Software Design Specification for "Farm Direct"

Project Title:

Farm Direct – Online Platform for Canadian Farmers to Sell Directly to Customers

Team Members:

- Dev Pinakinbhai Patel
- Hetanshu Hemant Patel
- Kush Janak Patel
- Pranav Shrivastava

Table of Contents

1. Introduction

The purpose of this document is to provide a comprehensive Software Design Specification for the "Farm Direct" mobile application. Farm Direct is a digital platform designed to facilitate direct transactions between farmers and consumers in Canada, addressing challenges like limited market access, high distribution costs, and visibility issues faced by farmers. By enabling farmers to list and sell products directly, the app contributes to a sustainable and efficient marketplace for local produce.

2. System Overview

The "Farm Direct" platform offers a centralized mobile app where Canadian farmers can showcase their products to a broad consumer base. Key functionalities include:

- **User Registration and Role Selection**: Differentiated workflows for Farmers and Consumers.
- **Product Browsing and Search**: Product categories and advanced filtering for easy navigation.
- **Farmer Profiles and Product Listings**: Each farmer's profile features product availability, location, and pricing.
- **Product Details**: Detailed information about individual products, including rate, quantity, and delivery options.
- **Community and Engagement**: Features to support consumer engagement, such as product reviews and forums.

The app is built using Flutter for cross-platform compatibility, and it interfaces with a backend REST API for data management.

3. Design Requirements

3.1 Functional Requirements

- **User Registration and Authentication**:
    - Users must register with phone numbers, validated via OTP, and select a role (Farmer or Consumer).
    - Only authenticated users have access to specific app functionalities, like product listings and messaging.
- **Product Browsing and Category Filtering**:
    - Users can browse products under categories such as Vegetables, Fruits, Dairy, etc.

    - Search options allow users to filter results based on product type, location, or specific farmer characteristics.

- **Farmer Product Listings**:
    - Farmers can create profiles and list products including details like price per unit, quantity, date listed, and location.
- **Product Details and Contact Information**:
    - Each product detail screen displays comprehensive information, including the farmer's name, product details, and contact methods.
- **Secure Transactional Features** (Future Scope):
    - While secure payment features are not implemented, placeholders will be added in the current version to support future integration.

3.2 Non-Functional Requirements

- **Reliability**: The platform should be available 99% of the time and handle downtime gracefully.
- **Usability**: The UI should be intuitive, designed for both tech-savvy users and farmers with limited digital literacy.
- **Performance**: Key screens, like Product Listings, should load within 2 seconds.
- **Scalability**: The app should support expansion to additional categories, features, and larger datasets.

4. System Architecture

4.1 Architectural Pattern

The Farm Direct app follows a **Client-Server Architecture** using a REST API backend to manage communication between the client (mobile app) and server (data storage and processing).

4.2 Architectural Components

1. **Frontend (Flutter Mobile Application)**:
    - Implements user interface, navigation, and input validation.
    - Communicates with the backend via HTTP requests to the REST API.
2. **Backend API (Java with Spring Boot)**:
    - Handles user requests, business logic, and CRUD operations.
    - Implements endpoints for user registration, product listing, farmer details, and search functionalities.

3. **Database (MongoDB)**:
    - Stores structured data on users, farmers, and product listings.
    - Scales horizontally to handle increased user activity and data entries.

5. Module Descriptions

5.1 Home Screen (home_screen.dart)

- **Purpose**: The entry point for users to explore categories and navigate through the app.
- **Components**:
    - **Category Grid**: Displays available product categories as clickable icons (e.g., Fruits, Vegetables).
    - **Search Field**: Allows users to quickly search for product categories.
    - **Bottom Navigation**: Provides easy access to Home, Search, Orders, and Profile sections.

5.2 Farmer List Screen (farmer_list_screen.dart)

- **Purpose**: Displays a list of farmers offering a selected product.
- **Features**:
    - **Farmer Search**: Filters farmers by name or location.
    - **Sort Options**: Sorts farmers based on pricing (low to high, high to low).
    - **FarmerCard**: Summarizes each farmer's information, including name, location, image, and product rate, linking to ProductDetailScreen for detailed information.

5.3 Product Category Screen (product_page.dart)

- **Purpose**: Displays products for the selected category with options for searching and browsing.
- **Features**:
    - **Product Search**: Users can search for specific products within the category.
    - **Product Cards**: A grid layout displays available products; each card links to the FarmerListScreen.

5.4 Product Detail Screen (product_detail_screen.dart)

- **Purpose**: Provides detailed information about a selected product and the farmer offering it.
- **Features**:

- **Product Information**: Includes the product name, quantity, rate, and date listed.
- **Farmer Contact Details**: Displays the farmer's contact information, location, and delivery options.
- **Delivery Details**: Shows availability of delivery service.

5.5 Signup Screen (signup_screen.dart)

- **Purpose**: Manages user registration and OTP verification for new users.
- **Features**:
  - **OTP Verification**: Sends a verification code to the user's phone number.
  - **User Type Selection**: Allows the user to choose between Farmer and Consumer roles, setting specific permissions and UI flows for each.

6. User Interface Design

6.1 Design Principles

The UI design adheres to the following principles:

- **Simplicity**: A minimalistic design with clear navigation and a straightforward layout.
- **Accessibility**: Large fonts, high-contrast colors, and clear icons for users with limited tech experience.
- **Consistency**: A uniform color scheme and component styling to enhance usability.

6.2 Screen Breakdown

1. **Home Screen**: Displays product categories and search options.
2. **Product Category Screen**: Lists products within a selected category.
3. **Farmer List Screen**: Shows farmers selling a specific product, along with sorting and filtering options.
4. **Product Detail Screen**: Detailed view of a product's information, including contact options for the farmer.
5. **Signup Screen**: Includes OTP verification, user type selection, and phone/email fields for new users.

6.3 Color Scheme

- **Primary Color**: Green (#008000) - represents agriculture and sustainability.
- **Secondary Color**: White and light grey for background and contrast.
- **Accent Colors**: Muted colors for buttons and icons to maintain focus on content.

