# **TechXpress E-commerce Platform with ASP.NET Core**

## **1. Project Overview**

### **The TechXpress E-commerce Platform is a full-featured online marketplace for selling electronics such as laptops, mobiles, and cameras. It provides customers with the ability to browse products, add them to a shopping cart, and complete purchases using integrated payment gateways. An admin panel is included for managing products, categories, and orders.**

### **This project follows an NTier Architecture to separate concerns across the Presentation (UI), Business Logic, and Data Access layers. By incorporating well-known design patterns (Repository Pattern, Unit of Work, Dependency Injection), the solution ensures a high level of maintainability, scalability, and testability.**

### 

## **2. Project Objectives**

### **Build a scalable and maintainable e-commerce application for selling electronics.**

### **Leverage design patterns (NTier, Repository, Unit of Work, Dependency Injection) to ensure clear separation of concerns.**

### **Implement robust user authentication and role-based access control using ASP.NET Identity.**

### **Provide a secure payment integration with Stripe to handle online transactions.**

### **Offer an intuitive admin panel for product, category, and order management.**

### **Deploy the application to the cloud (Microsoft Azure or Hostinger) for live usage.**

### 

## **3. Project Scope**

### **Frontend (Presentation Layer)**

### **ASP.NET Core MVC for views and controllers.**

### **Use of Bootstrap, jQuery, and Toaster JS for a modern, responsive UI.**

### **DataTables integration for search, filtering, sorting, and pagination in the admin panel.**

### **Business Logic Layer (Services)**

### **Encapsulates operations such as adding/removing products from the cart, calculating totals, and processing orders.**

### **Manages rules for promotions, discounts, and order flow.**

### **Data Access Layer**

### **Uses Entity Framework Core for database interactions.**

### **Implements Repository Pattern for clean data operations (CRUD).**

### **Unit of Work ensures all changes in a single transaction are handled atomically.**

### **Authentication and Authorization**

### **ASP.NET Identity for user registration, login, and role management (Customer, Admin).**

### **Role-Based Access Control (RBAC) to restrict admin features.**

### **Payment Gateway Integration**

### **Stripe is used to securely handle online payments.**

### **Deployment and Testing**

### **Deployment on Microsoft Azure or Hostinger.**

### **Comprehensive testing of features such as browsing, cart management, order placement, and admin controls.**

### 

## **4. Design Patterns and Architecture**

### **NTier Architecture**

### **Presentation Layer (TechXpress.Web)**

### **Hosts the ASP.NET MVC application, views, controllers, and client-side logic (jQuery, DataTables).**

### **Business Logic Layer (TechXpress.Services)**

### **Houses business rules, including product operations, cart calculations, and order processing.**

### **Data Access Layer (TechXpress.Data)**

### **Uses Entity Framework Core to manage database interactions, applying the Repository and Unit of Work patterns.**

### **Repository Pattern**

### **Provides a clean abstraction for data operations (CRUD).**

### **Includes generic repositories for common operations and specialized repositories (e.g., ProductRepository, CategoryRepository).**

### **Unit of Work Pattern**

### **Ensures multiple related changes (e.g., creating an order and updating stock) occur in a single transaction.**

### **Reduces data inconsistencies by grouping all database actions into one commit.**

### **Dependency Injection**

### **Injects services (repositories, business logic) into controllers for improved testability and loose coupling.**

### 

## **5. Technologies to Use**

### **ASP.NET Core MVC**

### **Entity Framework Core**

### **ASP.NET Identity (for user authentication and authorization)**

### **Stripe (for payment integration)**

### **jQuery and DataTables (for dynamic UI components)**

### **Toaster JS (for notifications)**

### **Microsoft Azure or Hostinger (for deployment)**

### 

## **6. Project Timeline**

| **Phase** | **Tasks** | **Duration** |
| --- | --- | --- |
| **Planning** | **Define requirements, scope, and architecture.** | **1 Week** |
| **Frontend Development** | **Implement UI using ASP.NET Core MVC, Bootstrap, and jQuery.** | **2 Weeks** |
| **Backend Development** | **Develop Business Logic and Data Access layers.** | **3 Weeks** |
| **Authentication** | **Implement user authentication and role-based access control.** | **1 Week** |
| **Payment Integration** | **Integrate Stripe for secure payment processing.** | **1 Week** |
| **Testing** | **Perform unit testing, integration testing, and user acceptance testing.** | **2 Weeks** |
| **Deployment** | **Deploy the application to Microsoft Azure or Hostinger.** | **1 Week** |

