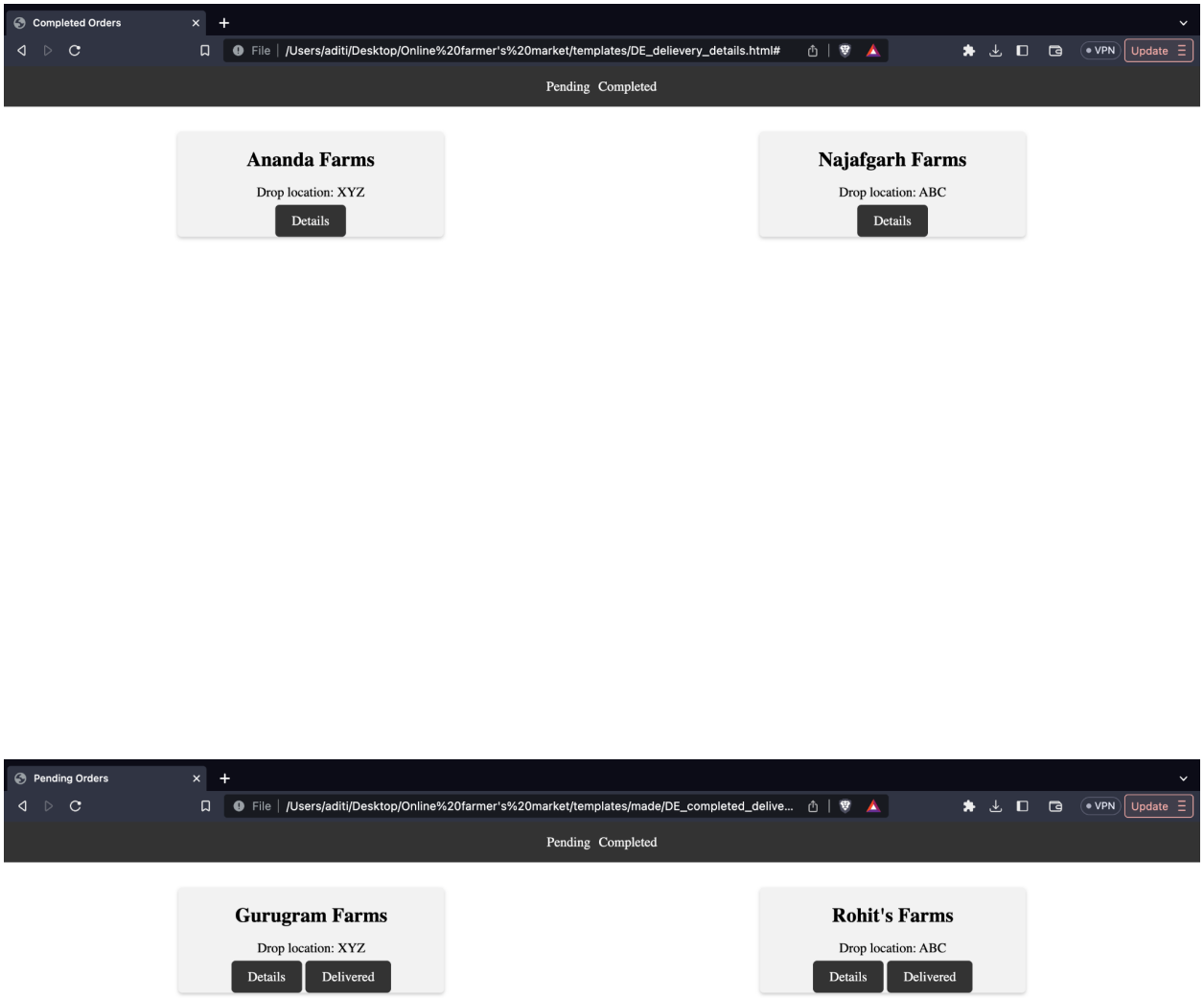

Customer Signin
Customer Signup
Farm Signin
Farm Signup
DE Signin
DE Signup
Admin Signin
Admin Signup
File /Users/aditi/Desktop/Online%20farmer's%20market/templates/made/DE_signin.html
VPN
Online Farmer's market
Please sign in
Email address
Password
Sign in
Don't have an account? Sign-up

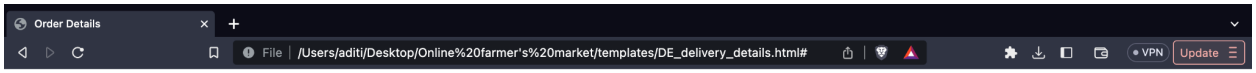

Customer Signin
Customer Signup
Farm Signin
Farm Signup
DE Signin
DE Signup
Admin Signin
Admin Signup
File /Users/aditi/Desktop/Online%20farmer's%20market/templates/made/DE_signup.html
VPN
Online Farmer's Market
Create an account
Name
Email address
Phone number
Address
Mode of Transport
Password
Sign up
Already have an account? Sign in


Customer Signin
Customer Signup
Farm Signin
Farm Signup
DE Signin
DE Signup
Admin Signin
Admin Signup
File
/Users/aditi/Desktop/Online%20farmer's%20market/templates/made/admin_signin.html
VPN
Online Farmer's market
Please sign in
Email address
Password
Sign in
Don't have an account? Sign-up


