PROJECT REPORT

FARMER MARKET DIRECT CONNECT

A Web Platform for Direct Farmer-Buyer Connection

1. PROJECT OVERVIEW

Farmer Market Direct Connect is a comprehensive web-based platform designed to eliminate intermediaries in the agricultural supply chain by providing direct communication channels between farmers and buyers.

Key Highlights:

• Three distinct user roles: Farmer, Buyer, and Admin

• Product listing and management system for farmers

• Advanced browsing and inquiry system for buyers

• Administrative oversight and monitoring capabilities

• RESTful API for future mobile integration

• Session-based secure authentication

• Responsive Bootstrap UI for all devices

Problem Addressed:

Farmers in India lose 30-40% of their profit to middlemen, while buyers pay inflated prices. This platform enables direct connection, ensuring fair pricing and transparent transactions for both parties.

Target Users:

• Farmers - List and sell agricultural products

• Buyers - Browse and purchase directly from farmers

• Administrators - Monitor and manage platform operations

2. OBJECTIVE

Primary Objectives:

1. Create a digital marketplace connecting farmers directly with buyers, eliminating middlemen

2. Implement secure role-based authentication system for three user types (Farmer, Buyer, Admin)

3. Provide comprehensive product management system with image upload, pricing, and inventory tracking

4. Develop inquiry-based communication system between farmers and buyers

5. Build responsive, user-friendly interface accessible on all devices

6. Demonstrate all 15 practical concepts from Web Technology with .NET curriculum

7. Create RESTful API for potential mobile application integration

8. Implement caching and performance optimization for better user experience

Technical Objectives:

• Apply ASP.NET Core MVC architecture pattern

• Implement Entity Framework Core for database operations

• Use SQLite for lightweight, portable database solution

• Create modular, maintainable, and scalable code structure

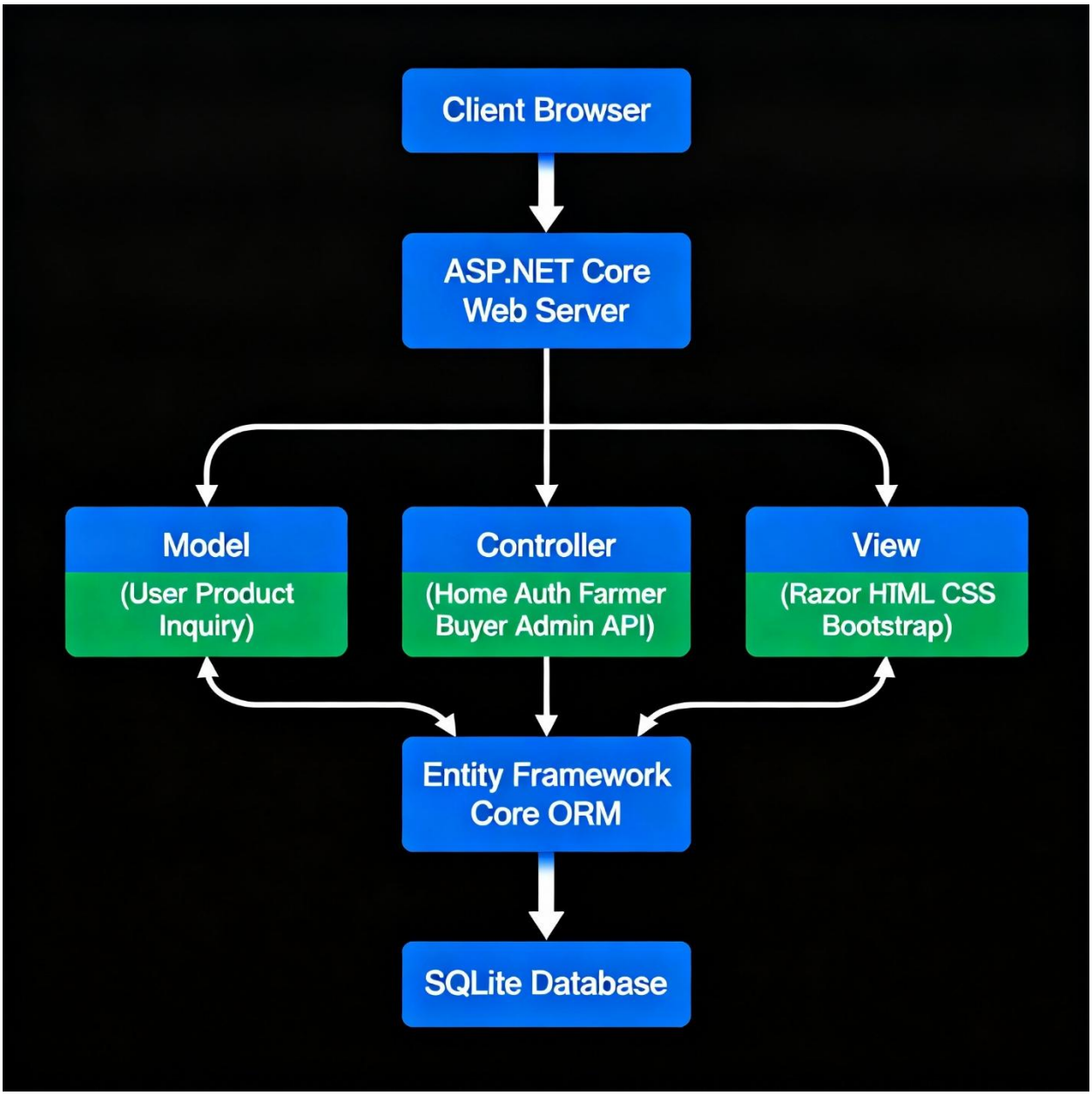
• Follow modern web development best practices