### 

## **7. Team Roles and Responsibilities**

| **Role** | **Responsibilities** |
| --- | --- |
| **Project Manager** | **Oversee project progress, manage timelines, and ensure deliverables are met.** |
| **Frontend Developer** | **Develop the user interface using ASP.NET Core MVC, Bootstrap, and jQuery.** |
| **Backend Developer** | **Implement business logic, data access, and authentication features.** |
| **QA Engineer** | **Conduct testing to ensure the application is bug-free and meets requirements.** |
| **DevOps Engineer** | **Handle deployment and ensure the application runs smoothly on the cloud.** |

### 

## **8. Risk Management**

| **Risk** | **Mitigation Strategy** |
| --- | --- |
| **Delays in development** | **Regular progress tracking and agile sprints to ensure timely delivery.** |
| **Payment gateway integration issues** | **Use Stripe's sandbox environment for testing and follow their documentation closely.** |
| **Security vulnerabilities** | **Implement secure coding practices and conduct regular security audits.** |
| **Deployment failures** | **Test deployment on a staging environment before going live.** |

### 

## **9. Deliverables**

### **Fully functional e-commerce platform with a responsive UI.**

### **Admin panel for managing products, categories, and orders.**

### **Secure payment integration using Stripe.**

### **Source code and documentation.**

### **Deployed application on Microsoft Azure or Hostinger.**

### 

## **10. Conclusion**

### **The TechXpress E-commerce Platform aims to provide a seamless online shopping experience for customers while offering robust tools for administrators to manage the platform efficiently. By leveraging modern technologies and design patterns, the project ensures scalability, maintainability, and security.**

### **Project: Electronics E-Commerce Store with Payment Integration**

### **1-Project Plan**

### **•**

### **Timeline (Gantt chart)**

### 

### 

**Task Assignment & Roles:**

Development Team Responsibilities:

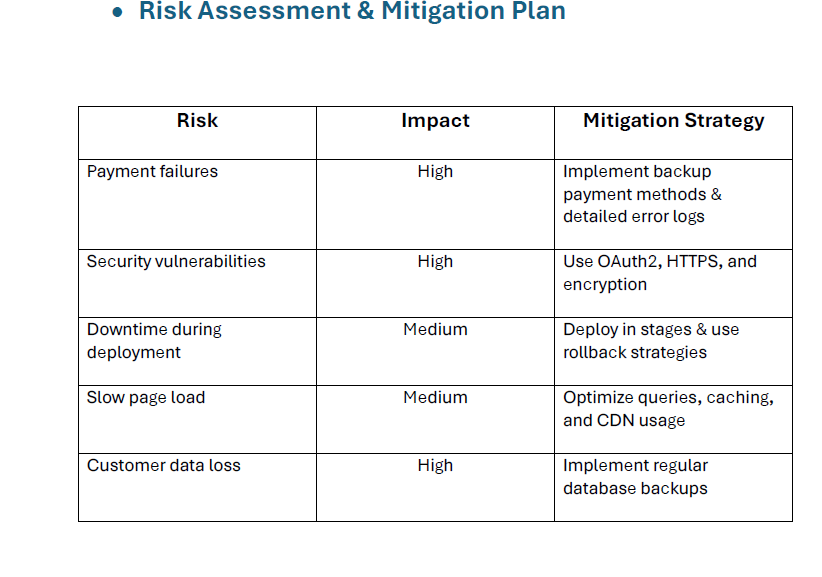
•Frontend Developer: Implement UI/UX, product pages, and shopping cart.

•Backend Developer: Build APIs, database, and order processing logic.

•Payment Integration Specialist: Integrate Stripe/PayPal for transactions.

•QA Engineer: Perform testing & bug fixing.

•DevOps Engineer: Handle deployment & server setup.



**4. Risk Assessment & Mitigation Plan:**

Identified Risks & Mitigation Strategies:

1. Scope Creep

a. Risk: Continuous addition of new features may extend the project timeline

and increase complexity.

b. Mitigation: Clearly define project scope and use change control processes.

2. Security Vulnerabilities

a. Risk: Unauthorized access, data breaches, or cyberattacks.

b. Mitigation: Implement security best practices, use encryption, and conduct

regular security audits.

