Project Proposal

1- Project Overview

TechXpress is a scalable and maintainable e-commerce platform designed to facilitate the online sale of electronic products. The system will enable customers to browse, search, and purchase products while allowing administrators to manage inventory, orders, and user roles efficiently.

1.1 Objectives

- Develop a feature-rich e-commerce website using **ASP.NET Core MVC**.
- Implement **NTier Architecture, Repository Pattern, and Unit of Work** to ensure separation of concerns and maintainability.
- Integrate **Stripe** for secure online payments.
- Deploy the application on **Microsoft Azure** for scalability and reliability.

2. Scope

2.1 Functional Scope

The system will provide the following core functionalities:

For Customers:

- Browse products by category and apply filters.
- Add products to the shopping cart and modify quantities.
- Securely complete purchases using Stripe.
- View order history and track order status.
- Manage user profiles and account details.

For Administrators:

- Manage products, categories, and inventory.
- Oversee and update order statuses.
- Access sales reports and analytics.
- Control user roles and permissions.

2.2 Non-Functional Scope

- **Performance:** Ensure search results load within **2 seconds** under normal load conditions.

- **Security:** Implement **AES-256 encryption**, HTTPS, and prevent SQL injection & XSS attacks.
- **Scalability:** The system should support future growth with horizontal scaling.
- **Usability:** A responsive and user-friendly interface designed with Bootstrap and JQuery.

3. Technologies and Tools

- **Backend:** ASP.NET Core MVC, Entity Framework Core.
- **Frontend:** HTML, CSS, JavaScript, JQuery, Bootstrap.
- **Database:** Microsoft SQL Server (hosted on Azure SQL Database).
- **Authentication:** ASP.NET Identity with role-based access control (RBAC).
- **Payment Gateway:** Stripe API.
- **Hosting:** Microsoft Azure / Hostinger.

4. Project Plan

The project will follow an **Agile development** approach with weekly sprints. Regular progress reviews will ensure timely delivery of features.

Week	Tasks
Week 1	Project setup, database design, authentication system
Week 2	Shopping cart, role-based access control, admin panel
Week 3	Order processing, Stripe integration, user profile management
Week 4	UI enhancements, testing, and deployment

5. Task Assignment & Roles

- **Project Manager:** Oversees project timeline, task assignments, and risk mitigation.
- **Frontend Developer:** Designs and implements the UI using Bootstrap and JQuery.
- **Backend Developer:** Develops business logic, database interactions, and APIs.
- **Database Administrator:** Manages database design and optimizations.
- **QA Engineer:** Conducts testing for security, performance, and usability.

6. Risk Assessment & Mitigation Plan

Risk	Impact	Mitigation Strategy

Risk	Impact	Mitigation Strategy
Payment Gateway Issues	High	Implement fallback payment options and thorough testing
Security Vulnerabilities	High	Conduct regular security audits and apply best security practices
Performance Bottlenecks	Medium	Optimize database queries and implement caching strategies
Scope Creep	Medium	Strict requirement management and stakeholder alignment
Deployment Failures	Low	Maintain detailed deployment documentation and CI/CD pipelines

7. KPIs (Key Performance Indicators)

- **System Performance:** Response time for product searches under **2 seconds**.
- **Transaction Success Rate:** Minimum **98%** successful transactions via Stripe.
- **User Engagement:** Increase in returning users by **30%** within 6 months.
- **Error Rate:** Less than **1%** critical system failures post-deployment.
- **Scalability:** System should handle at least **100 concurrent users** without degradation.

8. Assumptions and Constraints

- The project will be completed within **4 weeks**.
- The system will be hosted on **Microsoft Azure**.
- Integration with external email services (e.g., SendGrid) is **not required** at this stage.
- The Stripe payment gateway is assumed to be functional within the development environment.

9. Deliverables

- **Fully functional e-commerce platform** with authentication and role-based access.
- **Admin dashboard** for product and order management.
- **Secure payment processing** via Stripe.
- **Deployment on Microsoft Azure** with complete documentation.

Project Plan

Develop a scalable and maintainable e-commerce platform for selling electronics using ASP.NET Core.

It will implement design patterns like NTier Architecture, Repository Pattern, and Unit of Work.

Key Features:

- User-friendly browsing and shopping experience.
- Integrated payment system (Stripe).
- Admin panel for product, category, and order management.
- Role-based access control.
- Deployment on Microsoft Azure or Hostinger.

Technologies & Tools

- Backend: ASP.NET Core MVC, Entity Framework Core, ASP.NET Identity
- Frontend: JQuery, DataTables, Bootstrap, Toaster JS
- Payment Integration: Stripe
- Deployment: Microsoft Azure/Hostinger

Design Patterns

1. NTier Architecture: Presentation, Business Logic, and Data Access layers.
2. Repository Pattern: Abstracts data access.
3. Unit of Work Pattern: Ensures atomic transactions.
4. Dependency Injection: Promotes loose coupling and testability.

4-Week Development Plan
Week 1: Initial Setup & Product Listings

Tasks:

- Set up the NTier architecture:
  - Presentation Layer: ASP.NET MVC with views, controllers, JQuery, DataTables.
  - Business Logic Layer: Manage business rules (e.g., product addition, order calculations).
  - Data Access Layer: Implement repositories and Unit of Work for data management.
- Database Design: Use Entity Framework Core to define entities like Products, Categories, Orders

- Authentication Setup: Integrate ASP.NET Identity for registration, login, and roles.

Deliverables:

- Complete NTier setup with functional layers.
- Basic product listing page with search and filters.
- Implemented database schema with repository and Unit of Work patterns.
- User authentication system.

Week 2: Shopping Cart, Role-Based Access, and Admin Panel Tasks:

- Shopping Cart: Use Session State to store and manage cart data.
- Role Management: Define Customer and Admin roles with restricted access.
- Admin Dashboard: Implement product, category, and order management using DataTables.

Deliverables:

- Working shopping cart with product addition and review features.
- Role-based access control system.
- Admin dashboard with CRUD operations for products and orders.

Week 3: Order Placement, Payments, and User Profiles

Tasks:

- Order Placement: Ensure atomic transactions for order processing.
- Stripe Integration: Implement secure payment processing.
- User Profiles: Enable users to view order history and manage profiles.

Deliverables:

- Integrated order placement with Stripe payments.
- Functional customer profile and order history section.
- Thoroughly tested transaction process.

Week 4: UI Enhancements, Testing, and Deployment Tasks:

- UI Enhancements: Polish the UI using Bootstrap and custom CSS.
- DataTables Enhancements: Enable sorting, filtering, and pagination for admin management.
- Final Testing: Conduct end-to-end testing for the entire system.
- Deployment: Deploy to Microsoft Azure/Hostinger and finalize documentation.

Deliverables:

- Responsive and polished UI for frontend and admin panels.
- DataTables functionalities fully implemented.
- Fully tested and deployed application.
- Completed documentation for setup and usage.

Risk Management

- Delays in Payment Integration: Start early and ensure multiple testing phases.
- Deployment Issues: Prepare contingency plans with detailed documentation and testing.
- Role-Based Access Bugs: Conduct rigorous testing for various role scenarios.

Next Steps

- Set up the initial development environment.
- Define the database schema and relationships.
- Break down tasks into detailed sprints for better tracking.