7. Data Flow and API Endpoints

7.1 Data Flow

1. **User Registration/Login**: The mobile app sends user data to the backend for OTP generation and verification.
2. **Category and Product Requests**: The client requests available product categories and farmer data.
3. **Search and Filter**: Data is sent from the client for search and filter operations, with results fetched dynamically.
4. **Product Details**: The client requests specific farmer and product information.

7.2 Key API Endpoints

- **POST /register**: Register a new user with phone and OTP verification.
- **POST /login**: Authenticates the user and initiates a session.
- **GET /categories**: Retrieves a list of available product categories.
- **GET /products**: Fetches products based on a category or search query.
- **GET /farmers**: Retrieves a list of farmers selling a specific product.
- **GET /farmer/{id}**: Fetches detailed information for a selected farmer.

8. Database Design

8.1 Entities

- **User Entity**:
    - user_id: Unique identifier
    - first_name, last_name: User's name
    - user_type: Either Farmer or Consumer
    - phone_number: Phone number for OTP verification
    - email: Optional contact email
- **Product Entity**:
    - product_id: Unique identifier for each product
    - name, category: Basic product information
    - quantity, rate: Quantity available and price per unit

    - farmer_id: Links product to a specific farmer
- **Farmer Entity**:
    - farmer_id: Unique identifier for each farmer
    - name, location: Farmer's basic details
    - contact_info: Contact details like phone and email
    - products: List of products associated with the farmer

8.2 Relationships

- **User-Product**: Many-to-Many (Consumers can buy multiple products, and each product can be sold by different farmers).
- **Farmer-Product**: One-to-Many (Each farmer can list multiple products).

9. Data Dictionary

1. User

Attribute	Data Type	Description	Constraints
userID	Integer	Unique identifier for each user in the system.	Primary Key, Auto-increment
name	String	Full name of the user.	Not Null, Max 50 characters
email	String	Email address used for login and notifications.	Unique, Not Null, Valid email format
password	String	Encrypted password for user authentication.	Not Null, Min 8 characters
role	String	Role of the user, either 'Farmer', 'Customer', or 'Admin'.	Not Null, Enum ('Farmer', 'Customer', 'Admin')

---

2. Farmer

Attribute	Data Type	Description	Constraints
farmID	Integer	Unique identifier for each farmer.	Primary Key, Auto-increment
farmLocation	String	Location of the farmer's farm.	Not Null, Max 100 characters

---

3. Customer

Attribute	Data Type	Description	Constraints
customerID	Integer	Unique identifier for each customer.	Primary Key, Auto-increment
cartID	Integer	Identifier for the customer’s cart, linking to the Cart entity.	Foreign Key (references Cart.cartID)

---

4. Admin

Attribute	Data Type	Description	Constraints
adminID	Integer	Unique identifier for each admin.	Primary Key, Auto-increment

5. Product

Attribute	Data Type	Description	Constraints
productID	Integer	Unique identifier for each product.	Primary Key, Auto-increment
productName	String	Name of the product.	Not Null, Max 50 characters
description	String	Brief description of the product.	Not Null, Max 255 characters
price	Float	Price of the product per unit.	Not Null, > 0
quantity	Integer	Quantity available for sale.	Not Null, >= 0
category	String	Category of the product (e.g., Fruits, Vegetables, Dairy).	Not Null, Max 30 characters

6. Order

Attribute	Data Type	Description	Constraints
orderID	Integer	Unique identifier for each order.	Primary Key, Auto-increment
customerID	Integer	Identifier of the customer placing the order.	Foreign Key (references Customer.customerID)
orderDate	Date	Date when the order was placed.	Not Null
totalAmount	Float	Total cost of the order.	Not Null, > 0
status	String	Status of the order (e.g., 'Pending', 'Shipped', 'Delivered').	Not Null, Enum ('Pending', 'Shipped', 'Delivered')

7. Cart

Attribute	Data Type	Description	Constraints
cartID	Integer	Unique identifier for each cart.	Primary Key, Auto-increment
customerID	Integer	Identifier of the customer owning the cart.	Foreign Key (references Customer.customerID)
totalAmount	Float	Total value of items in the cart.	Default 0, >= 0

