Table of Contents

# Dairy Drop - Documentation (First Deliverable)

**Project Title:** Dairy Drop - Online Dairy Products E-Commerce Platform

**Group ID:** F25PROJECT88363

**Student ID:** BC230205728

**Supervisor Name:** Muhammad Hassaan

**Email:** m.hassaan@vu.edu.pk

**Date:** 05-Dec-2025

## Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 02-Nov-2025 | 0.1 | Initial draft submission | BC230205728 |
| 05-Dec-2025 | 1.0 | Complete revision with detailed requirements | BC230205728 |

## Table of Contents

1. [Introduction of Document](#Xdffda7aad17513f81cad9ea554b3c4451930025)
2. [Scope (of the Project)](#X6cbcb9c616fae0370816ca279e6bc8686a4265f)
3. [Functional Requirements](#Xe7303fedc1caa0f98a54d3d573fddb57a70f32c)
4. [Non-Functional Requirements](#Xa059f3edd26bb652ad8373400c47e55bfc01b65)
5. [Use Case Diagram](#X98ceaed315979a46ffe72fdcfa04fd3afa6636c)
6. [Work Plan](#Xedc59b38aee921599a1a06d84af15ecd6c6e2d3)
7. [Entity Relationship Diagram (ERD)](#X0dffae920e159e3a5a87c83e3b9d9957bdcfb09)
8. [Database Design](#Xf39b8f1f42479e81da005ed623c5066659178f6)
9. [Interface Design](#X69da497d4b01c37e1680d08fbd601149de364c9)

## 1. Introduction of Document

This document serves as the comprehensive planning and design documentation for the Dairy Drop project, a web-based e-commerce platform for dairy products. As the first deliverable, it establishes the foundation for the entire project implementation. The primary purpose of this document is to clearly identify all functional and non-functional requirements, ensuring a unified understanding among all stakeholders. It acts as a design blueprint by providing detailed system design diagrams, such as Use Case models, Entity Relationship Diagrams (ERDs), and database schemas to guide the development process. Furthermore, this document establishes a realistic project plan with clear timelines, milestones, and deliverables. It functions as a vital communication tool to facilitate feedback and validation between the developer and supervisor before coding begins, while also serving to mitigate risk by identifying technical challenges and dependencies early to reduce implementation hurdles.

The benefits of undertaking this thorough documentation phase are significant. Primarily, it drives cost reduction, as fixing requirement issues during the planning stage is significantly cheaper than addressing post-development changes. It promotes time efficiency by preventing scope creep, reducing necessary rework, and maintaining a strict focus on defined objectives. Additionally, the process ensures stakeholder alignment regarding expectations for system capabilities and limitations. Detailed specifications naturally lead to quality improvement, resulting in implementations that better meet user needs. Finally, this documentation builds a strong maintenance foundation, serving as a reference point for future maintenance, updates, and scaling.

In adherence to industry-standard software documentation practices, this document follows a structured flow. It begins with the project scope, moves through a detailed requirements analysis, presents visual system models, outlines the development timeline, and concludes with comprehensive database and interface design specifications.

## 1. Scope of the Project

The Dairy Drop project is scoped as a **MERN stack** e-commerce platform designed to connect dairy producers directly with consumers for convenient online ordering and delivery of fresh dairy products. The core deliverable includes a fully **responsive** web application featuring secure user authentication (JWT), comprehensive product browsing with advanced filtering, a persistent shopping cart, and a streamlined checkout utilizing **Cash on Delivery (COD)**. Key functionality encompasses complete order lifecycle tracking (Pending to Delivered), order history, and a product review system. Administrative features provide secure CRUD operations for product, order, and user management, along with review moderation. The project is bounded by a single-city geographic deployment, limited to approximately 10,000 active users and 500 daily orders, with a strict **9-week development timeline** (Nov 1, 2025 - Feb 1, 2026). Functionality is strictly defined to exclude online payment gateways, subscription features, multi-vendor support, native mobile apps, and third-party logistics integration, ensuring focus on the defined core business logic.

## 3. Functional Requirements

Functional requirements define what the system will do - specific behaviors, features, and operations the platform must support.

### 3.1 User Authentication and Account Management

| **ID** | **Requirement** | **Description** | **Acceptance Criteria** |
| --- | --- | --- | --- |
| **FR-1.1** | User Registration | Users create accounts with name, email, phone, and password. System validates inputs, checks for duplicates, and hashes passwords using bcrypt. | No duplicate email/phone; password encrypted; email validation; success confirmation |
| **FR-1.2** | User Login | Users log in with email/phone and password. System generates JWT token valid for 24 hours upon successful authentication. | Valid credentials grant access; JWT token generated; invalid attempts logged; password never in plain text |
| **FR-1.3** | User Logout | Users securely log out, clearing JWT token and session data from client storage. | Token cleared; session terminated; user redirected; cannot access protected routes |
| **FR-1.4** | Profile Management | Users view and edit personal information (name, phone, email) and manage multiple delivery addresses with one set as default. | Email uniqueness maintained; changes reflected immediately; addresses saved correctly |

### 3.2 Product Browsing and Discovery

| **ID** | **Requirement** | **Description** | **Acceptance Criteria** |
| --- | --- | --- | --- |
| **FR-2.1** | View Product Catalog | Display all active products in grid/list format with images, names, prices, and ratings. Support pagination. | Products display accurately; out-of-stock marked; efficient loading |
| **FR-2.2** | Filter Products | Filter by category (Milk, Yogurt, Cheese, Butter, Cream, Ghee, Ice Cream), price range, fat content, brand, and availability. | Multiple filters apply simultaneously; dynamic updates; clear filters option |
| **FR-2.3** | Search Products | Search products by keywords with partial matching and relevance ranking (case-insensitive). | Partial matches included; relevant results; “no results” message when appropriate |
| **FR-2.4** | View Product Details | Display comprehensive product information: images, description, nutritional info, price, stock status, reviews, and related products. | All information accurate; reviews sorted by date; related products relevant |