Customer Signin
Customer Signup
Farm Signin
Farm Signup
DE Signin
DE Signup
Admin Signin
Admin Signup
File
/Users/aditi/Desktop/Online%20farmer's%20market/templates/made/admin_signup.html
VPN
Online Farmer's Market
Create an account
Email address
Password
Sign up
Already have an account? Sign in

Screenshots(Delivery executive)





Order Details

Pick up location: Ananda Farms

Drop location: XYZ

Farmer's name: Rahul

Phone number: 555-555-5555

Customer's name: Seema

Phone number: 555-555-5555

Total price : 1260

Payment: CoD

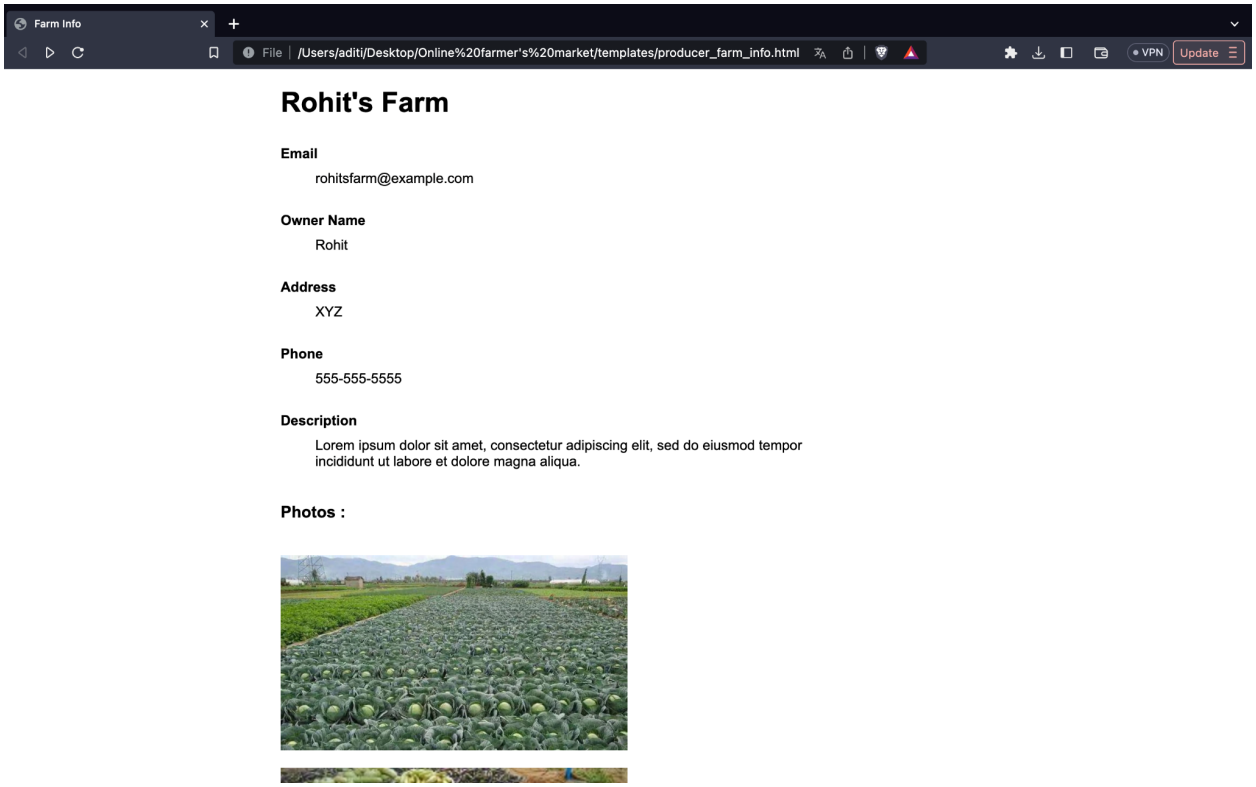
Items to be Delivered

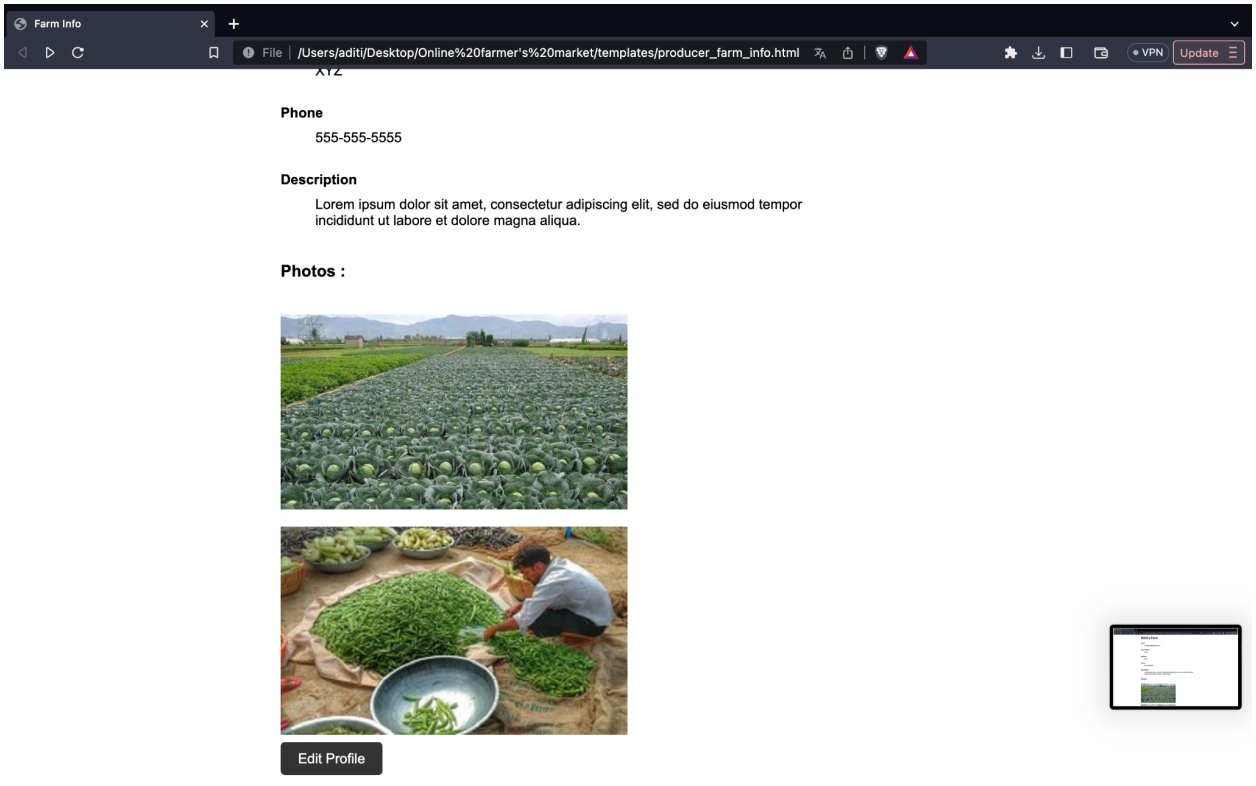
Item	Quantity	Price
Apples	10	1200
Oranges	5	60

Pick up Location

Drop Location

Screenshots(Producer)





Edit Farm Info

File | /Users/aditi/Desktop/Online%20farmer's%20market/templates/producer_edit_farm_info.html

VPN Update

Edit Farm Info

Email:

rohitsfarm@example.com

Owner Name:

Rohit

Address:

XYZ

Phone:

555-555-5555

Description:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

New Image URL:

https://example.com/image.jpg

Save Changes Cancel

Farm Products

File | /Users/aditi/Desktop/Online%20farmer's%20market/templates/producer_farm_products.html

VPN Update

Farm Products

Tomato

Rs. 10 10 kg

Tags: tomato, organic, local, vegetables

This is a description of the product. It can be multiple lines long.

Edit Product Details

Lady finger

Rs. 15 20 kg

Tags: lady finger, organic, local, vegetables

This is a description of the product. It can be multiple lines long.

Edit Product Details

Add a New Product

Edit Product

File | /Users/aditi/Desktop/Online%20farmer's%20market/templates/producer_edit_product.html...

VPN Update

Edit Product

Change Image

Name:
Tomato

Price:
10

Quantity:
10 kg

Tags:
tomato, organic, local, vegetables

Description:
This is a description of the product. It can be multiple lines long.

Save Changes Cancel

Add Product

File | /Users/aditi/Desktop/Online%20farmer's%20market/templates/producer_addProduct.html

VPN Update

Add Product

Name:

Price:

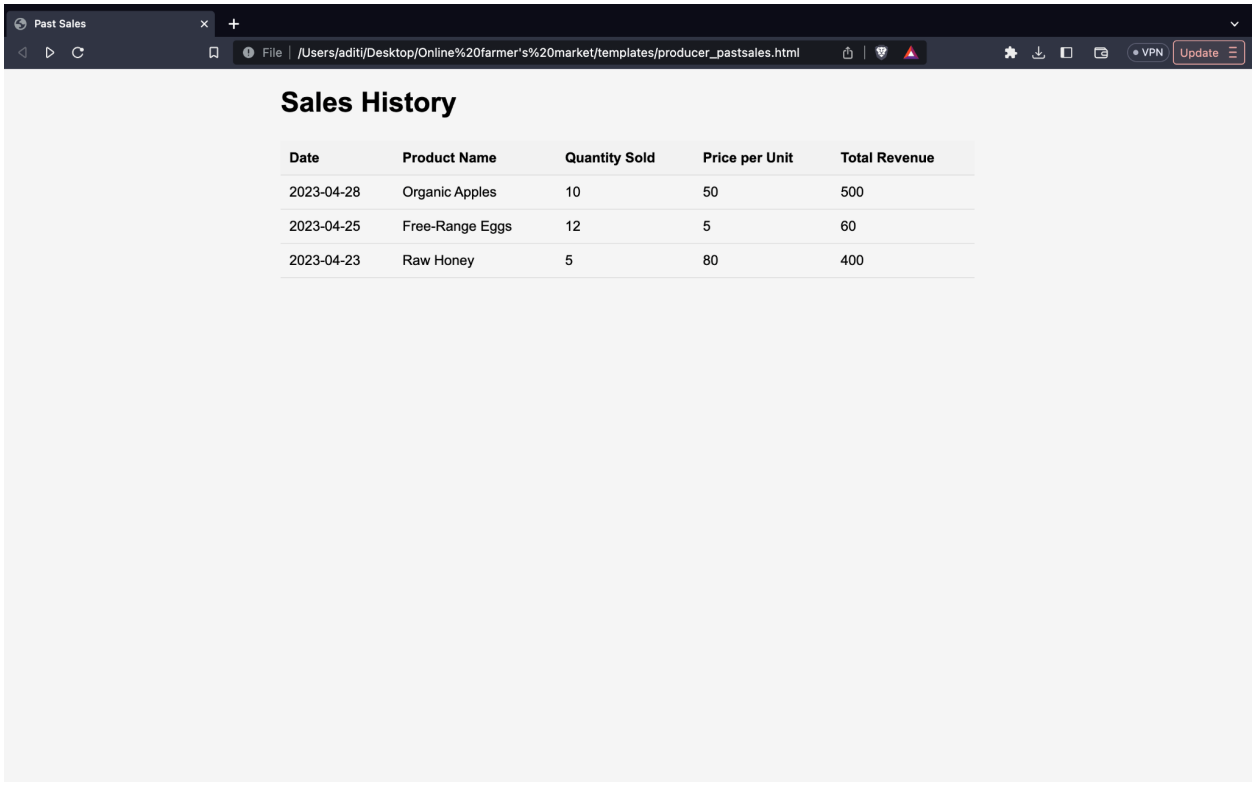
Quantity:

Tags:

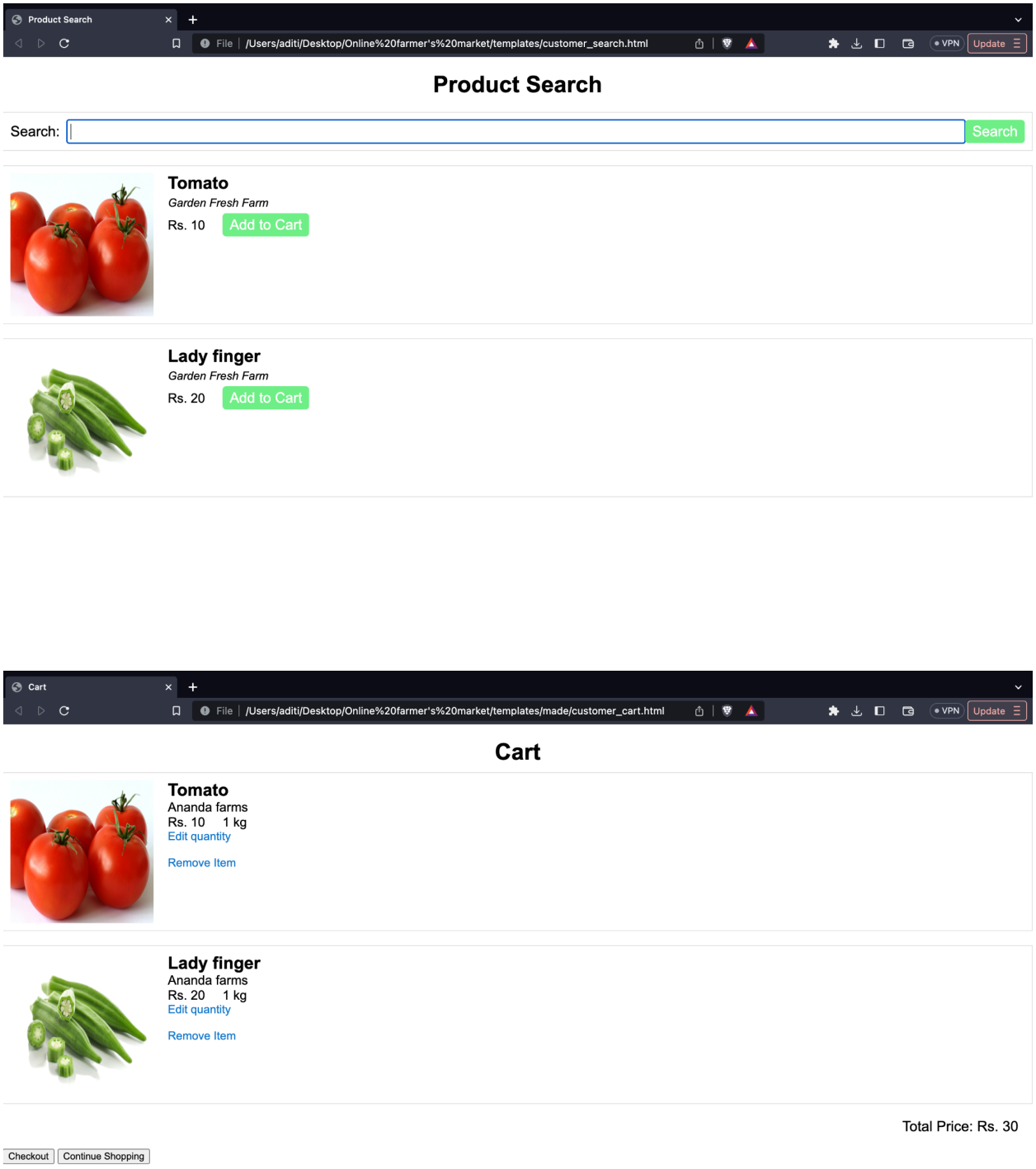
Description:

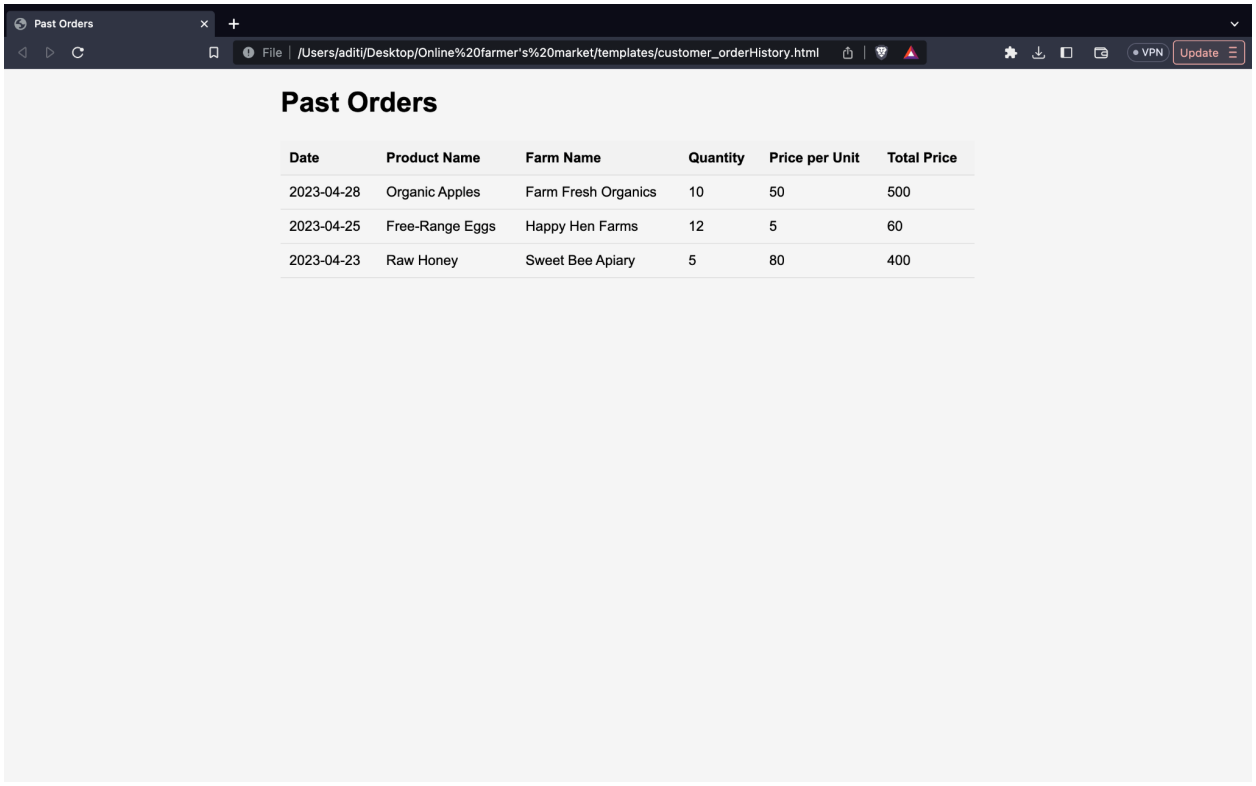
Image : Upload

Add Product Cancel


Past Sales
File | /Users/aditi/Desktop/Online%20farmer's%20market/templates/producer_pastsales.html
VPN
Update
Sales History
Date
Product Name
Quantity Sold
Price per Unit
Total Revenue
2023-04-28
Organic Apples
10
50
500
2023-04-25
Free-Range Eggs
12
5
60
2023-04-23
Raw Honey
5
80
400

Screenshots(Customer)




Past Orders
File | /Users/aditi/Desktop/Online%20farmer's%20market/templates/customer_orderHistory.html
VPN
Update
Past Orders
Date
Product Name
Farm Name
Quantity
Price per Unit
Total Price
2023-04-28
Organic Apples
Farm Fresh Organics
10
50
500
2023-04-25
Free-Range Eggs
Happy Hen Farms
12
5
60
2023-04-23
Raw Honey
Sweet Bee Apiary
5
80
400

Data design - Databases

1. **Admin database**
   - adm_id : Primary key, int type
   - adm_email
   - adm_password

2. **Customer database**
   - cid : Primary key, int type
   - cemail
   - cpassword
   - cname
   - caddress
   - cphone

3. **farm database**
   - fid : Primary key, int type
   - femail
   - fpassword
   - fname
   - ownername
   - faddress
   - fphone
   - fdescription

4. **DE database**
   - did : Primary key, int type
   - demail
   - dpassword
   - dname
   - daddress
   - dphone
   - mode_of_transport

5. **product database**
   - pid : Primary key, int type
   - fid
   - pname
   - pprice
   - pquantity
   - ptags(for searching)
   - pdescription

6. **cart database**
   - cart_id : Primary key, int type
   - cid
   - pid
   - fid
   - quantity
   - total_price
   - status