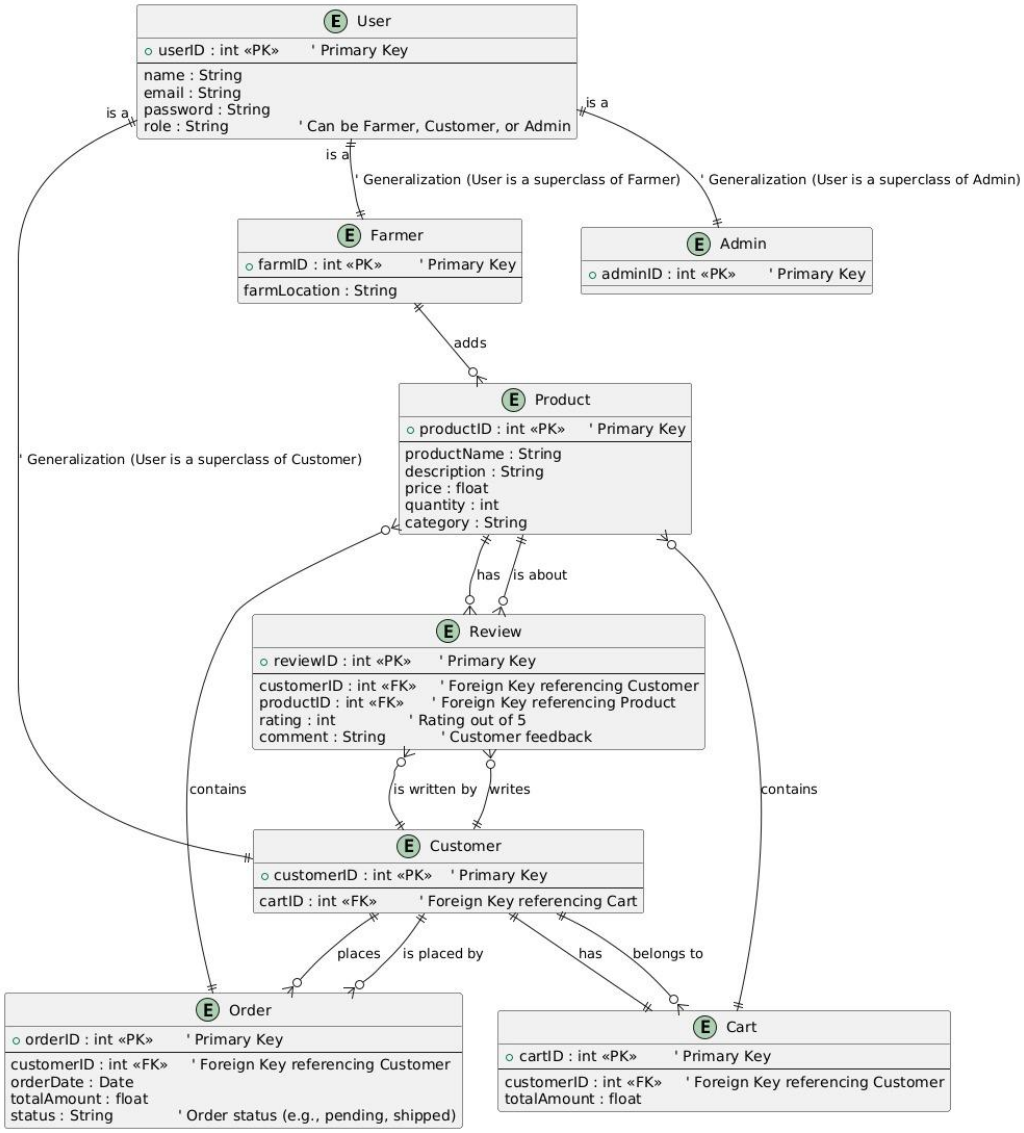
---

8. Review

Attribute	Data Type	Description	Constraints
reviewID	Integer	Unique identifier for each review.	Primary Key, Auto-increment
customerID	Integer	Identifier of the customer writing the review.	Foreign Key (references Customer.customerID)
productID	Integer	Identifier of the product being reviewed.	Foreign Key (references Product.productID)
rating	Integer	Rating given to the product (1 to 5).	Not Null, Range 1-5
comment	String	Textual feedback from the customer.	Optional, Max 255 characters

10. System Design Diagrams

10.1 ER Diagram



Entities and Attributes:

1. **User**:
   - Attributes: userID (PK), name, email, password, role (Farmer, Customer, Admin).
2. **Farmer**:
   - Inherits User attributes; additional attribute: farmLocation.
3. **Admin**:
   - Inherits User attributes; additional attribute: adminID.
4. **Customer**:
   - Inherits User attributes; additional attribute: customerID (FK to Cart).
5. **Product**:
   - Attributes: productID (PK), productName, description, price, quantity, category.
6. **Review**:

- Attributes: reviewID (PK), rating, comment; FKs: customerID (Customer), productID (Product).

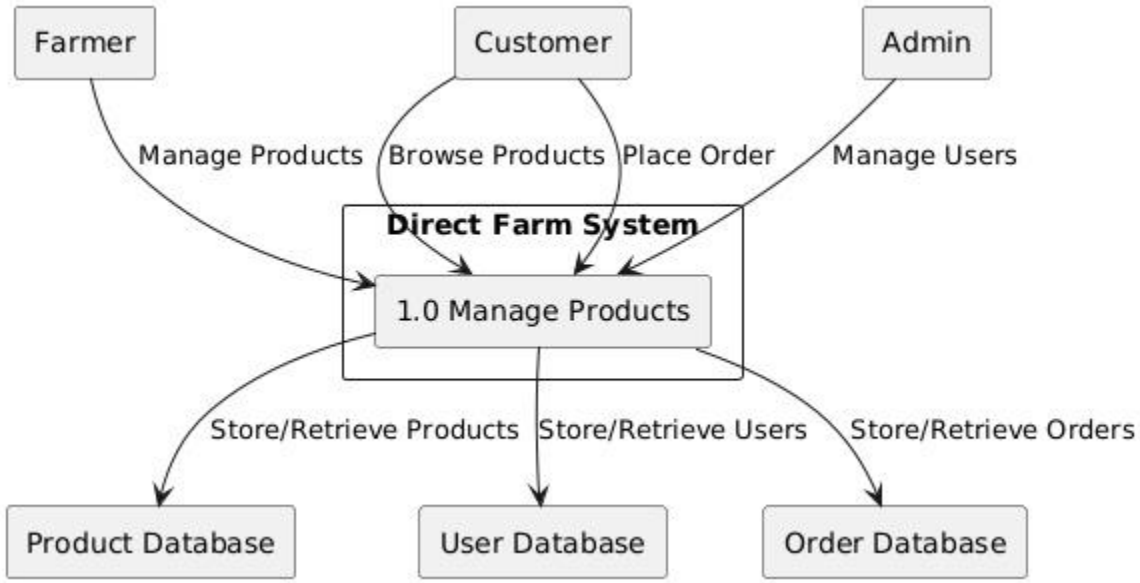
7. **Cart**:
   - Attributes: cartID (PK), totalAmount; FK: customerID (Customer).
8. **Order**:
   - Attributes: orderID (PK), orderDate, totalAmount, status; FK: customerID (Customer).

Relationships:

1. **User-Farmer-Admin-Customer Hierarchy**:
   - User is the superclass of Farmer, Customer, and Admin.
2. **Farmer Adds Product**:
   - Farmer adds Product for sale.
3. **Customer Places Order**:
   - Customer places Order, linked by FK.
4. **Customer Belongs to Cart**:
   - Each Customer has a Cart.
5. **Order Contains Products**:
   - An Order includes one or more Product entries.
6. **Product Has Reviews**:
   - Review links Customer feedback to Product.

Summary: The diagram illustrates entity relationships, including different user roles (Farmer, Customer, Admin), Product management, Cart for shopping, and Order for purchases. Inheritance and relationships streamline the interactions, ensuring each role's specific operations are represented effectively.