3. Performance Issues

a. Risk: Slow response times and scalability challenges under high traffic.

b. Mitigation: Optimize database queries, use caching strategies, and

implement load balancing.

4. Payment Gateway Failures

a. Risk: Payment transactions failing due to API downtime or integration issues.

b. Mitigation: Use multiple payment gateway providers and implement fallback

mechanisms.

**Key Performance Indicators (KPIs):**

•System uptime: Maintain 99.9% availability.

•Response time: Pages should load under 3 seconds.

•Checkout success rate: At least 95% of transactions should complete without failure.

•User adoption: Target 500+ active users in the first 3 months.

### **TechXpress E-commerce Platform – Requirements Gathering**

#### **1. Stakeholder Analysis**

As part of the project planning phase, I identified the key stakeholders and their specific needs to ensure the system meets all functional and business requirements:

* Customers need a seamless shopping experience to browse, purchase electronics, and make secure payments.
* Admins require a comprehensive admin panel to manage products, categories, and orders efficiently.
* The system administrator is responsible for managing security, user roles, and system performance.
* The payment provider (Stripe) handles online transactions and ensures secure payment processing.
* The hosting provider (Microsoft Azure/Hostinger) ensures application uptime, scalability, and performance.

#### **2. User Stories & Use Cases**

To capture the functional aspects of the system, I created the following user stories and use cases:

* Customers should be able to search for products by category or price to find what they need quickly.
* Customers should have the ability to add products to their shopping cart for review and later purchase.
* The checkout process must be secure, allowing customers to complete purchases using Stripe.
* Customers should have access to their order history through their accounts to track shipping status.
* Admins need the ability to add, edit, and delete products for efficient inventory management.
* Admins should be able to track and update order statuses to ensure smooth order fulfillment.
* Admins should manage product categories and promotional offers to enhance the shopping experience.
* The system administrator should manage user roles and permissions to maintain security and access control.
* The system administrator should monitor system performance to ensure stability and optimal operation.

#### **3. Functional Requirements**

I outlined the core functionalities that the system must provide:

* Product browsing and search functionality.
* Shopping cart management with session handling.
* Secure checkout process integrated with Stripe payment gateway.
* User authentication and role-based access control for customers and admins.
* Admin panel for managing products, categories, and orders.
* Automated email notifications for order updates.
* Customer profile section for order tracking and account management.

#### **4. Non-functional Requirements**

Beyond functionality, I also defined critical non-functional requirements to ensure performance, security, and scalability:

* The system should process requests within two seconds for a smooth user experience.
* Passwords should be encrypted using ASP.NET Identity, and payment transactions must be securely handled.
* The interface should be intuitive and user-friendly, utilizing Bootstrap and Toaster JS for an engaging experience.
* The platform must maintain 99.9% uptime, hosted on Azure or Hostinger.
* The architecture follows N-Tier Structure and Repository Pattern to allow future growth and expansion.

### **TechXpress E-commerce Platform – System Analysis & Design**

## **1. Problem Statement & Objectives**

### **Problem Statement**

The current lack of a specialized and scalable e-commerce platform for electronics limits the ability of businesses to manage products, orders, and secure transactions effectively. Customers also need a seamless shopping experience with robust security and order tracking.

### **Objectives**

* Develop a **scalable** and **secure** e-commerce platform for electronics.
* Implement a **role-based system** to manage customers and administrators.
* Integrate **Stripe** for secure online transactions.
* Provide an **admin dashboard** for managing products, categories, and orders.
* Design a **user-friendly interface** with an optimized shopping experience.

