GROUP COURSEWORK

Adithya Phanish

James Govey

Roch Daraz

Ryan Ribeiro

Shayan Haghighi

Tejavardhan Jeenu

Thomas Cartwright

Thomas Muir

University of Birmingham

Software Engineering and Professional Practice

Contents

Requirements Engineering

A1

---

1. Introduction

FarmFresh is a visionary application designed with the primary purpose of redefining the way farmers and consumers interact. With an overarching goal of promoting sustainability, ethical practices, and local sourcing, the application seeks to establish a direct and transparent connection between producers and consumers. This is achieved by providing a platform where farmers can showcase their products and practices, and consumers can make informed, eco-conscious choices. *FarmFresh* contributes to a more sustainable and interconnected food ecosystem.

Product Scope:

Farmers, acting as the backbone of the system, have the capability to create detailed profiles functioning as virtual storefronts. These profiles encompass personal information, farming practices, and a diverse product portfolio. Farmer profiles may include:

- Name of farm/owner
- Their crops or livestock
- When these items are in season (if applicable)
- Where they are located
- Size of the farm
- Pricing for their products
- Information on any events they have organized/are participating in

Farmers will also have a customer rating to their account, based on user feedback provided by customers. The platform supports the dynamic management and updating of product listings, with each product with its:

- Description
- Pricing
- (Seasonal) Availability
- Image
- Labels (Organic, non-GMO etc.)

The consumers and can explore farmer profiles, gaining insights into their practices and allowing them to place orders on items as they choose. The user account information includes:

- Account number
- Username & password
- Contact details
- Location (if access is allowed)

Furthermore, farmers can contribute to transparency by inputting environmental impact metrics related to their practices, which the system will use to calculate statistics on the environmental effect of the application.

Delivery partners can efficiently register and coordinate with farmers and consumers. The system is required to provide customers the ability to choose delivery partners based on their environmental practices and ensure transparency by providing real time tracking of orders. The application is required to act as a centralised hub for communication and logistics management.

The system also incorporates a centralised calendar that displays upcoming events that farmers have organised. Customers can explore event details such as: location participating farmers, event themes, date and similar events. This subsystem will notify farmers and consumers about upcoming events in their area, and after the event, consumers can submit reviews and post event highlights.

Assumptions for Related Subsystems:

- The system assumes a secure user authentication mechanism to protect farmer, consumer, and delivery partner accounts. It assumes the implementation of encryption protocols to secure sensitive data
- The system assumes successful integration with external services, particularly eco-friendly delivery partners. It assumes compatibility with industry-standard data formats for efficient data retrieval, processing, and storage
- The system assumes that financial transactions are handled outside the Farm2Table application. The focus is on connecting farmers and consumers, and actual transactions are expected to occur through established financial channels.

A2

State the functional and non-functional requirements of your system. Use sensible phrasing, grouping and prioritisation as discussed in the sessions.

Functional Requirements

1. Local Farmer Profiles

A farmer should be able to create a farmer profile account that showcases their farming practices and commitment to sustainability. Consumers can then view the farmer profile, browse products and support sustainable agriculture.

a) The farm management system should be accessible either through a mobile application or a web application
b) The system shall implement a registration system that allows farmers to create farmer profile accounts.
    a. The system should display a user-friendly interface form that retrieves registration information from farmers.
    b. The registration interface must adapt to different screen sizes and devices, including tablets, smartphones and desktops.
c) The system shall provide farmers a section to describe their farming practices and commitment to sustainability
    a. The system should implement a WYSIWYG editor that allows farmers to compose their farm descriptions.
    b. A farmer should be able to display farming practices to consumers through the WYSIWYG editor such as the farm's soil conditions, crop rotation and irrigation methods
    c. The system could enable farmers to embed media such as videos and images in their description that showcases their farming practices.
d) The system must allow farmers to update their farming practices, farm descriptions and sustainability certifications
e) The system shall allow farmers to display sustainability certifications on their profiles. Examples include LEAF Marque and RSPCA Assured.
    a. The system must display an official badge that represents a sustainability certification in the farm's account.

2. Product Listings

A customer can explore a wide range of products including fruits, vegetables, poultry and plants.

a) The system must allow farmers to set labels for their products
    a. Product labels include “Organic”, “Seasonal”, “Farm Fresh”, “Free-range”, “Native Species” and “Non-GMO”
b) The system must display the rating of a product
c) The system shall allow farmers to set the visibility preference for a product
    a. The system shall allow products to be listed as having public visibility, member-only visibility and farmer visibility
d) The system must allow farmers to add and remove a product available for sale
f) The system shall allow users to search for a product without being logged in
    a. The system could narrow down a search by applying specific product filters
        i. For example, the system could allow users to search for a product based on a product labels “Organic”, “Free-range”, etc.
    b. The system could allow a listing to be sorted either by price, rating or relevance
g) The system must ensure products listed are in the sterling pound currency
h) The system should allow farmers to upload a picture of a product

3. Environmental Impact Tracking

The Environmental Impact Tracking system enables farmers to monitor carbon emissions, water and waste management usage for each of their farms.

a) The system shall display heat maps that illustrate the carbon emissions in a farm
    a. The system should use different colours to indicate the intensity of emissions
    b. The system should connect to real-time sensors that continuously stream carbon emission data
    c. The system should update and change the colour of the heat map according to the intensity of emissions
b) The system should integrate water usage sensors
    a. The system should calculate water usage by crop or field
    b. The system could provide the farmer the latest weather report including temperature (°C), humidity (RH) and chance of precipitation
c) The system must allow farmers to categorise waste generated on the farm so that it can be displayed to users.
    a. The system allows farms to classify their waste type as “organic”, “recyclable”, “non-organic” and “hazardous”

4. Geolocation Service

a) The system must be capable of handling geographic data across the entire United Kingdom
    a. The system should ensure that geographic data covers rural areas and urban areas
b) The system should provide users an interactable map interface that allows customers to explore agricultural resources and view farm locations
c) The system must allow users to share their precise geographical location coordinates (longitude and latitude)
    a. The system shall send a notification permission to a user to obtain their location
    b. The system should run in the background to provide continuous location updates
d) The system should implement proximity search algorithms
    a. The system should calculate the closest farm given a user location coordinate
    b. The system could implement the Haversine formula to determine the shortest distance between a user current location and a farm.
e) The system shall track and store a user location history

5. Analytics Management

a) The system should store details about a user's device
    a. The system should store operating system, device type, browser version
    b. The system should collect cookies from web browsers
    c. The system shall store login details, item search history, items in the shopping cart, number of pages visited on a website and language preferences.
b) The system shall monitor the number of clicks on each product listing
c) The system could measure the amount of time a user spends viewing a product listing
    a. The system should track time spent on each individual product
    b. The system shall record timestamps of type mm/dd/yyyy-hh:mm:ss for when the user the starts to view a product.
d) The system should monitor user scrolling within pages
    a. The system should identify specific areas and products users tend to scroll
e) The system could incorporate statistical models to predict future price of products given user demand

6. **Seasonal Availability**

a) The system must ensure farmers specify the seasonal availability of each product listing
    a. The system should provide farmers with options to customise availability labels of seasonal products such as “In Season” and “Out of Season”
b) The system shall provide farmers the ability to preview historical seasonal availability information for each product
    a. The system shall provide farmers a detailed dashboard that shows the availability history over different seasons and years.
    b. For example, “Organic Strawberries” were available between Spring (April 17, 2022 - May 31, 2022)
c) The system should notify farmers to review and update seasonal availability of products before the start of each season
    a. The system should automatically set review reminders for each product listing
    b. The system could allow farmers to set their preferred communication channel to receive the seasonal availability notification (email or SMS)

7. Feedback and Ratings

a) The system shall integrate feedback and ratings with customer accounts to allow users to track their contributions
b) The system shall provide a user-friendly interface that allows consumers to provide feedback and ratings for products
    a. The system must ensure customers are logged in before leaving a rating
    b. The system must allow farmers to be rated based on quality of product, freshness and delivery speed
c) The system could allow customers to submit ratings and feedback anonymously
d) The system shall provide customers an option to provide feedback along with ratings
    a. The system should set a reasonable character limit for written feedback
    b. The system could allow customers to attach images of products they purchased allowing for a more detailed feedback comment
e) The system must ensure that feedback and ratings are updated in real-time on product listings
f) The system could filter spam or inappropriate feedback before it becomes publicly visible.

Non-Functional Requirements

1. **Security**

a) The system shall implement Role-based access control (RBAC)
    a. The system should define roles within the Farm System based on job responsibility and authority. Roles include “Admin”, “User”, “Manager”, etc.
b) The system must ensure all collected user data from cookies (login details, search history, items in shopping cart) is encrypted
    a. The system should use different encryption keys for different types of data
    b. The system should rotate encryption keys every 17 weeks for added security.
c) The system must encrypt geographical data of users using industry standard encryption algorithms
    a. The system shall apply a stream cypher (ChaCha20) to the user’s stored device location (longitude/latitude)
d) The system should enforce a strong password policy during signup
    a. A special character is an element of "!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"
    b. Passwords should have at least 12 characters, 3 digits and a special character
    c. The system shall add a *salt* to each password as part of the hashing process
    d. The system shall use Argon2id with a minimum configuration of 19 MiB of memory, an iteration count of 2, and 1 degree of parallelism.

2. **Scalability**

(a) The system must support autoscaling of ReplicaSets within a Kubernetes cluster through Vertical Pod Autoscaler (VPA)
    a. The system should set the CPU Utilisation Threshold to 80%. If the farm’s servers are operating above the threshold, the system should scale up by doubling the number of pods running.
    b. If the farm’s servers are operating below the threshold, the system should scale down by removing 30% of the number of pods running.

(b) The VPA must ensure that pods have sufficient memory resources
    a. The system should increase the memory utilisation of a container by 20% if the container's memory usage consistently spikes during certain periods.

(c) (Load balancing): Nginx should distribute user traffic from the web application across the server pool using Round Robin algorithm to prevent server overload

3. **Availability**

a) The system must have an uptime of 99.92% over a 90-day period (exclude planned maintenance)
    a. The farm server should be operational and available for at least 95% of the time
    b. The system should be able to handle a 45% increase in user location queries during peak hours.
b) The farm server's performance should comply with the level of service outlined in the Service Level Agreement (SLA)
c) The system should ensure that real-time sensor data from crops is available and updated at least every 15 minutes
d) The system shall send app notifications to farmers every hour containing the latest weather report in the region.

3.1 **Availability for Geolocation Service system**

a) The Geolocation Service system should be operational at least 98% of the time
b) The system must display a farm's location within 2 seconds of receiving a displayFarmLocation() request
c) The system should compute the Haversine function with a maximum response time of 800 milliseconds.
d) The system should be accurate in capturing a user's latitude and longitude coordinates, with an error rate of less than 1.5%
e) The system shall convert latitude and longitude coordinates from degrees to radians with an accuracy of least 99.8%.

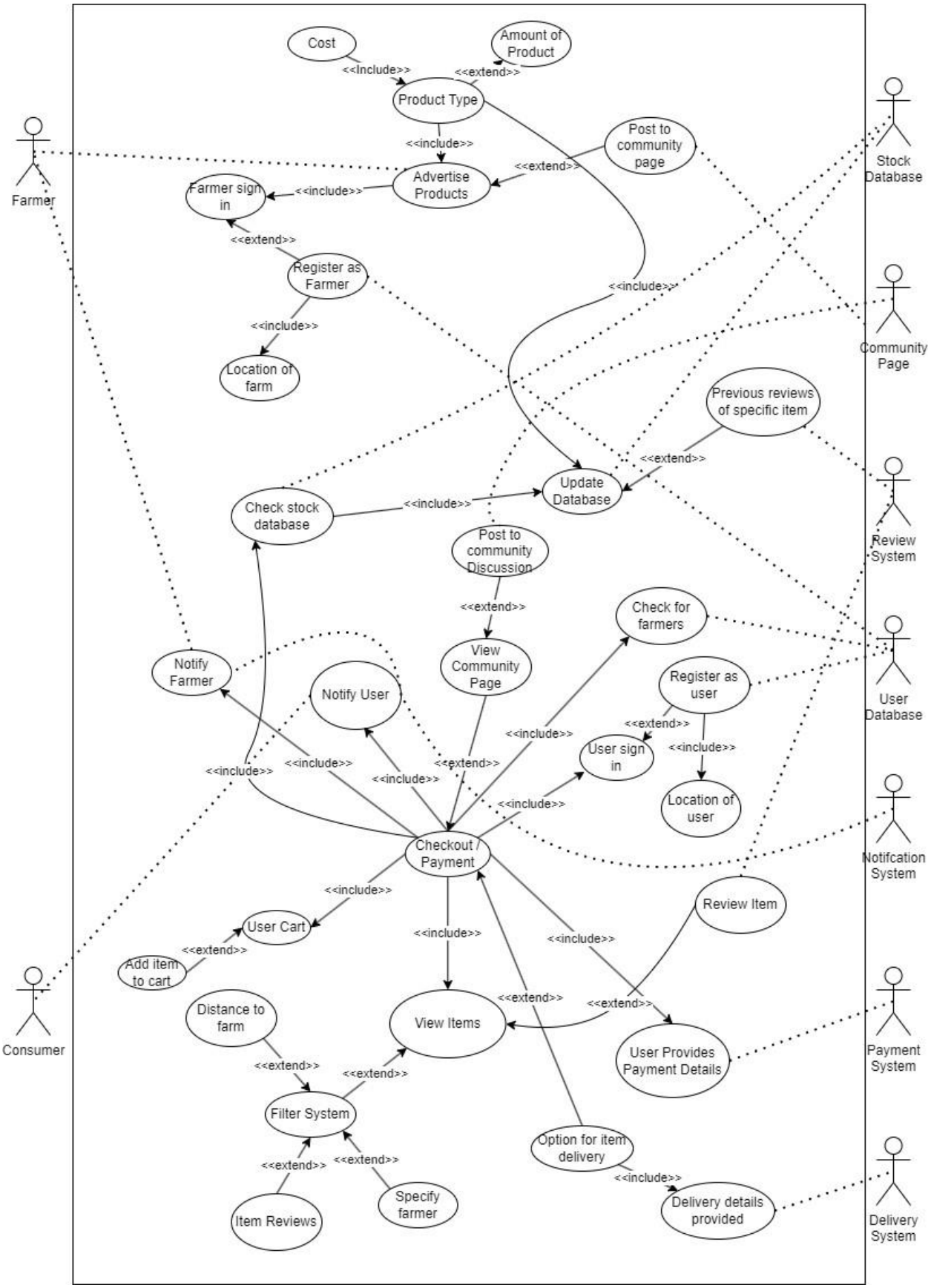
4. **Disaster Recovery**

(a) The system shall have a Recovery Time Objective (RTO) of 6 hours
    a. The phases of the RTO (data restoration, configuration, testing) should be optimised, aiming to resume normal operations within this timeframe
    b. The system should automate repetitive recovery tasks. For example, automating the restoration process of data in case of system failure.
(b) The farm server should identify and log an incident occurred within 15 minutes of the occurrence
(c) The farm's data centre should be powered by alternative power sources such as generators, with a minimum runtime of 48 hours
(d) The farm's databases should be backed up at least once every couple of days

Software Specification, Analysis and Design with UML

B1: Use Case Diagram

The Use Case Diagram for the local farmer product listing system is presented on the next page. All the actors included in the system are shown outside the boundary, within the boundary, identified use cases are displayed along with their relationship with other use cases, shown using arrows, and, if applicable, their relationship with actors, shown using dotted lines.