10.2 Data Flow Diagram 0



Components Explanation:

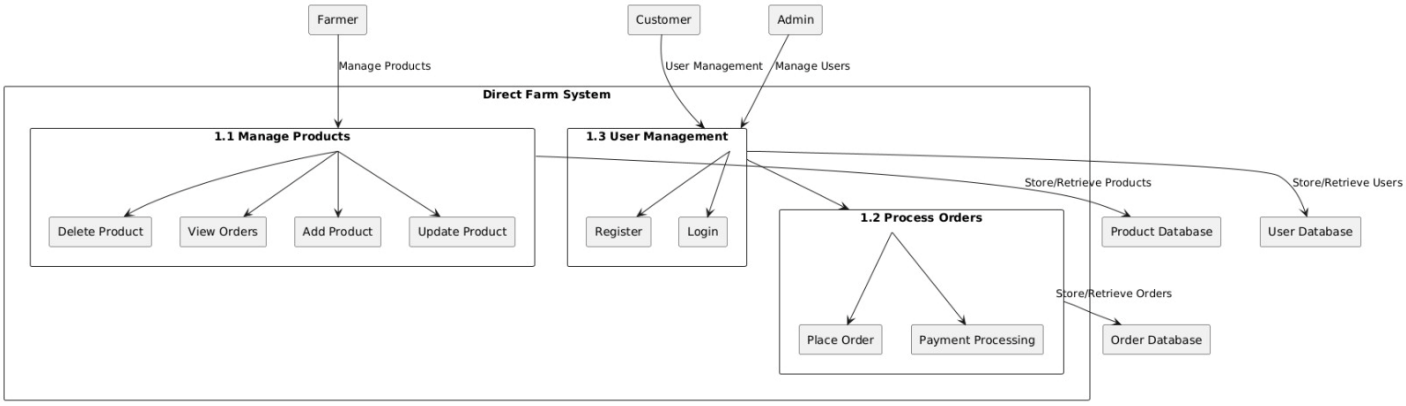
1. **Farm Direct System**:
    - This is the main system facilitating interactions between users and data stores, enabling the management of products, users, and orders.
2. **Entities**:
    - **Farmer**: Manages products by adding, updating, or removing them in the Product Database.
    - **Customer**: Browses products, adds them to the cart, and places orders, with order data stored in the Order Database.
    - **Admin**: Manages user accounts, adding or removing users as needed, with data stored in the User Database.
3. **Processes**:
    - **1.0 Manage Products**: Handles product management, including creation, updates, deletions, and retrievals for both Farmers and Customers.
        - **Farmers**: Use this to manage product listings (e.g., create, update, or delete).
        - **Customers**: Use this to browse products, add items to the cart, and initiate orders.
4. **Data Stores**:
    - **Product Database**: Stores all product-related information, including product name, description, price, and quantity.
    - **User Database**: Stores details of all users—Farmers, Customers, and Admins.
    - **Order Database**: Stores customer order details, including order status, total amount, and associated products.

Data Flows:

- **Manage Products (Farmer to System)**: Farmers provide product details for storage, retrieval, or updates in the Product Database.
- **Browse Products (Customer to System)**: Customers view available products in the Product Database.
- **Place Order (Customer to System)**: Customers place orders, which the system stores in the Order Database.
- **Manage Users (Admin to System)**: Admins manage user accounts, with details stored in the User Database.
- **Store/Retrieve Products, Users, Orders (System to Databases)**: The system interacts with the Product, User, and Order Databases to store and retrieve necessary data.

Summary: This Level 0 DFD offers a high-level view of the "Farm Direct" system's main processes, showing how Farmers, Customers, and Admins interact with the system. Farmers manage products, Customers browse and place orders, and Admins handle user accounts. Data is efficiently organized in the Product, User, and Order Databases to ensure seamless data flow and integrity.

10.3 Data Flow Diagram 1



External Entities:

1. **Farmer**: Manages products within the system.
2. **Customer**: Registers, logs in, and places orders for products.
3. **Admin**: Manages user accounts (e.g., customers and potentially farmers).

Major Processes:

1. **1.1 Manage Products**: Allows farmers to:
    - **Delete Product**: Remove products from the list.
    - **View Orders**: View orders placed by customers.
    - **Add Product**: Add new products to the system.
    - **Update Product**: Edit existing product details.
    - Interacts with the **Product Database** to store or retrieve product data.
2. **1.2 Process Orders**: Manages customer orders, including:
    - **Place Order**: Customers make orders.
    - **Payment Processing**: Processes payments for orders.
    - Interacts with the **Order Database** for storing or retrieving order information.
3. **1.3 User Management**: Handles user-related actions:
    - **Register**: New customers create accounts.
    - **Login**: Authenticates existing users.
    - Interacts with the **User Database** to store or retrieve user information.

Data Stores:

1. **Product Database**: Stores product details managed by farmers, accessed by the "Manage Products" process.
2. **Order Database**: Holds order information, accessed by the "Process Orders" process.
3. **User Database**: Contains user data (e.g., customers and potentially farmers), accessed by the "User Management" process.

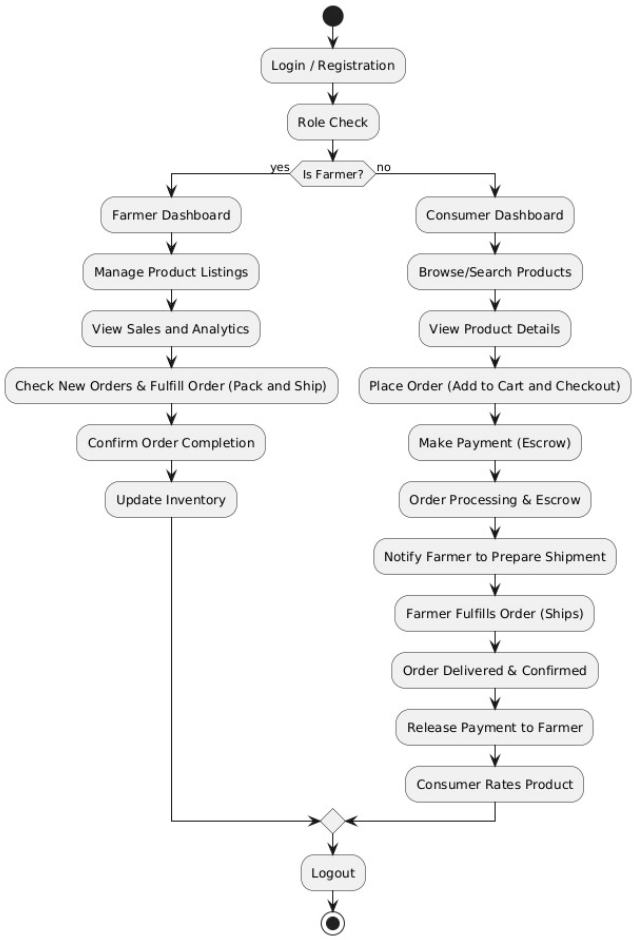
Data Flow Summary:

- **Farmers**: Manage products by adding, updating, viewing, and deleting them, with product data flowing to/from the **Product Database**.

- **Customers**: Register, log in, and place orders. Order data flows to/from the **Order Database**, and user data flows to/from the **User Database**.
- **Admins**: Manage users, with data flowing to/from the **User Database**.

Summary: This Level 1 DFD provides an overview of the "Farm Direct" system's internal data flows, showing how farmers, customers, and admins interact with processes and data stores. Farmers manage products, customers handle accounts and orders, and admins oversee user management. Each process interacts with relevant databases to ensure smooth information flow and data integrity.

10.4 Flowchart



Memory updated

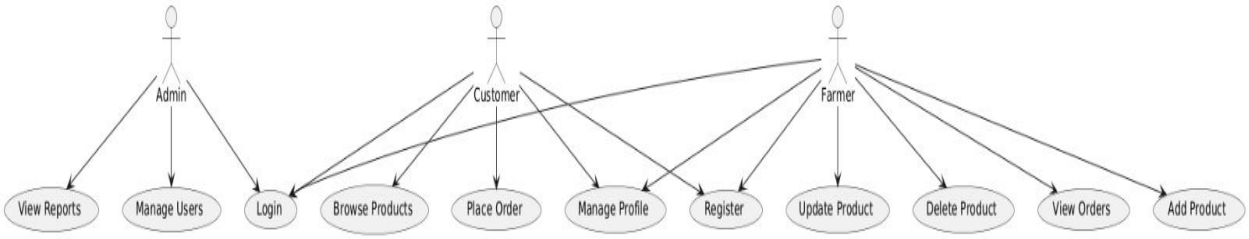
Got it! I'll make sure to keep the descriptions compact while following the format you provided. Here's the revised flowchart description for "Farm Direct":

Components Explanation:

1. **Farm Direct System:**
    - Facilitates connections between Farmers and Consumers, managing product listings, orders, and payments.
2. **User Roles:**
    - **Farmer:** Manages product listings, fulfills orders, and tracks sales.
    - **Consumer:** Searches for products, places orders, and makes payments.
3. **User Journey:**
    - **Start Point:**
        1. **Login / Registration:** Users register or log in to access the system.
        2. **Role Check:** System identifies the user as a Farmer or Consumer.
    - **Farmer's Pathway:**
        - Farmers access the Dashboard with functionalities:
            1. **Manage Product Listings:** Add, update, or delete products.
            2. **View Sales and Analytics:** Review sales metrics.
            3. **Check New Orders & Fulfill (Pack and Ship):** View and prepare incoming orders.
            4. **Confirm Order Completion:** Update order status post-fulfillment.
            5. **Update Inventory:** Adjust stock levels after orders.
    - **Consumer's Pathway:**
        - Consumers access the Dashboard with options:
            1. **Browse/Search Products:** Search and view available products.
            2. **View Product Details:** Access detailed product information.
            3. **Place Order (Add to Cart and Checkout):** Add items to cart and checkout.
            4. **Make Payment (Escrow):** Payments held in escrow until delivery.
            5. **Order Processing & Escrow:** System processes orders while holding payments.
            6. **Notify Farmer to Prepare Shipment:** Alerts farmer to prepare the order.
            7. **Farmer Fulfills Order (Ships):** Farmer ships the product.
            8. **Order Delivered & Confirmed:** Consumer confirms receipt.
            9. **Release Payment to Farmer:** Escrow releases payment post-confirmation.
            10. **Consumer Rates Product:** Consumers can rate products after delivery.
4. **End Point:**
    - **Logout:** Users log out after completing tasks.

Summary: This flowchart outlines the "Farm Direct" user journey, detailing how Farmers manage inventory and fulfill orders while Consumers search, order, and confirm deliveries. The escrow system secures transactions, releasing payments only upon delivery confirmation, enhancing trust between Farmers and Consumers.

10.5 Use Case



Components Explanation:

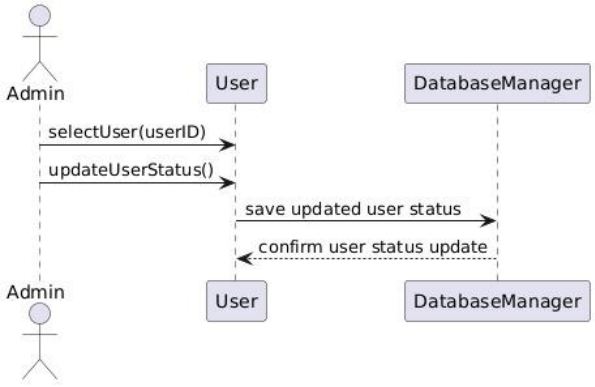
1. **Farm Direct System:**
    - Illustrates roles (Admin, Customer, Farmer) and their specific actions within the system.
2. **Actors:**
    - **Admin:** Administrative user with high-level system control.
    - **Customer:** Regular user who browses products and places orders.
    - **Farmer:** Supplier responsible for managing product listings.
3. **Use Cases:**
    - **Admin:**
        1. **View Reports:** Access reports for monitoring system performance, sales, and user activity.
        2. **Manage Users:** Add, update, or remove users (Customers and Farmers).
        3. **Login:** Access required to use the system features.
    - **Customer:**
        1. **Login:** Access the system to reach the dashboard.
        2. **Browse Products:** Search and view products listed by Farmers.
        3. **Place Order:** Add items to the cart and place orders.
        4. **Manage Profile:** Update personal information and payment details.
        5. **Register:** Create a new account for system access.
    - **Farmer:**
        1. **Login:** Access the Farmer Dashboard.
        2. **Update Product:** Edit product details (price, description, quantity).
        3. **Delete Product:** Remove products from listings.
        4. **View Orders:** Check incoming orders from Customers.
        5. **Add Product:** List new products for sale.
        6. **Manage Profile:** Update personal profile information.
        7. **Register:** Create a new account to start listing products.