Roles and Responsibilities

Backend Developer

- Implement NTier Architecture (Presentation, Business Logic, Data Access layers).
- Design database schema using Entity Framework Core.
- Develop generic/specific repositories and Unit of Work pattern.
- Integrate Stripe payment gateway and ensure atomic transactions.
- Set up ASP.NET Identity for authentication and role management (RBAC).
- Handle shopping cart business logic (totals, promotions) and session management.

Frontend Developer

- Create responsive UI using Bootstrap, custom CSS, and JQuery.
- Develop product listings with search/filters and integrate DataTables for admin panels.
- Implement dynamic elements (live search, interactive features).
- Design admin dashboard layouts and user profile pages.

Full-Stack Developer

- Bridge frontend and backend for shopping cart integration.
- Develop user profile pages with order history and account management.
- Assist in connecting UI components (e.g., admin dashboard) with backend repositories.

DevOps Engineer

- Deploy the application to Microsoft Azure/Hostinger.
- Configure CI/CD pipelines (if applicable).
- Ensure scalability and monitor performance post-deployment.

QA Tester

- Test shopping cart functionality, payment flows, and RBAC permissions.
- Validate end-to-end user journeys (registration to order placement).
- Perform stress testing and ensure cross-browser compatibility.

Project Manager

- Oversee timelines and deliverables for each week.
- Coordinate collaboration between roles (e.g., backend-frontend integration).
- Finalize project documentation (architecture, user manuals).

Weekly Task Breakdown

Week 1

Backend Developer: NTier setup, EF Core schema design, repository/Unit of Work implementation.
Frontend Developer: Basic product listing page with search/filters.
Backend/Security Developer: ASP.NET Identity setup for user authentication.

Week 2

Backend Developer: Shopping cart logic, RBAC implementation.
Frontend Developer: Admin dashboard UI with DataTables.
Full-Stack Developer: Integrate cart UI with backend session management.

Week 3

Backend Developer: Order placement logic, Stripe API integration.
Full-Stack Developer: User profile page and order history integration.
QA Tester: Begin testing payment processing and order workflows.

Week 4

Frontend Developer: Polish UI with Bootstrap and JQuery enhancements.
DevOps Engineer: Deploy to Azure/Hostinger.
QA Tester: Conduct final testing (functionality, responsiveness).
Project Manager: Compile documentation and ensure all deliverables are met.

Collaboration Points

- Backend + Frontend Developers: Admin dashboard integration (DataTables with repositories).
- Backend + Frontend Developers: Shopping cart UI-business logic synchronization.
- Full-Stack + Backend Developers: User profile updates and order history data retrieval.
- Project Manager + All Roles: Weekly sync-ups to track progress and resolve blockers.

Risk Assessment & Mitigation Plan

Risk 1: Poor system architecture leading to performance issues.

- *Mitigation:* Follow NTier Architecture, Repository Pattern, and Unit of Work to ensure a scalable and maintainable codebase.
  Conduct regular code reviews and optimizations.

Risk 2: Security vulnerabilities in authentication and payment integration.

- *Mitigation:* Use ASP.NET Identity for secure authentication and role-based access control. Implement security best practices such as password hashing, multi-factor authentication, and OAuth. Secure payment transactions with Stripe's recommended security measures.

Risk 3: Database integrity issues and data loss.

- *Mitigation:* Use Entity Framework Core with Unit of Work to ensure atomic transactions. Implement regular database backups and enforce data validation rules.

Risk 4: Compatibility and responsiveness issues across devices.

- *Mitigation:* Utilize Bootstrap for a responsive UI and conduct cross-browser/device testing throughout development.

Risk 5: Poor session management affecting shopping cart functionality.

- *Mitigation:* Use Session State for cart storage and implement session expiration strategies to prevent stale data.

Risk 6: Ineffective role-based access control (RBAC) allowing unauthorized actions.

- *Mitigation:* Properly configure ASP.NET Identity roles (Customer, Admin) and implement access control policies in controllers.

Risk 7: Deployment failures causing downtime.

- *Mitigation:* Use Microsoft Azure/Hostinger for hosting with continuous integration and deployment (CI/CD) pipelines. Conduct deployment tests on a staging server before going live.

Risk 8: Payment failures due to Stripe API errors.

- *Mitigation:* Implement detailed error logging and fallback mechanisms for failed transactions. Notify users in case of issues and provide retry options.

Risk 9: Inventory mismanagement leading to overselling.

- *Mitigation:* Implement a real-time inventory management system that updates stock levels upon successful order placement.

Risk 10: Poor user adoption due to unoptimized UI/UX.

- *Mitigation:* Conduct user testing, gather feedback, and implement improvements. Enhance UI using jQuery and interactive elements for better user engagement.

Risk 11: Non-compliance with data protection regulations (e.g., GDPR, PCI DSS).

- *Mitigation:* Ensure encryption of sensitive data, obtain user consent for data collection, and follow Stripe's PCI compliance guidelines.

Risk 12: Unauthorized access due to weak security measures.

- *Mitigation:* Use HTTPS, secure API endpoints, implement rate limiting, and regularly update security patches.

2. Severity Chart

🟥 Risk 1	🟥 Risk 2	🟥 Risk 3	🟧 Risk 4
🟧 Risk 5	🟥 Risk 6	🟧 Risk 7	🟩 Risk 8
🟧 Risk 9	🟩 Risk 10	🟥 Risk 11	🟥 Risk 12

Key Performance Indicators (KPIs) for TechXpress E-commerce Platform

- 1. **Sales and Revenue Metrics**
    - **Total Revenue**: The total amount of money generated from product sales.
    - **Average Order Value (AOV)**: Total revenue divided by the number of orders.
    - **Conversion Rate**: Percentage of website visitors who complete a purchase.
    - **Customer Lifetime Value (CLV)**: Estimated total revenue from a single customer over their lifetime.
    - **Cart Abandonment Rate**: Percentage of users who add items to the cart but do not complete the purchase.
    - **Refund and Return Rate**: Percentage of products returned by customers.
    - **Product Performance Metrics**: Sales per product category (laptops, mobiles, cameras) to identify top-selling items.

2. Customer Experience & Engagement Metrics

- **Customer Satisfaction Score (CSAT)**: Based on customer surveys and feedback.
- **Net Promoter Score (NPS)**: Measures customer willingness to recommend the platform to others.
- **Customer Retention Rate**: Percentage of customers who make repeat purchases.
- **Average Response Time (Support)**: Time taken to respond to customer inquiries.
- **User Reviews and Ratings**: Average product ratings and number of reviews per product.
- **Checkout Process Efficiency**: Time taken for users to complete the checkout process.

3. Operational & Order Fulfillment Metrics

- **Order Processing Time**: Average time taken to process an order from checkout to shipment.
- **On-Time Delivery Rate**: Percentage of orders delivered within the promised timeframe.
- **Stock Availability**: Number of products in stock versus out-of-stock items.
- **Order Cancellation Rate**: Percentage of canceled orders before shipment.
- **Payment Success Rate**: Percentage of successful transactions processed via Stripe.
- **Fraud Detection Rate**: Number of fraudulent transactions detected and prevented.

4. Website Performance & Technical Metrics

- **Website Uptime**: Percentage of time the website is available and operational.
- **Page Load Time**: Average time taken for a page to load.