### 3.3 Shopping Cart and Checkout

| **ID** | **Requirement** | **Description** | **Acceptance Criteria** |
| --- | --- | --- | --- |
| **FR-3.1** | Add to Cart | Users add products with selected quantity. System validates stock and updates cart with real-time totals. | Cannot exceed stock; cart persists for authenticated users; guest cart in localStorage |
| **FR-3.2** | View Cart | Display all cart items with images, names, prices, quantities, subtotals, and cart total. | Accurate information; correct price calculations; empty cart message |
| **FR-3.3** | Update Cart | Users modify item quantities or remove items. System recalculates totals in real-time. | Quantity cannot exceed stock; totals update instantly; validation messages |
| **FR-3.4** | Checkout Process | Authenticated users select delivery address, review order summary, and confirm payment method (COD). | Only authenticated users checkout; address required; validation checks |
| **FR-3.5** | Place Order | Create order in database, reduce stock, clear cart, generate confirmation with order ID. | Order stored correctly; inventory updated; cart cleared; confirmation sent |

### 3.4 Order Management (User)

| **ID** | **Requirement** | **Description** | **Acceptance Criteria** |
| --- | --- | --- | --- |
| **FR-4.1** | View Order History | Display all user orders sorted by date with order ID, date, status, total, and details button. | All orders displayed; chronological sorting; filter by status option |
| **FR-4.2** | Track Order Status | Display real-time order status with timeline: Pending → Confirmed → Packed → Out for Delivery → Delivered. | Status updates reflected immediately; clear timeline visualization; timestamps shown |
| **FR-4.3** | Cancel Order | Users cancel pending orders, restoring product stock and updating order status. | Only pending orders cancellable; stock restored; confirmation message |

### 3.5 Product Reviews and Ratings

| **ID** | **Requirement** | **Description** | **Acceptance Criteria** |
| --- | --- | --- | --- |
| **FR-5.1** | Submit Review | Users submit ratings (1-5 stars) and optional comments (10-500 characters) for products. | Rating required; one review per user per product; product average updated |
| **FR-5.2** | View Reviews | Display all product reviews with user names, ratings, comments, and dates. Show average rating. | Reviews accurate; average calculated correctly; sorted by date |
| **FR-5.3** | Edit/Delete Review | Users edit or delete their own reviews. System recalculates product ratings. | Only own reviews editable; ratings updated; deleted reviews unrecoverable |

### 3.6 Admin Authentication and Dashboard

| **ID** | **Requirement** | **Description** | **Acceptance Criteria** |
| --- | --- | --- | --- |
| **FR-6.1** | Admin Login | Admins log in with email and password. System generates JWT with admin role. | Only valid admin credentials grant access; admin routes protected; secure session |
| **FR-6.2** | Dashboard Overview | Display key metrics: total users, products, orders, pending orders, revenue, and recent orders list. | Metrics accurate and real-time; actionable insights provided |

### 3.7 Admin Product Management

| **ID** | **Requirement** | **Description** | **Acceptance Criteria** |
| --- | --- | --- | --- |
| **FR-7.1** | Add Product | Admin creates products with name, category, description, price, stock, images, brand, nutritional info, and shelf life. | All fields validated; product immediately available; images stored correctly |
| **FR-7.2** | Edit Product | Admin updates any product field including images. Changes reflect immediately in catalog. | Changes visible instantly; product ID unchanged; previous images retained unless replaced |
| **FR-7.3** | Delete Product | Admin removes products from catalog. Cannot delete products in pending orders. | Product hidden from catalog; pending orders protected; historical orders retain info |
| **FR-7.4** | Manage Stock | Admin updates inventory levels. Out-of-stock products automatically marked unavailable. | Stock accurate; unavailable products can’t be added to cart; immediate updates |

### 3.8 Admin Order Management

| **ID** | **Requirement** | **Description** | **Acceptance Criteria** |
| --- | --- | --- | --- |
| **FR-8.1** | View All Orders | Display all orders with filtering (status, date, customer) and sorting. Support pagination. | All orders visible; filters work correctly; pagination for large lists |
| **FR-8.2** | Update Order Status | Admin changes order status with forward-only progression. Status changes timestamped and logged. | Status only progresses forward; timestamps recorded; customer sees updates |
| **FR-8.3** | View Order Details | Display complete order information: customer details, delivery address, items, total, and status history. | All information complete; contact details accessible; status update controls available |

### 3.9 Admin User and Review Management

| **ID** | **Requirement** | **Description** | **Acceptance Criteria** |
| --- | --- | --- | --- |
| **FR-9.1** | View Users | Display all registered users with name, email, phone, registration date, status, and order count. | All users displayed; search functionality; accurate data |
| **FR-9.2** | Manage User Status | Admin activates or deactivates user accounts. Deactivated users cannot log in. | Status updates correctly; deactivated users blocked; reactivation possible |
| **FR-9.3** | Manage Reviews | Admin views all reviews and deletes inappropriate ones. Product ratings recalculate automatically. | All reviews accessible; deletion permanent; ratings recalculated; action logged |

## 4. Non-Functional Requirements

Non-functional requirements define how the system performs its functions, focusing on quality attributes.

### 4.1 Security

| **Requirement** | **Specification** | **Acceptance Criteria** |
| --- | --- | --- |
| **Authentication** | JWT-based with 24-hour expiration; bcrypt password hashing (10 salt rounds); secure token storage | Passwords never stored in plain text; tokens validated on protected routes; expiration enforced |
| **Data Protection** | HTTPS for all transmissions; environment variables for sensitive configs; role-based access control (RBAC) | SSL/TLS implemented; env vars used; roles enforced; no sensitive data in logs |
| **Input Validation** | Server-side validation for all inputs; XSS and NoSQL injection prevention; file upload restrictions (images only) | Injection attacks blocked; invalid data rejected; file types restricted |

### 4.2 Performance

| **Requirement** | **Specification** | **Acceptance Criteria** |
| --- | --- | --- |
| **Response Time** | Page load <3s on broadband; API response <500ms for 95% of requests; search results <1s | Measured response times meet thresholds; no noticeable lag |
| **Scalability** | Support 10,000 concurrent users; manage 100,000 products; handle 500 orders/day; stateless backend for horizontal scaling | Load testing demonstrates capacity; stable performance with large datasets |
| **Optimization** | Redux caching; static asset caching; code splitting and lazy loading; MongoDB query optimization with indexes | Repeat visits faster; reduced API calls; optimized bundle size |