Summary: This use case diagram highlights interactions between user roles within the "Farm Direct" system, showcasing specific permissions for each role:

- **Admin** oversees system operations and user management.
- **Customers** engage with products, manage orders, and update profiles.
- **Farmers** handle product listings, order fulfillment, and profile management.

The diagram underscores the structured permissions and responsibilities assigned to each role, ensuring secure and organized functionality within the system.

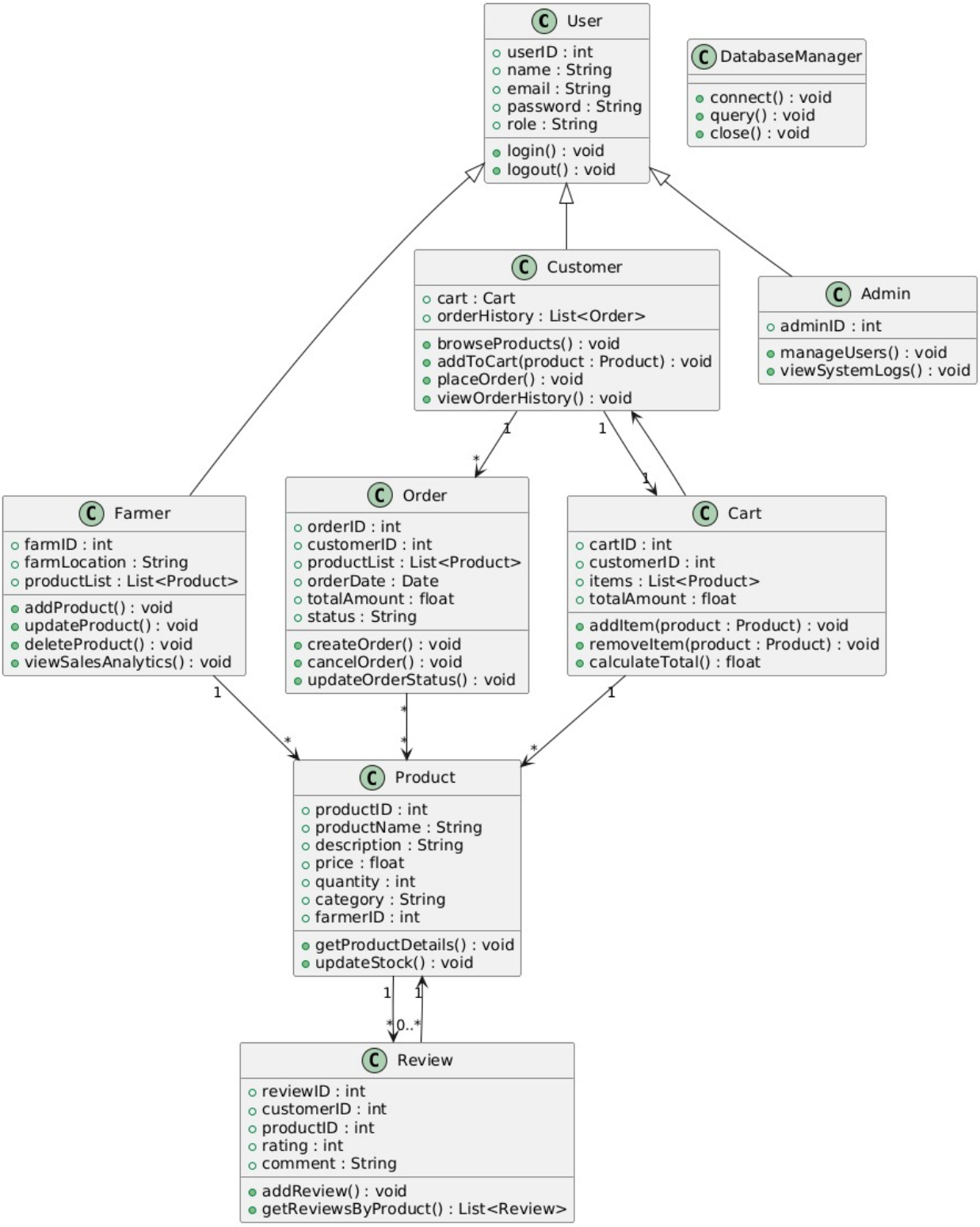
10.6 Admin Managing User



1. **Farm Direct System:**
    - Illustrates the process of an "Admin" managing a "User" with the assistance of a "DatabaseManager."
2. **Sequence of Actions:**
    1. **selectUser(userID):**
        - Admin initiates the selection of a user by their ID, sending a request to the User component to retrieve the specified user.
    2. **updateUserStatus():**
        - Admin sends a request to update the user's status (e.g., activate, deactivate, change permissions).
    3. **save updated user status:**
        - The User component interacts with the DatabaseManager to save the updated status in the database.
    4. **confirm user status update:**
        - DatabaseManager confirms the successful update of the user status.
    5. **Response back to Admin:**
        - The confirmation is relayed back to the User component, which informs the Admin of the successful status update.

Summary: This sequence diagram details the process for "Farm Direct," where the Admin selects a user, updates their status, and ensures that the changes are securely recorded in the database. Each step involves clear communication between the Admin, User component, and DatabaseManager, ensuring accurate and reliable management of user statuses.