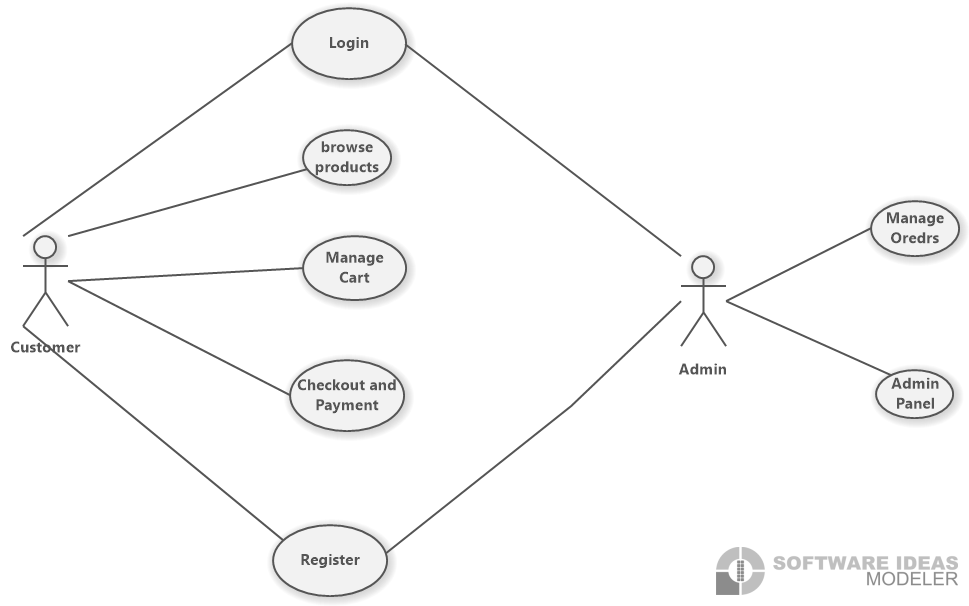
## **2. Use Case Diagram & Descriptions**

### **Use Case Diagram**

A visual representation of how different actors interact with the system.

#### **Actors**

* **Customer**: Can browse products, add items to the cart, place orders, and make payments.
* **Admin**: Manages products, categories, and orders.
* **System Administrator**: Oversees system security and user management.
* **Stripe API**: Handles payment transactions.



### **Functional Requirements**

* User authentication and authorization (customers, admins).
* Product browsing, search, and filtering.
* Shopping cart and order management.
* Secure checkout process with Stripe integration.
* Admin panel for inventory and order management.
* Email notifications for order updates.
* User profile management and order history tracking.

### **Non-Functional Requirements**

| **Functional Requirements** | **Non-Functional Requirements** |
| --- | --- |
| Product search & filter | System response within 2 sec |
| Secure checkout via Stripe | 99.9% uptime on Azure/Hostinger |
| Admin panel for orders | Encrypted passwords via ASP.NET Identity |
| Email notifications | Scalable NTier architecture |

## **4. Software Architecture**

### **Architecture Overview:**

The system follows a **NTier Architecture**:

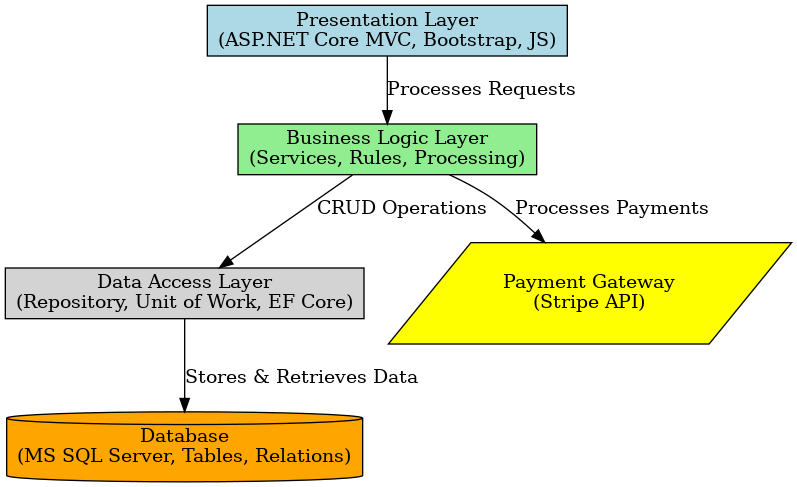
1. **Presentation Layer (TechXpress.Web)** – Handles UI using **ASP.NET Core MVC, Bootstrap, JavaScript**.
2. **Business Logic Layer (TechXpress.Services)** – Implements business rules and logic.
3. **Data Access Layer (TechXpress.Data)** – Manages database interactions via **Entity Framework Core**.