7. **order database**
   - ord_id : Primary key, int type
   - cart_id
   - cid
   - pid
   - fid
   - did
   - payment_id
   - status(ordered, on the way, delivered, returning, returned)
   - order_date
   - delivery_date
   - quantity
   - payment_mode
   - total_price

models.py

application > models.py > Order

```python
class Admin(db.Model):
    adm_id=db.Column(db.Integer, primary_key=True)
    adm_email=db.Column(db.String(), nullable=False)
    adm_password=db.Column(db.String(), nullable=False)

class Customer(db.Model):
    cid = db.Column(db.Integer, primary_key=True)
    cemail = db.Column(db.String(), nullable=False)
    cpassword = db.Column(db.String(), nullable=False)
    cname = db.Column(db.String(), nullable=False)
    caddress = db.Column(db.String(), nullable=False)
    cphone = db.Column(db.String(), nullable=False)

class Farm(db.Model):
    fid = db.Column(db.Integer, primary_key=True)
    femail = db.Column(db.String(), nullable=False)
    fpassword = db.Column(db.String(), nullable=False)
    fname = db.Column(db.String(), nullable=False)
    ownername = db.Column(db.String(), nullable=False)
    faddress = db.Column(db.String(), nullable=False)
    fphone = db.Column(db.String(), nullable=False)
    fdescription = db.Column(db.String(), nullable=False)

class DeliveyExe(db.Model):
    did = db.Column(db.Integer, primary_key=True)
    demail = db.Column(db.String(), nullable=False)
    dpassword = db.Column(db.String(), nullable=False)
    dname = db.Column(db.String(), nullable=False)
    daddress = db.Column(db.String(), nullable=False)
    dphone = db.Column(db.String(), nullable=False)
    MoT=db.Column(db.String(), nullable=False)

class Product(db.Model):
    pid=db.Column(db.Integer, primary_key=True)
    fid=db.Column(db.Integer, db.ForeignKey('farm.fid'), nullable=False)
    pname=db.Column(db.String(), nullable=False)
    pprice=db.Column(db.Integer, nullable=False)
    pquantity=db.Column(db.Integer, nullable=False)
    ptags=db.Column(db.String(), nullable=False)
    pdescription=db.Column(db.String(), nullable=False)
    cart_items = db.relationship('Cart', backref='product', lazy=True)

class Cart(db.Model):
    cart_id = db.Column(db.Integer, primary_key=True)
    cid = db.Column(db.Integer, db.ForeignKey('customer.cid'), nullable=False)
    product = db.relationship('Product', backref='carts')  ##
    fid = db.Column(db.Integer, db.ForeignKey('farm.fid'), nullable=False)
    quantity = db.Column(db.Integer, nullable=False)
    total_price = db.Column(db.Integer, nullable=False)
    status = db.Column(db.String(), nullable=False, default='pending')

class Order(db.Model):
    ord_id = db.Column(db.Integer, primary_key=True)
    cart_id = db.Column(db.Integer, db.ForeignKey('cart.cart_id'), nullable=False)
    cid = db.Column(db.Integer, db.ForeignKey('customer.cid'), nullable=False)
    pid = db.Column(db.Integer, db.ForeignKey('product.pid'), nullable=False)
    fid = db.Column(db.Integer, db.ForeignKey('farm.fid'), nullable=False)
    did = db.Column(db.Integer, db.ForeignKey('delivey_exe.did'), nullable=False)
    payment_id = db.Column(db.String(), nullable=False)
    status = db.Column(db.String(), nullable=False)
    order_date = db.Column(db.DateTime(timezone=True), default=func.now(), nullable=False)
    delivery_date = db.Column(db.DateTime(timezone=True))
    quantity = db.Column(db.Integer, nullable=False)
    payment_mode=db.Column(db.String(), nullable=False)
    total_price = db.Column(db.Integer, nullable=False)
db.create_all()
```

Pseudocode

main.py

```python
import os
from flask import Flask
from application import config
from application.config import LocalDevelopmentConfig
from application.database import db


def create_app():
    app=Flask(__name__,template_folder="templates")
    if os.getenv('ENV',"development")=="production":
        #if no value setup for ENV variable then it will setup to "development"
        raise Exception("Currently no production config is setup.")
    else:
        print("Starting Local Development")
        app.config.from_object(LocalDevelopmentConfig)

    db.init_app(app)
    app.app_context().push()
    return app

app=create_app()

from application.controllers import main
app.register_blueprint(main)

if __name__=='__main__':
    app.run(host='0.0.0.0',debug=True,port=8080)
```

requirements.txt

```
alembic==0.9.10
Flask==2.2.2
Flask-Login==0.4.1
Flask-Migrate==2.2.1
Flask-SQLAlchemy==3.0.2
Flask-JWT==0.3.2
python-dateutil==2.7.3
python-editor==1.0.3
SQLAlchemy==1.4.44
Werkzeug==2.2.2
requests==2.28.1
flask-swagger-ui==4.11.1
coverage==6.4
py-healthcheck==1.10.1
Flask-Testing==0.8.1
Jinja2==3.0.2
```

HTML template(with CSS)

models.py | controllers.py 4 | producer_signup.html

templates > made > producer_signup.html > html > head > style

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Farm Signup</title>
        <link href="/assets/css/bootstrap.min.css" rel="stylesheet">
        <link href="/assets/css/signin.css" rel="stylesheet">
        <script src="/assets/js/jquery-1.11.1.min.js"></script>
        <script src="/assets/js/bootstrap.min.js"></script>
        <style>
            body {
              display: flex;
              justify-content: center;
              align-items: center;
              height: 100vh;
              background-image: url('https://thumbs.dreamstime.com/b/indian-rural-farmer-using-laptop-149929405.jpg');
              background-size: cover;
            }

            .container {
                display: flex;
                flex-direction: column;
                justify-content: center;
                align-items: center;
                background-color: #fff; /* Add a white background color */
                padding: 20px; /* Add padding to the container */
                border-radius: 10px;
            }


            .form-signup {
                display: flex;
                flex-direction: column;
                align-items: center;
            }