10.7 Class Diagram



Components Explanation:

1. **Farm Direct System:**

- Represents the structure of an online marketplace with user roles and interactions.

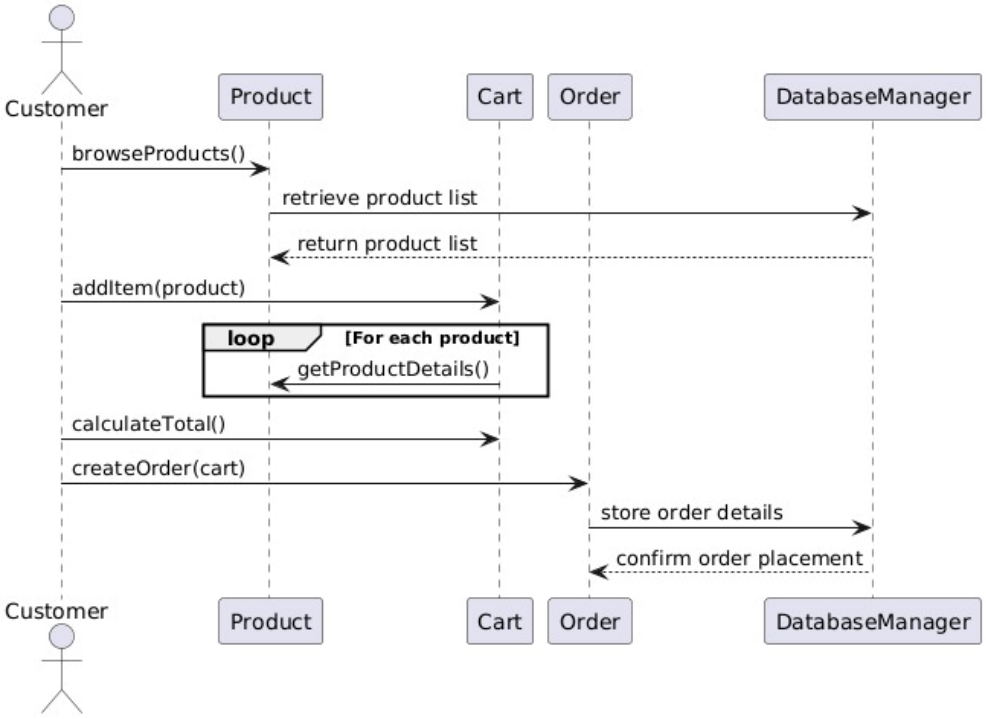
2. **Classes:**
    1. **User**
        - **Attributes:**
            - userID: Unique identifier.
            - name: User's name.
            - email: User's email.
            - password: User's authentication password.
            - role: User's role (Customer, Farmer, Admin).
        - **Methods:**
            - login(): Logs the user into the system.
            - logout(): Logs the user out of the system.
    2. **Customer (inherits from User)**
        - **Attributes:**
            - cart: Instance of Cart class.
            - orderHistory: List of past Order objects.
        - **Methods:**
            - browseProducts(): Allows browsing available products.
            - addToCart(product): Adds product to cart.
            - placeOrder(): Places an order for cart items.
            - viewOrderHistory(): Views previous orders.
    3. **Farmer (inherits from User)**
        - **Attributes:**
            - farmID: Unique identifier for the farm.
            - farmLocation: Location of the farm.
            - productList: List of added products.
        - **Methods:**
            - addProduct(): Adds a new product.
            - updateProduct(): Updates existing product details.
            - deleteProduct(): Removes a product.
            - viewSalesAnalytics(): Views sales data.
    4. **Admin (inherits from User)**
        - **Attributes:**
            - adminID: Unique identifier for admin.
        - **Methods:**
            - manageUsers(): Manages user accounts.
            - viewSystemLogs(): Views system logs.
    5. **DatabaseManager**
        - **Methods:**
            - connect(): Connects to the database.
            - query(): Executes database queries.
            - close(): Closes the database connection.
    6. **Order**
        - **Attributes:**
            - orderID: Unique identifier for the order.
            - customerID: ID of the customer.
            - productList: List of products in the order.
            - orderDate: Date of the order.

        - totalAmount: Total order cost.
        - status: Order status (e.g., pending, completed).
    - **Methods:**
        - createOrder(): Creates a new order.
        - cancelOrder(): Cancels an existing order.
        - updateOrderStatus(): Updates order status.
7. **Cart**
    - **Attributes:**
        - cartID: Unique identifier for the cart.
        - customerID: ID of the customer.
        - items: List of products in the cart.
        - totalAmount: Total amount of cart items.
    - **Methods:**
        - addItem(product): Adds product to cart.
        - removeItem(product): Removes product from cart.
        - calculateTotal(): Calculates total amount.
8. **Product**
    - **Attributes:**
        - productID: Unique identifier.
        - productName: Name of the product.
        - description: Product description.
        - price: Product price.
        - quantity: Available quantity.
        - category: Product category.
        - farmerID: ID of the listing farmer.
    - **Methods:**
        - getProductDetails(): Retrieves product information.
        - updateStock(): Updates stock quantity.
9. **Review**
    - **Attributes:**
        - reviewID: Unique identifier for the review.
        - customerID: ID of the reviewing customer.
        - productID: ID of the reviewed product.
        - rating: Customer rating.
        - comment: Additional comments.
    - **Methods:**
        - addReview(): Adds a product review.
        - getReviewsByProduct(): Retrieves reviews for a product.

3. **Relationships:**
    - **Inheritance:** Customer, Farmer, and Admin inherit from User, sharing common attributes and methods.
    - **Associations:**
        - **Customer** has a one-to-one relationship with **Cart** and a one-to-many relationship with **Order**.
        - **Order** has a one-to-many relationship with **Product**.
        - **Product** has a one-to-many relationship with **Review**.
        - **Product** is associated with **Farmer**, linking products to the farmers who listed them.

Summary: This class diagram outlines the structure of the "Farm Direct" system, detailing user interactions with products, orders, and reviews. Farmers manage products, Customers place orders, and Admins oversee user management, providing a comprehensive framework for the online marketplace.