- **Bounce Rate**: Percentage of visitors who leave the website after viewing only one page.
- **Mobile Responsiveness Score**: Usability and functionality on mobile devices.
- **Error Rate**: Number of website crashes, payment failures, and critical errors reported.
- **Security Incidents**: Number of detected security threats or breaches.

5. Marketing & Traffic Metrics

- **Website Traffic**: Total number of visitors to the platform.

- **Traffic Source Breakdown**: Percentage of traffic from organic search, paid ads, social media, and direct visits.
- **Click-Through Rate (CTR)**: Percentage of users clicking on ads or marketing campaigns.
- **Customer Acquisition Cost (CAC)**: Cost of acquiring a new customer through marketing efforts.
- **Email Campaign Performance**: Open rates, click rates, and conversion rates from email marketing.
- **Social Media Engagement**: Likes, shares, comments, and followers across platforms.

6. Admin Panel & Management Metrics

- **Product Management Efficiency**: Number of products added, updated, or removed per month.
- **Order Management Efficiency**: Number of orders processed and approved per day.
- **Admin Activity Log**: Monitoring actions taken by admins to track system modifications.
- **Role-Based Access Effectiveness**: Measuring unauthorized access attempts and security breaches.

7. Final Deployment & Scaling Metrics

- **Hosting and Server Costs**: Monthly costs for Microsoft Azure/Hostinger hosting.
- **Scalability Performance**: Ability to handle traffic spikes without downtime.
- **Load Balancing Efficiency**: Distribution of traffic across servers.
- **User Growth Rate**: Percentage increase in registered users over time.
- **DataTables Performance**: Efficiency and speed of DataTables in handling admin panel data.

2. Literature Review

2.1 Feedback & Evaluation

The success of the TechXpress project will be assessed based on several key factors, including the clarity of objectives, technical implementation, and adherence to software engineering principles. Feedback from the lecturer and peers will be crucial in refining the project.

Areas of evaluation may include:

- Project Objectives & Scope: Are the goals clearly defined and achievable within the given timeframe?
- Technical Implementation: How well are the core functionalities implemented using ASP.NET Core, Entity Framework, and design patterns like NTier Architecture and Unit of Work?
- User Experience & UI Design: Is the platform user-friendly and visually appealing?
- Performance & Security: Are best practices followed for optimization and secure payment integration (Stripe)?
- Scalability & Maintainability: Can the system handle future expansion and modifications effectively?

2.2 Suggested Improvements

Based on standard software development best practices, the following enhancements could improve the project:

- UI/UX Enhancements: Consider integrating modern front-end frameworks like React or Vue.js for a more dynamic user experience.
- Advanced Search & Filtering: Implement Elasticsearch or similar tools to enhance product discovery.
- AI-Driven Recommendations: Introduce machine learning algorithms to suggest products based on user behavior.
- Automated Testing: Expand test coverage using unit tests (xUnit), integration tests, and Selenium for UI testing.
- Scalability Enhancements: Implement microservices architecture to separate key functionalities for better scalability and maintainability.

2.3 Final Grading Criteria

To ensure a fair and comprehensive evaluation, the project will be graded based on the following criteria:

Category	Weight (%)	Evaluation Criteria
Documentation	20%	Clarity and completeness of project proposal, technical documentation, and risk assessment.
Implementation	40%	Functionality of core features, proper use of ASP.NET Core, Entity Framework, and design patterns.
Testing	20%	Effectiveness of unit and integration tests, debugging, and performance testing.
Presentation	20%	Quality of the project demonstration, ability to answer questions, innovation, and creativity.

Project Requirement

1. Introduction

- **Purpose**: The purpose of this document is to define the requirements for the **TechXpress E-commerce Platform**, a scalable and maintainable web application for selling electronics. The platform will allow users to browse products, add items to a shopping cart, and complete purchases using integrated payment gateways. It will also include an admin panel for managing products, categories, and orders.
- **Scope**: The platform will be developed using **ASP.NET Core MVC**, **Entity Framework Core**, and **Stripe** for payment integration. It will follow **NTier Architecture**, **Repository Pattern**, and **Unit of Work Pattern** for efficient development and separation of concerns.

2. Functional Requirements

The functional requirements describe the features and behaviors of the system. These are categorized by user roles and system functionalities.

2.1 User Roles

1. **Customer**:
    - Browse products by category (e.g., laptops, mobiles, cameras).
    - Search for products using keywords and filters (e.g., price range, brand).
    - Add products to the shopping cart.
    - View and manage the shopping cart (update quantities, remove items).
    - Proceed to checkout and complete the purchase using Stripe.
    - View order history and track order status.
    - Manage user profile (update personal details, change password).
2. **Admin**:
    - Manage product listings (add, update, delete products).
    - Manage product categories (add, update, delete categories).
    - View and manage orders (update order status, cancel orders).
    - View sales reports and analytics.
    - Manage user roles and permissions.

2.2 Product Management

1. The system shall allow admins to add new products with details such as:
    - Product name, description, price, stock quantity, category, and image.
2. The system shall allow admins to update product details.
3. The system shall allow admins to delete products.

4. The system shall display products to customers with filters and sorting options (e.g., price low to high, popularity).

2.3 Shopping Cart

1. The system shall allow customers to add products to the shopping cart.
2. The system shall display the shopping cart with the following details:
    - Product name, price, quantity, and total price.
3. The system shall allow customers to update the quantity of items in the cart.
4. The system shall allow customers to remove items from the cart.

5. The system shall calculate the total price of items in the cart, including taxes and discounts (if applicable).

2.4 Checkout and Payment

1. The system shall allow customers to proceed to checkout from the shopping cart.
2. The system shall collect shipping and payment details during checkout.
3. The system shall integrate with **Stripe** to process payments securely.
4. The system shall display an order confirmation page with order details and a unique order ID.
5. The system shall send an email confirmation to the customer after a successful purchase.

2.5 Order Management

1. The system shall allow customers to view their order history.
2. The system shall allow customers to track the status of their orders (e.g., pending, shipped, delivered).
3. The system shall allow admins to update the status of orders.
4. The system shall allow admins to cancel orders.

2.6 User Authentication and Authorization

1. The system shall allow users to register and log in using **ASP.NET Identity**.
2. The system shall enforce role-based access control (RBAC):
    - Customers can only access customer-related features.
    - Admins can access the admin panel and manage products, categories, and orders.

Non-Functional Requirements

These requirements describe the quality attributes of the system.

1. **Performance**:
    - Product search results shall load within **2 seconds** under normal load.
2. **Security**:
    - The system shall encrypt sensitive data (e.g., passwords, payment details) using **AES-256 encryption**.
    - The system shall prevent SQL injection and cross-site scripting (XSS) attacks.
    - The system shall enforce HTTPS for all communication.
3. **Usability**:
    - The system shall provide clear error messages and validation feedback to users.
4. **Scalability**:
    - The system shall be designed to scale horizontally to support future growth in users and products.

4. User Requirements

1. **Customer Experience**:
    - Customers shall be able to easily browse and search for products.
    - Customers shall have a seamless checkout experience with minimal steps.
    - Customers shall receive timely notifications about their order status.
2. **Admin Experience**:
    - Admins shall have an intuitive dashboard for managing products, categories, and orders.
    - Admins shall be able to generate sales reports and analytics.