Cost
Amount of Product
<<Include>>
<<extend>>
Product Type
<<include>>
Post to community page
<<extend>>
Advertise Products
Farmer sign in
<<include>>
<<extend>>
Register as Farmer
<<include>>
Location of farm
<<include>>
Farmer
Stock Database
Community Page
Review System
User Database
Notifcation System
Payment System
Delivery System
Consumer
Previous reviews of specific item
<<extend>>
Update Database
Check stock database
<<include>>
Post to community Discussion
<<extend>>
View Community Page
Check for farmers
Notify Farmer
Notify User
Register as user
<<extend>>
User sign in
<<include>>
Location of user
<<include>>
<<include>>
<<include>>
<<extend>>
<<include>>
<<include>>
<<include>>
Checkout / Payment
<<include>>
User Cart
<<extend>>
Add item to cart
<<include>>
<<include>>
Review Item
Distance to farm
View Items
<<extend>>
<<extend>>
User Provides Payment Details
<<extend>>
<<extend>>
Filter System
<<extend>>
<<extend>>
Item Reviews
Specify farmer
Option for item delivery
<<include>>
Delivery details provided

B2: Critical Use Cases

New farmer gets setup in the application

Preconditions:

1. They must not already be logged in as a farmer
2. The individual must not already be registered

Flow of events:

1. The use case begins when the farmer chooses the option to register from the farmer sign in page.
2. The user must select their farm's location from a map or dropdown menu
3. The system checks that a farmer with a certain identifier ie. Email, location, name, phone number, a combination, does not already exist in the user database, to avoid duplicate users.
4. If they do not exist, the farmer may enter their details, including their farm location, and proceed to sign up
5. A new farmer profile with the provided details is created

Postconditions:

1. Farmer added to user database
2. Farmer is now logged in and can proceed to add stock and customise their profile

New User (consumer) get set up in the app and searches for a local farm.

Preconditions:

1. They must not already be logged in as a user

Flow of events:

1. The use case begins when the user chooses the option to register from the sign-in page.
2. The system checks that a user with the same email address or phone number does not exist to avoid duplicate users. Not as much of an issue as farms so we needn't implement further checks like with the farm
3. If they do not exist, the user may enter their details, including their location, and proceed to sign-up
4. The user can begin a search for a particular farm by using the filter system, they specify the farmer using the FarmName filter to find their local farm.

Postconditions:

1. The user is added to user database
2. The user is now logged in to the application

User orders a product

Preconditions:

1. User is logged in
2. User has the items in their cart
3. Product is in stock (Check stock database) against cart items

Flow of events:

1. If the basket has items, the use case begins when the user chooses the checkout option from the 'Basket' page (Shopping cart).
2. Otherwise, the user will need to select some items to add to the basket, from the page of their local farm. Step 1 repeats
3. The system checks the stock of the items in the card once more, when the user selects this option.
4. If the item is not available, it will be removed from the cart, followed by an apology to the user
5. Otherwise, the user proceeds to a payment/shipping page where they may select the address to which the items are shipped, and a payment method, accompanied by appropriate error checking for these forms.

Postconditions:

1. Stock decremented in database
2. Order created and tracked in the database, with records being updated accordingly

Farmer makes a community post

Preconditions:

1. Farmer is logged in

Flow of events:

1. The use case begins when the farmer navigates to the community page of the application
2. The farmer selects the option to create a new post
3. The farmer is prompted to enter the content of their post, including text, links, images and videos
4. Posts are checked for spam/violations of guidelines
5. If approved, the post is published to the local community's board

Postconditions:

1. Post added to community board and visible to other users that are a part of the community
2. For analytical purposes, engagement with the community is recorded for each farmer, with the corresponding response from the community. This data can be used to improve the application in order to help our farmers connect better with the community.
3. Users of the community who have enabled notifications, will receive a notification through the medium of their choice.

B3: Use case stories

New farmer gets set up in the app:

A farmer registers as a farmer in the app using the user database system. They enter email and confirm that they have access to that email with a 2 factor authentication code. They then enter a password and confirm it with secondary field that must match exactly. The password must also be at least 12 characters long.

The farmer next sets their farm location by pinpointing on a map. Our Backend system stores the farm location (as geocoded coordinates) in the database along with the farmers farmerID.

The farmer adds a new product to their farm that consumers will be able to purchase. To do this, they will first of all select a pre-existing product or add a new one. Then they will specify the available quantities and corresponding prices (our system will deal with sales and keeping track of stock, however a farmer can also manually update the stock level at any point). Next they will have the option to set the season time for that product so they can schedule it to become available if it isn't ready yet, or schedule it to stop being sold once the season's over. The Backend system adds this to the database linked to the farmID and farmerID and then advertises this to users who are either following that farmer's farm or are looking for that specific product.

(Farmer repeats this process for however many products they wish to sell)

New user (consumer) gets set up in the app and searches for local farm:

A new user registers by entering their email and confirming with an emailed code, then enters password and confirms (same as for farmers).

They then will be asked to set their location either by allowing access to location services on their device, or by entering their address, or by pinpointing on a map, all of these are turned into geocoded coordinates that are stored with the users profile in the database

The user can then navigate to explore the farms page and either just clicks search or filters by an item they want, and then presses search. The backend system then takes the user's location and calculates the distance between them and all stored farms. If the user specified a product, it will also filter out farms that don't sell the product. It will then return a list of farms ordered by distance from the user. The user can than can then follow any farms they are interested in or buy a product straight away.

User orders a product:

User finds a product they would like to buy from a specific farm and adds the product to their cart, also specifying the quantity by typing in the quantity box or incrementing/decrementing using the adjustment arrows. Backend stock system double checks that that quantity of product is available from that farm before adding the item to the cart as quantities could have changed since loading the page (If not available will show error message)

The user may go and add more products to their cart before checking out.

Next the user goes to the checkout page and specifies if they want the product(s) to be delivered or the want to pick up. If delivery is selected, the user provides their delivery address which is validated to be a real address. The user enters their payment details and completes the transaction which includes one last stock check to ensure there's never a situation where a user has paid but there isn't enough stock. The stock system then decrements all stock for products that were purchased by their purchased quantities.

The backend system creates a new order and stores this in the database. The corresponding farmer is then notified of the order and told when it needs to be ready for pickup by/sent for delivery.

Once the order has been picked up by the user or delivered to the user, it is marked as complete.

Farmer makes community post:

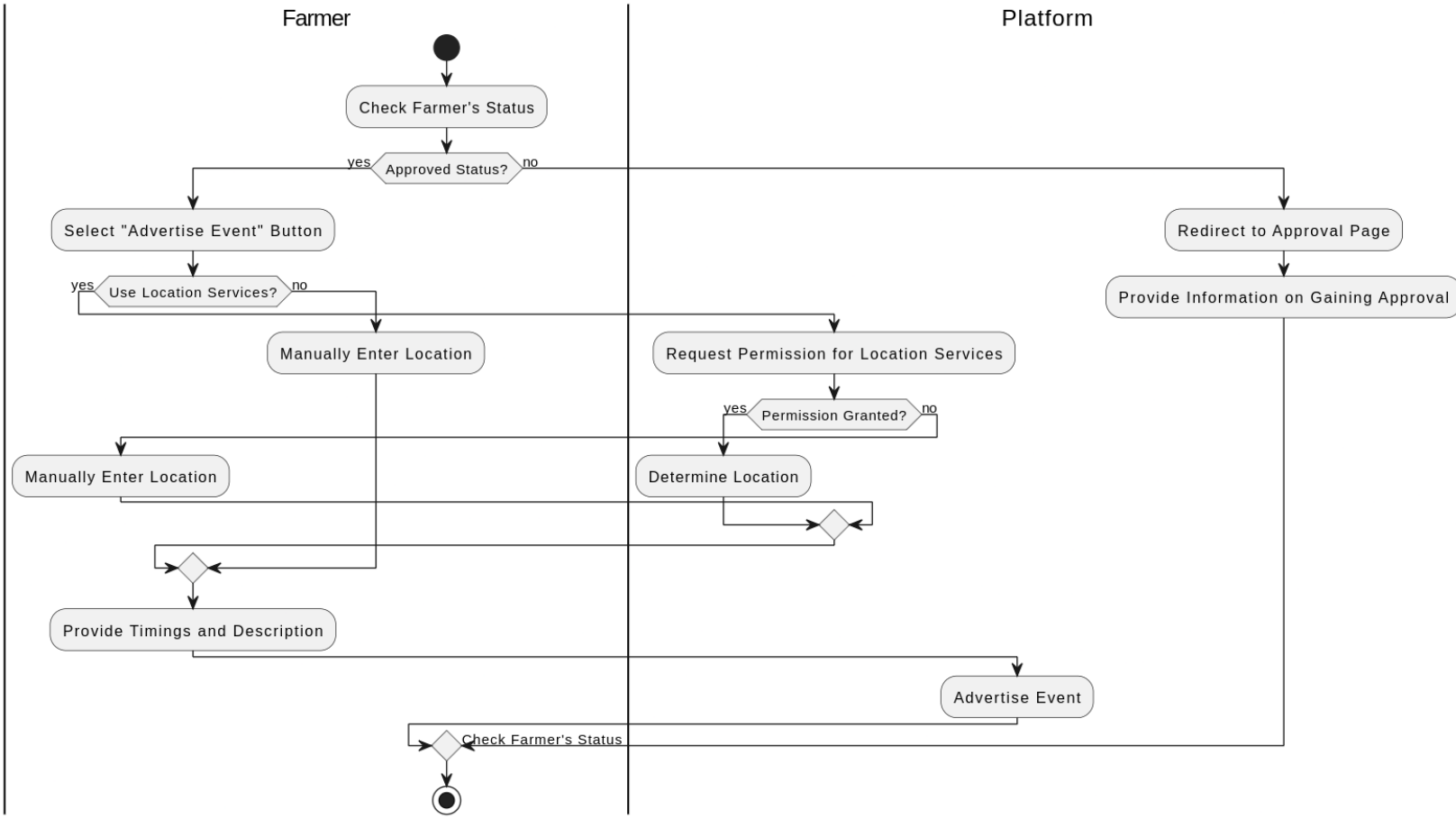
A farmer navigates to community post section and clicks New community post. They either add new product(s) to advertise or chooses existing product(s) they sell. They can then either leave the price as it was or adjusts for community post deal.

Next, the farmer confirms the farm location and whether delivery is accepted or only pickup is available as this could be different to normal for the deal.

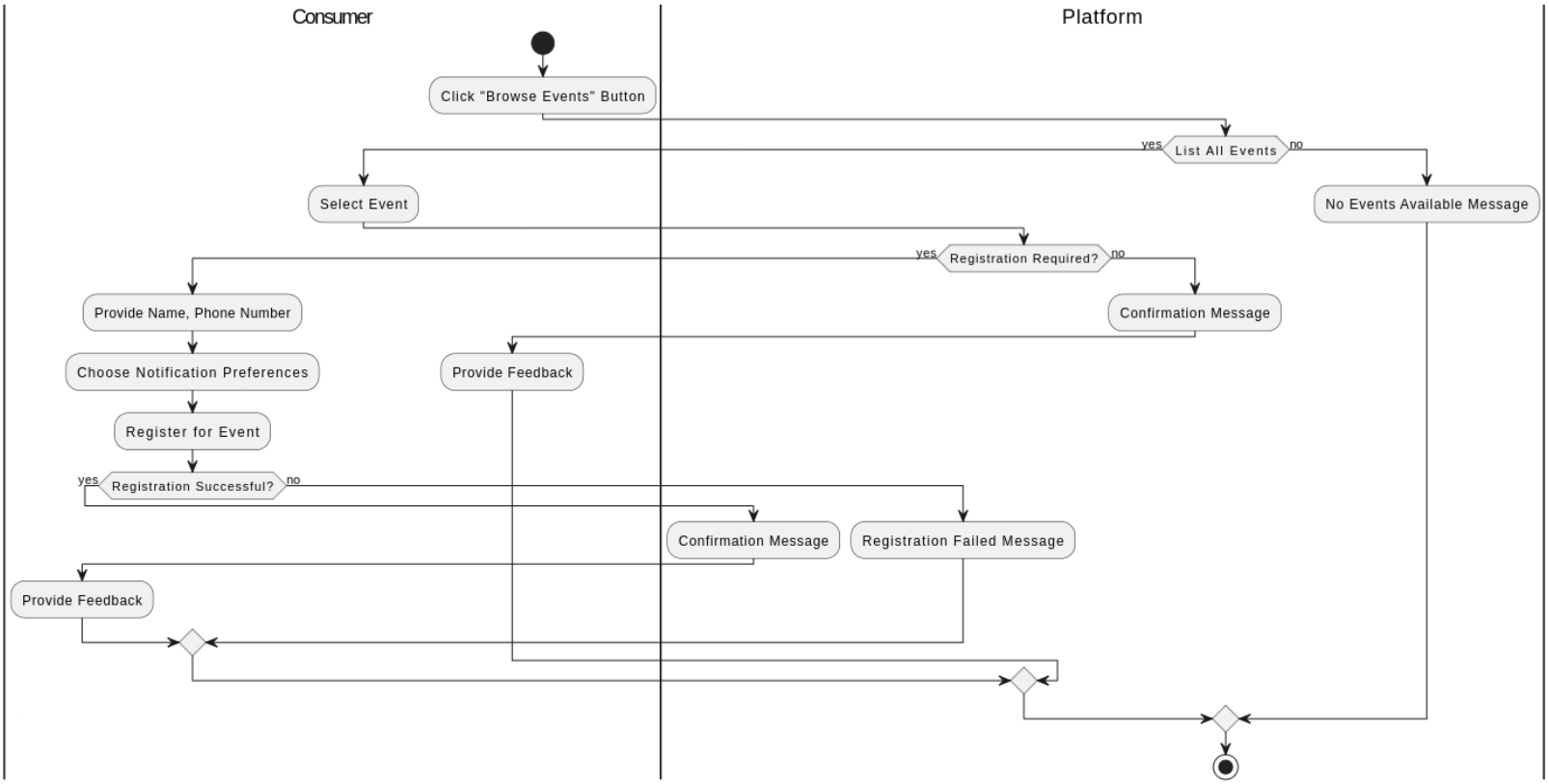
The farmer writes the text for the community post and clicks 'send community post'. The backend system checks this text post for any language against the guidelines and then if

clean stores the post in the database and submits to the community page. Any users signed up for notifications to that farm or for the product(s) will get a push notification if they are signed into the mobile app and have notifications enabled.

B4: Farmer adding an event (e.g. farmers market) activity diagram



User subscribing to an event activity diagram



B5: Class Analysis

B5: Noun Analysis

By identifying relevant noun and verb phrases, potential classes, and potential attributes and methods for these classes

Class	Use		Class	Use
FarmFresh			Name of farm	Attribute (farmers)
farmers	Class (sub-class of profile)		Order price	Attribute (orders)
consumers	Class (sub-class of profile)		events	Class
sustainability	Too vague		product	Class
connection			product listings	Attribute (farmers)
platform	Similar class exists		orders	Class
Food ecosystem	Too vague		Account name	Attribute (profiles)
profiles	Class		Username	Attribute (profiles)
Virtual storefront	Class exists		password	Attribute (profiles)
(information on) farming practices	Attribute (farmers)		Contact details	Attribute (farmers)
Product portfolio	Attribute (farmers)		location	Attribute (farmers)
Image of farm	Attribute (farmers)		Seasonal availability	Attribute (product)
Dynamic management	Too vague		Centralised hub	Too vague
Description of farm	Attribute (farmers)		Crops	Attribute (famers)
Review	Class		Livestock	Attribute (farmers)
Transparency	Too vague		Environmental impact metrics	Class
Delivery partners	Class		Management of product listings	Class

calendar	Class		Event location	Attribute (event)
Event theme	Attribute (event)		Participating famers	Attribute (event)
Post event highlights	Attribute (event)		Real time tracking	Attribute (consumer)
Labels	Attribute (product)		Pricing	Attribute (product)

B5: Verb Analysis

Class	Possible Class Method		Class	Possible Class Method
Create Profile	Profile class		Gain insights	Too vague
feedback	Customer class		Input environmental metrics	Farmer class
Integrate with external services	Too vague		register	Profile class
Explore profiles	Profile class		Coordinate with farmers	Too vague
Place orders	Customer class		Interact with consumers	Too vague
Secure user authentication	Outside of scope		Updating of product listings	Update product class
Management of product listings	Product class		Protect farmer	Too vague
Calculate statistics	Environmental Metrics class		Choose delivery partner	Order class
Manage logistics	Too vague		Coordinate with famers	Too vague
Handle financial transactions	Outside of scope		Update product listing	Update product class
Explore event details	Calendar class		Notify farmers	Calendar class
Notify consumers	Calendar class		Submit review	Review class
Submit post-event highlights	Calendar class		Ensure transparency	Too vague

B5: CRC Cards

CRC cards were then created visually represent the potential classes identified during the noun-verb analysis

Update Products	
Responsibilities	**Collaborators**
• Manage the updating of product information by farmers. • Provide methods to modify product descriptions, pricing, availability, and images. • Collaborate with the "Farmer" class to associate updates with specific products.	• Farmers • Products