3. TECHNOLOGY STACK

FRONTEND	HTML5	Markup Language
	CSS3	Styling
	JavaScript (ES6+)	Client-side Logic
	Bootstrap 5.3	UI Framework
	Font Awesome 6.x	Icons
BACKEND	ASP.NET Core MVC	8.0 - Web Framework
	C#	12.0 - Language
	Entity Framework Core	8.0 - ORM
	LINQ	Data Querying
DATABASE	SQLite	3.x - Database
	EF Core Migrations	Schema Management
TOOLS	Visual Studio Code	Code Editor
	.NET CLI	Build & Run
	Git	Version Control
	GitHub	Repository Hosting
	DB Browser for SQLite	Database Management

4. SYSTEM ARCHITECTURE

4.1 Architecture Diagram



4.2 Application Flow

---

1. User accesses application through web browser
2. Request reaches ASP.NET Core routing system
3. Controller receives request and processes business logic
4. Model interacts with database through Entity Framework
5. Controller passes data to View using ViewBag/ViewData/TempData
6. View renders HTML with data and sends response to browser
7. User sees the final page with dynamic content

4.3 MVC Pattern Implementation

MODEL: Represents data and business logic

- User.cs, Product.cs, Inquiry.cs
- FarmerMarketContext.cs (Database Context)

VIEW: Presents data to user (UI Layer)

- Razor Views (.cshtml files)
- Layout, Partial Views
- HTML, CSS, JavaScript

CONTROLLER: Handles user requests and application flow

- HomeController, AuthController
- FarmerController, BuyerController
- AdminController, ApiController

5. MODULES / FUNCTIONAL COMPONENTS

Module Name	Description
Authentication Module	User registration, login, logout with session management. Role-based access control for Farmer, Buyer, Admin.
Farmer Module	Dashboard showing product statistics. Add, edit, delete products with image upload. View and manage buyer inquiries. Inventory tracking and management.
Buyer Module	Browse products with category filters. Search and sort functionality. View detailed product information. Send inquiries to farmers. Track inquiry status and responses.
Admin Module	Platform statistics dashboard. Manage users (view, delete). Monitor products and inquiries. Content moderation and platform oversight capabilities.
Product Management	Complete CRUD operations for products. Image upload and storage.

	Price and inventory tracking.
	Category-based organization.
	Organic product tagging.
Inquiry System	Buyer can send product inquiries.
	Farmers receive and view inquiries.
	Status tracking (Pending/Responded).
	Contact information exchange.
Home & Navigation	Landing page with time-based greetings.
	Platform statistics display.
	Navigation menu with role-based options.
	About and Privacy pages.
RESTful API	GET /api/api/products - All products
	GET /api/api/products/{id} - Single
	GET /api/api/farmers - All farmers
	GET /api/api/stats - Statistics
	JSON responses for mobile integration.
Caching Module	Memory cache for product listings.
	Dashboard statistics caching.
	Improved performance with reduced database calls.
	Cache invalidation on data updates.