            .form-signup-heading {
              margin-bottom: 20px;
            }

            .form-control {
              font-size: 24px;
            }

            .btn {
              font-size: 24px;
            }

            .heading {
              font-size: 36px;
              font-weight: bold;
              text-align: center;
              margin-top: 20px;
              margin-bottom: 20px;
            }
        </style>
    </head>
    <body>
        <div class="container">
            <h1 class="heading">Online Farmer's Market</h1>
            <br>
            <form class="form-signup" role="form" action="/auth/signup" method="post">
                <h1 class="form-signup-heading">Create an account</h1>
                <input type="text" class="form-control" placeholder="Farm Name" name="fname" required autofocus><br>
                <input type="text" class="form-control" placeholder="Owner Name" name="oname" required autofocus><br>
                <input type="text" class="form-control" placeholder="Email address" name="email" required><br>
                <input type="tel" class="form-control" placeholder="Phone number" name="phone" required><br>
                <input type="text" class="form-control" placeholder="Address" name="address" required><br>
                <input type="text" class="form-control" placeholder="Farm description" name="fdesc" required autofocus><br>
                <input type="password" class="form-control" placeholder="Password" name="password" required>
                <input type="hidden" name="csrf_token" value="{{ csrf_token }}"><br><br>
                <button class="btn btn-lg btn-primary btn-block" type="submit">Sign up</button>
            </form>
            <p class="text-center">Already have an account? <a href="/signin">Sign in</a></p>
        </div>
    </body>
</html>

```

Timeline Chart

A timeline chart is an effective way to visualize a process using chronological order. Often used for managing a project's schedule, timeline charts function as a sort of calendar of events within a specific period of time.

S. no.	Process/Phase	Start Date	End Date
1	Problem Statement	Jan 7, 2023	Jan 10, 2023
2	Initial Requirements	Jan 13, 2023	Jan 18, 2023
3	Requirements Analysis	Jan 21, 2023	Jan 28, 2023
4	Final requirements	Feb2 , 2023	Feb 7, 2023
5	Process Model	Feb 10, 2023	Feb 15, 2023
6	Use case diagram	Feb 18, 2023	Feb 22, 2023
7	Data flow diagram	Feb 23 , 2023	Feb 28 , 2023
8	Sequence Diagram	March 1, 2023	March 8, 2023

9	Function Point Analysis	March 14, 2023	March 18, 2023
10	Data Design	April 7, 2023	April 15, 2023
11	Pseudocode	April 22, 2023	April 25, 2023
12	Coding	April 28, 2023	May 4, 2023

Function Point Analysis

FPA is a standard metric for the relative size and complexity of a software system, originally developed by Alan Albrecht of IBM in the late 1970s. Function Points (FPs) can be used to estimate the relative size and complexity of software in the early stages of development- analysis and design.
It assesses the functionality delivered to its users, based on the user's external view of the functional requirements. It measures the logical view of an application not the physically implemented view or the internal technical view.
The FPA technique is used to analyze the functionality delivered by software and Unadjusted Function Point (UFP) is the unit of measurement.

Objectives of FPA:

1. The objective of FPA is to measure functionality that the user requests and receives.
2. The objective of FPA is to measure software development and maintenance independently of technology used for implementation.
3. It should be simple enough to minimize the overhead of the measurement process.
4. It should be a consistent measure among various projects and organizations.

Types of FPA:

Transactional Functional Type –

1. External Input (EI): EI processes data or control information that comes from outside the application's boundary. The EI is an elementary process.
2. External Output (EO): EO is an elementary process that generates data or control information sent outside the application's boundary.
3. External Inquiries (EQ): EQ is an elementary process made up of an input-output combination that results in data retrieval.

Data Functional Type -

4. Internal Logical File (ILF): A user identifiable group of logically related data or control information maintained within the boundary of the application.
5. External Interface File (EIF): A group of user recognizable logically related data allusion to the software but maintained within the boundary of another software.

Benefits of FPA:

• FPA is a tool to determine the size of a purchased application package by counting all the functions included in the package.
• It is a tool to help users discover the benefit of an application package to their organization by counting functions that specifically match their requirements.
• It is a tool to measure the units of a software product to support quality and productivity analysis.

COMPLEXITY ADJUSTMENT VALUE TABLE

S.No.	Parameters	Adjustment Factors
1.	Does the system require reliable backup and recovery?	4
2.	Are specialized data communications required to transfer information to & from applications?	3
3.	Are there distributed processing functions?	3
4.	Is performance critical?	4
5.	Will the system run in an existing, heavily utilized operational environment?	3
6.	Does the on-line data entry require the input transaction to be built over multiple screens or operations?	3
7.	Does the system require online data entry?	4
8.	Is the code designed to be reusable?	3
9.	Is the system designed for multiple installations in different organizations?	4
10.	Are the inputs, outputs, inquiries complex?	2
11.	Are the ILFS updated online?	3
12.	Is the internal processing complex?	4
13.	Are conversion and installation included in the design?	1

14.	Is the application designed to facilitate change and ease of use by the user?	4

Scale varies from 0 to 5 according to the character of Complexity Adjustment
Factor (CAF). Below table shows scale:
0 = No influence
1 = Incidental
2 = Moderate
3 = Average
4 = Significant
5 = Essential

Formula for calculating Complexity Adjustment Factor (CAF):

$$CAF = 0.65 + 0.01 * \sum fi$$

Calculating ∑ fi

∑ fi = f1 + f2 + f3 +f4 + f5 + f6 + f7 + f8 + f9 + f10 + f11 + f12 + f13 + f14
= 4 + 3 + 3 + 4 + 3 + 3 + 4 + 3 + 4 + 2 + 3 + 4 + 1 + 4
= 45

Calculating CAF

CAF = 0.65 + 0.01 * ∑fi
= 0.65 + 0.01*45
= 0.65 + 0.45
= 1.1

Unadjusted Function Point (UFP)

Calculation: The weight factor is assumed to be average.

Function Type	Estimated count	Weight Factor			Function Type Total
		Simple	Average	Complex	
EI	27	3	4	6	108
EO	18	4	5	7	90
EQ	20	3	4	6	80
ILF	12	7	10	15	120
EIF	3	5	7	10	21

Total Unadjusted Function Point Count =

108+90+80+120+21 = 419

Calculating Function Point Count

AFP = UFP * CAF

AFP = 419*1.1

AFP = 460.9

Where AFP = Adjusted Function Point

UFP = Unadjusted Function Point

CAF = Complexity Adjustment Factor

CYCLOMATIC COMPLEXITY

It is a software metric that measures the logical complexity of the program code. It counts the number of decisions in the given program code. It measures the number of linearly independent paths through the program code. The lesser the cyclomatic complexity the more efficient is our code.

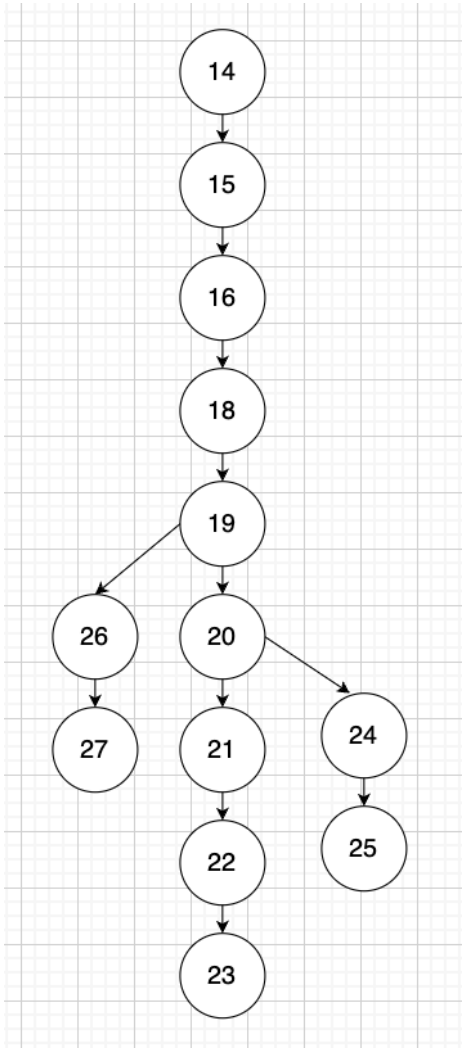
Code :

application > controllers.py > ...