### 4.3 Usability

| **Requirement** | **Specification** | **Acceptance Criteria** |
| --- | --- | --- |
| **UI Design** | Modern Tailwind CSS design; consistent colors, typography, spacing; visual feedback for actions; WCAG 2.1 Level AA compliance | Intuitive navigation; visually appealing; accessible to screen readers |
| **Responsive Design** | Support desktop (1920px+), laptop (1366-1919px), tablet (768-1365px), mobile (320-767px); touch-friendly on mobile | Renders correctly on all devices; no horizontal scrolling; appropriate touch targets (44x44px) |
| **Learnability** | Intuitive labels; tooltips for complex features; clear error messages; familiar e-commerce patterns | First-time users complete purchase in <5 minutes; max 3 clicks to add to cart |

### 4.4 Reliability

| **Requirement** | **Specification** | **Acceptance Criteria** |
| --- | --- | --- |
| **Availability** | 99.5% uptime (~3.65 hours downtime/month); scheduled maintenance during low-traffic hours; graceful degradation | Uptime monitored and meets target; planned downtime communicated; auto-recovery from transient errors |
| **Data Integrity** | ACID-compliant transactions where applicable; atomic inventory updates; daily database backups; dual validation (client & server) | No data corruption; accurate stock levels; no lost/duplicate orders; tested backup/restore |
| **Error Handling** | Try-catch blocks for critical operations; user-friendly error messages; server-side error logging; fallback UI; custom 404/500 pages | System never crashes; helpful error messages; errors logged for review; critical errors alert admins |

### 4.5 Maintainability

| **Requirement** | **Specification** | **Acceptance Criteria** |
| --- | --- | --- |
| **Code Quality** | React/Node.js best practices; modular components (single responsibility); meaningful naming; code comments; ESLint/Prettier formatting; DRY principle | Code passes linting; clear structure; understandable by new developers |
| **Documentation** | README with setup instructions; API endpoint documentation; database schema docs; component documentation | New developer can setup environment; all APIs documented with inputs/outputs; schema clearly documented |
| **Version Control** | Git with meaningful commits; main/development branching; regular commits; .gitignore for sensitive files | All code tracked in Git; clear commit history; no sensitive data in repository |

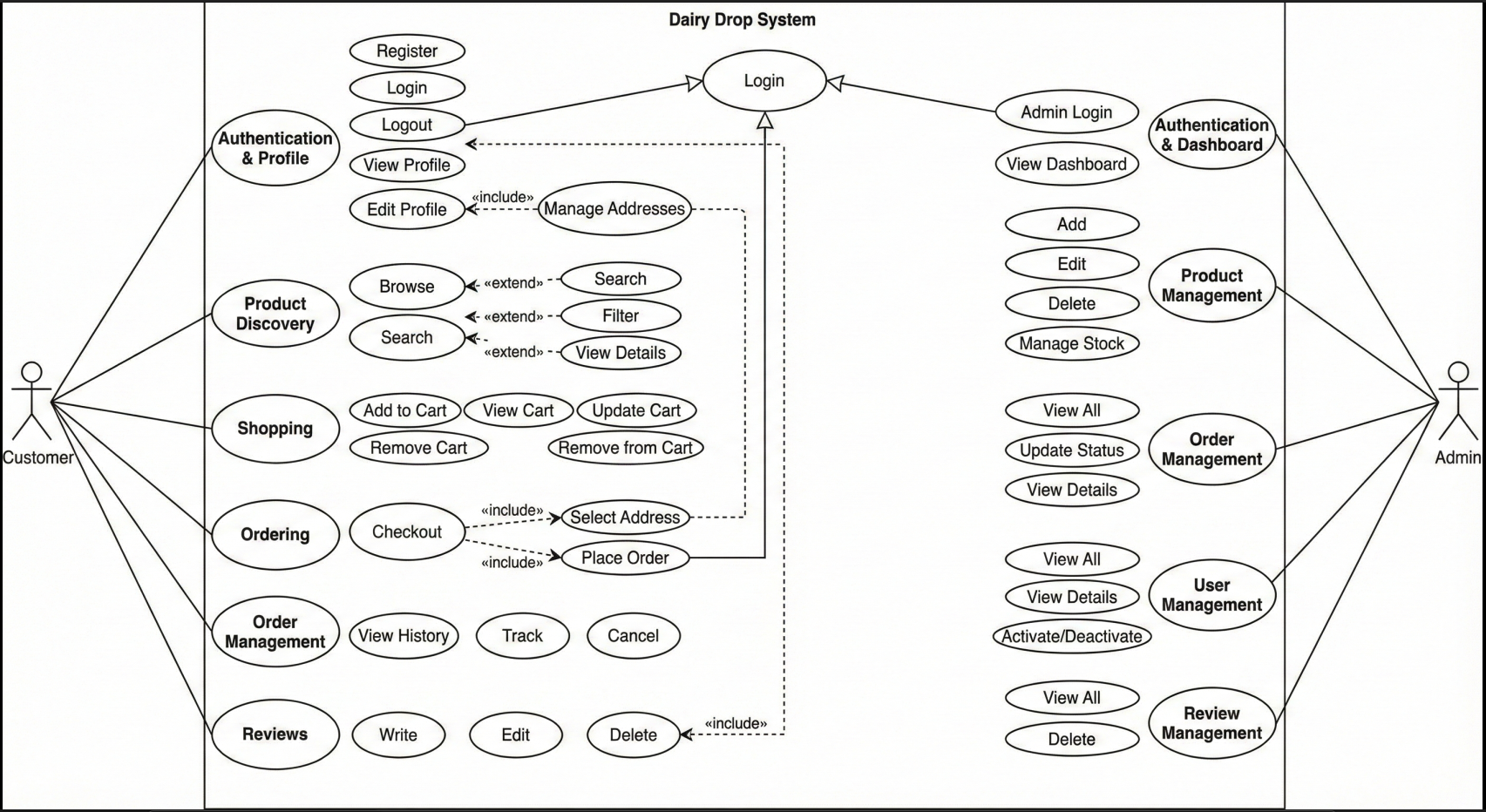
### 4.6 Compatibility

| **Requirement** | **Specification** | **Acceptance Criteria** |
| --- | --- | --- |
| **Browser Support** | Latest versions of Chrome, Firefox, Safari, Edge; graceful degradation for older versions; no IE 11 support | All features work in specified browsers; consistent UI rendering; no critical browser-specific bugs |
| **Device Support** | Desktop (Windows, macOS, Linux); tablets (iOS, Android); smartphones (iOS, Android); minimum 320px width | Core functionality works on all devices; UI adapts to device capabilities |