6. DATABASE DESIGN

6.1 Database Tables

TABLE: USERS

Column Name	Data Type	Description
UserId	INT	Primary Key (Auto)
Name	VARCHAR (100)	User's full name
Email	VARCHAR (100)	Unique email address
Password	VARCHAR (255)	Encrypted password
Phone	VARCHAR (15)	Contact number
UserType	VARCHAR (20)	Farmer/Buyer/Admin
Address	TEXT	Full address
City	VARCHAR (50)	City name
State	VARCHAR (50)	State name
CreatedAt	DATETIME	Registration timestamp

TABLE: PRODUCTS

Column Name	Data Type	Description
ProductId	INT	Primary Key (Auto)
FarmerId	INT	Foreign Key → Users
ProductName	VARCHAR (100)	Product name
Category	VARCHAR (50)	Vegetable/Fruit/Grain/etc.
Price	DECIMAL (10,2)	Price per unit
Unit	VARCHAR (20)	Kg/Quintal/Liter/etc.
AvailableQty	DECIMAL (10,2)	Available quantity
Description	TEXT	Product description
ImagePath	VARCHAR (255)	Product image URL
IsOrganic	BOOLEAN	Organic flag (True/False)
CreatedAt	DATETIME	Product added timestamp

TABLE: INQUIRIES

Column Name	Data Type	Description
InquiryId	INT	Primary Key (Auto)
ProductId	INT	Foreign Key → Products
BuyerId	INT	Foreign Key → Users
RequiredQty	DECIMAL (10,2)	Quantity needed
Message	TEXT	Inquiry message
Status	VARCHAR (20)	Pending/Responded/Closed
CreatedAt	DATETIME	Inquiry sent timestamp

6.2 Relationships

• Users (Farmer) ——— 1:N ——► Products

One farmer can have multiple products

• Products ——— 1:N ——► Inquiries

One product can have multiple inquiries

• Users (Buyer) ——— 1:N ——► Inquiries

One buyer can send multiple inquiries

7. IMPLEMENTATION / WORKING

7.1 User Registration & Login Flow

Step 1: User visits home page and clicks "Register"

Step 2: Fills registration form with required details

Step 3: Selects user type (Farmer or Buyer)

Step 4: System validates data using Model Validation

Step 5: Data stored in Users table with encrypted password

Step 6: User redirected to Login page

Step 7: User enters email and password

Step 8: System validates credentials against database

Step 9: On success, creates session with UserId and UserType

Step 10: User redirected to role-specific dashboard

7.2 Farmer Product Management Flow

Step 1: Farmer logs in and reaches Dashboard

Step 2: Clicks "Add Product" button

Step 3: Fills product form (name, category, price, quantity, etc.)

Step 4: Uploads product image (async operation)

Step 5: System validates all fields

Step 6: Product saved in Products table with FarmerId

Step 7: Image stored in wwwroot/images/products folder

Step 8: Cache cleared for product listings

Step 9: Success message displayed via TempData

Step 10: Farmer redirected back to Dashboard

Step 11: Product appears in Dashboard table

Step 12: Farmer can Edit or Delete products anytime

7.3 Buyer Product Browsing & Inquiry Flow

Step 1: Buyer visits "Browse Products" page

Step 2: Products fetched from cache (if available)

Step 3: Buyer applies filters (category, search, sort)

Step 4: Query string parameters passed to controller

Step 5: Filtered products displayed in card grid

Step 6: Buyer clicks "View Details" on a product

Step 7: Detailed product page shown with farmer info

Step 8: Buyer fills inquiry form with required quantity

Step 9: Inquiry saved in Inquiries table

Step 10: Farmer notified (visible in their Inquiries page)

Step 11: Buyer can track inquiry status in "My Inquiries"