Products	
Responsibilities	**Collaborators**
• Provide information about the types of products offered by farmers, such as seasonal availability, stock, price etc.	• Farmers • Orders • Update Products

Farmer	
Responsibilities	**Collaborators**
• Create detailed profiles to showcase farming practices and commitment to sustainability • Showcase products on the platform • Establish a direct connection with consumers • Share information about farming practices and sustainability	• Update Products • Consumers • Reviews • Events • Products

Delivery Partners	
Responsibilities	**Collaborators**
• Provide eco-friendly delivery services	• Order

Consumer	
Responsibilities	**Collaborators**
• Explore and discover local farmers • Explore local farmers' products • Make informed purchasing decisions based on comprehensive product listings	• Farmers • Reviews • Order • Calendar

Events	
Responsibilities	**Collaborators**
• Manage information about real-life market events organized by farmers. • Store details about event schedules, locations, and participating farmers. • Provide methods for creating and updating events.	• Farmers • Calendar

Reviews	
Responsibilities	**Collaborators**
• Capture and manage user reviews for farmers. • Store review content, ratings, and timestamps. • Provide methods for submitting and retrieving reviews.	• Farmers • Consumers

Payment	
Responsibilities	**Collaborators**
• Handle payment transactions within the Farm2Table system	• Order • Consumer

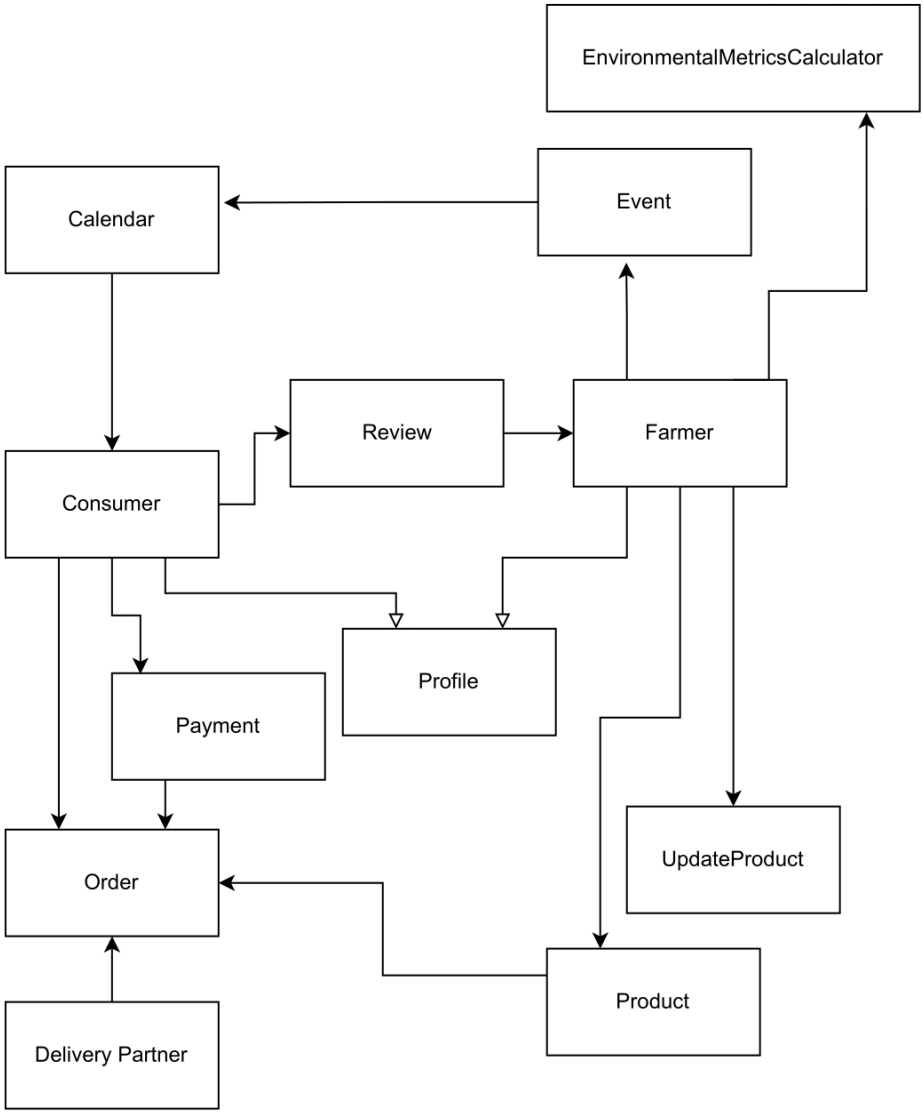
Calendar	
Responsibilities	Collaborators
• Manage scheduling and organization of events within the system. • Provide methods for creating, updating, and retrieving event details.	• Events • Consumer

Environmental Metrics Calculator	
Responsibilities	Collaborators
• Calculate and manage environmental impact metrics for farmers.	• Farmer

Order	
Responsibilities	Collaborators
• Manage details of customer orders within the FarmFresh system. • Store information about the ordered product, associated consumer, and delivery status.	• Delivery Partner • Consumer • Product

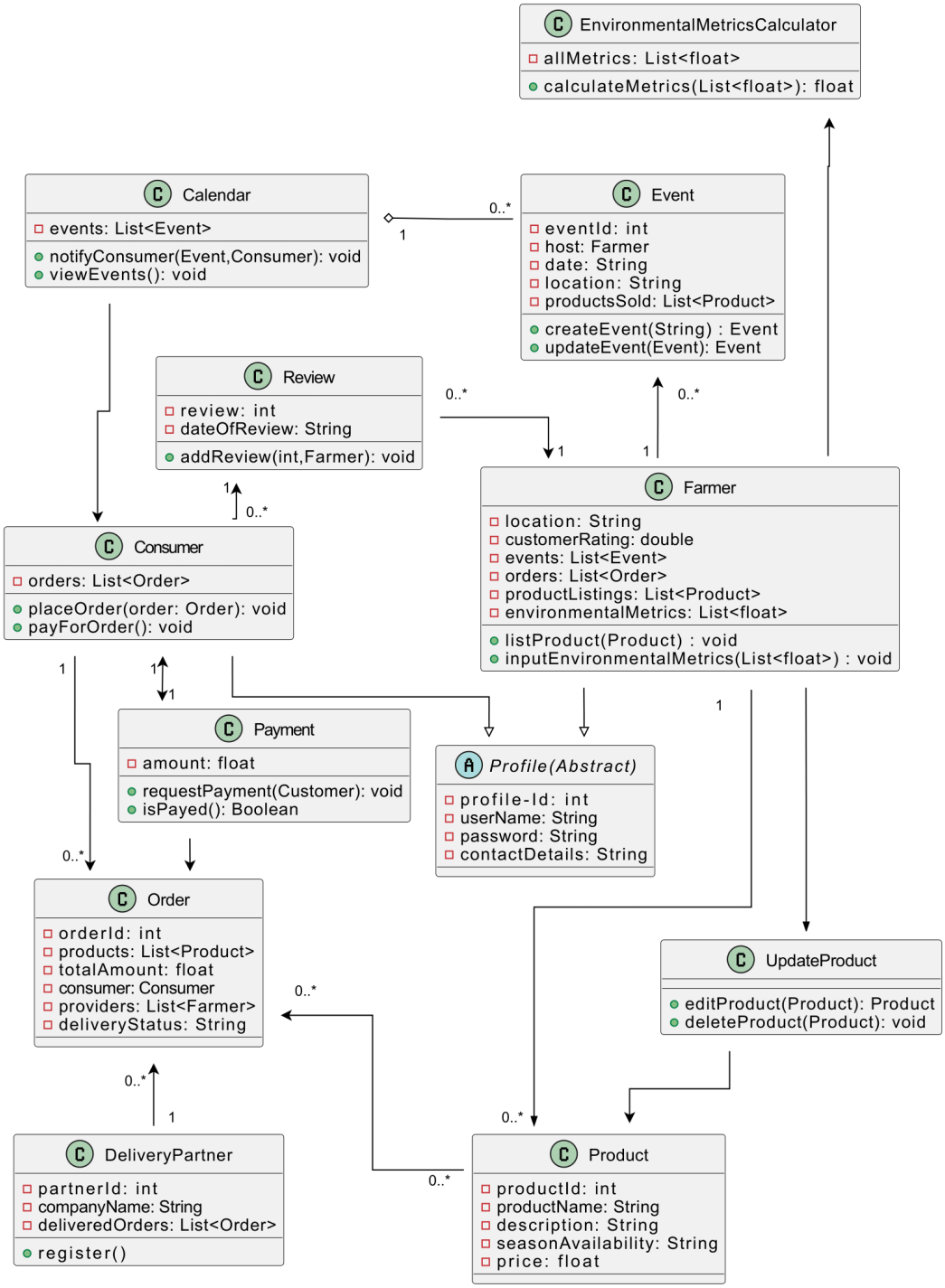
B5: First Cut Class Diagram

CRC cards and noun verb analysis were then used to create a first cut class diagram, showing how the classes work together. See diagram on next page.



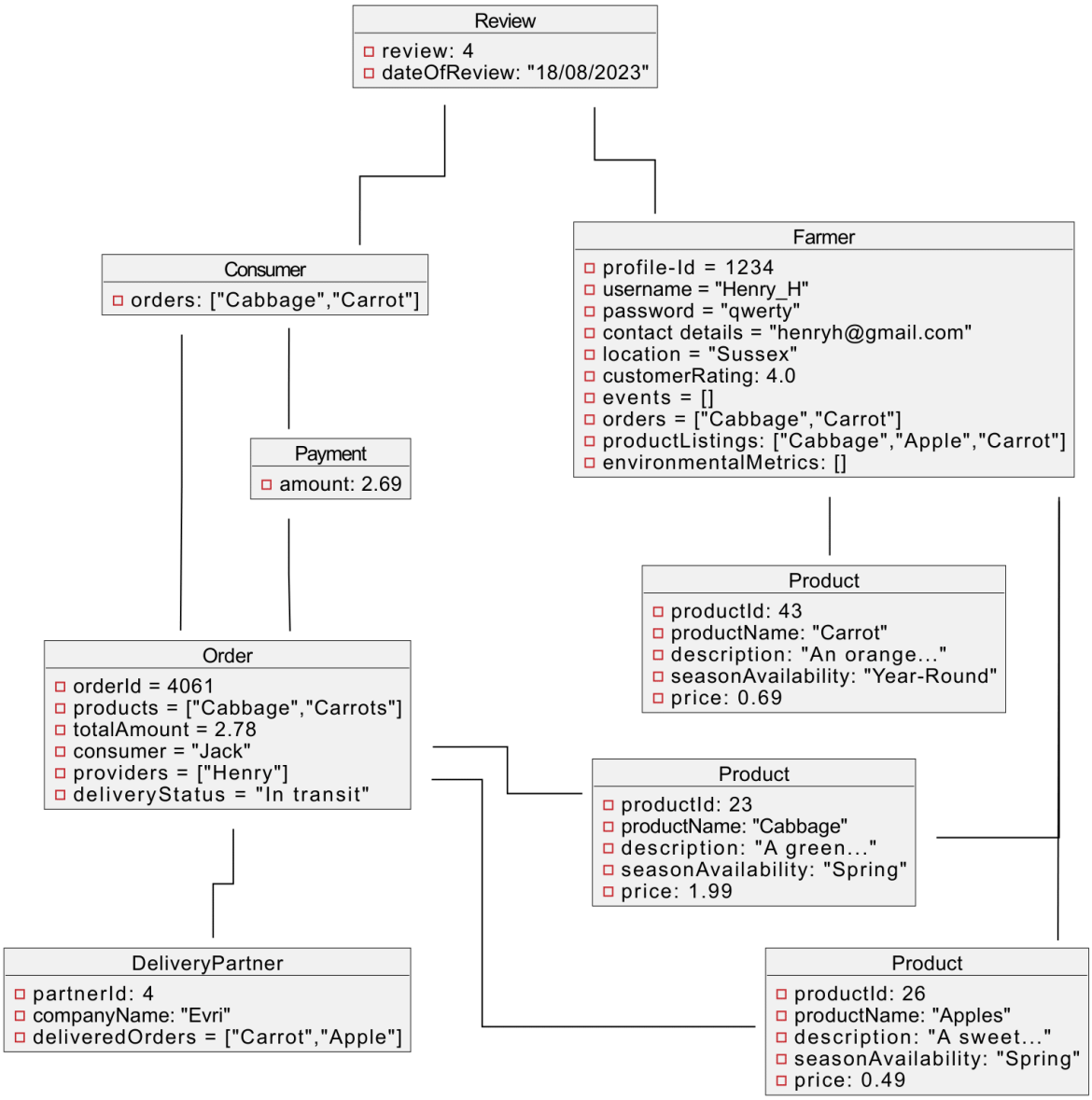
B5: Class Diagram

From this first cut diagram, the full classes, with attributes and methods, were created.
See next page for diagram.

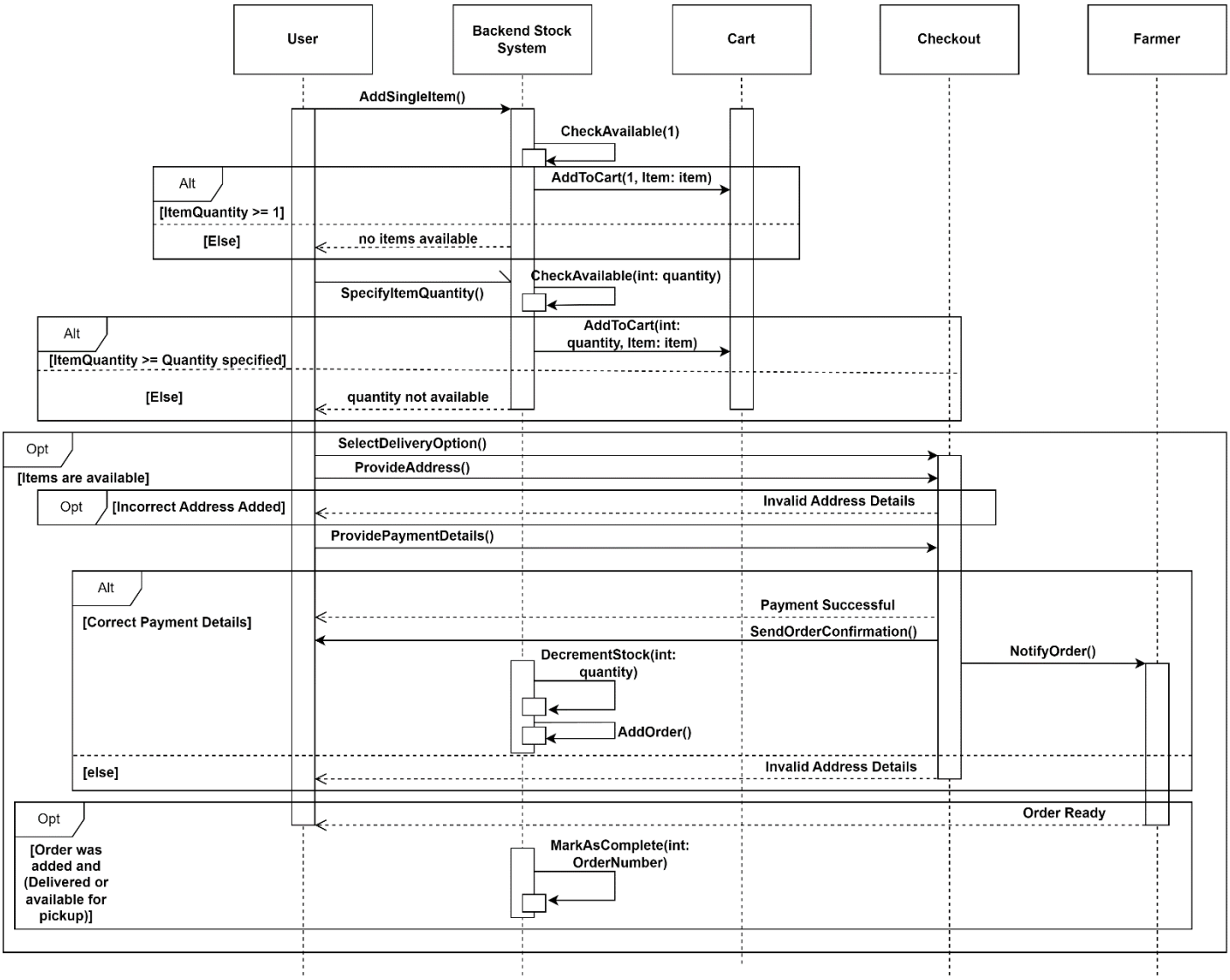

EnvironmentalMetricsCalculator
allMetrics: List<float>
calculateMetrics(List<float>): float
Calendar
events: List<Event>
notifyConsumer(Event,Consumer): void
viewEvents(): void
Event
eventId: int
host: Farmer
date: String
location: String
productsSold: List<Product>
createEvent(String) : Event
updateEvent(Event): Event
Review
review: int
dateOfReview: String
addReview(int,Farmer): void
Farmer
location: String
customerRating: double
events: List<Event>
orders: List<Order>
productListings: List<Product>
environmentalMetrics: List<float>
listProduct(Product) : void
inputEnvironmentalMetrics(List<float>) : void
Consumer
orders: List<Order>
placeOrder(order: Order): void
payForOrder(): void
Payment
amount: float
requestPayment(Customer): void
isPayed(): Boolean
Profile(Abstract)
profile-Id: int
userName: String
password: String
contactDetails: String
Order
orderId: int
products: List<Product>
totalAmount: float
consumer: Consumer
providers: List<Farmer>
deliveryStatus: String
UpdateProduct
editProduct(Product): Product
deleteProduct(Product): void
DeliveryPartner
partnerId: int
companyName: String
deliveredOrders: List<Order>
register()
Product
productId: int
productName: String
description: String
seasonAvailability: String
price: float
0..*
1

B6: Object Diagram

From the class diagram a snapshot of the system attributes were created as an object diagram.