### 4.7 Standards and Compliance

| **Requirement** | **Specification** | **Acceptance Criteria** |
| --- | --- | --- |
| **Web Standards** | Valid HTML5, CSS3, ES6+ JavaScript; RESTful API principles; correct HTTP status codes; proper CORS; SEO best practices | HTML/CSS passes W3C validation; API follows REST conventions; basic SEO elements present |

## 5. Use Case Diagram

****

### 5.1 Introduction

A Use Case Diagram visually represents system functionality by showing interactions between actors (users) and use cases (functional requirements). For Dairy Drop, it illustrates what the system does from a user’s perspective.

### 5.2 Actors

1. **Customer (User):** Primary actor who browses and purchases dairy products (registered and guest users)
2. **Admin (System Administrator):** Manages platform including products, orders, and users with elevated privileges

### 5.3 Use Cases

#### **Customer Use Cases (23):**

**Authentication & Profile:** 1. Register Account 2. Login 3. Logout 4. View Profile 5. Edit Profile 6. Manage Addresses

**Product Discovery:** 7. Browse Products 8. Search Products 9. Filter Products 10. View Product Details

**Shopping:** 11. Add to Cart 12. View Cart 13. Update Cart 14. Remove from Cart

**Ordering:** 15. Checkout 16. Select Delivery Address 17. Place Order

**Order Management:** 18. View Order History 19. Track Order 20. Cancel Order

**Reviews:** 21. Write Review 22. Edit Review 23. Delete Review

#### **Admin Use Cases (15):**

**Authentication & Dashboard:** 1. Admin Login 2. View Dashboard

**Product Management:** 3. Add Product 4. Edit Product 5. Delete Product 6. Manage Stock

**Order Management:** 7. View All Orders 8. Update Order Status 9. View Order Details

**User Management:** 10. View All Users 11. View User Details 12. Activate/Deactivate User

**Review Management:** 13. View All Reviews 14. Delete Review

## 6. Work Plan

### 6.1 Gantt Chart

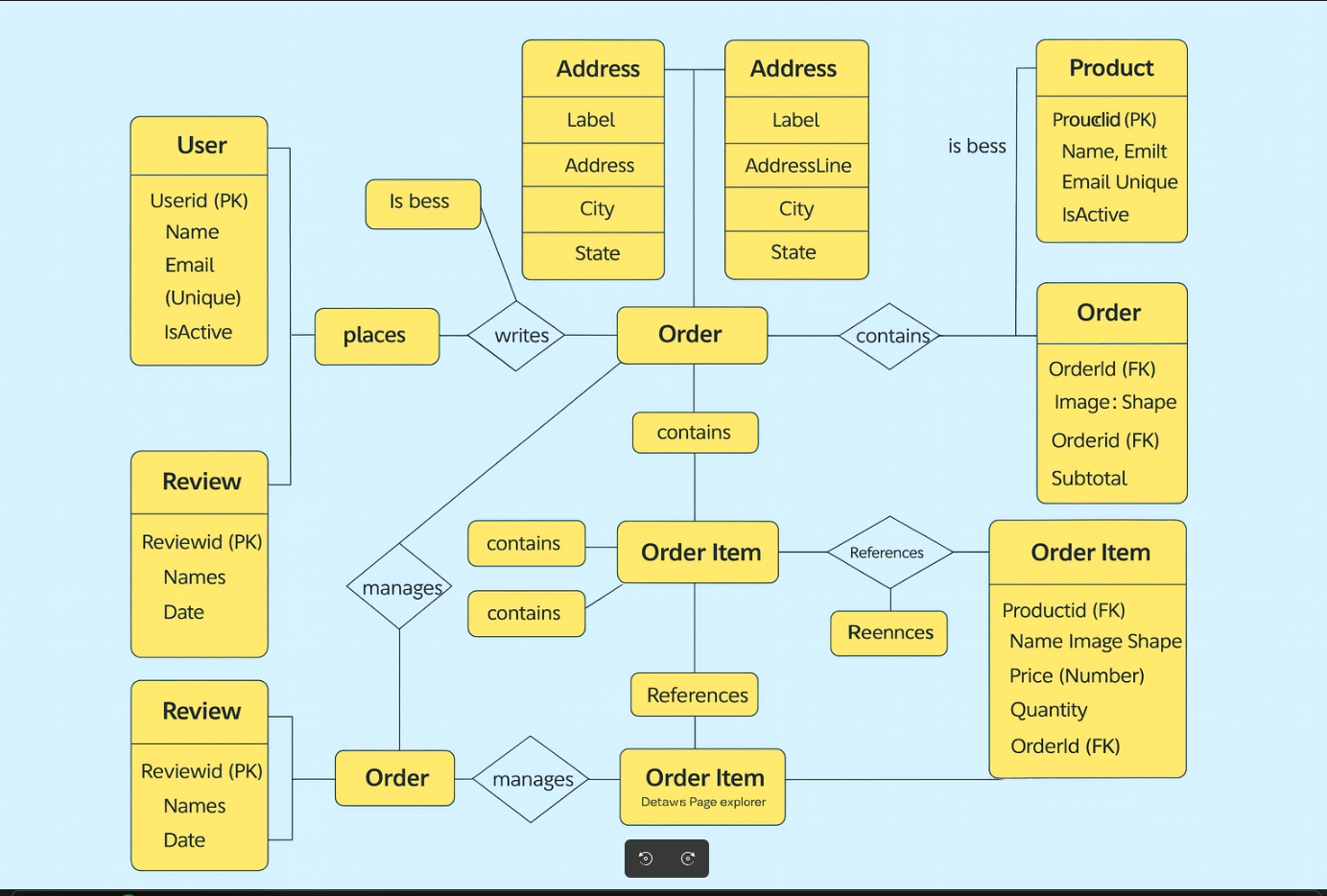
| **Task/Phase** | **Duration** | **Start Date** | **End Date** | **Deliverables** |
| --- | --- | --- | --- | --- |
| **Phase 1: Planning & Requirements** | 1 week | 01-Nov-2025 | 07-Nov-2025 | Requirements document, Use Case Diagram, ERD |
| - Requirement gathering | 3 days | 01-Nov-2025 | 03-Nov-2025 | Functional & non-functional requirements |
| - System design planning | 2 days | 04-Nov-2025 | 05-Nov-2025 | System architecture document |
| - Use case and ERD creation | 2 days | 06-Nov-2025 | 07-Nov-2025 | Use Case Diagram, ERD |
| **Phase 2: UI/UX Design** | 1 week | 08-Nov-2025 | 14-Nov-2025 | Wireframes, mockups, design system |
| - Wireframing | 4 days | 08-Nov-2025 | 11-Nov-2025 | Low-fidelity wireframes |
| - UI component design | 3 days | 12-Nov-2025 | 14-Nov-2025 | High-fidelity mockups, style guide |
| **Phase 3: Backend Development** | 2 weeks | 15-Nov-2025 | 28-Nov-2025 | Fully functional backend API |
| - Project setup | 1 day | 15-Nov-2025 | 15-Nov-2025 | Node.js + Express + MongoDB setup |
| - Database schema & models | 2 days | 16-Nov-2025 | 17-Nov-2025 | Mongoose models |
| - Authentication (JWT + bcrypt) | 3 days | 18-Nov-2025 | 20-Nov-2025 | Register, login, auth middleware |
| - User & product APIs | 3 days | 21-Nov-2025 | 23-Nov-2025 | CRUD endpoints |
| - Cart & order APIs | 3 days | 24-Nov-2025 | 26-Nov-2025 | Cart management, order placement |
| - Review & admin APIs | 2 days | 27-Nov-2025 | 28-Nov-2025 | Review submission, admin management |
| **Phase 4: Frontend Development** | 2.5 weeks | 29-Nov-2025 | 16-Dec-2025 | Fully functional frontend |
| - React setup & routing | 1 day | 29-Nov-2025 | 29-Nov-2025 | React + React Router |
| - Redux Toolkit setup | 2 days | 30-Nov-2025 | 01-Dec-2025 | Redux store, slices, API integration |
| - Authentication UI | 2 days | 02-Dec-2025 | 03-Dec-2025 | Login/Register pages |