### **Software Architecture Diagram:**

Below is a high-level system architecture representation:

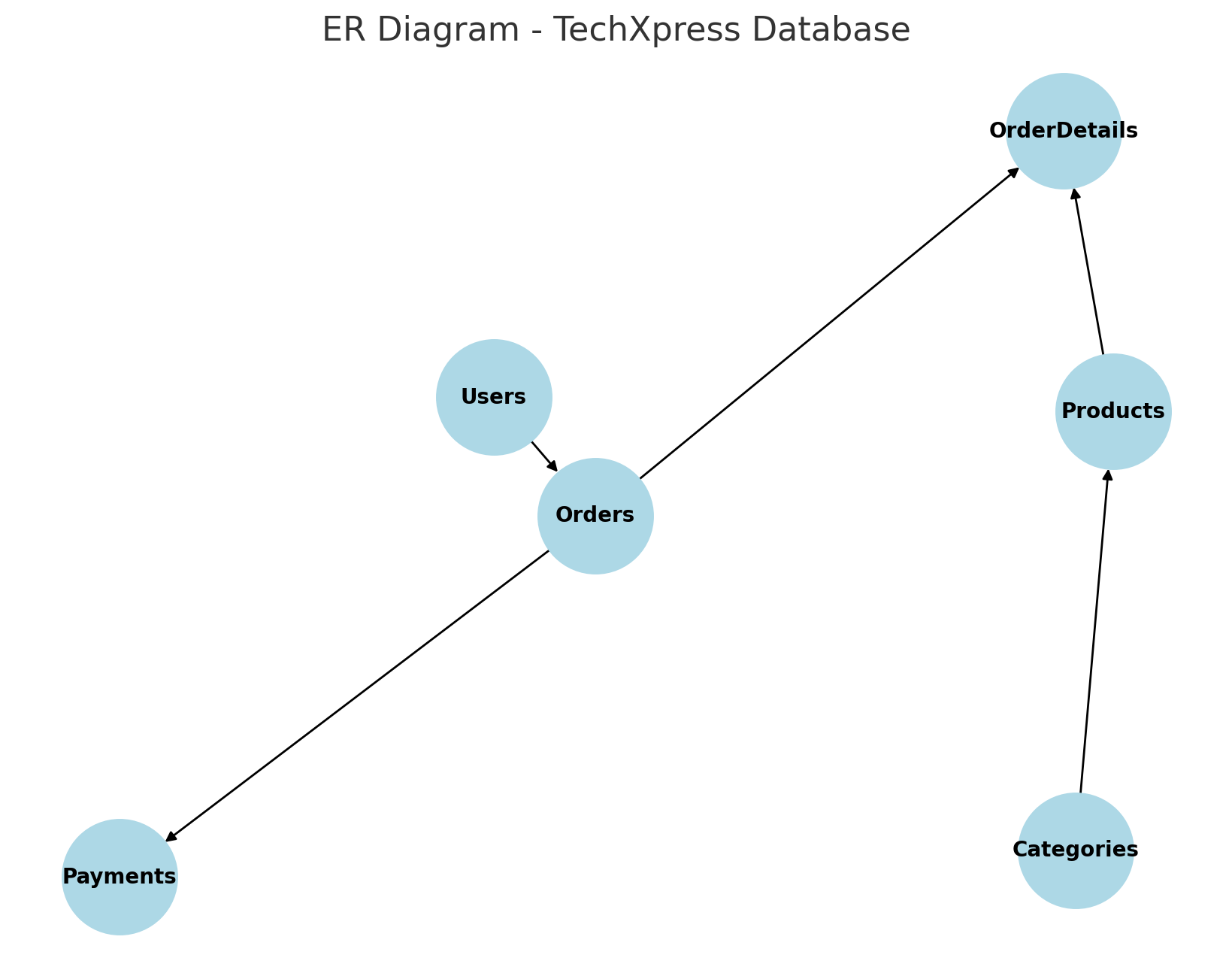
### **Design Patterns**

* **Repository Pattern**: Abstracts database operations.
* **Unit of Work**: Ensures atomic transactions.
* **Dependency Injection**: Enhances modularity and testability.



## **5. Database Design & Data Modeling**

### **ER Diagram (Entity-Relationship Diagram)**

A structured model illustrating relationships between **Products, Users, Orders, Categories**, and **Payments**.

### **Logical & Physical Schema**

#### **Tables & Attributes**

* **Users** (UserID, Name, Email, Password, Role)
* **Products** (ProductID, Name, Description, Price, CategoryID, StockQuantity)
* **Categories** (CategoryID, Name)
* **Orders** (OrderID, UserID, TotalAmount, PaymentStatus, OrderDate)
* **OrderDetails** (OrderDetailID, OrderID, ProductID, Quantity, Price)
* **Payments** (PaymentID, OrderID, TransactionID, PaymentMethod, Status)