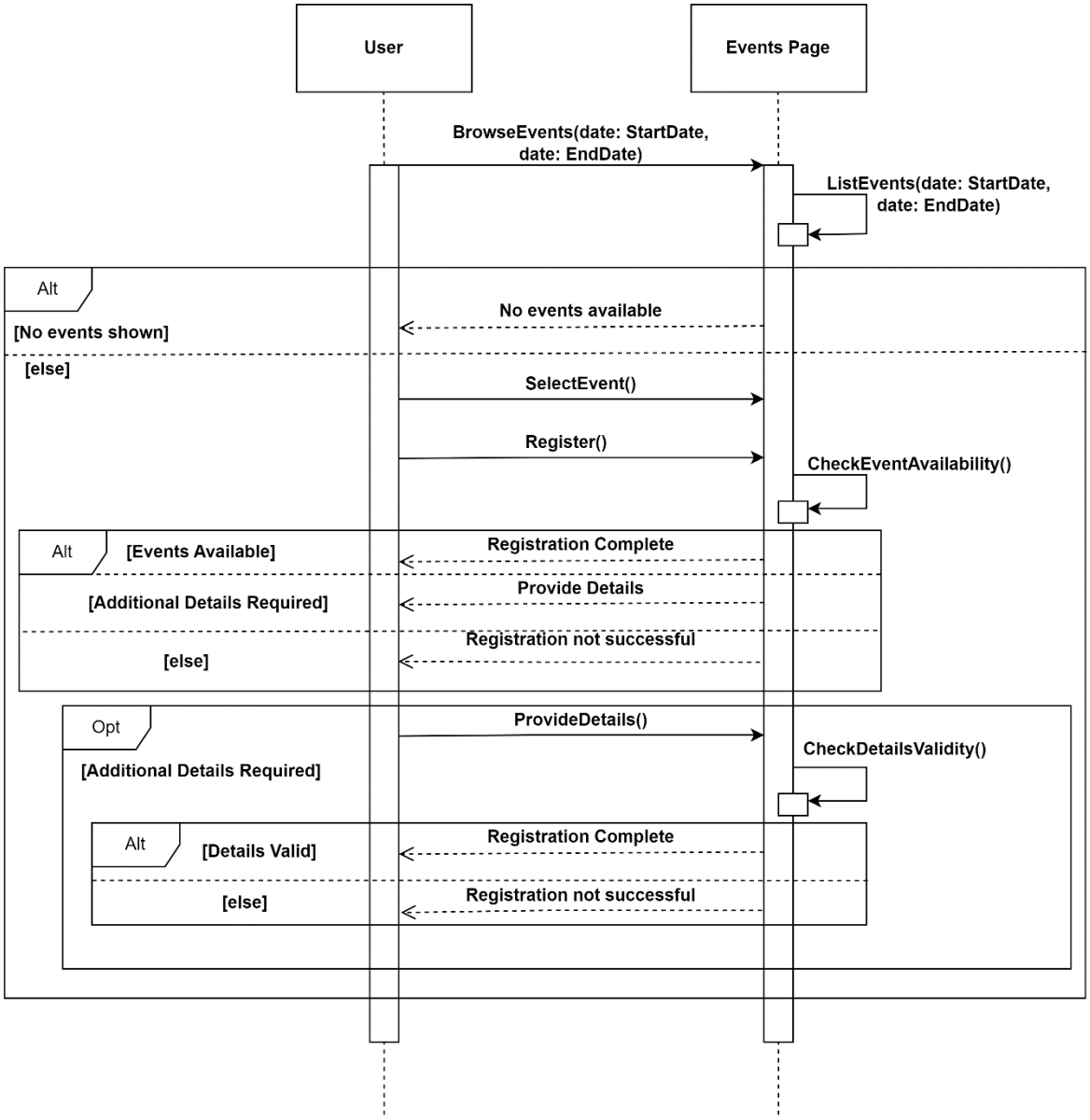


B7: Sequence Diagram for a User ordering a Product



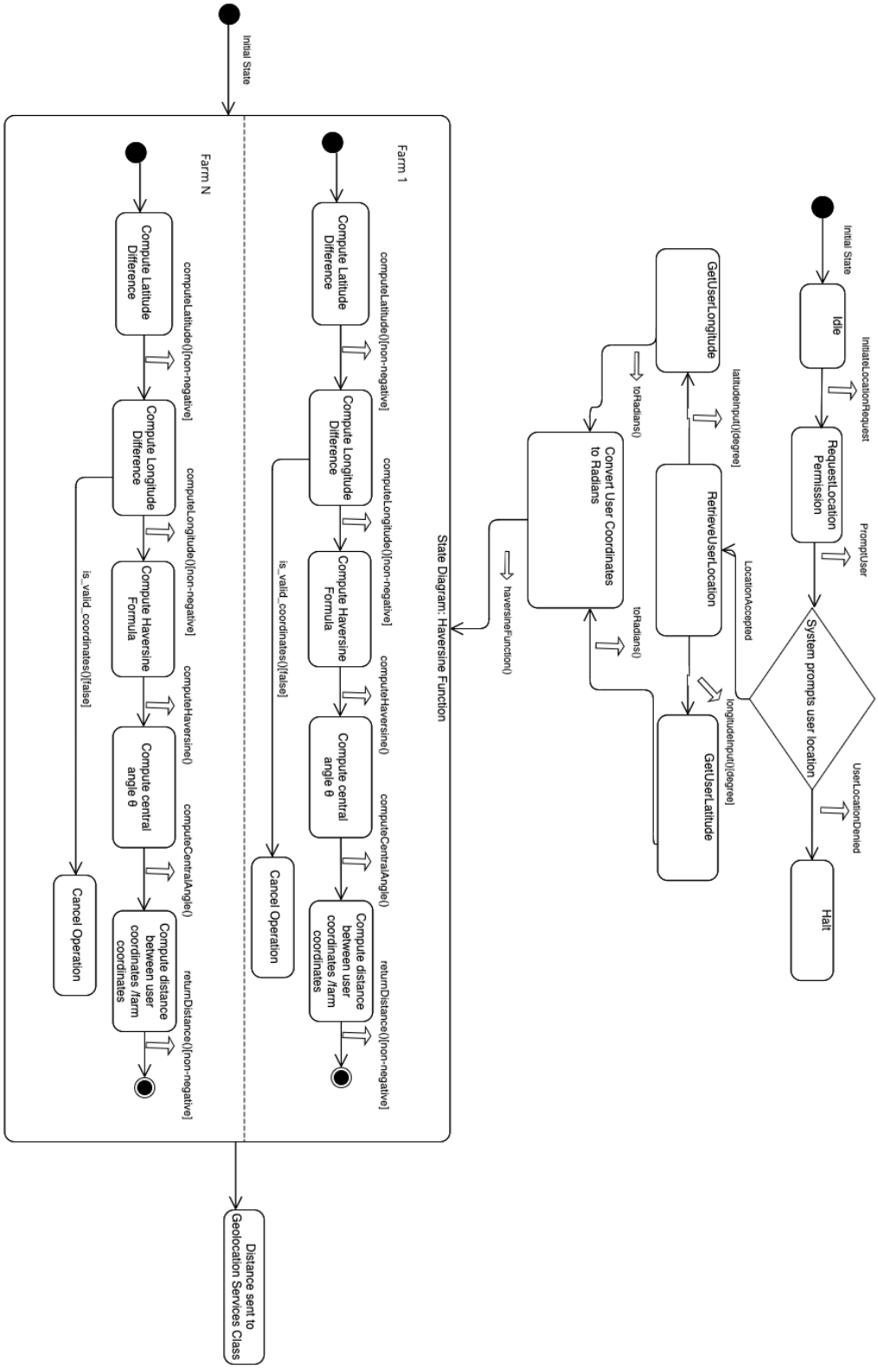
Here when I mention SendOrderConfirmation(), I made the assumption that it is an Email sent to the User utilising the account details of the user to identify the Email to send to.

Sequence Diagram for a User subscribing to an event



B8: State Diagram

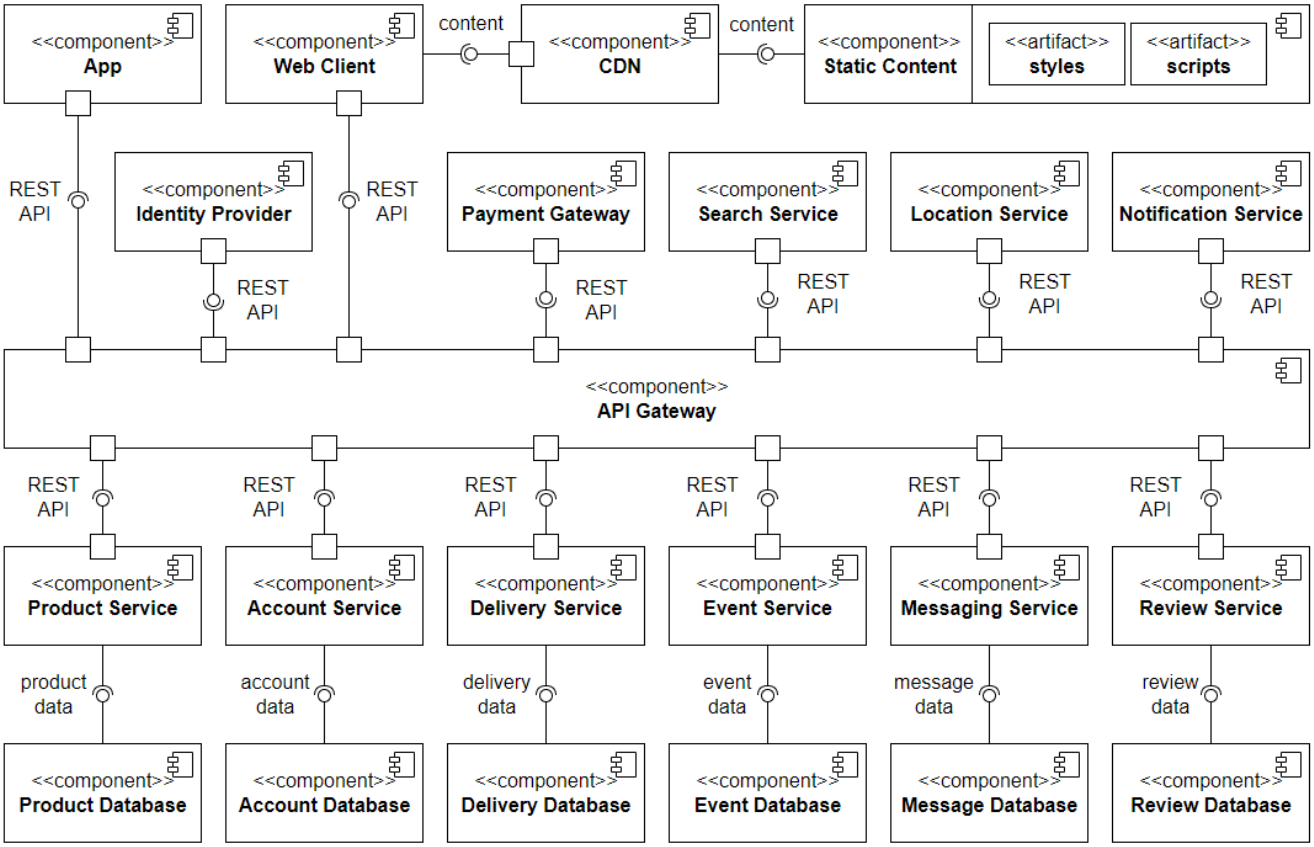
The Geolocation Services state machine diagrams illustrate the states involved in managing location-based functionalities within the system, retrieving user device location information and calculating concurrently the distances between a user and each farm coordinate N stored in the database. In the Haversine Function diagram, we assume each farm coordinate is stored in a database as key-value pair. Key: name of the farm, value is the corresponding latitude-longitude. The diagram incorporates error-handling states to manage issues during the geolocation process, and ultimately sends the distance result to the Geolocation class. See next page for diagram.


Initial State
Idle
InitiateLocationRequest
RequestLocation Permission
PromptUser
System prompts user location
UserLocationDenied
Halt
LocationAccepted
latitudeInput()[degree]
longitudeInput()[degree]
GetUserLongitude
RetrieveUserLocation
GetUserLatitude
toRadians()
toRadians()
Convert User Coordinates to Radians
haversineFunction()
State Diagram: Haversine Function
Initial State
Farm 1
Compute Latitude Difference
computeLatitude()[non-negative]
Compute Longitude Difference
computeLongitude()[non-negative]
Compute Haversine Formula
computeHaversine()
Compute central angle θ
computeCentralAngle()
Compute distance between user coordinates /farm coordinates
returnDistance()[non-negative]
is_valid_coordinates()[false]
Cancel Operation
Farm N
Compute Latitude Difference
computeLatitude()[non-negative]
Compute Longitude Difference
computeLongitude()[non-negative]
Compute Haversine Formula
computeHaversine()
Compute central angle θ
computeCentralAngle()
Compute distance between user coordinates /farm coordinates
returnDistance()[non-negative]
is_valid_coordinates()[false]
Cancel Operation
Distance sent to Geolocation Services Class