```python
from flask import render_template, request
from flask import current_app as app
from flask import redirect, url_for, flash
from application.models import Farm, Customer, Admin, DeliveryExe, Product, Order, Sales
from application.database import db
from flask import Blueprint
from werkzeug.security import generate_password_hash, check_password_hash
from flask_login import login_user, login_required, logout_user, current_user

main = Blueprint('main', __name__)

@app.route('/ulogin', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form.get('email')
        password = request.form.get('password')

        user = Customer.query.filter_by(email=email).first()
        if user:
            if check_password_hash(user.password, password):
                flash('Logged in successfully!', category='success')
                login_user(user, remember=True)
                return redirect(url_for('views.home'))
            else:
                flash('Incorrect password, try again.', category='error')
        else:
            flash('Email does not exist.', category='error')

    return render_template("login.html", user=current_user)


@app.route('/ulogout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('auth.login'))


@app.route('/usign-up', methods=['GET', 'POST'])
def sign_up():
    if request.method == 'POST':
        email = request.form.get('email')
        cname = request.form.get('firstName')
        password1 = request.form.get('password1')
        password2 = request.form.get('password2')

        user = Customer.query.filter_by(cemail=email).first()
        if user:
            flash('Email already exists.', category='error')
        elif len(email) < 4:
            flash('Email must be greater than 3 characters.', category='error')
        elif len(cname) < 2:
            flash('First name must be greater than 1 character.', category='error')
        elif password1 != password2:
            flash('Passwords don\'t match.', category='error')
```

```
            flash('Passwords don\'t match.', category='error')
        elif len(password1) < 7:
            flash('Password must be at least 7 characters.', category='error')
        else:
            new_user = Customer(cemail=email, cname=cname, password=generate_password_hash(
                password1, method='sha256'))
            db.session.add(new_user)
            db.session.commit()
            login_user(new_user, remember=True)
            flash('Account created!', category='success')
            return redirect(url_for('views.home'))

    return render_template("sign_up.html", user=current_user)


```

Login :



Independent paths:
a. 14, 15, 16, 18, 19, 20, 21, 22, 23
b. 14, 15, 16, 18, 19, 20, 24, 25
c. 14, 15, 16, 18, 19, 26,27

Cyclomatic Complexity:

1. Number of regions: 1
2. V(G) = E - N + 2C *(* C : number of component *)*
   = 12 - 13 + 2*1
   = 1
3. V(G) = P + 1*C
   = 0+1 = 1

Cyclomatic Complexity is 1.