6. External Interfaces

1. **Stripe API**:
   - The system shall integrate with Stripe for payment processing.
   - The system shall handle payment success and failure responses from Stripe.

     // not required paid service

2. **Email Service**:

   - The system shall send email notifications (e.g., order confirmation, password reset) using an external email service (e.g., SendGrid).

     //not required

7. Assumptions and Constraints

1. **Assumptions**:
   - The system will be deployed on **Microsoft Azure**.
2. **Constraints**:
   - The project must be completed within **4 weeks**.

Database Requirements for TechXpress E-commerce Platform

1. Database Overview

The database will store all data related to products, categories, users, orders, and payments. It will be designed using Entity Framework Core and follow the Repository Pattern and Unit of Work Pattern for efficient data access and management. The database will be hosted on Microsoft Azure SQL Database for scalability and reliability.

2. Database Schema

The database schema will consist of the following tables:

2.1 Users Table

- **Purpose: Stores user information for authentication and role-based access control.**
- **Fields:**
  - **UserId (Primary Key, GUID)**
  - **FirstName (String, Not Null)**
  - **LastName (String, Not Null)**
  - **Email (String, Unique, Not Null)**
  - **PasswordHash (String, Not Null)**
  - **Role (String, Not Null) – Values: Customer, Admin**
  - **CreatedAt (DateTime, Not Null)**
  - **LastLogin (DateTime, Nullable)**
  - **IsActive (Boolean, Default: True)**

2.2 Products Table

- **Purpose: Stores product details for display and management.**
- **Fields:**
    - **ProductId (Primary Key, GUID)**
    - **Name (String, Not Null)**
    - **Description (String, Not Null)**
    - **Price (Decimal, Not Null)**
    - **StockQuantity (Integer, Not Null)**
    - **CategoryId (Foreign Key, GUID, Not Null)**
    - **ImageUrl (String, Nullable)**
    - **CreatedAt (DateTime, Not Null)**
    - **UpdatedAt (DateTime, Nullable)**
    - **IsActive (Boolean, Default: True)**

2.3 Categories Table

- **Purpose: Organizes products into categories for better navigation.**
- **Fields:**
    - **CategoryId (Primary Key, GUID)**
    - **Name (String, Not Null)**
    - **Description (String, Nullable)**
    - **CreatedAt (DateTime, Not Null)**
    - **UpdatedAt (DateTime, Nullable)**
    - **IsActive (Boolean, Default: True)**

2.4 Orders Table

- **Purpose: Tracks customer orders and their status.**
- **Fields:**
    - **OrderId (Primary Key, GUID)**
    - **UserId (Foreign Key, GUID, Not Null)**
    - **OrderDate (DateTime, Not Null)**
    - **TotalAmount (Decimal, Not Null)**
    - **OrderStatus (String, Not Null) – Values: Pending, Processing, Shipped, Delivered, Cancelled**
    - **PaymentStatus (String, Not Null) – Values: Paid, Unpaid**
    - **ShippingAddress (String, Not Null)**
    - **TransactionId (String, Nullable) – Stores Stripe transaction ID.**

2.5 OrderDetails Table

- **Purpose: Stores individual items within an order.**
- **Fields:**
    - **OrderDetailId (Primary Key, GUID)**
    - **OrderId (Foreign Key, GUID, Not Null)**
    - **ProductId (Foreign Key, GUID, Not Null)**
    - **Quantity (Integer, Not Null)**
    - **UnitPrice (Decimal, Not Null)**
    - **TotalPrice (Decimal, Not Null)**

2.6 ShoppingCart Table

- **Purpose: Temporarily stores items added to the cart by users.**

- **Fields:**
    - **CartId (Primary Key, GUID)**
    - **UserId (Foreign Key, GUID, Not Null)**
    - **ProductId (Foreign Key, GUID, Not Null)**
    - **Quantity (Integer, Not Null)**
    - **AddedAt (DateTime, Not Null)**

---

3. Relationships

1. **Users ↔ Orders: One-to-Many (One user can have many orders).**
2. **Orders ↔ OrderDetails: One-to-Many (One order can have multiple items).**
3. **Products ↔ Categories: Many-to-One (Many products belong to one category).**
4. **Products ↔ ShoppingCart: Many-to-Many (Many products can be in many carts).**
5. **Products ↔ OrderDetails: Many-to-Many (Many products can be in many orders).**

4. Data Validation Rules

1. **Product Price:**
    - **Must be a positive decimal value.**
    - **Cannot be null.**
2. **Stock Quantity:**
    - **Must be a non-negative integer.**
    - **Cannot be null.**
3. **Email:**
    - **Must be unique.**
    - **Must follow a valid email format (e.g., user@example.com).**

4. **Order Status:**
    - **Must be one of the predefined values: Pending, Processing, Shipped, Delivered, Cancelled.**
5. **Payment Status:**
    - **Must be one of the predefined values: Paid, Unpaid.**

---

5. Indexes

1. **Users Table:**
    - **Index on Email for fast lookup during login.**
2. **Products Table:**
    - **Index on CategoryId for efficient filtering by category.**
    - **Index on Name for fast search functionality.**
3. **Orders Table:**
    - **Index on UserId for quick retrieval of user orders.**
    - **Index on OrderDate for sorting and reporting.**

---

6. Security Requirements

1. **Data Encryption:**
    - **Sensitive data (e.g., PasswordHash, TransactionId) will be encrypted using AES- 256 encryption.**
2. **Access Control:**

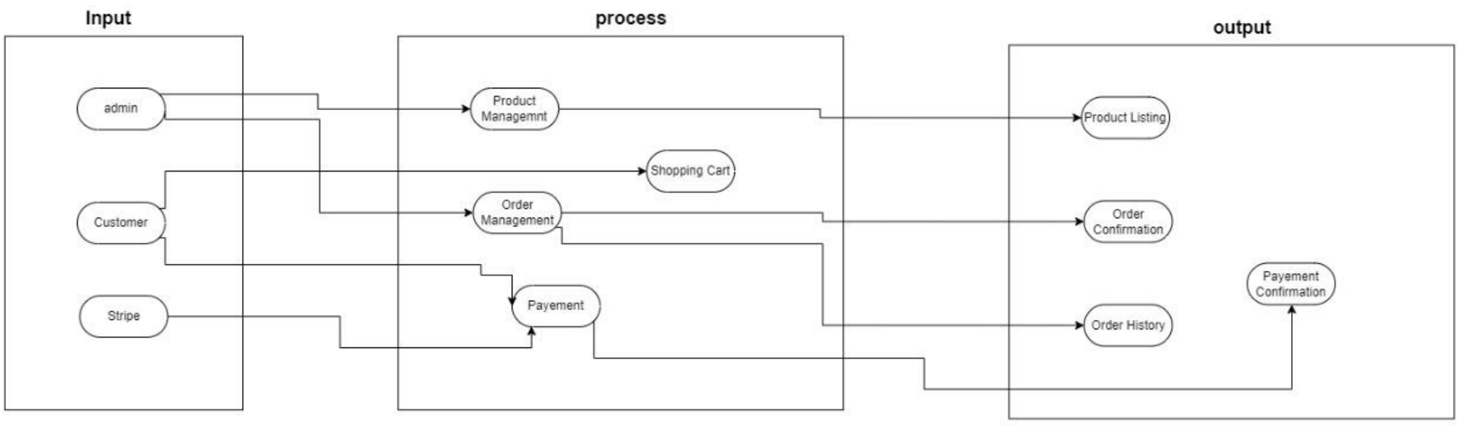
    - **Only authorized users (e.g., admins) can access and modify product and order data.**
    - **Customers can only access their own orders and shopping cart.**
3. **Audit Logs:**
    - **All changes to critical tables (e.g., Products, Orders) will be logged for auditing purposes.**