Software Architecture Style, Modelling and Evaluation

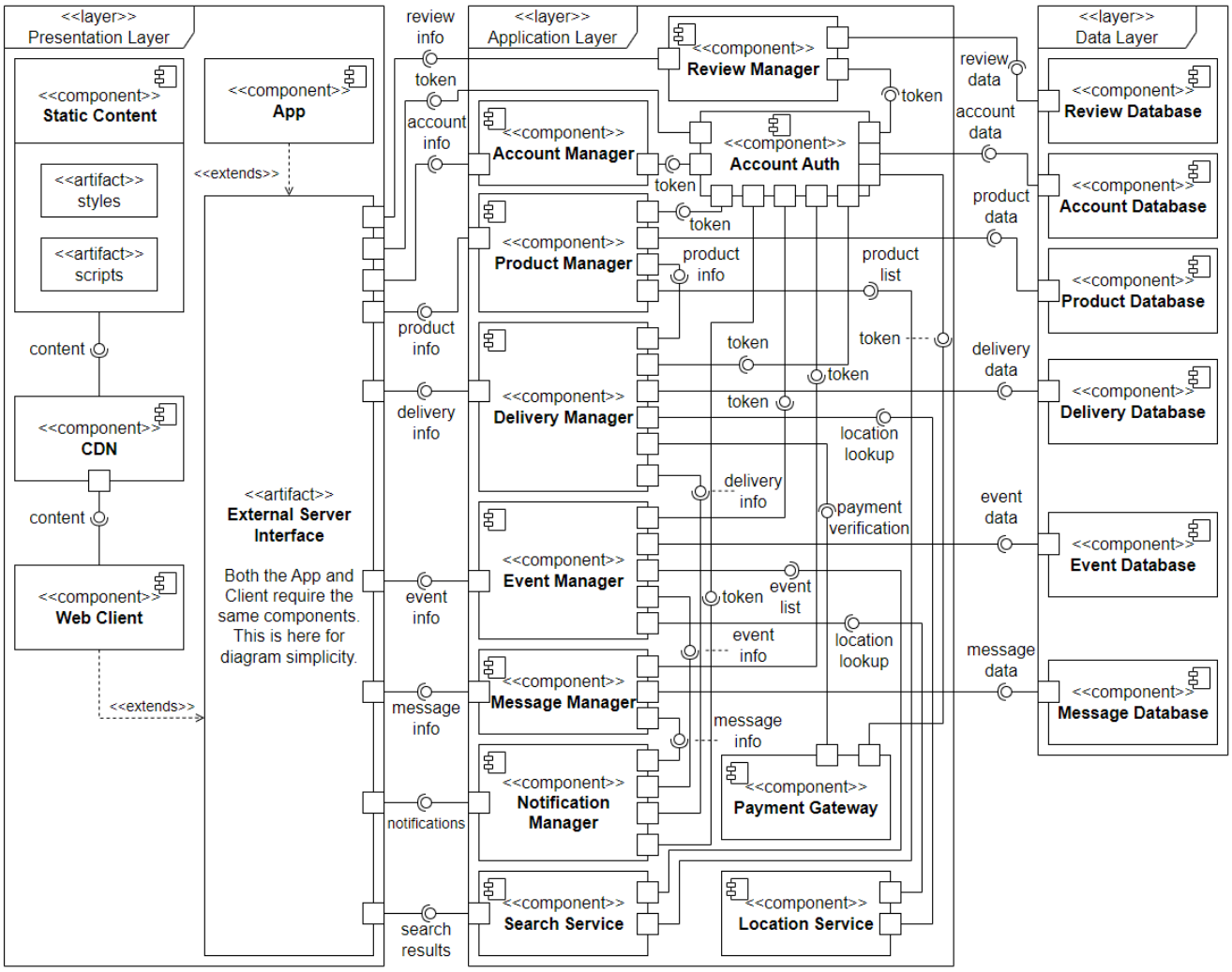
C1: Candidate Architecture Component Diagrams

- Microservices Architecture



The component diagram for the microservice architecture design of our system detailing the interconnectivity of all the components within, serving the users via mobile app or the web client. The system will communicate information via a REST API and pass this information through a main API Gateway server which routes calls to their respective services which may or may not then request further information from their respective databases. The identity provider in the image serves as authentication for the user allowing calls to be forwarded as necessary.

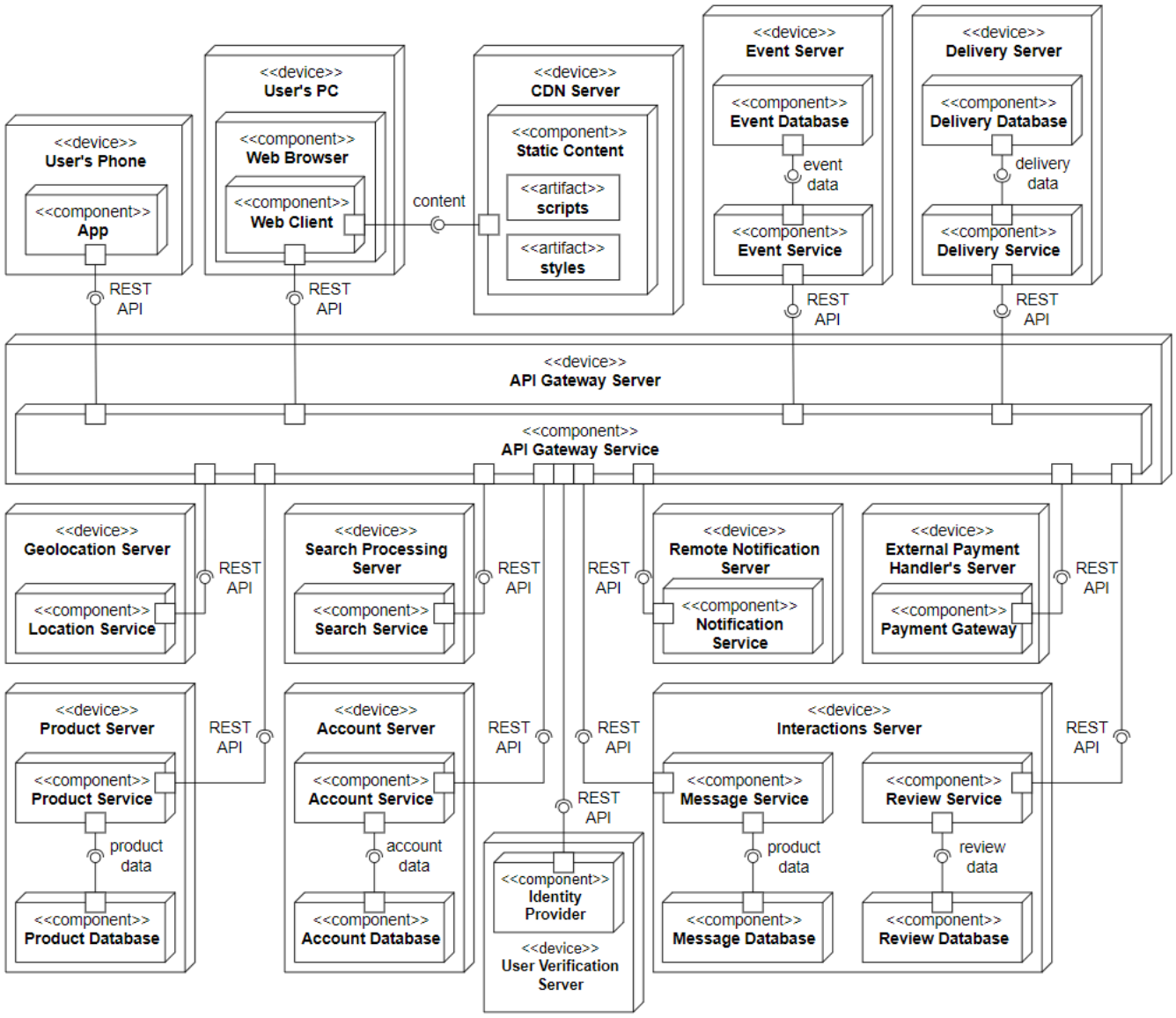
- Three-Tier Architecture



The component diagram for the three-tier architecture design of our system detailing the interconnectivity of the components within. For simplicity, as all connections to the web client and app would be equivalent, they have been generalised and merged into an "External Server Interface". The client would receive information from the respective managers which would, in most cases, require authorisation via the account authenticator which provides a token once the user logs in. The managers' role is to process information, which is usually provided by the respective databases, and provide it to the user.

C2: Candidate Architecture Deployment Diagrams

- Microservices Architecture



The deployment diagram for the microservice architecture design of our system. The interconnectivity of the systems is identical to that of the respective component diagram, but the deployment diagram features the hardware upon which the various components will reside. In this design, the static webpage content is assumed to be uploaded directly to a CDN server which would then detect the changes and propagate them to the other CDN servers rather than uploading changes to a single main static content server.

The user would initially log in to the website which would be verified by the identity provider permitting the user's web client or app to make calls to all the other services

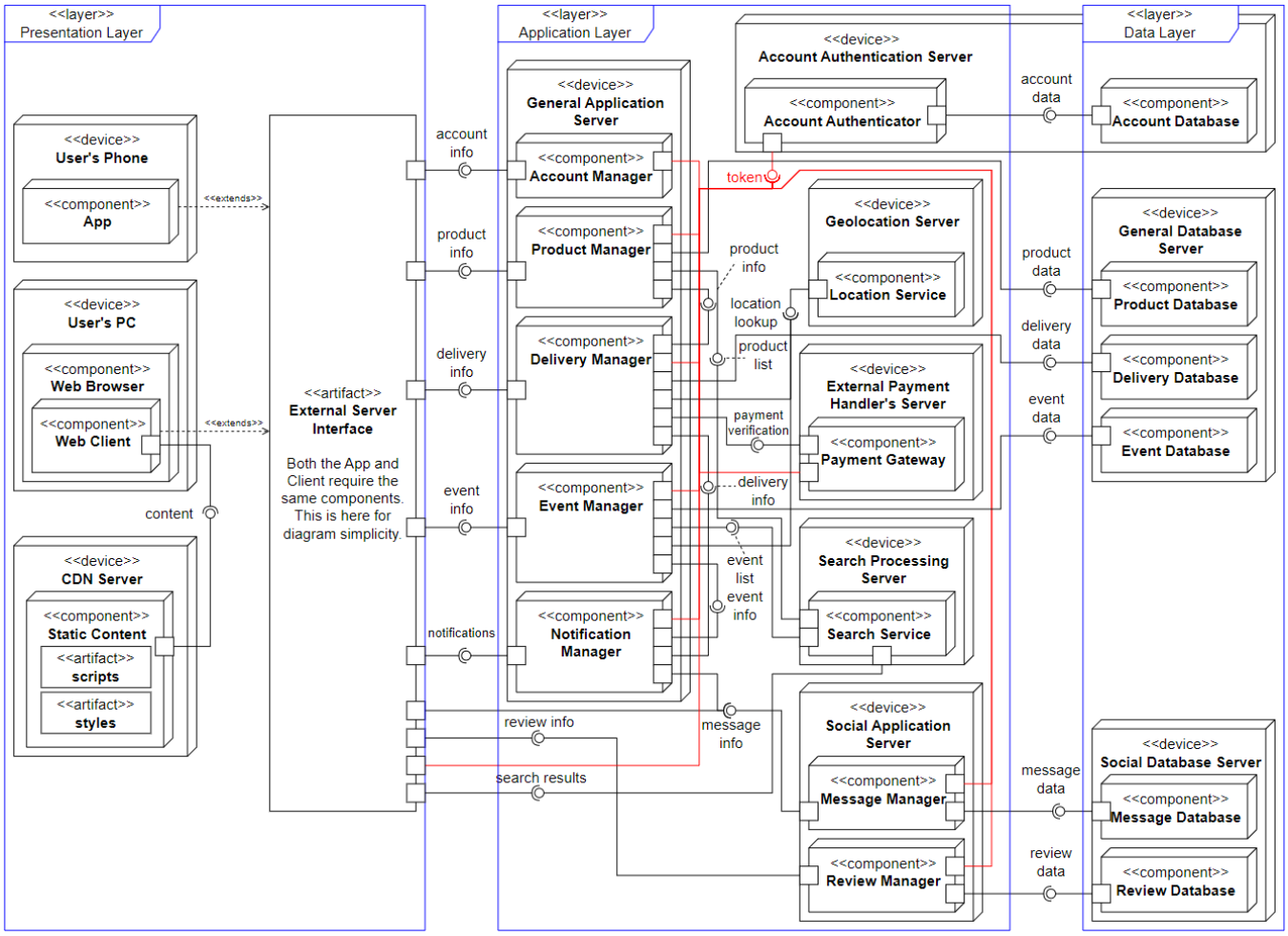
available via the main API gateway which would route these calls, this would be a very important part of the system with a very large amount of throughput which should be optimised as best as possible both in software and providing dedicated hardware for it to run on.

Some services do not require databases, these are the notification service, location service and the search service as these all operate in conjunction with other services which either already store the necessary data or where a separate database of our own wouldn't be necessary – this pertains to the location service and payment gateway since these would both be provided by a third party.

The account service's role is both to provide the correct credentials for the identity provider to authenticate a user and to keep track and allow the management of all the general information around a user's account – things that would appear on their profile as well as their settings and preferences.

Most services are separated into their own physical servers for the sake of easier maintainability and to allow optimisation of hardware for each service except for the message and review services as these both operate on almost identical types of data – text with timestamps differing only by the addition of ratings for the review service, this allows the databases and both services to still be easily optimised by virtue of their similarity.

- Three Tier Architecture



The deployment diagram for the three-tier architecture design of our system. The interconnectivity of the system is, again, identical to that of the respective component diagram with the addition being the layout of the components with respect to the hardware they will reside in. Minor colour coding is used for diagram clarity; layers are blue and any connections that are passing the authentication token are red.

Like with the microservice architecture, the user would try and log on through their web client or app which would be verified by the account authenticator which has exclusive access to the account database (including credentials). The account authenticator would reside on a separate account authentication server along with the account database for security purposes, all servers should be secure and monitored but this is especially the case with the user authentication server and this means that the account database does not need to at any point communicate over the internet as all communication can be done locally within the server between the authenticator and account database.

In the same way the microservice architecture has an identity provider, the three-tier architecture would pass around a token generated for a session when a user logs in that allows their client to access all the necessary services.

Unlike the microservice architecture, all the databases are on separate physical servers to their respective services by nature of the three-tier design, this provides benefits to both security and an alternative form of hardware optimisation to that described in the microservices section – the database servers only need to be designed to store large amounts of data and be able to find/serve it fast with a far lesser regard for processing speed as that is the role of the managers, and their hardware will, in turn, be optimised for such.

The location service and payment gateway run on their own separate servers as they would be provided by third parties. The search service runs on a separate server as this is a specific potentially processing heavy task that would greatly benefit from a separately optimised server for the sake of search query processing.

As with the microservice architecture design, the message and review components are grouped into their respective application (for the services) and database servers categorised as ‘social’. In this case, all the other services supported by a database are also all hosted on a general application server and database server for the respective components.

C3: Architecture Evaluation

In a microservices architecture, the system is broken down into smaller, independent services that communicate via APIs. This promotes flexibility, scalability, and resilience. Each service can be developed, deployed, and scaled independently, allowing for faster innovation and easier maintenance. However, the complexity of managing multiple services, ensuring their communication, and handling eventual consistency between services can be a challenge.

The use of a main API Gateway server in microservices simplifies client communication, and the identity provider allows for secure authentication, ensuring controlled access. This architecture facilitates quicker development cycles, promotes team autonomy, and allows for the use of diverse technologies within different services.

The three-tier architecture separates the system into presentation, application, and data tiers. This architecture simplifies development and maintenance, as components are organized into clear layers. It's easier to understand and typically involves less complexity compared to microservices. However, scaling can be more challenging as all components within a tier may need to scale together, leading to potential resource inefficiency.

In this architecture, the account authenticator handles user authentication and authorization, providing tokens for secure access. The managers process information from databases and deliver it to clients. It's a straightforward approach but might become a bottleneck as the system grows due to its monolithic nature.

Microservices offer flexibility, scalability, and resilience but come with increased complexity in managing distributed systems and handling inter-service communication. Three-tier architecture simplifies development but might face challenges in scaling and can become a single point of failure due to its monolithic structure.

Given the system's nature and requirements mentioned, the microservices architecture is the preferred option. Its ability to scale independently, accommodate varied technologies, and handle rapid changes aligns with the modern demands of system design. Despite its complexities, the microservices architecture is more adaptable to evolving needs and promotes better resource utilization compared to the three-tier architecture. It provides greater scalability, as well as easier fault isolation and maintenance due to the fact that all services are separated from each other. The use of an API Gateway and identity provider in microservices ensures secure and controlled access, vital for modern applications.

In conclusion, while both architectures have their merits, the flexibility, scalability, maintainability and adaptability of microservices make it the more suitable choice.