7.4 Admin Monitoring Flow

Step 1: Admin logs in with admin credentials

Step 2: Dashboard displays platform statistics

Step 3: Statistics fetched using LINQ queries

Step 4: Admin can view all users, products, inquiries

Step 5: Admin can delete inappropriate content

Step 6: All actions logged with timestamps

7.5 API Request Flow

Step 1: Client sends HTTP GET request to API endpoint

Step 2: ApiController receives request

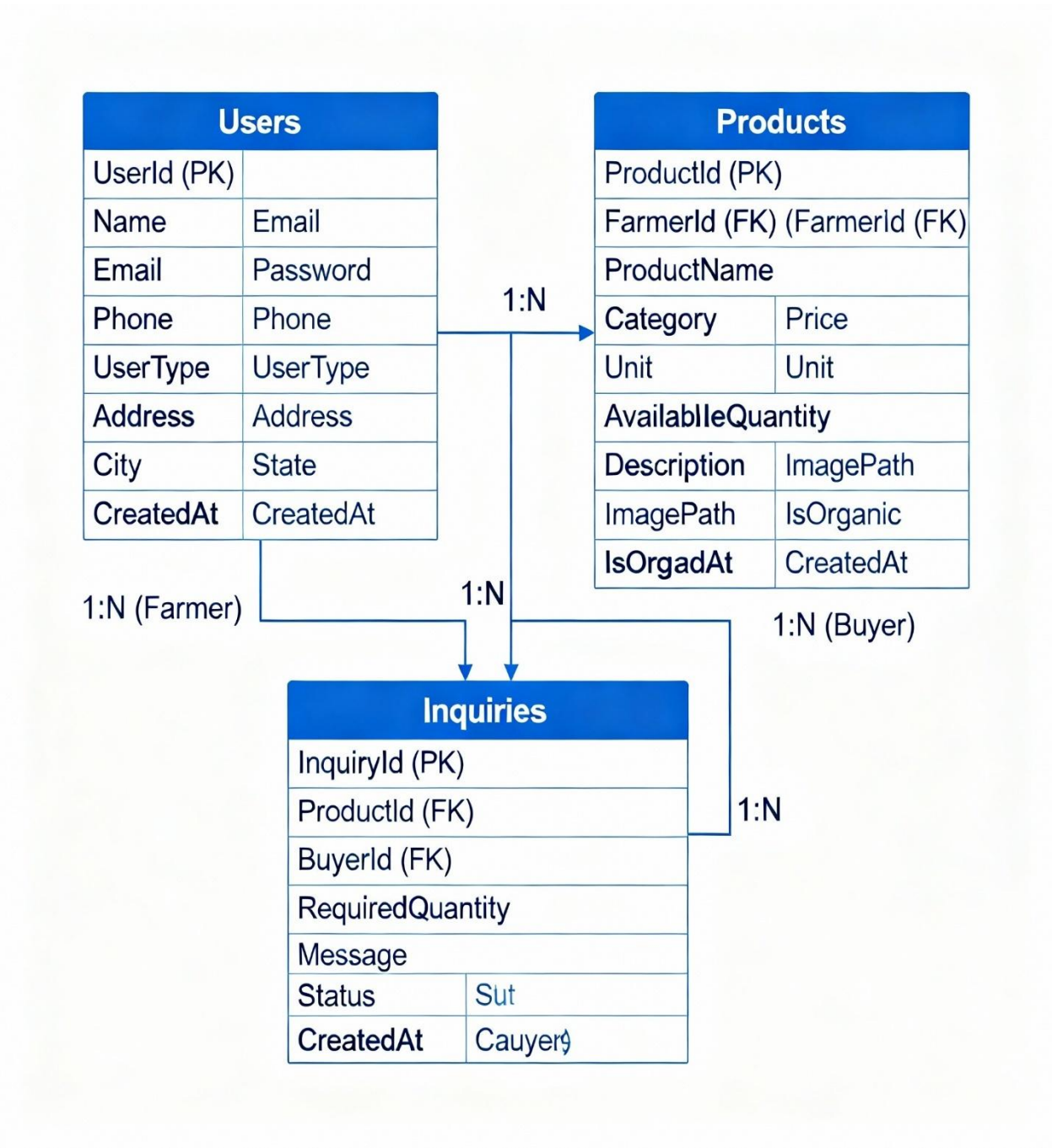
Step 3: Data fetched from database using Entity Framework