10.8 Customer Browsing products and placing an order



Components Explanation:

1. **Farm Direct Sequence Diagram:**
    - Illustrates the process of a Customer browsing products, adding them to their cart, calculating totals, and placing an order.
2. **Sequence Steps:**
    1. **browseProducts():**
        - The Customer initiates the browsing process, sending a request to the Product component for available products.
    2. **retrieve product list and return product list:**
        - The Product component retrieves the product list from the database and returns it to the Customer.
    3. **addItem(product):**
        - The Customer adds a selected product to their Cart.
        - The Cart component requests product details from the Product component for each added product.
    4. **loop [For each product] getProductDetails():**

- Within a loop, the Cart component retrieves details (e.g., name, price, availability) for each product from the Product component.

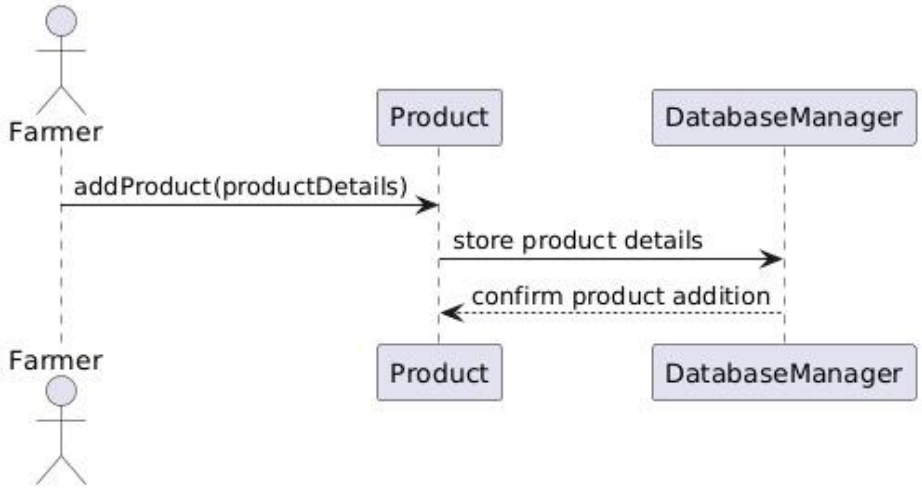
5. **calculateTotal():**
   - The Customer requests the total price of items in the cart.
   - The Cart component calculates the total based on the added products.
6. **createOrder(cart):**
   - After confirming the cart total, the Customer initiates the order placement by sending a request to the Order component.
7. **store order details:**
   - The Order component communicates with the DatabaseManager to store order details in the database.
8. **confirm order placement:**
   - The DatabaseManager confirms the order placement back to the Order component, which relays the confirmation to the Customer.

Summary:

- The diagram captures the Customer journey of browsing products, managing a Cart, and placing an order.
- The Product component provides necessary product information.
- The Cart component manages items and calculates the total cost.
- The Order component creates and stores order details in the database, with the DatabaseManager ensuring data integrity and confirmation.

This flow represents a typical e-commerce interaction in "Farm Direct," facilitating a seamless customer experience in browsing, cart management, and order placement.

10.9 Farmer adding a product



This diagram is a **sequence diagram** showing the process by which a **Farmer** adds a product, and how the system handles this addition in interaction with other components like **Product** and **DatabaseManager**.

Here's a breakdown of the process:

1. **Farmer Initiates Product Addition**:
    - The **Farmer** initiates the action by calling the addProduct(productDetails) function on the **Product** component. Here, productDetails likely contain information such as the product name, quantity, price, etc.
2. **Product Component Stores Details**:
    - The **Product** component then sends a request to the **DatabaseManager** to store product details. This step involves saving the product information to the database or another storage system.
3. **DatabaseManager Confirms Storage**:
    - After storing the product details, the **DatabaseManager** sends a confirmation message back to the **Product** component, indicated by the confirm product addition.
4. **Completion**:
    - Once the product addition is confirmed, the process ends. The **Product** component might then update the **Farmer** with a success message or simply complete the process silently, depending on the implementation.

This diagram outlines a straightforward product addition workflow, where the **Farmer** interacts with the system to add product information, which is managed by **Product** and **DatabaseManager** components to ensure data is stored properly.

11. Security and Privacy

- **Authentication**: OTP-based phone verification for registration.
- **Data Privacy**: User information is stored securely and accessed only by authorized users.
- **Access Control**: Only authenticated users can access features like viewing product details and farmer information.

12. Performance Considerations

- **Database Indexing**: Index frequently queried fields like product_id and category in MongoDB for efficient data retrieval.
- **Image Loading Optimization**: Cache images locally to reduce loading times for commonly accessed images.
- **Optimized Data Loading**: Limit data payloads by fetching only necessary fields initially and loading detailed data on demand.

13. Testing Plan

13.1 Unit Testing

- **Signup and OTP Verification**: Ensure OTPs are generated, sent, and verified accurately.
- **Product Search and Filter**: Confirm search and filter features work as expected with various inputs.

13.2 Integration Testing

- **API Endpoints**: Verify data retrieved from endpoints matches expected results.
- **Navigation Testing**: Ensure smooth transitions between screens without data loss.

13.3 User Acceptance Testing (UAT)

- **Ease of Use**: Gather feedback from target users on the app's navigation and usability.
- **Load Testing**: Simulate multiple concurrent users to test system performance.

14. Deployment and Maintenance

Deployment

The Farm Direct app will be deployed on both iOS and Android platforms, with backend services hosted on a cloud provider to ensure scalability and reliability.

Maintenance

- **Error Logging**: Implement logging for tracking and debugging.
- **Updates**: Frequent updates based on user feedback and bug reports to enhance usability.

15. Appendices

- **Technology Stack**: Flutter, MongoDB, Spring Boot for backend API
- **Development Tools**: Android Studio, Visual Studio Code, GitHub for version control
- **Project Management**: GitHub Issues for task tracking, GitHub Actions for CI/CD integration

This Software Design Specification provides a comprehensive guide for developing and refining the Farm Direct app. Let me know if any section needs more detail or if there's another focus area you'd like to expand.