D: Software Testing

Requirement	**Objectives**	**Test Type**	**Exit Criteria** / Expected Result
Local Farmer Profiles (Functional)	**Account Creation Testing:** • Verify that the registration system allows farmers to create farmer profile accounts. **User Interface Form Testing:** • Confirm that the system displays a user-friendly interface form for farmers to provide registration information. **Responsive Design Testing:** • Ensure that the registration interface adapts to different screen sizes and devices, including tablets, smartphones, and desktops. **Description Section Testing:** • Test the system's ability to provide farmers with a section to describe their farming practices and commitment to sustainability. **WYSIWYG Editor Functionality Testing:** • Validate that the system implements a WYSIWYG editor that allows farmers to compose their farm descriptions. **Farming Practices Display Testing:** • Confirm that farmers can display farming practices, such as soil conditions, crop rotation, and irrigation methods, through the WYSIWYG editor. **Media Embedding Testing:** • Verify that the system enables farmers to embed media, such as videos and images, in their descriptions to showcase their farming practices. **Update Functionality Testing:**	**Automated Testing:** • Employed for account creation testing, user interface form testing, responsive design testing, WYSIWYG editor functionality testing, farming practices display testing, media embedding testing, update functionality testing, certification display testing, and badge representation testing. **Manual Testing:** • Used for subjective assessments, such as the user-friendliness of the interface and the accuracy of sustainability certification display. **Usability Testing:** • External, unfamiliar users brought in to verify ease of creating an account as well as setting up profile details. As well as ease of uploading and verifying their certification.	**Account Creation Successful:** • Farmers can successfully create farmer profile accounts. **User-Friendly Interface Validated:** • The system displays a user-friendly interface form for farmers. **Responsive Design Implemented:** • The registration interface adapts to different screen sizes and devices. **Description Section Functional:** • Farmers have a section to describe their farming practices and commitment to sustainability. **WYSIWYG Editor Works as Intended:** • The system's WYSIWYG editor allows farmers to compose farm descriptions. **Farming Practices Displayed Correctly:** • Farmers can display farming practices, including soil conditions, crop rotation, and irrigation methods. **Media Embedding Functions Properly:** • Farmers can embed media (videos and images) in their descriptions to showcase farming practices. **Update Functionality Verified:** • Farmers can update their farming practices, farm descriptions, and sustainability certifications. **Certification Displayed on Profiles:** • The system allows farmers to display sustainability certifications on their profiles. **Badge Represents Certification:** • An official badge representing a sustainability

	• Ensure that farmers can update their farming practices, farm descriptions, and sustainability certifications. **Certification Display Testing:** • Confirm that the system allows farmers to display sustainability certifications on their profiles. **Badge Representation Testing:** • Validate that an official badge representing a sustainability certification is displayed in the farm's account.		certification is displayed in the farm's account. **Account Management Usability Verified:** • New user, from the homepage, should be able to create an account with a provided email address and their own password and verify the email within 1m 30s (time logging into/browsing email inbox is excluded) • New user, already logged in to a farmer account with no additional profile information, should be able to accurately fill out all pre-provided details within 2 minutes. • New user, already logged into their farmer account and a scanner and/or modern phone provided, should be able to upload 3 provided **physical paper** documents of certification to the system of approvable quality (they will be checked instantly for the sake of this test) within 10 minutes.
Product Listings (Functional)	**Product Listing Exploration Testing:** • Verify that customers can explore a wide range of products, including fruits, vegetables, poultry, and plants. **Product Label Setting Testing:** • Confirm that farmers can set labels for their products, including "Organic," "Seasonal," "Farm Fresh," "Free-range," "Native Species," and "Non-GMO." **Product Rating Display Testing:** • Ensure that the system displays the rating of each product accurately. **Visibility Preference Setting Testing:** • Verify that farmers can set the visibility preference for a product, including public	**Automated Testing:** • Employed for product label setting testing, product rating display testing, visibility preference setting testing, product availability testing, user search testing, sorting options testing, and currency verification testing. **Manual Testing:** • Used for product listing exploration testing, product filter application testing, and product image upload testing. **Usability Testing:** • External, unfamiliar users brought in to verify ease of setting up product listings accurately and with any required details as well as to	**Successful Product Exploration:** • Customers can explore a wide range of products, including fruits, vegetables, poultry, and plants. **Product Labels Set Correctly:** • Farmers can set labels for their products, and the system accurately displays these labels. **Accurate Product Rating Display:** • The system correctly displays the rating of each product. **Visibility Preferences Set Successfully:** • Farmers can set visibility preferences for products, and the system accurately reflects these preferences.

	visibility, member-only visibility, and farmer visibility. **Product Availability Testing:** • Confirm that farmers can add and remove a product, indicating its availability for sale. **User Search Testing:** • Validate that users can search for a product without being logged in. **Product Filter Application Testing:** • Test the system's ability to narrow down a search by applying specific product filters, such as "Organic," "Free-range," etc. **Sorting Options Testing:** • Confirm that the system allows users to sort product listings by price, rating, or relevance. **Currency Verification Testing:** • Ensure that products listed are in the sterling pound currency. **Product Image Upload Testing:** • Confirm that farmers can successfully upload a picture of a product.	verify ease of browsing and filtering of products.	**Product Availability Functions as Expected:** • Farmers can add and remove products, indicating their availability for sale. **User Search Operational:** • Users can successfully search for products without being logged in. **Effective Product Filtering:** • The system accurately narrows down searches based on specific product filters. **Sorting Options Work Correctly:** • Users can sort product listings by price, rating, or relevance. **Currency in Sterling Pound Verified:** • Products listed are in the sterling pound currency. **Product Image Upload Successful:** • Farmers can upload pictures of products successfully. **Listing/Browsing Usability Verified:** • New user, logged in as farmer on their product page, should be able to accurately list 2 pre-prepared product listings (with labels and varied visibilities) within 2 minutes. • New user, on product page, should be able to find 3 specific given product listings within 1m30s as well as demonstrate how to browse with specific label filters within 15s of being asked to do so and then how to sort by different product properties within a further 15s.
GeoLocation Service (Functional)	**Geographic Data Coverage Testing:** • Verify that the system is capable of handling geographic data across the entire United Kingdom. • Ensure that geographic data covers both rural and urban areas.	**Manual Testing:** • Used for interactable map interface testing, user location sharing testing, and location history tracking testing. **Automated Testing:**	**Geographic Data Coverage Confirmed:** • The system effectively handles geographic data across the entire United Kingdom, covering both rural and urban areas.

	Interactable Map Interface Testing: • Test the system's map interface to ensure users can interactively explore agricultural resources and view farm locations. **User Location Sharing Testing:** • Confirm that the system allows users to share their precise geographical location coordinates (longitude and latitude). **Location Notification Testing:** • Validate that the system sends a notification to users to obtain their location. • Confirm that users can respond to the notification to share their location. **Background Service Testing:** • Test the system's ability to run in the background and provide continuous location updates. **Proximity Search Algorithm Testing:** • Verify the implementation of proximity search algorithms to calculate the closest farm given a user location coordinate. **Haversine Formula Implementation Testing:** • Test the system's use of the Haversine formula to determine the shortest distance between a user's current location and a farm. **Location History Tracking Testing:** • Ensure that the system accurately tracks and stores a user's location history.	• Employed for geographic data coverage testing, location notification testing, background service testing, proximity search algorithm testing, and Haversine formula implementation testing.	**Interactable Map Interface Functioning:** • Users can interactively explore agricultural resources and view farm locations using the map interface. **User Location Sharing Successful:** • Users can share their precise geographical location coordinates. **Location Notification Process Verified:** • The system successfully sends notifications to users to obtain their location, and users can respond accordingly. **Background Service Operational:** • The system runs in the background and provides continuous location updates. **Proximity Search Algorithms Working:** • Proximity search algorithms accurately calculate the closest farm given a user location coordinate. **Haversine Formula Implemented Correctly:** • The Haversine formula is correctly implemented to determine the shortest distance between a user's current location and a farm. **Location History Tracking Implemented:** • The system accurately tracks and stores a user's location history.
Feedback and Ratings (Functional)	**Integration Testing with Customer Accounts:** • Verify that feedback and ratings are integrated with customer accounts, allowing users to track their contributions accurately. **User Authentication Testing:** • Confirm that the system ensures customers are	**Automated Testing:** • Employed for integration testing, user interface testing, and real-time update testing. **Manual Testing:** • Used for user authentication testing, farmer rating system testing, anonymous feedback testing, and feedback and rating options testing.	**Integration with Customer Accounts Successful:** • Feedback and ratings are accurately integrated with customer accounts. **Interface Usability Confirmed:** • New user, logged in to buyer account on farmer profile, should be able to leave a pre prepared farmer review

	logged in before allowing them to leave a rating. **Farmer Rating System Testing:** • Validate that the system allows customers to rate farmers based on the quality of the product, freshness, and delivery speed. **Anonymous Feedback Testing:** • Test the system's ability to allow customers to submit ratings and feedback anonymously. **Feedback and Rating Options Testing:** • Verify that customers have the option to provide feedback along with ratings. • Confirm the system enforces a reasonable character limit for written feedback. **Multilingual Support Testing:** • Ensure the feedback system supports input in multiple languages. **Image Attachment Testing:** • Test the system's ability to allow customers to attach images of products for more detailed feedback. **Real-time Update Testing:** • Validate that feedback and ratings are updated in real-time on product listings. **Spam and Inappropriate Content Filtering:** • Confirm that the system filters spam or inappropriate feedback before it becomes publicly visible.	**Security Testing:** • Performed to identify and address potential vulnerabilities, especially in the filtering of spam or inappropriate content. **User Interface Usability Testing:** • External, unfamiliar users brought in to verify the system provides a user-friendly interface for consumers to easily provide feedback and ratings for products, anonymous or otherwise.	accurately within 1 minute and prior to submission, should be able to explain how to make it anonymous within 15 seconds. **User Authentication Enforced:** • Customers are required to be logged in before leaving a rating. **Farmer Rating System Functioning:** • Customers can successfully rate farmers based on product quality, freshness, and delivery speed. **Anonymous Feedback Option Implemented:** • Customers can submit ratings and feedback anonymously if the system supports this feature. **Feedback Options and Limits Enforced:** • Customers can provide feedback along with ratings, and the system enforces a reasonable character limit for written feedback. **Multilingual Support Verified:** • The feedback system supports input in multiple languages. **Image Attachment Functionality Confirmed:** • Customers can attach images of products for more detailed feedback. **Real-time Update Successful:** • Feedback and ratings are updated in real-time on product listings. **Spam and Inappropriate Content Filter Implemented:** • The system effectively filters spam or inappropriate feedback before it becomes publicly visible.

Analytics Management	**Data Storage Testing:** • Verify that the system accurately stores details about a user's device, including operating system, device type, and browser version. • Confirm the correct collection and storage of cookies from web browsers. **User Activity Tracking Testing:** • Ensure that the system successfully stores login details, item search history, items in the shopping cart, number of pages visited, and language preferences. • Validate the accuracy of the recorded timestamps for user interactions with product listings. **Product Interaction Testing:** • Verify that the system accurately monitors the number of clicks on each product listing. • Confirm the system's ability to measure the time a user spends viewing a product listing. • Test the tracking of time spent on each individual product. **User Interaction Analytics Testing:** • Ensure the system monitors user scrolling within pages. • Confirm the identification of specific areas and products that users tend to scroll. **Statistical Model Testing:** • Validate the incorporation of statistical models to predict the future price of products based on user demand. **Performance Testing:** • Test the system's performance under various loads to ensure it can handle a significant number of user interactions. **Security Testing:**	**Automated Testing:** • Employed for data storage validation, user activity tracking, product interaction, and statistical model testing. **Manual Testing:** • Used for user interaction analytics testing, especially to validate user scrolling behaviours. **Performance Testing:** • Conducted to evaluate the system's responsiveness and scalability. **Security Testing:** • Performed to identify and address potential vulnerabilities in the storage and handling of user data.	**Data Accuracy Confirmed:** • All stored user device details, browser information, and user activity data are accurate. **User Interaction Features Function as Expected:** • User interactions, such as clicks on product listings and time spent on products, function as intended. **Statistical Models Perform Well:** • Statistical models accurately predict future product prices based on user demand. **Performance Metrics Met:** • The system meets predefined performance criteria under various loads. **Security Measures Implemented:** • All identified security vulnerabilities are addressed, and user data is securely stored.

	• Ensure that sensitive user data, such as login details and personal preferences, is securely stored and protected.		
Environmental Impact Tracking	**Functionality Testing:** • Verify that the system accurately displays heat maps for carbon emissions. • Ensure that the colours used in the heat maps appropriately represent the intensity of emissions. • Validate the real-time streaming of carbon emission data from connected sensors. • Confirm that the heat map updates dynamically based on the changing intensity of emissions. **Integration Testing:** • Test the integration of water usage sensors to ensure accurate and reliable data retrieval. • Validate the calculation of water usage by crop or field based on integrated sensors. • Verify the integration of real-time weather data, including temperature, humidity, and precipitation. **Waste Management Testing:** • Confirm that farmers can successfully categorize waste into types such as "organic," "recyclable," "non-organic," and "hazardous." • Validate the system's ability to display categorized waste data to users. **Performance Testing:** • Test the system's response time when updating heat maps and displaying real-time sensor data.	**Manual Testing:** • Used for usability testing and to ensure the intuitive categorization of waste. **Automated Testing:** • Employed for functionality, integration, and performance testing to ensure efficiency and repeatability. **Security Testing:** • Conducted to identify and address potential vulnerabilities in the system. **Usability Testing:** • External, unfamiliar users brought in to verify that the system provides a user-friendly interface for farmers to interact with environmental data related to their water consumption and carbon emissions. • Verify that the categorization of waste is intuitive for users to set and view.	**All Critical Functionality Passes:** • Heat maps display accurately. • Real-time sensor data is integrated successfully. • Water usage calculations are accurate. • Waste categorization and display functions as expected. **Performance Metrics Met:** • Response times for critical functions meet predefined performance criteria. **Security Vulnerabilities Addressed:** • All identified security vulnerabilities are addressed and resolved. **Usability Verified:** • New user, logged in as farmer on the environmental tracking page, should be able to accurately give the total and average per hour consumption of water and emission rate of carbon for 3 different days in the past week as well as the consumption of both over the last 4 hours of the current day within 3 minutes. • New user, from the environmental tracking page, should be able to name 3 locations each for areas of high emissions, moderate emissions and low emissions of carbon and should also be able to give an estimation of the emission rate at each location with up to a 10% error from the actual value for each of these. All this

	• Verify the system's ability to handle concurrent requests from multiple farms. **Security Testing:** • Ensure that the system has adequate security measures to protect environmental data and farmer information. • Validate access controls to ensure that only authorized individuals can view and modify data.		should be done within 3 minutes. • New user, logged in from waste tracking page, should be able to categorize 4 provided pieces of waste into each of the 4 categories within 1 minute and give the categorization of 4 already categorized pieces of waste within a further 1 minute. • User feedback from usability testing is positive, and any identified issues are addressed.
Seasonal Availability	**Seasonal Availability Configuration:** • Verify that farmers can successfully specify the seasonal availability of each product listing. • Confirm the customization options for availability labels, such as "In Season" and "Out of Season." **Historical Availability Preview:** • Ensure that farmers can preview historical seasonal availability information for each product. • Validate the accuracy of historical data displayed. **Dashboard Functionality:** • Confirm that the system provides farmers with a detailed dashboard showing availability history over different seasons and years. • Verify the correct display of specific product availability examples, like "Organic Strawberries." **Notification System Testing:** • Validate that the system notifies farmers to review and update seasonal availability before the start of each season. • Confirm the automatic setting of review reminders for each product listing. **Communication Preferences Testing:**	**Manual Testing:** • Used for seasonal availability configuration, historical availability preview, and dashboard functionality. **Automated Testing:** • Employed for notification system testing and communication preferences testing. **Integration Testing:** • Conducted to verify the integration between the seasonal availability module and other system components. **Usability Testing:** • External, unfamiliar individuals brought in to analyse ease of use and understanding for setting the seasonal availability of a product, being able to see and discern products which are in and out of season as well as view and explain the historical seasonal availability of a product via the dashboard.	**Seasonal Availability Configuration Successful:** • Farmers can accurately specify the seasonal availability of each product listing. **Historical Data Accuracy Confirmed:** • Historical seasonal availability information is correctly displayed and matches expected values. **Dashboard Displays Accurate Data:** • The dashboard shows availability history over different seasons and years, including specific product examples. **Notification System Functions as Intended:** • Farmers receive timely notifications to review and update seasonal availability. • Review reminders are automatically set for each product listing. **Communication Preferences Implemented:** • Farmers can successfully set their preferred communication channel for seasonal availability notifications. • Notifications are sent through the chosen communication channel. **Integration Successful:**

	• Verify that farmers can set their preferred communication channel (email or SMS) to receive seasonal availability notifications. • Confirm that notifications are sent through the chosen communication channel. **Integration Testing:** • Test the integration between the seasonal availability module and other system components to ensure seamless functionality.		• The seasonal availability module is seamlessly integrated with other system components. **Interface Intuitive and Understandable:** • New user, already logged in on their product listings page, can set the seasonal availability of three of their products within 1m 30s. • New user, from the products page, should be able to accurately name the seasonality of 10 products that located on the first page within 2 minutes and accurately describe the historical seasonality of 2 within 1m – 1m30s (depending on length of history).
Security	**Role-Based Access Control (RBAC) Testing:** • Verify that the system implements Role-Based Access Control and defines roles such as "Admin," "User," "Manager," etc. **Geographical Data Encryption Testing:** • Confirm that the system encrypts geographical data of users using industry-standard encryption algorithms. **Stream Cipher Application Testing:** • Ensure that a stream cipher (ChaCha20) is applied to the user's stored device location (longitude/latitude). **User Data Encryption Testing:** • Verify that all collected user data, including login details and item search history, is encrypted. **Different Encryption Keys Usage Testing:** • Confirm that the system uses different encryption keys for different types of data.	**Automated Testing:** • Employed for role-based access control testing, geographical data encryption testing, stream cipher application testing, user data encryption testing, different encryption keys usage testing, encryption key rotation testing, TLS client authentication testing, server configuration testing, password policy enforcement testing and hashing algorithm configuration testing. **Manual Testing:** • Used for special character requirement testing and salt usage testing.	**Successful RBAC Implementation:** • The system effectively implements Role-Based Access Control with defined roles. **Geographical Data Encryption Confirmed:** • User geographical data is encrypted using industry-standard encryption algorithms. **Stream Cipher Successfully Applied:** • A stream cipher (ChaCha20) is successfully applied to the user's stored device location. **User Data Encryption Verified:** • All collected user data, including login details and item search history, is encrypted. **Different Encryption Keys in Use:** • The system uses different encryption keys for different types of data. **Encryption Key Rotation Successful:** • Encryption keys are rotated every 17 weeks for added security.