Step 4: Data serialized to JSON format

Step 5: JSON response sent back to client

Step 6: Mobile app or external system can consume API

8. ER DIAGRAM


Users
UserId (PK)
Name
Email
Email
Password
Phone
Phone
UserType
UserType
Address
Address
City
State
CreatedAt
CreatedAt
Products
ProductId (PK)
FarmerId (FK) (FarmerId (FK)
ProductName
Category
Price
Unit
Unit
AvailablleQuantity
Description
ImagePath
ImagePath
IsOrganic
IsOrgadAt
CreatedAt
1:N
1:N (Farmer)
1:N
1:N (Buyer)
Inquiries
InquiryId (PK)
ProductId (FK)
BuyerId (FK)
RequiredQuantity
Message
Status
Sut
CreatedAt
Cauyer
1:N

Relationships:

• USERS (Farmer) ──►◄ PRODUCTS (One-to-Many)

- One Farmer can list multiple Products

- Each Product belongs to one Farmer

• PRODUCTS ──►◄ INQUIRIES (One-to-Many)

- One Product can receive multiple Inquiries

- Each Inquiry is for one Product

• USERS (Buyer) ──►◄ INQUIRIES (One-to-Many)

- One Buyer can send multiple Inquiries

- Each Inquiry is from one Buyer

9. TESTING SUMMARY

No.	Functionality	Expected Result	Status
1.	User Registration	User account created, data stored in DB	✓ PASS
2.	User Login	Session created, user redirected to dashboard	✓ PASS
3.	Add Product (Farmer)	Product saved with image, shown in list	✓ PASS
4.	Edit Product	Product details updated successfully	✓ PASS
5.	Delete Product	Product removed from database and UI	✓ PASS
6.	Browse Products (Buyer)	All products displayed with filters working	✓ PASS
7.	Search Products	Products filtered by search term	✓ PASS

8.	Category Filter	Products filtered by selected category	✓ PASS
9.	Sort Products	Products sorted by price(low/high)	✓ PASS
10.	View Product Details	Complete product info with farmer contact	✓ PASS
11.	Send Inquiry	Inquiry saved, visible to farmer	✓ PASS
12.	View My Inquiries	All buyer inquiries displayed with status	✓ PASS
13.	Farmer View Inquiries	All product inquiries shown with buyer info	✓ PASS
14.	Admin Dashboard	Statistics displayed correctly	✓ PASS
15.	Admin Manage Users	All users listed, can delete non-admin users	✓ PASS
16.	Admin Manage Products	All products listed, can delete any product	✓ PASS
17.	API Endpoint - Products	JSON response with all products	✓ PASS

18.	API Endpoint - Stats	JSON response with platform statistics	✓ PASS
19.	Image Upload	Image stored and displayed correctly	✓ PASS
20.	Session Management	User stays logged in, logout clears session	✓ PASS
21.	Model Validation	Invalid data shows error messages	✓ PASS
22.	Responsive UI	Works on mobile, tablet, desktop	✓ PASS
23.	Time-based Greeting	Shows correct greeting based on time of day	✓ PASS
24.	Cache Implementation	Faster page loads, cache clears on update	✓ PASS
25.	Logout Functionality	Session cleared, user redirected to home	✓ PASS

TESTING SUMMARY:

Total Test Cases: 25	Passed: 25	Failed: 0	Success Rate: 100%

10. CONCLUSION

The Farmer Market Direct Connect project successfully demonstrates a comprehensive solution to the agricultural supply chain inefficiency problem by creating a direct digital platform for farmer-buyer interaction.

Key Achievements:

✓ Successfully implemented all 15 practical concepts from the Web Technology with .NET curriculum

✓ Created a fully functional three-role system (Farmer, Buyer, Admin) with distinct responsibilities and access controls

✓ Developed responsive, user-friendly interface using Bootstrap that works seamlessly across all devices

✓ Implemented secure session-based authentication with role-based access control

✓ Built RESTful API for future mobile application integration

✓ Used Entity Framework Core with code-first approach for efficient database management

✓ Applied performance optimization through memory caching

✓ Demonstrated async operations for improved user experience

Project Impact:

This platform has the potential to revolutionize the agricultural market by:

• Increasing farmer profits by 30-40% through middleman elimination

• Reducing buyer costs with direct sourcing

• Providing transparent pricing and product information

• Enabling direct communication and trust-building

• Supporting sustainable agriculture practices

Technical Learning:

Through this project, I gained hands-on experience in:

• ASP.NET Core MVC architecture and design patterns

• Entity Framework Core ORM and database migrations

• RESTful API development and JSON responses

• Session management and authentication systems

• Bootstrap framework for responsive design

• Git version control and GitHub collaboration

• Async programming and performance optimization

The project successfully meets all academic requirements while solving a real-world problem, making it both educationally valuable and practically applicable.