| - Product pages | 4 days | 04-Dec-2025 | 07-Dec-2025 | Catalog, filters, details |
| - Cart & checkout | 4 days | 08-Dec-2025 | 11-Dec-2025 | Cart page, checkout flow |
| - Profile & orders | 3 days | 12-Dec-2025 | 14-Dec-2025 | Profile, order history, tracking |
| - Admin dashboard | 2 days | 15-Dec-2025 | 16-Dec-2025 | Admin management UI |
| **Phase 5: Integration & Testing** | 1.5 weeks | 17-Dec-2025 | 26-Dec-2025 | Tested, debugged application |
| - Frontend-backend integration | 3 days | 17-Dec-2025 | 19-Dec-2025 | Full stack integration |
| - Functional testing | 3 days | 20-Dec-2025 | 22-Dec-2025 | Test features, identify bugs |
| - Bug fixes | 3 days | 23-Dec-2025 | 25-Dec-2025 | Resolve issues |
| - Performance optimization | 1 day | 26-Dec-2025 | 26-Dec-2025 | Optimize load times |
| **Phase 6: Documentation & Deployment** | 6 days | 27-Dec-2025 | 01-Feb-2026 | Deployed app, final documentation |
| - Final documentation | 2 days | 27-Dec-2025 | 28-Dec-2025 | Complete project docs |
| - Deployment | 2 days | 29-Dec-2025 | 30-Dec-2025 | Live application |
| - Production testing | 1 day | 31-Dec-2025 | 31-Dec-2025 | Production validation |
| - Presentation prep | 1 day | 01-Feb-2026 | 01-Feb-2026 | Slides and demo |

### 6.2 Milestones

| **Milestone** | **Date** | **Criteria** |
| --- | --- | --- |
| Requirements Finalized | 07-Nov-2025 | Approved requirements and design documents |
| UI/UX Design Completed | 14-Nov-2025 | Approved mockups and design system |
| Backend API Completed | 28-Nov-2025 | All APIs functional and tested (Postman) |
| Frontend Completed | 16-Dec-2025 | All UI components functional |
| Integration & Testing Done | 26-Dec-2025 | Bug-free, fully integrated application |
| Production Deployment | 30-Dec-2025 | Application live and accessible |
| Final Submission | 01-Feb-2026 | Documentation and presentation ready |

## 7. Entity Relationship Diagram (ERD)

### 7.1 Introduction



### 7.2 Entities and Attributes

#### **7.2.1 User Entity**

| **Attribute** | **Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| userId (PK) | ObjectId | Unique identifier | Primary Key, Auto-generated |
| name | String | Full name | Required, 2-100 chars |
| email | String | Email address | Required, Unique, Valid format |
| phone | String | Phone number | Required, Unique, 10-15 digits |
| password | String | Hashed password | Required, Bcrypt hashed |
| addresses | Array | Delivery addresses | Optional, embedded sub-documents |
| isActive | Boolean | Account status | Default: true |
| createdAt | Date | Creation timestamp | Auto-generated |
| updatedAt | Date | Update timestamp | Auto-generated |

**Address Sub-Document:** addressLine1, addressLine2, city, state, postalCode, phone, label, isDefault

#### **7.2.2 Product Entity**

| **Attribute** | **Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| productId (PK) | ObjectId | Unique identifier | Primary Key, Auto-generated |
| name | String | Product name | Required, 2-200 chars |
| category | String | Product category | Required, Enum (Milk, Yogurt, Cheese, Butter, Cream, Ghee, Ice Cream, Other) |
| description | String | Description | Required, up to 1000 chars |
| price | Number | Price | Required, Positive |
| stock | Number | Available quantity | Required, Non-negative |
| images | Array | Image URLs | Required, at least 1 |
| brand | String | Brand name | Optional |
| nutritionalInfo | Object | Nutritional details | Optional |
| shelfLife | String | Shelf life | Optional |
| averageRating | Number | Average rating | Default: 0, 0-5 |
| totalReviews | Number | Review count | Default: 0 |
| isActive | Boolean | Availability | Default: true |
| createdAt | Date | Creation timestamp | Auto-generated |
| updatedAt | Date | Update timestamp | Auto-generated |