8. Performance Requirements

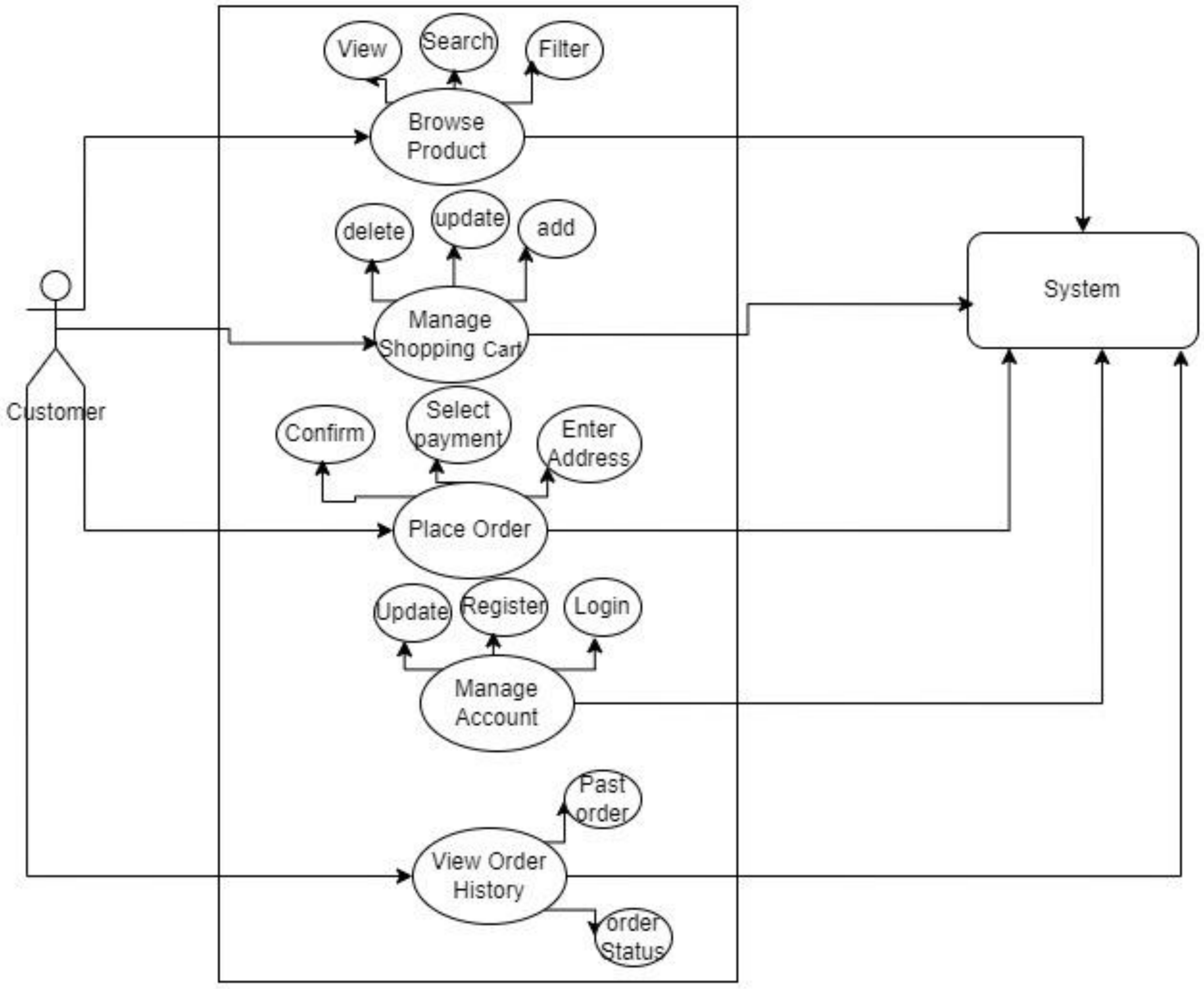
1. **Response Time:**
    - **Product search queries must return results within 2 seconds under normal load.**
    - **Order placement transactions must complete within 5 seconds.**