11. FUTURE SCOPE

1. MOBILE APPLICATION DEVELOPMENT

• Build native Android and iOS apps using the existing RESTful API

• Implement push notifications for new inquiries and responses

• Add GPS-based location tracking for nearby farmers

• Enable offline mode for product browsing

2. PAYMENT GATEWAY INTEGRATION

• Integrate Razorpay or PayTM for secure online transactions

• Implement order placement and payment processing

• Add invoice generation and payment history tracking

• Enable wallet functionality for faster transactions

3. ADVANCED COMMUNICATION FEATURES

• Real-time chat system between farmers and buyers

• Video call integration for product demonstration

• SMS and WhatsApp integration for instant notifications

• Email notification system for inquiry updates

4. AI-POWERED FEATURES

• Price prediction based on market trends

• Crop recommendation system for farmers

• Demand forecasting for better inventory management

• Chatbot for customer support and queries

5. MULTI-LANGUAGE SUPPORT

• Add Hindi, Gujarati, and other regional languages

• Voice-based navigation for less tech-savvy users

• Language switching option in user settings

• Regional content customization

6. LOGISTICS INTEGRATION

• Partner with delivery services for product transportation

• Real-time order tracking and delivery status

• Calculate delivery charges based on distance

• Pickup and drop-off point management

7. ANALYTICS AND REPORTING

• Sales analytics dashboard for farmers

• Market trend analysis and reports

• Seasonal demand predictions

• Revenue tracking and profit calculations

8. SOCIAL FEATURES

• User reviews and ratings for farmers and products

• Community forum for agricultural discussions

• Success stories and testimonials

• Social media integration for sharing products

12. SYSTEM REQUIREMENTS

12.1 Hardware Requirements

Component	Specification
Processor	Intel Core i3 or equivalent
RAM	4 GB minimum (8 GB recommended)
Hard Disk	10 GB free space
Display	1024 x 768 resolution or higher
Internet Connection	Broadband connection required

12.2 Software Requirements

Software	Version
Operating System	Windows 10/11, macOS, Linux
.NET SDK	8.0 or higher
Web Browser	Chrome, Firefox, Edge (Latest)
Code Editor	Visual Studio Code (Recommended)
Database	SQLite 3.x (included in project)
Git	Latest version (for version ctrl)

12.3 Development Tools (Optional)

• DB Browser for SQLite - For database management

• Postman - For API testing

• Git Bash - For Git operations

• Visual Studio 2022 - Alternative IDE

13. CHALLENGES & LEARNINGS

Challenges Faced:

1. DATABASE SELECTION

Initially attempted to use MySQL with XAMPP but encountered configuration issues and port conflicts. Successfully resolved by switching to SQLite, which provided a lightweight, portable solution without requiring separate server installation.

2. IMAGE UPLOAD IMPLEMENTATION

Implementing async file upload while maintaining database consistency was challenging. Learned to use IFormFile with async operations and proper file path management in wwwroot folder.

3. SESSION MANAGEMENT

Understanding session lifecycle and implementing role-based access control across multiple controllers required careful planning.

Resolved by creating consistent session checking mechanism.

4. CACHING STRATEGY

Implementing memory cache while ensuring data consistency after CRUD operations required cache invalidation logic. Learned to balance performance with data freshness.

5. BOOTSTRAP RESPONSIVE DESIGN

Making the application truly responsive across all devices, especially complex tables and forms, required extensive testing and CSS customization.

Key Learnings:

• Importance of choosing appropriate technology stack based on project requirements and constraints

• Value of MVC architecture in creating maintainable and scalable applications

• Significance of user experience and responsive design in modern web applications

• Power of Entity Framework Core in simplifying database operations and reducing boilerplate code

• Effective use of Git and GitHub for version control and code management

• Practical application of theoretical concepts learned in classroom

• Problem-solving skills developed through debugging and issue resolution

• Time management and project planning for meeting deadlines

This project provided invaluable hands-on experience in full-stack web development and reinforced the importance of practical implementation alongside theoretical knowledge.

END OF REPORT