#### **7.2.3 Order Entity**

| **Attribute** | **Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| orderId (PK) | ObjectId | Unique identifier | Primary Key, Auto-generated |
| userId (FK) | ObjectId | Reference to User | Required, Foreign Key |
| products | Array | Ordered products | Required, embedded sub-documents |
| deliveryAddress | Object | Delivery address | Required, embedded |
| totalAmount | Number | Total order amount | Required, Positive |
| status | String | Order status | Required, Enum (Pending, Confirmed, Packed, Out for Delivery, Delivered, Cancelled) |
| paymentMethod | String | Payment method | Default: “Cash on Delivery” |
| orderDate | Date | Order timestamp | Auto-generated |
| statusHistory | Array | Status change logs | Auto-generated |
| updatedAt | Date | Update timestamp | Auto-generated |

**Product Item Sub-Document:** productId, name, price, quantity, subtotal

#### **7.2.4 Review Entity**

| **Attribute** | **Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| reviewId (PK) | ObjectId | Unique identifier | Primary Key, Auto-generated |
| userId (FK) | ObjectId | Reference to User | Required, Foreign Key |
| productId (FK) | ObjectId | Reference to Product | Required, Foreign Key |
| rating | Number | Star rating | Required, Integer 1-5 |
| comment | String | Review text | Optional, up to 500 chars |
| createdAt | Date | Creation timestamp | Auto-generated |
| updatedAt | Date | Update timestamp | Auto-generated |

#### **7.2.5 Admin Entity**

| **Attribute** | **Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| adminId (PK) | ObjectId | Unique identifier | Primary Key, Auto-generated |
| name | String | Admin name | Required |
| email | String | Email address | Required, Unique, Valid format |
| password | String | Hashed password | Required, Bcrypt hashed |
| role | String | Admin role | Default: “admin” |
| createdAt | Date | Creation timestamp | Auto-generated |
| updatedAt | Date | Update timestamp | Auto-generated |

### 7.3 Entity Relationships

| **Relationship** | **Type** | **Description** |
| --- | --- | --- |
| User ↔ Order | 1:N | One user can place many orders |
| Order ↔ Product | M:N | Orders contain multiple products (embedded in Order) |
| User ↔ Review | 1:N | One user can write many reviews |
| Product ↔ Review | 1:N | One product can have many reviews |
| Admin ↔ Product | 1:N | Admin manages products (implicit) |
| Admin ↔ Order | 1:N | Admin manages orders (implicit) |

### 7.4 ERD Visual Representation

**Note:** Create the ERD using draw.io, Lucidchart, Microsoft Visio, or similar tools showing: - 5 entity rectangles (User, Product, Order, Review, Admin) - Attributes listed in each entity - Primary Keys (PK) and Foreign Keys (FK) marked - Relationship lines with cardinality (1:N, M:N)

## 8. Database Design

### 8.1 Database Technology

**MongoDB** - NoSQL document database chosen for flexible schema, scalability, performance, native JSON storage, and seamless MERN stack integration.

### 

#### **8.2.1 Users Collection**

{  
 \_id: ObjectId,  
 name: String (required, 2-100 chars),  
 email: String (required, unique, valid format),  
 phone: String (required, unique, 10-15 digits),  
 password: String (required, bcrypt hashed),  
 addresses: [  
 {  
 addressLine1: String (required),  
 addressLine2: String,  
 city: String (required),  
 state: String (required),  
 postalCode: String (required),  
 phone: String (required),  
 label: String (default: "Home"),  
 isDefault: Boolean (default: false)  
 }  
 ],  
 isActive: Boolean (default: true),  
 createdAt: Date,  
 updatedAt: Date  
}

**Indexes:** email (unique), phone (unique), createdAt

**Example:**

{  
 "\_id": "507f1f77bcf86cd799439011",  
 "name": "Ahmad Ali",  
 "email": "ahmad.ali@example.com",  
 "phone": "03001234567",  
 "password": "$2b$10$hashedpassword...",  
 "addresses": [  
 {  
 "addressLine1": "House 123, Street 5",  
 "city": "Islamabad",  
 "state": "ICT",  
 "postalCode": "44000",  
 "phone": "03001234567",  
 "label": "Home",  
 "isDefault": true  
 }  
 ],  
 "isActive": true,  
 "createdAt": "2025-11-01T10:30:00Z"  
}

#### **8.2.2 Products Collection**

{  
 \_id: ObjectId,  
 name: String (required, 2-200 chars),  
 category: String (required, enum),  
 description: String (required, max 1000 chars),  
 price: Number (required, min 0),  
 stock: Number (required, min 0),  
 images: [String] (required, min 1),  
 brand: String,  
 nutritionalInfo: {  
 calories: Number,  
 fat: String,  
 protein: String,  
 carbohydrates: String,  
 calcium: String  
 },  
 shelfLife: String,  
 averageRating: Number (default: 0, 0-5),  
 totalReviews: Number (default: 0),  
 isActive: Boolean (default: true),  
 createdAt: Date,  
 updatedAt: Date  
}

**Indexes:** category, price, averageRating, name (text), isActive

**Example:**

{  
 "\_id": "507f1f77bcf86cd799439012",  
 "name": "Organic Whole Milk 1L",  
 "category": "Milk",  
 "description": "Fresh organic whole milk from grass-fed cows.",  
 "price": 250,  
 "stock": 150,  
 "images": ["/uploads/organic-milk.jpg"],  
 "brand": "Pure Dairy Farms",  
 "nutritionalInfo": {  
 "calories": 150,  
 "fat": "8g",  
 "protein": "8g"  
 },  
 "averageRating": 4.5,  
 "totalReviews": 28,  
 "isActive": true  
}