Diagrams

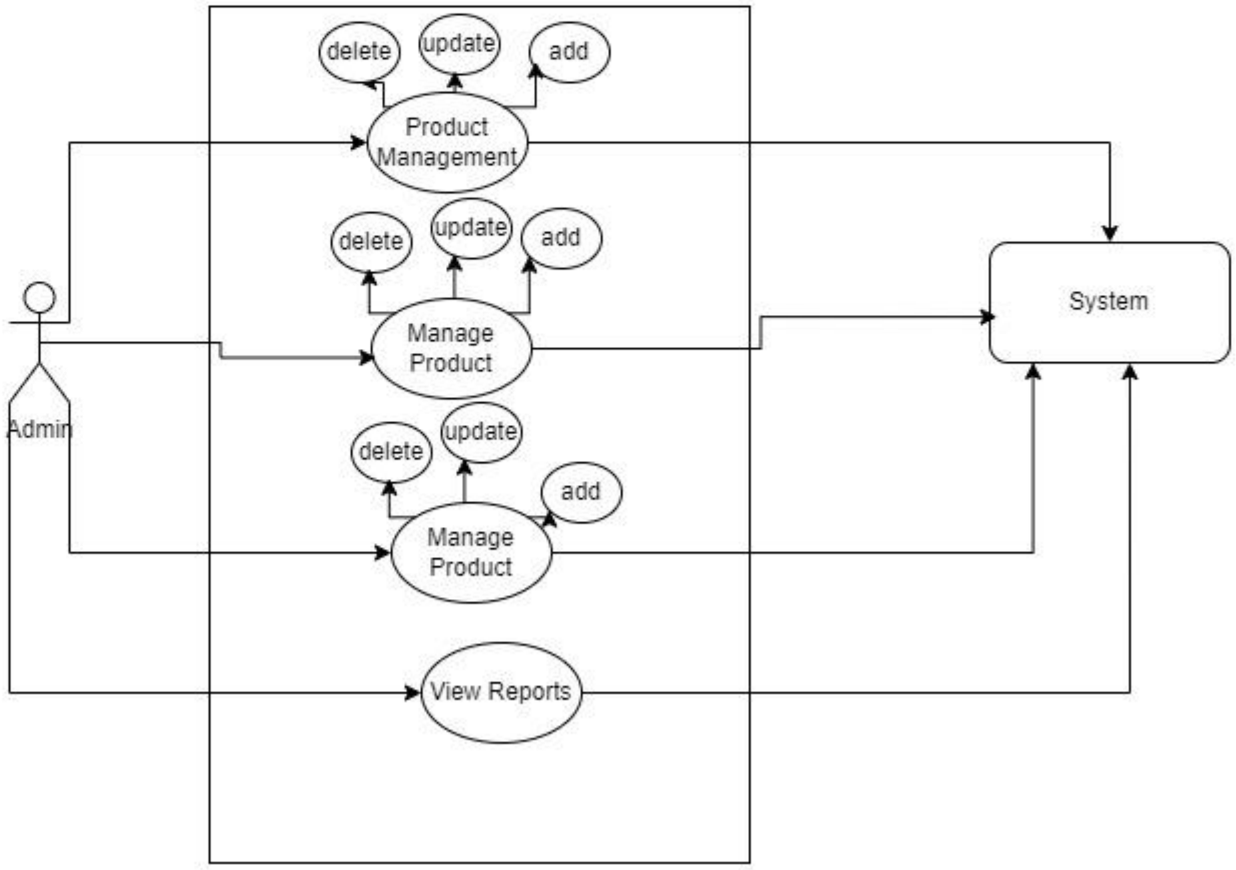
1-DFD



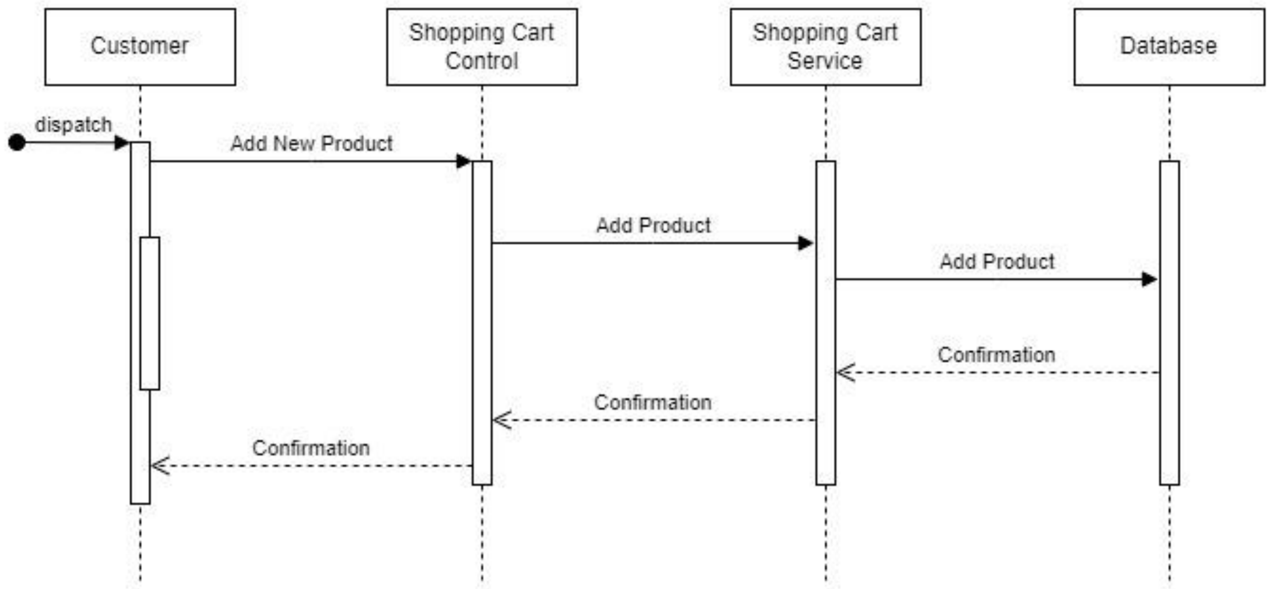
2- Use Case Customer



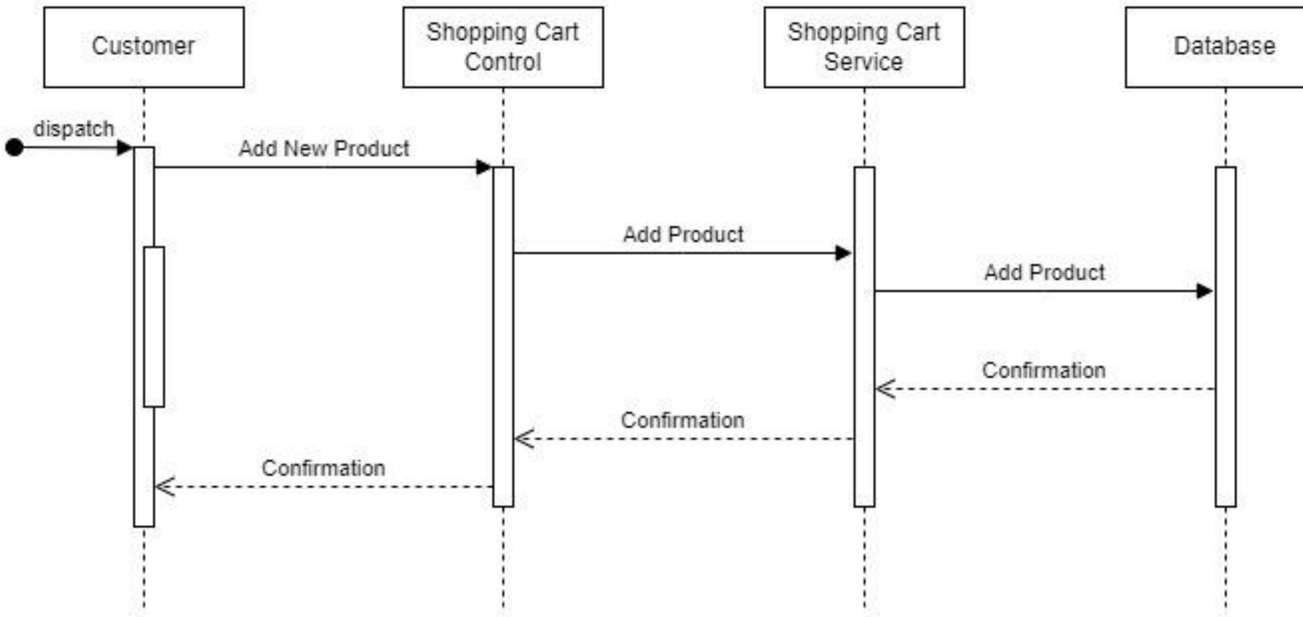
3- Use Case Admin



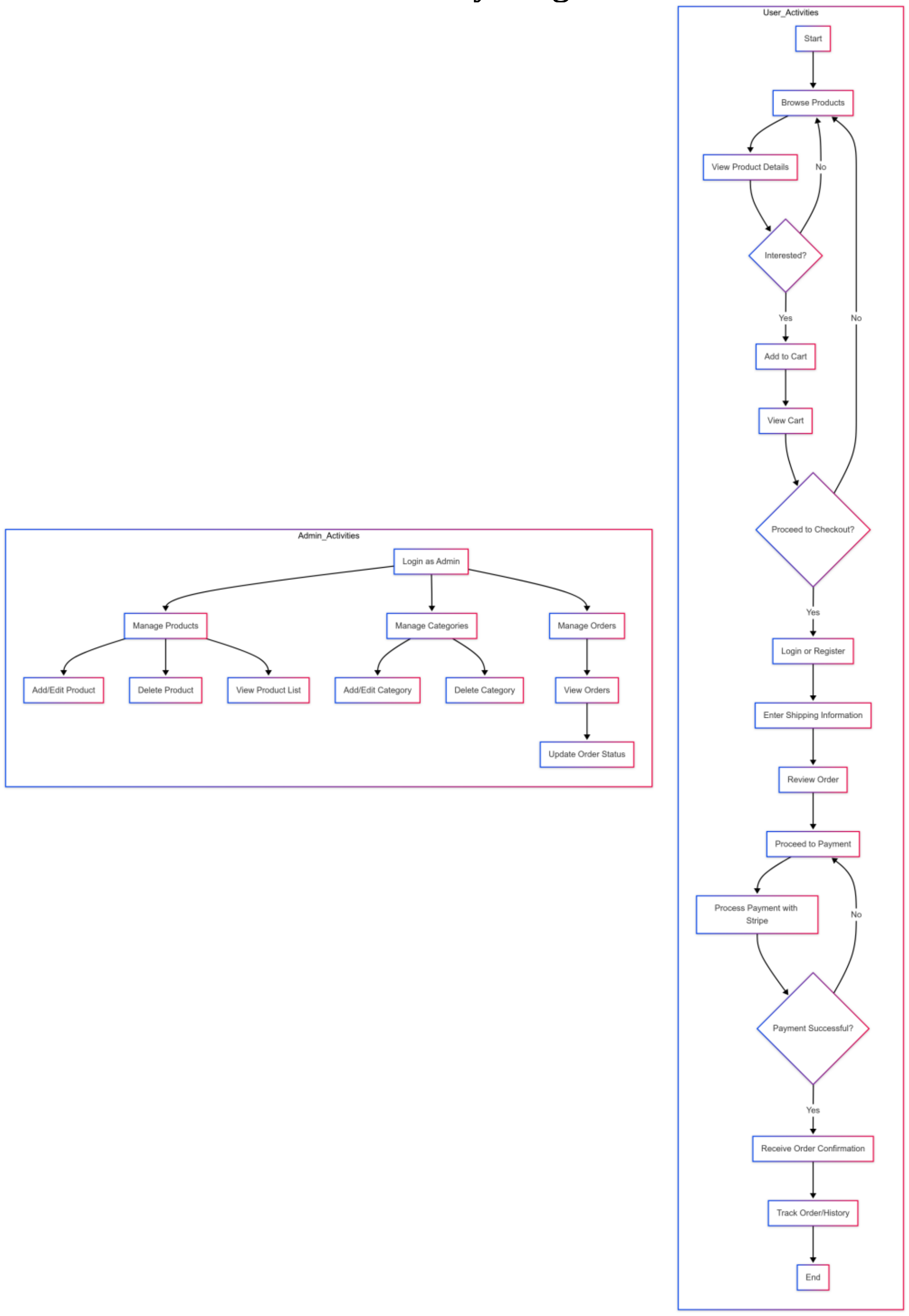
4- Sequence Diagrams Customer



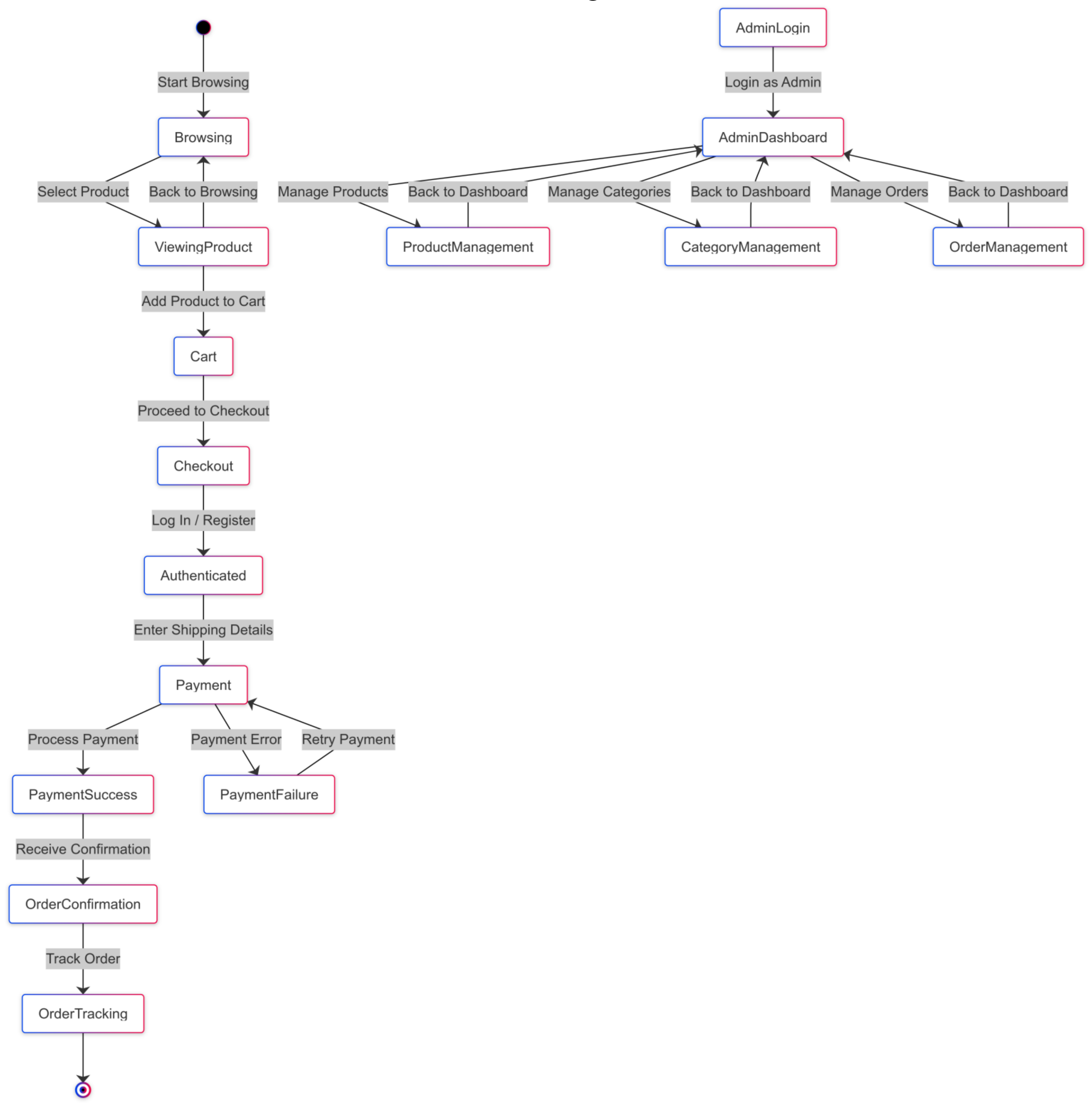
4. Sequence Diagrams Admin



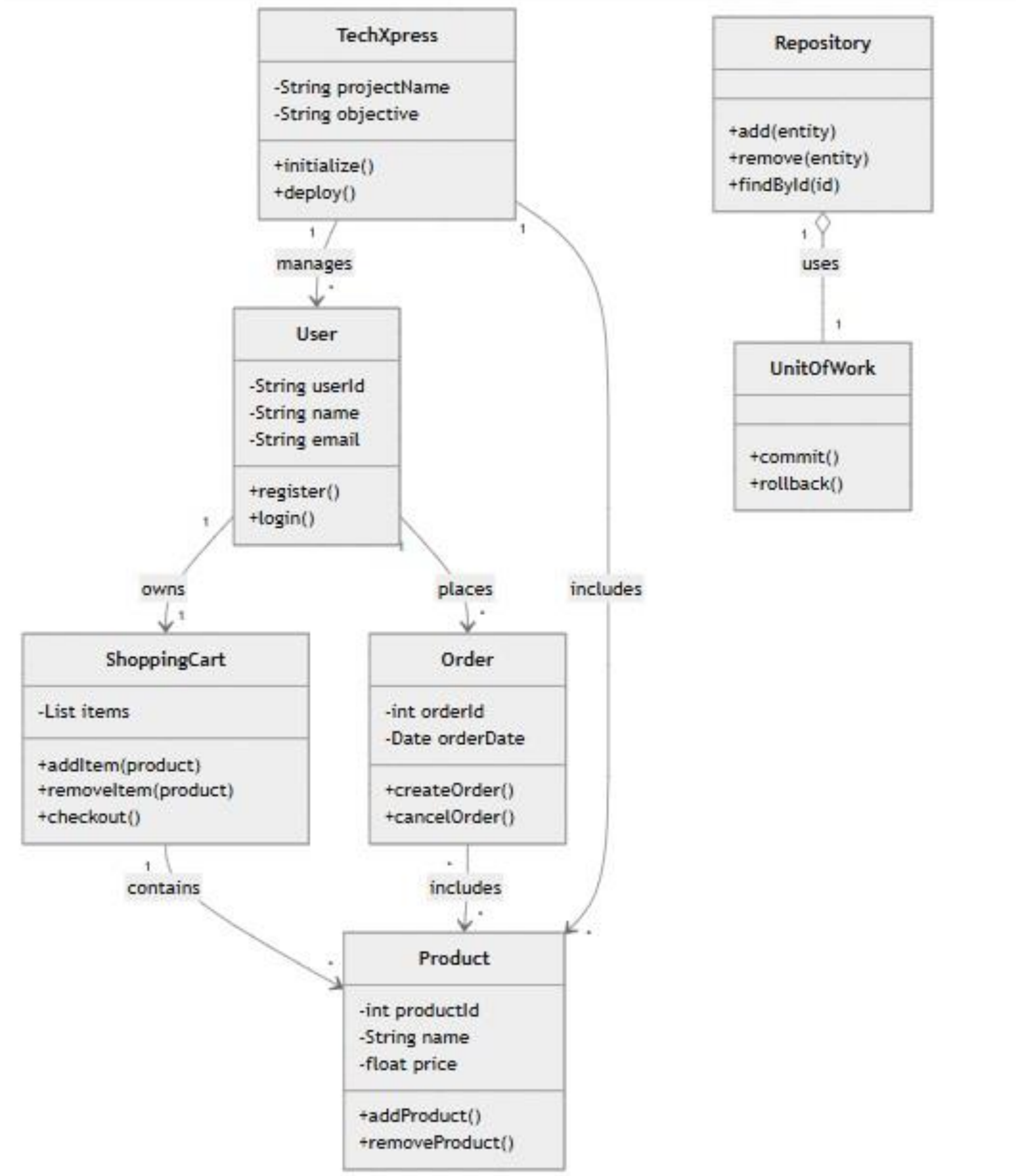