	Encryption Key Rotation Testing: • Validate that encryption keys are rotated every 17 weeks for added security. **TLS Client Authentication Testing:** • Confirm that farmers have client certificates in their web browsers for TLS client authentication. **Server Configuration Testing:** • Verify that the server hosting farmer profiles is configured to request client authentication during the handshake. **Password Policy Enforcement Testing:** • Ensure that the system enforces a strong password policy. **Special Character Requirement Testing:** • Confirm that passwords must have at least 12 characters, 3 digits, and a special character. **Hashing Algorithm Configuration Testing:** • Verify that the system uses Argon2id with a minimum configuration of 19 MiB of memory, an iteration count of 2, and 1 degree of parallelism. **Salt Usage Testing:** • Confirm that the system adds a unique salt to each password as part of the hashing process.		**TLS Client Authentication Operational:** • Farmers have client certificates, and the server is configured to request client authentication during the handshake. **Password Policy Enforcement Validated:** • The system enforces a strong password policy, including the special character requirement. **Hashing Algorithm Configuration Confirmed:** • The system uses Argon2id with the specified minimum configuration. **Salt Usage Verified:** • The system adds a unique salt to each password as part of the hashing process.
Availability	**Uptime Testing:** • Verify that the system achieves an uptime of 99.92% over a 90-day period, excluding planned maintenance. **Farm Server Availability Testing:** • Ensure that the farm server is operational and available for at least 95% of the time. **Concurrent User Handling Testing:**	**Automated Testing:** • Employed for uptime testing, concurrent user handling testing, incident identification and logging testing, performance compliance testing, location display testing, Haversine function computation testing, latitude and longitude accuracy testing, coordinate conversion	**Uptime Target Achieved:** • The system achieves an uptime of 99.92% over the 90-day period, excluding planned maintenance. **Farm Server Availability Validated:** • The farm server is operational and available for at least 95% of the time. **Concurrent User Handling Confirmed:**

	• Confirm that the farm server can handle a 25% increase in concurrent users during peak hours. **Incident Identification and Logging Testing:** • Validate that the farm server identifies and logs incidents within 15 minutes of their occurrence. **Performance Compliance Testing:** • Confirm that the farm server's performance complies with the level of service outlined in the Service Level Agreement (SLA). **Location Display Testing:** • Ensure that the system displays a farm's location within 2 seconds of receiving a request. **Haversine Function Computation Testing:** • Verify that the system computes the Haversine function with a maximum response time of 800 milliseconds. **Latitude and Longitude Accuracy Testing:** • Confirm that the system accurately captures a user's latitude and longitude coordinates with an error rate of less than 1.5%. **Coordinate Conversion Accuracy Testing:** • Validate that the system converts latitude and longitude coordinates from degrees to radians with an accuracy of at least 99.8%. **Location Query Handling Testing:** • Ensure that the system can handle a 45% increase in location queries during peak hours.	accuracy testing, and location query handling testing. **Manual Testing:** • Used for farm server availability testing.	• The farm server can handle a 25% increase in concurrent users during peak hours. **Incident Identification and Logging Successful:** • The farm server identifies and logs incidents within 15 minutes of their occurrence. **Performance Compliance Verified:** • The farm server's performance complies with the level of service outlined in the SLA. **Location Display Timeliness Validated:** • The system displays a farm's location within 2 seconds of receiving a request. **Haversine Function Response Time Confirmed:** • The system computes the Haversine function with a maximum response time of 800 milliseconds. **Latitude and Longitude Accuracy Verified:** • The system accurately captures a user's latitude and longitude coordinates with an error rate of less than 1.5%. **Coordinate Conversion Accuracy Validated:** • The system converts latitude and longitude coordinates from degrees to radians with an accuracy of at least 99.8%. **Location Query Handling Confirmed:** • The system can handle a 45% increase in location queries during peak hours.

Scalability	**Page Functionality Testing:** • Verify that the upcoming events page (for farmer's markets, etc.) functions as intended. **Notification System Testing:** • Confirm that the notification system for upcoming events and new listings works seamlessly. **Email and Push Notification Testing:** • Validate that email notifications and in-app push notifications are sent and received successfully and in order. **Store Page Functionality Testing:** • Ensure that the store page (for delivery, etc.) functions as intended. **Filter/Search System Testing:** • Test the filter and search system on the store page for accuracy and efficiency. **Data Analytics Functionality Testing:** • Verify the functionality of the data analytics module, including the ability to sell data to companies. **Individual Farmer Pages Testing:** • Confirm that each farm(er) has a dedicated page with relevant information. **Location-Based Search Testing:** • Validate the ability to find a local farmer based on the user's location. **Impact Tracking Testing:** • Test the system's ability to track the impact of each farmer's practices, including water usage and carbon footprint. **Account Management Testing:** • Confirm that users can change their email, contact number, and password. **Farmer Accreditation/Authentication Testing:**	**Automated Testing:** • Employed for notification system testing, email and push notification testing, filter/search system testing, data analytics functionality testing, account management testing, accreditation/authentication testing, impact tracking testing, review system testing, feedback and support system testing, and cross-platform testing. **Manual Testing:** • Used for page functionality testing, individual farmer pages testing, location-based search testing, and general messaging functionality testing.	**Upcoming Events Page Operational:** • The upcoming events page functions as intended. **Notification System Working:** • The notification system for upcoming events and new listings works seamlessly. **Email and Push Notifications Sent and Received:** • Email notifications and in-app push notifications are sent and received successfully and in order. **Store Page Functions Properly:** • The store page, including delivery options, functions as intended. **Filter/Search System Accurate and Efficient:** • The filter and search system on the store page is accurate and efficient. **Data Analytics Module Operational:** • The data analytics module, including the ability to sell data to companies, functions properly. **Individual Farmer Pages Displayed Correctly:** • Each farm(er) has a dedicated page with relevant information. **Location-Based Search Successful:** • Users can find a local farmer based on their location. **Impact Tracking Functionality Verified:** • The system accurately tracks the impact of each farmer's practices, including water usage and carbon footprint. **Account Management Features Confirmed:** • Users can successfully change their email, contact number, and password. **Accreditation/Authentication Process Validated:** • The accreditation/authentication process for farmers is successful.

	• Validate the accreditation/authentication process for farmers. **User Location Settings Testing:** • Test users' ability to set their county/borough in the account settings. **Review System Testing:** • Ensure the review system allows users to provide feedback on farms. **Feedback and Support System Testing:** • Confirm the functionality of the feedback and support system. **General Messaging Functionality Testing:** • Validate the general messaging system between users, farmers, and farming enthusiasts. **Read Receipts and Online Status Testing:** • Confirm that read receipts are accurate and display when users were last online. **Typing Indicators Testing:** • Test the system's ability to display typing indicators during messaging. **Cross-Platform Testing:** • Validate that the system is accessible and functions seamlessly across web pages and mobile apps.		**User Location Settings Implemented:** • Users can set their county/borough in the account settings. **Review System Works Properly:** • The review system allows users to provide feedback on farms. **Feedback and Support System Functional:** • The feedback and support system works as intended. **Messaging System Operational:** • The general messaging system between users, farmers, and farming enthusiasts functions properly. **Read Receipts and Online Status Accurate:** • Read receipts are accurate, and the system displays when users were last online. **Typing Indicators Function as Intended:** • The system displays typing indicators during messaging. **Cross-Platform Compatibility Verified:** • The system is accessible and functions seamlessly across web pages and mobile apps.

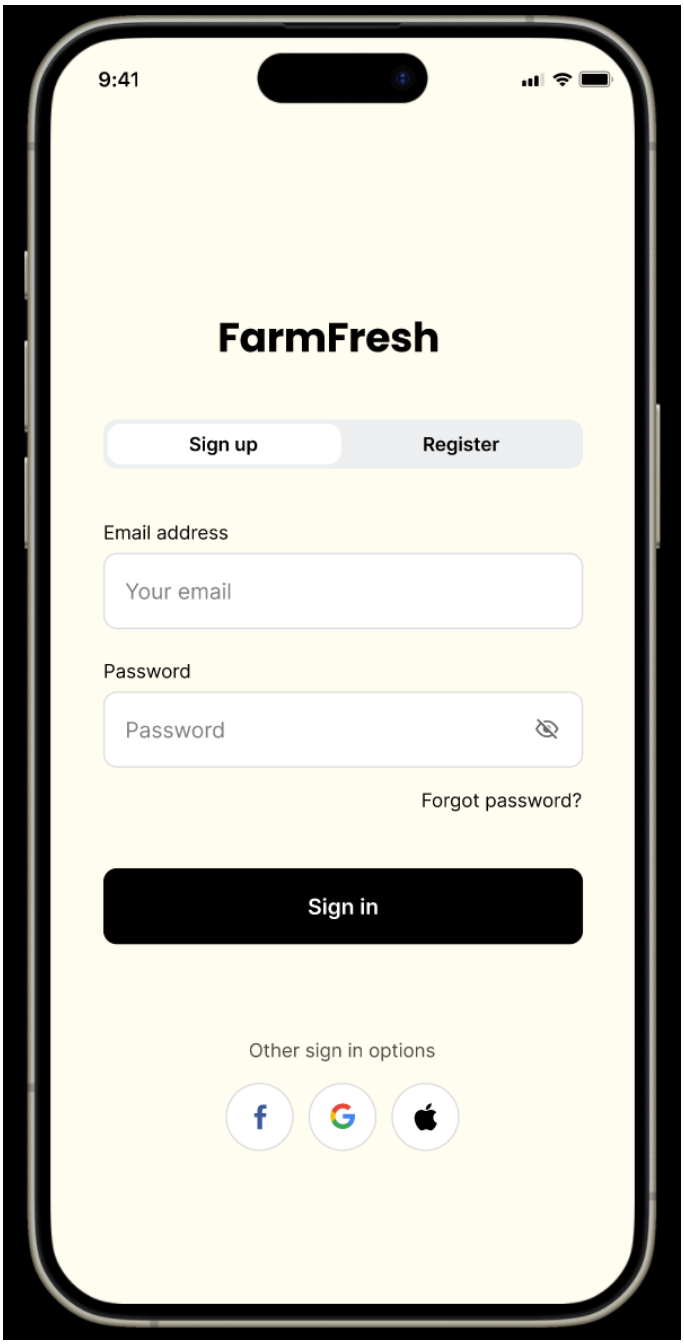
E: Usability and Prototyping:

Interactive Figma Prototype:

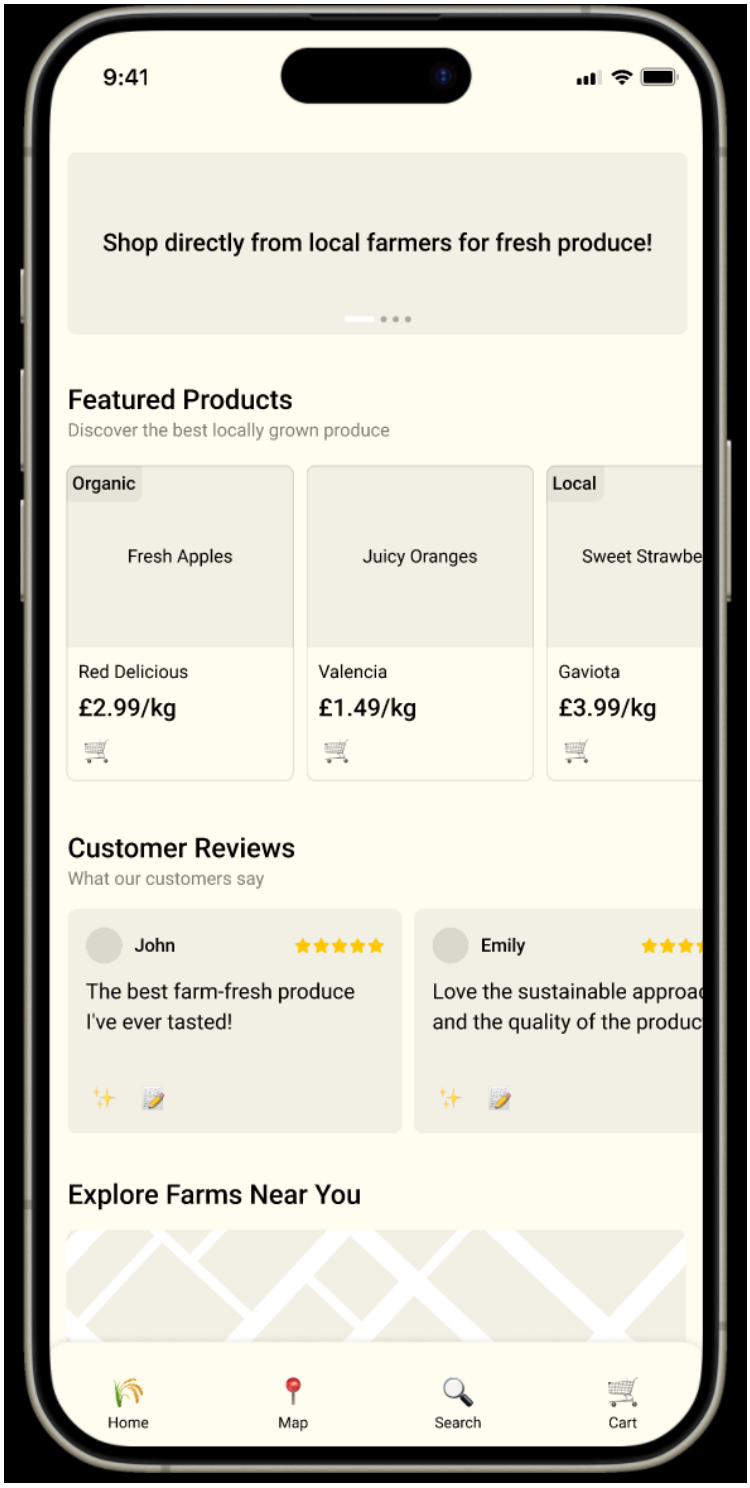
LINK

Splashscreen

Sign In/Up Page

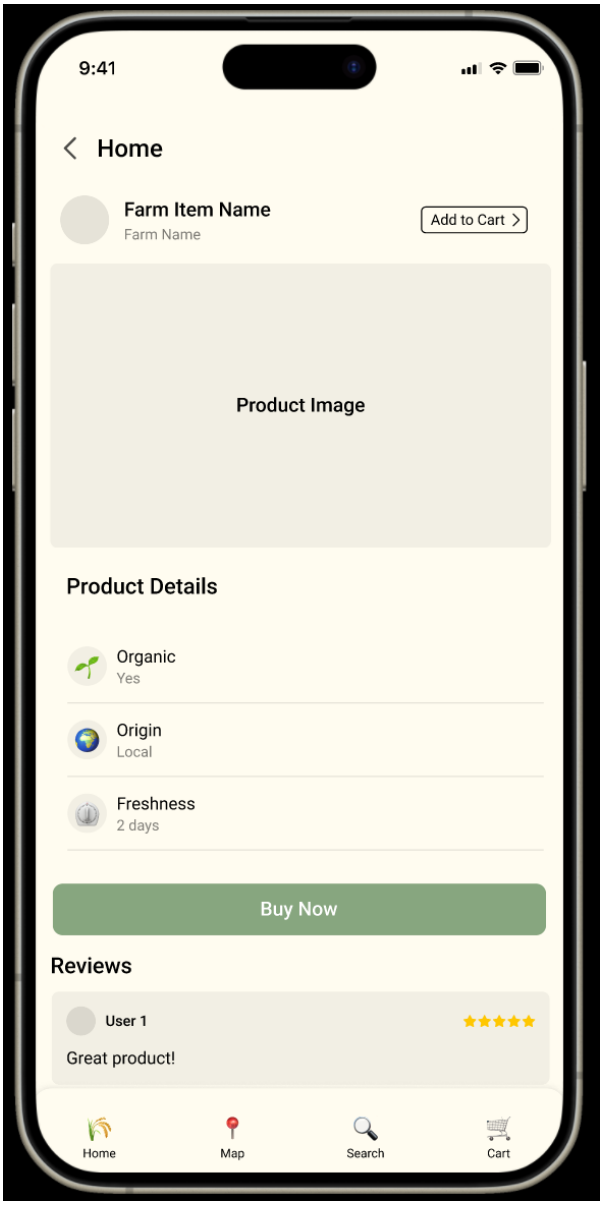


Home Page

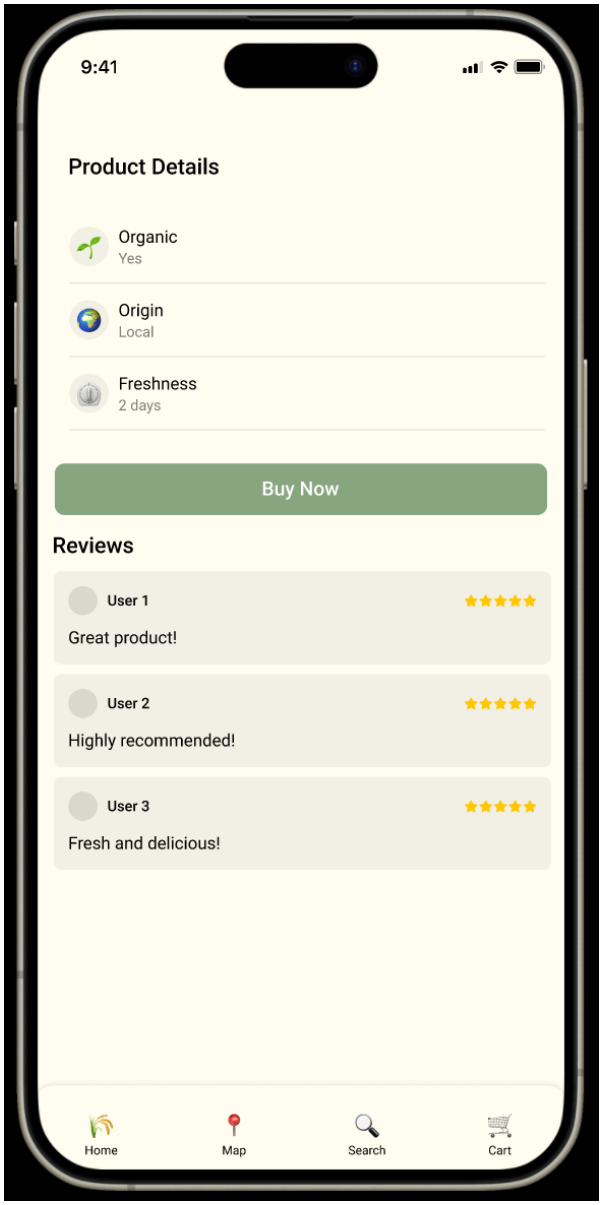

9:41
Shop directly from local farmers for fresh produce!
Featured Products
Discover the best locally grown produce
Organic
Fresh Apples
Red Delicious
£2.99/kg
Juicy Oranges
Valencia
£1.49/kg
Local
Sweet Strawbe
Gaviota
£3.99/kg
Customer Reviews
What our customers say
John
The best farm-fresh produce I've ever tasted!
Emily
Love the sustainable approac and the quality of the produc
Explore Farms Near You
Home
Map
Search
Cart

Product Page

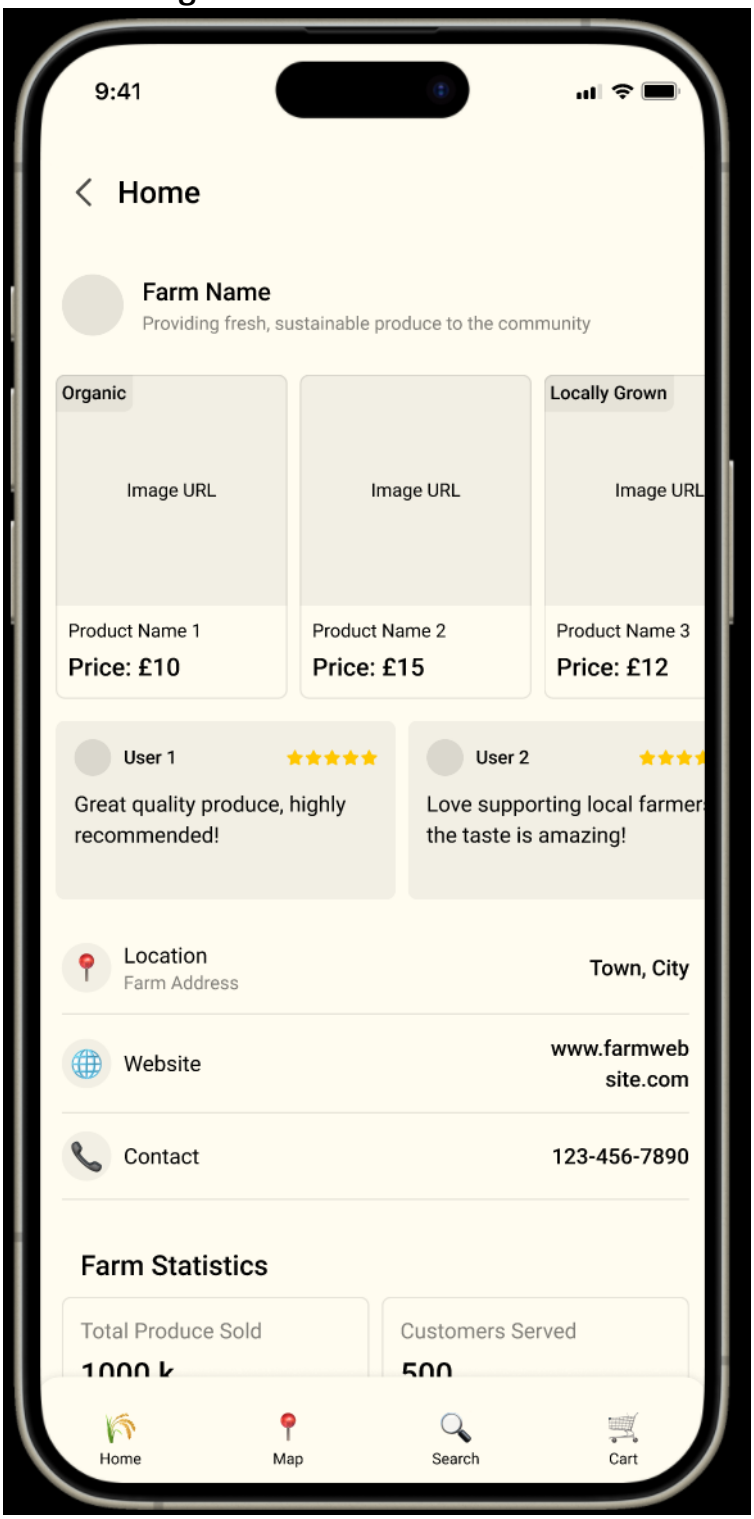
(Top)



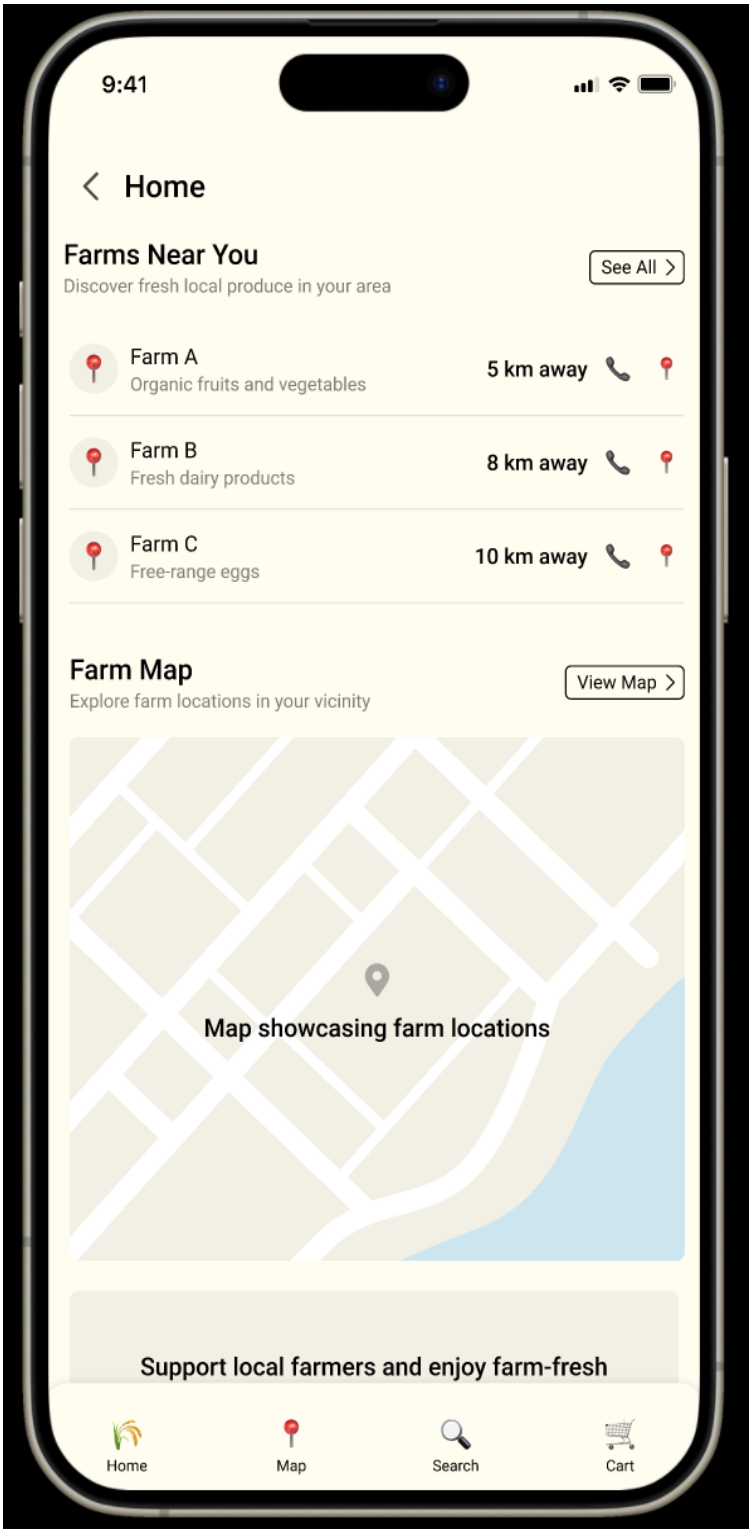
(Scrolled down)



Farm Page


9:41
Home
Farm Name
Providing fresh, sustainable produce to the community
Organic
Image URL
Product Name 1
Price: £10
Image URL
Product Name 2
Price: £15
Locally Grown
Image URL
Product Name 3
Price: £12
User 1
Great quality produce, highly recommended!
User 2
Love supporting local farmer
the taste is amazing!
Location
Farm Address
Town, City
Website
www.farmweb
site.com
Contact
123-456-7890
Farm Statistics
Total Produce Sold
1000 k
Customers Served
500
Home
Map
Search
Cart

Farm Finder Page



F: Ethics and Professional Practice

Firstly, the project acts consistently with the public interest. The project aims to connect consumers and farmers, making it easier for the public to purchase food, as well as promoting healthier eating as the majority of goods the farmers will be encouraged to sell are fresh local produce. Additionally, the project makes purchasing food less expensive due to the direct link between the farmer and the consumer, eliminating the middleman and reducing overall costs. The project also promotes sustainability as it encourages connecting consumers and farmers located in closer proximity, reducing food miles and the carbon footprint of the food consumers eat.

The project serves both farmers and consumers in their best interests, as it does not take commission from farmers - increasing their profits - and is honest to the consumers about the origins of the food they have ordered. The software also clearly communicates key information to both farmers and consumers using a notification system.

The project is fair and does not discriminate: any person can use the application to order food from farmers, the design of the application tries to maximise accessibility, any farmer can advertise and sell their products on the application, and the program does not favour any one farmer over another.

The project respects user's privacy and honours their confidentiality. The application must store personal data, for example user's name and location, therefore the application is transparent with users signing up about what personal data is being stored and how it is being used, and they must agree to a privacy policy statement. Only the personal data required to provide the service is collected, and the user can choose to obtain and amend inaccurate information. All personal data is deleted when a user deletes their account. The software also protects every user's personal data and only discloses it with the appropriate authorities; it does not sell data to tracking and advertising companies.

The project aims to produce a product that meets the highest professional standards possible: the project is developed using the best software engineering tools such as UML diagrams, communication has remained clear and consistent throughout the projects development, and all group members are aware of the social context in which their work is deployed so they can use their best judgement to make the project as ethical as possible. The group members also only complete work in areas in which they are professionally competent, decreasing the likeliness of low standards. Developers are also encouraged to peer review other people's work to improve it further. Collaboration is also encouraged, for example, multiple group members worked on the software testing plan as each person could work on an individual requirement.

The project respects all colleagues and tries to uphold the principles of professional practice. Everyone working on the project is given an equal amount of work which is

discussed during regular team meetings, and all ideas are listened to and everything possible is done to support a struggling team member.

Project Management and Moderation

Before starting the coursework, we met up as a team to brainstorm ideas. We wanted to initially generate different thoughts from all team members, so we went round the group to mention in turn our initial ideas. After this process, we came to a mutual consensus on the high-level plan and design of our project.

We met up again to initiate the initial progress on the sections of the project, where at first we focused on assigning tasks from the different sections to each team member. We let anyone mention whether there was a particular question they preferred to answer, designated these and then assigned the rest. We have noted the assigned tasks below; however, due to the nature of a team project, we wanted to work collaboratively and as such we ensured we gave help to each other on different tasks. Assigning tasks helped distribute the workload, but we ensured that we helped each other in equal amounts to create consistency and fairness of workload distribution. Therefore each section may not have strictly only been worked on by the assigned member, but as a group we can agree that overall the project was split in an even manner.

To ensure we met the deadline, we met up weekly in person to collaborate, discuss and evaluate our progress on the project. Additionally, we set up online methods of communication to discuss the project as a group when we were not meeting in person, as well as to have those who could not meet in person video call in remotely. We booked a feedback session which proved useful, as we were able to note subtle changes in each section to make, in order to ameliorate the overall quality of the coursework.

Section designation:

- Adithya Phanish – 2403447 – AXP1143@student.bham.ac.uk: Section B4, C3
- James Govey – 2399707 – JXG317@student.bham.ac.uk B3, Part of E1, Testing
- Roch Daraz – 2455357 – RKD@student.bham.ac.uk Section C1, C2, Usability Tests in D
- Ryan Ribero – 2402973 – RXR274@student.bham.ac.uk Section A2, B8
- Shayan Haghighi -2407477 – sxh1498@student.bham.ac.uk Section B5,B6,E2
- Tejavardhan Jeenu – 2410624 - TXJ224@student.bham.ac.uk Section B7, Section D
- Thomas Cartwright – 2428097 – txc297@student.bham.ac.uk Section B1, F
- Thomas Muir – 2441267 – txm267@student.bham.ac.uk Section B2, E