#### **8.2.3 Orders Collection**

{  
 \_id: ObjectId,  
 userId: ObjectId (ref: users, required),  
 products: [  
 {  
 productId: ObjectId (ref: products, required),  
 name: String (required),  
 price: Number (required),  
 quantity: Number (required, min 1),  
 subtotal: Number (required)  
 }  
 ],  
 deliveryAddress: {  
 addressLine1: String (required),  
 city: String (required),  
 state: String (required),  
 postalCode: String (required),  
 phone: String (required)  
 },  
 totalAmount: Number (required, min 0),  
 status: String (required, enum),  
 paymentMethod: String (default: "COD"),  
 statusHistory: [  
 {  
 status: String (required),  
 timestamp: Date,  
 updatedBy: String  
 }  
 ],  
 orderDate: Date,  
 updatedAt: Date  
}

**Indexes:** userId, status, orderDate, compound {userId: 1, orderDate: -1}

**Example:**

{  
 "\_id": "507f1f77bcf86cd799439013",  
 "userId": "507f1f77bcf86cd799439011",  
 "products": [  
 {  
 "productId": "507f1f77bcf86cd799439012",  
 "name": "Organic Whole Milk 1L",  
 "price": 250,  
 "quantity": 2,  
 "subtotal": 500  
 }  
 ],  
 "deliveryAddress": {  
 "addressLine1": "House 123, Street 5",  
 "city": "Islamabad",  
 "state": "ICT",  
 "postalCode": "44000",  
 "phone": "03001234567"  
 },  
 "totalAmount": 500,  
 "status": "Delivered",  
 "paymentMethod": "Cash on Delivery",  
 "orderDate": "2025-12-03T10:00:00Z"  
}

#### **8.2.4 Reviews Collection**

{  
 \_id: ObjectId,  
 userId: ObjectId (ref: users, required),  
 productId: ObjectId (ref: products, required),  
 rating: Number (required, 1-5),  
 comment: String (max 500 chars),  
 createdAt: Date,  
 updatedAt: Date  
}

**Indexes:** productId, userId, compound {productId: 1, createdAt: -1}, unique compound {userId: 1, productId: 1}

**Example:**

{  
 "\_id": "507f1f77bcf86cd799439015",  
 "userId": "507f1f77bcf86cd799439011",  
 "productId": "507f1f77bcf86cd799439012",  
 "rating": 5,  
 "comment": "Excellent quality milk!",  
 "createdAt": "2025-12-02T16:20:00Z"  
}

#### **8.2.5 Admins Collection**

{  
 \_id: ObjectId,  
 name: String (required),  
 email: String (required, unique, valid format),  
 password: String (required, bcrypt hashed),  
 role: String (default: "admin"),  
 createdAt: Date,  
 updatedAt: Date  
}

**Indexes:** email (unique)

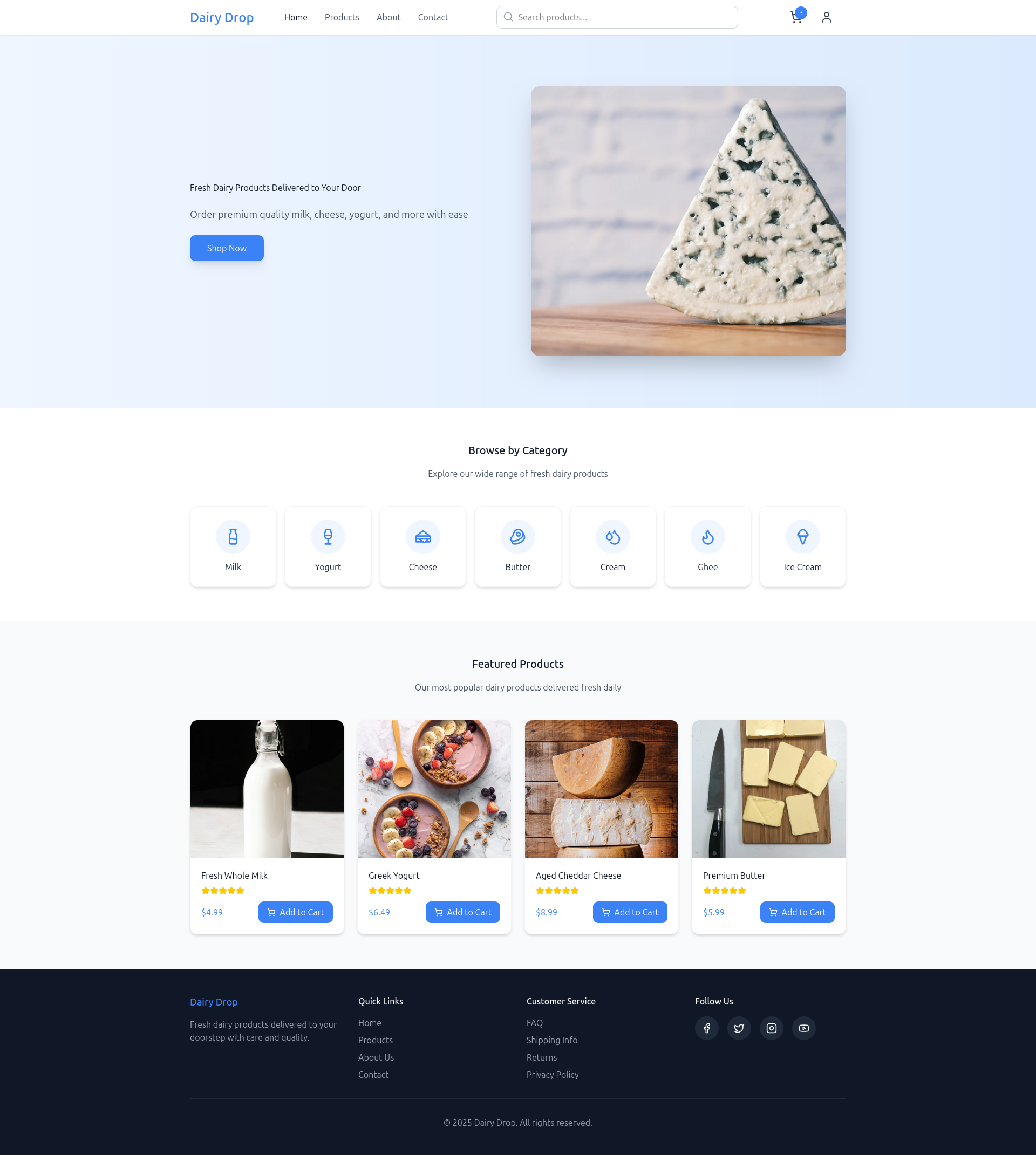
**Example:**

{  
 "\_id": "507f1f77bcf86cd799439016",  
 "name": "Admin User",  
 "email": "admin@dairydrop.com",  
 "password": "$2b$10$hashedpassword...",  
 "role": "admin"  
}