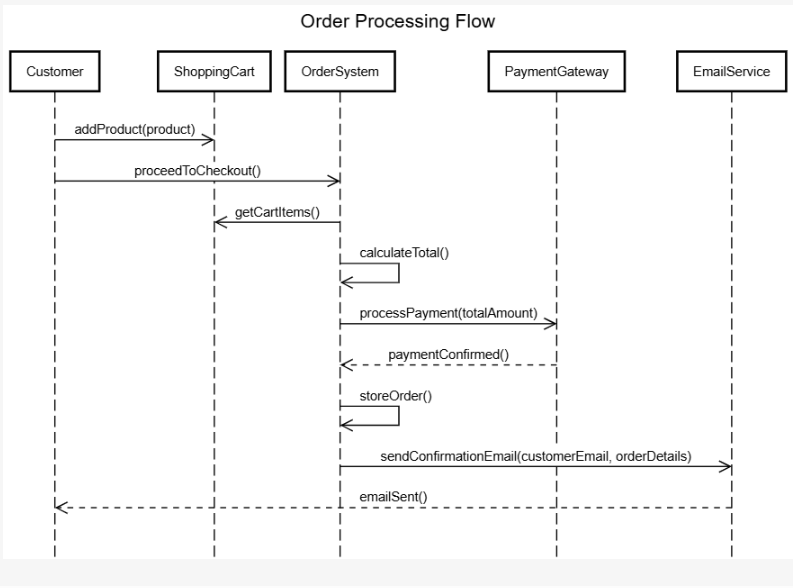
## **6. Data Flow & System Behavior**

### **DFD (Data Flow Diagram)** :llustrates the flow of data between users, the system, and external services like Stripe.