5. Activity Diagram



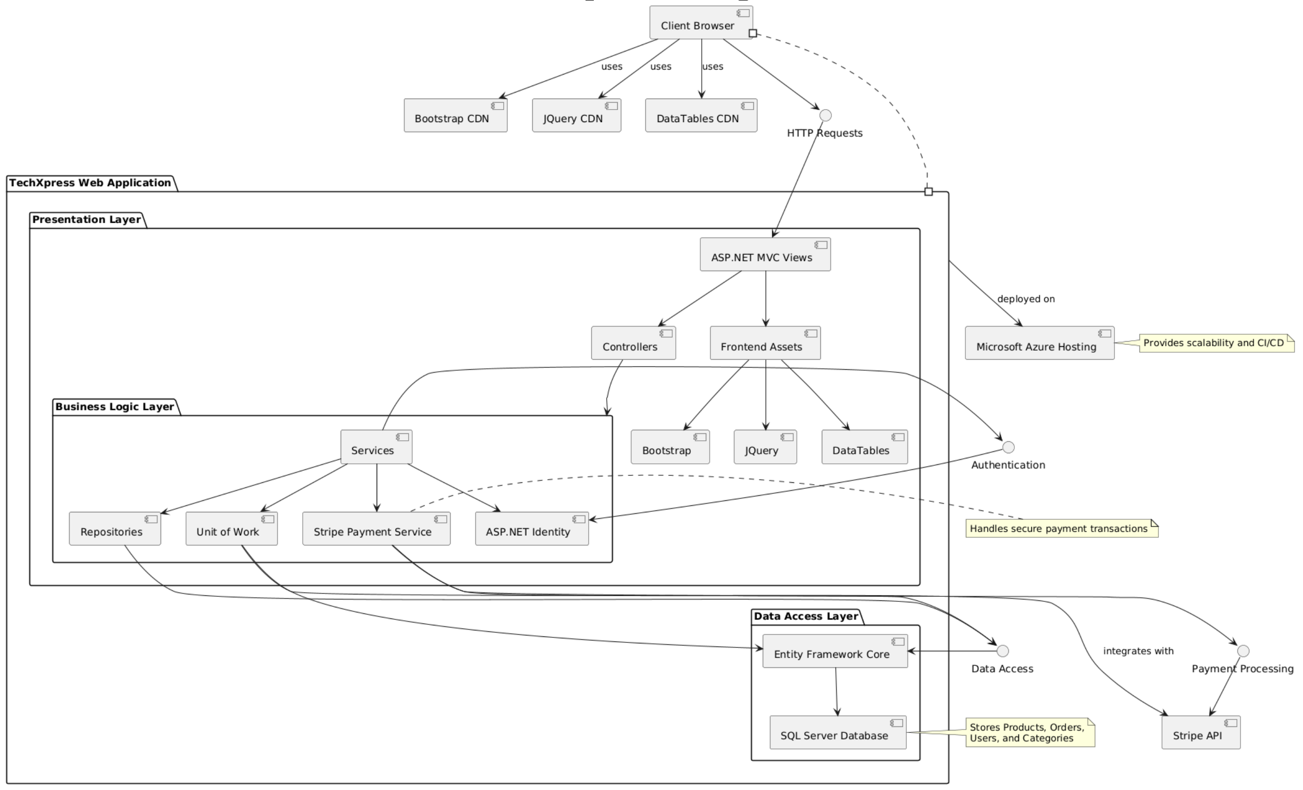
7-State Diagram



8-Class Diagram



9-Component Diagram


Client Browser
uses
uses
uses
Bootstrap CDN
JQuery CDN
DataTables CDN
HTTP Requests
TechXpress Web Application
Presentation Layer
ASP.NET MVC Views
Controllers
Frontend Assets
Bootstrap
JQuery
DataTables
deployed on
Microsoft Azure Hosting
Provides scalability and CI/CD
Business Logic Layer
Services
Authentication
Repositories
Unit of Work
Stripe Payment Service
ASP.NET Identity
Handles secure payment transactions
Data Access Layer
Entity Framework Core
Data Access
integrates with
Payment Processing
SQL Server Database
Stores Products, Orders, Users, and Categories
Stripe API