### 8.3 Database Implementation Notes

* **References:** Orders and Reviews reference Users and Products via ObjectId
* **Embedded Documents:** User addresses, order products, delivery addresses embedded for efficiency
* **Denormalization:** Order products snapshot data (name, price) to preserve historical accuracy
* **Validation:** Schema-level validation using Mongoose; unique, enum, range, and format constraints
* **Security:** Connection string in env vars; limited privileges; IP whitelist; TLS/SSL encryption

## 9. Interface Design

****

### 

### 9.1 Design Principles

* **Clean & Minimalist:** Focused, uncluttered design
* **Consistent:** Uniform colors, typography, spacing
* **Responsive:** Desktop, tablet, mobile support
* **Accessible:** WCAG 2.1 Level AA compliance
* **User-Centric:** Intuitive navigation, clear CTAs

### 9.2 Technology

* **Styling:** Tailwind CSS
* **Design Tools:** Figma / Adobe XD
* **Icons:** Heroicons / Font Awesome
* **Typography:** Inter, Poppins, or Roboto

### 9.3 Color Scheme

**Primary:** Blue (#3B82F6), Dark Blue (#1E40AF) **Secondary:** Green (#10B981), Red (#EF4444), Yellow (#F59E0B) **Neutral:** Dark Gray (#1F2937), Medium Gray (#6B7280), Light Gray (#F3F4F6), White (#FFFFFF)

### 9.4 Key Interface Screens

#### **9.4.1 Home Page**

* **Navigation Bar:** Logo, menu (Home, Products, About, Contact), search bar, cart icon with badge, user account
* **Hero Section:** Banner image, headline “Fresh Dairy Products Delivered to Your Door”, CTA “Shop Now”
* **Category Section:** Grid of category cards (Milk, Yogurt, Cheese, Butter, Cream, Ghee, Ice Cream)
* **Featured Products:** Product cards with image, name, price, rating, “Add to Cart” button
* **Footer:** Quick links, social media, copyright

#### **9.4.2 Product Listing Page**

* **Sidebar Filters:** Category, price range, brand, rating, availability; “Clear All” button
* **Main Area:** Sort dropdown, product grid (3-4 columns), pagination
* **Product Cards:** Image, name, category badge, price, rating, stock status, “Add to Cart”

#### **9.4.3 Product Details Page**

* **Left:** Image gallery with zoom, thumbnails
* **Right:** Name, brand, rating, price, stock status, description, quantity selector, “Add to Cart”
* **Tabs:** Description, Nutritional Info, Reviews
* **Related Products:** Horizontal scrollable list

#### **9.4.4 Shopping Cart Page**

* **Cart Items:** Table/card layout with image, name, price, quantity selector, subtotal, remove button
* **Cart Summary:** Subtotal, delivery charges, total, “Proceed to Checkout” button
* **Empty State:** “Your cart is empty” with “Start Shopping” button

#### **9.4.5 Checkout Page**

* **Progress:** Cart → Address → Review → Confirmation
* **Sections:** Delivery address selection/add, order summary, payment method (COD), “Place Order” button

#### **9.4.6 Order Confirmation Page**

* **Success Message:** Checkmark icon, “Order Placed Successfully!”, order ID
* **Summary:** Items, address, total, estimated delivery
* **Actions:** “Track Order”, “Continue Shopping”

#### **9.4.7 User Profile Page**

* **Sidebar Menu:** Profile Info, My Orders, Saved Addresses, Change Password, Logout
* **Main Area:** Dynamic content based on selection (profile edit, order list, address management, password change)

#### **9.4.8 Order Tracking Page**

* **Header:** Order ID, date, current status
* **Timeline:** Visual stepper (Pending → Confirmed → Packed → Out for Delivery → Delivered)
* **Details:** Items, address, total
* **Actions:** “Cancel Order” (if pending), “Contact Support”

#### **9.4.9 Admin Dashboard**

* **Metrics Cards:** Total users, products, orders, pending orders, revenue
* **Recent Orders Table:** Order details with “View” button
* **Alerts:** Low stock products
* **Quick Actions:** “Add Product”, “View Pending Orders”

#### **9.4.10 Admin Product Management**

* **Header:** “Product Management”, “Add New Product” button
* **Table:** Image, name, category, price, stock (color-coded), status toggle, edit/delete actions
* **Filters:** Search, category, status
* **Add/Edit Form:** All product fields with image upload

#### **9.4.11 Admin Order Management**

* **Header:** “Order Management”
* **Table:** Order ID, customer, date, status (color-coded), total, “View Details”
* **Filters:** Status, date range, search
* **Details Modal:** Customer info, address, items, status update dropdown

### 9.5 Responsive Breakpoints

* **Desktop (1920px+):** Full layout, multi-column grids
* **Laptop (1366-1919px):** Condensed layout, 3-column grids
* **Tablet (768-1365px):** Collapsible sidebars, 2-column grids
* **Mobile (320-767px):** Single column, hamburger menu, stacked cards

### 9.6 Interface Mockups

**Note:** Visual mockups for the above interfaces will be created using Figma/Adobe XD and inserted as images in the final document. Mockups will demonstrate layout, colors, typography, and interactive elements as described.

## Conclusion

This comprehensive documentation for the **Dairy Drop** project provides a complete blueprint for developing a modern, secure, and user-friendly e-commerce platform for dairy products using the **MERN stack** (MongoDB, Express.js, React, Node.js).

**End of Documentation**