### 

### **Sequence Diagrams**

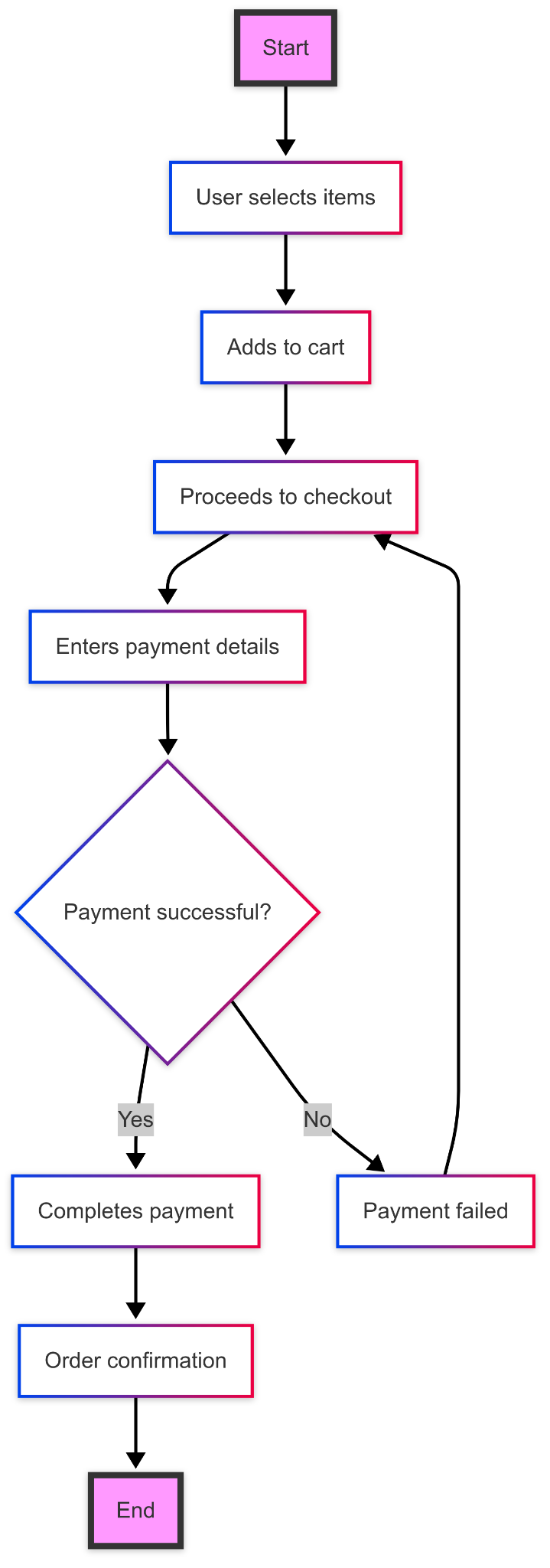
#### **Example: Order Processing Flow**



### **Activity Diagram**

#### **Example: Customer Checkout:**

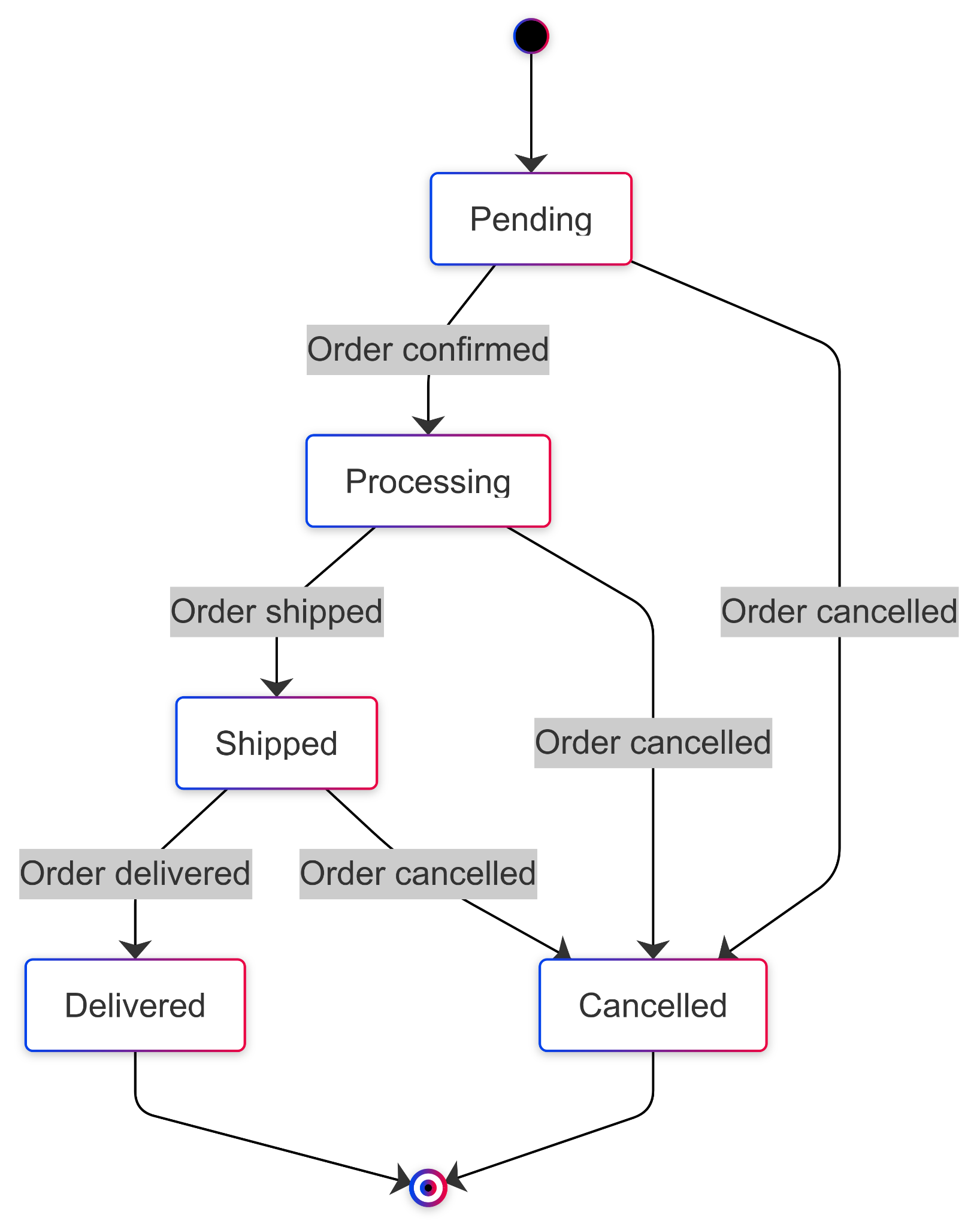
* User selects items → Adds to cart → Proceeds to checkout → Enters payment details → Completes payment → Order confirmation.



### **State Diagram**

Defines different states of an **Order**:

* **Pending** → **Processing** → **Shipped** → **Delivered** → **Cancelled**.



### 

### **Class Diagram**

Defines system structure, showing classes like **User, Product, Order, Payment, Cart, and Admin**, including their attributes and relationships.

