**TachXpress E-Commerce Platform  
  
  
  
  
Team Members  
Yasmeen Yehia**

**Malak Hisham**

**Belal Fawzy**

**Instructor**

**Karim Essam**

* **Project Planning & Management**

1. **Project Proposal:**

* **Overview**

The **TechXpress E-commerce Platform** is an **ASP.NET Core-based web application** designed to provide a **scalable and maintainable** solution for selling electronics. It will support **product browsing, cart management, order processing**, and **payment integration** using **Stripe**. Additionally, an **admin panel** will be included for managing products, categories, and orders.

* **Objectives**
  + Develop a **user-friendly** and **secure** e-commerce platform.
  + Implement **NTier Architecture** for **better maintainability**.
  + Ensure **secure authentication** **and authorization** using **ASP.NET Identity.**
  + Provide a **seamless checkout experience** with **Stripe payment integration.**
  + Deploy the system on **Microsoft Azure** for scalability.
* **Scope**
  + Product listing, filtering, and search.
  + Shopping cart and checkout process.
  + User authentication and role-based access control (RBAC).
  + Order placement and payment processing.
  + Admin panel for managing products and orders.

1. **Project Plan:**

* **Timeline (Gantt Chart)**
* **Milestones & Deliverables**

|  |  |  |
| --- | --- | --- |
| Milestone | Deliverables | Due Date |
| Initial Setup | **NTier Architecture, User Authentication** | End of Week 1 |
| Shopping Cart & Admin Panel | **Shopping Cart, Role-based Access** | End of Week 2 |
| Payment & Order Management | **Stripe Integration, Order History** | End of Week 3 |
| Final Testing & Deployment | **Fully Functional Platform** | End of Week 4 |

|  |  |
| --- | --- |
| Week | Tasks |
| Week 1 | **Initial setup, database design, authentication, product listing** |
| Week 2 | **Shopping cart, role-based access, admin panel** |
| Week 3 | **Order placement, Stripe integration, user profiles** |
| Week 4 | **UI enhancements, testing, deployment** |

1. **Task Assignment & Roles:**

|  |  |  |
| --- | --- | --- |
| Task | Assigned To | Description |
| Database Schema Design | **Backend (Belal)** | **Define tables, relationships using EF Core** |
| User Authentication | **Backend (Belal)** | **Implement ASP.NET Identity** |
| Product Listing | **Frontend Developer (Malak)** | **Create UI for browsing products** |
| Shopping Cart | **Full Stack Developer (Malak)** | **Develop cart management functionality** |
| Admin Panel | **Full Stack Developer (Yasmeen)** | **Create dashboard for product & order management** |
| Payment Integration | **Backend Developer (Yasmeen)** | **Integrate Stripe API for payments** |
| Testing | **QC Engineer (Belal)** | **Perform functional & usability testing** |
| Deployment | **DevOps Engineer (Yasmeen)** | Deploy system on Azure |

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

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Impact | Likelihood | Mitigation Strategy |
| Database Failure | **High** | **Medium** | **Regular backups, failover strategy** |
| Security Vulnerabilities | **High** | **High** | **Implement authentication, secure API calls** |
| Payment Gateway Issues | **High** | **Medium** | **Use sandbox testing, monitor transaction failures** |
| Scope Creep | **Medium** | **High** | **Define clear requirements & avoid unnecessary changes** |
| Deployment Errors | **High** | **Medium** | **Use CI/CD pipeline for smooth releases** |
| Performance Issues | **Medium** | **Medium** | **Optimize queries, caching strategies** |

1. **Key Performance Indicators (KPIs):**

|  |  |  |
| --- | --- | --- |
| KPI | Target | Measurement |
| System Uptime | **99.9%** | **Monitor server logs** |
| Response Time | **< 2 seconds** | **Load testing** |
| User Registration Rate | **500+ users in 1 month** | **Track sign-ups** |
| Successful Transactions | **95%+ success rate** | **Stripe payment logs** |
| Cart Abandonment Rate | **< 30%** | **Analyze user behavior** |
| Bug Resolution Time | **< 24 hours** | **Ticket tracking** |

* **Requirements Gathering**

1. **Stakeholders:**Customers, Admins, Delivery Staff, Payment Gateway, Business Owners **.**
2. **User Stories:**

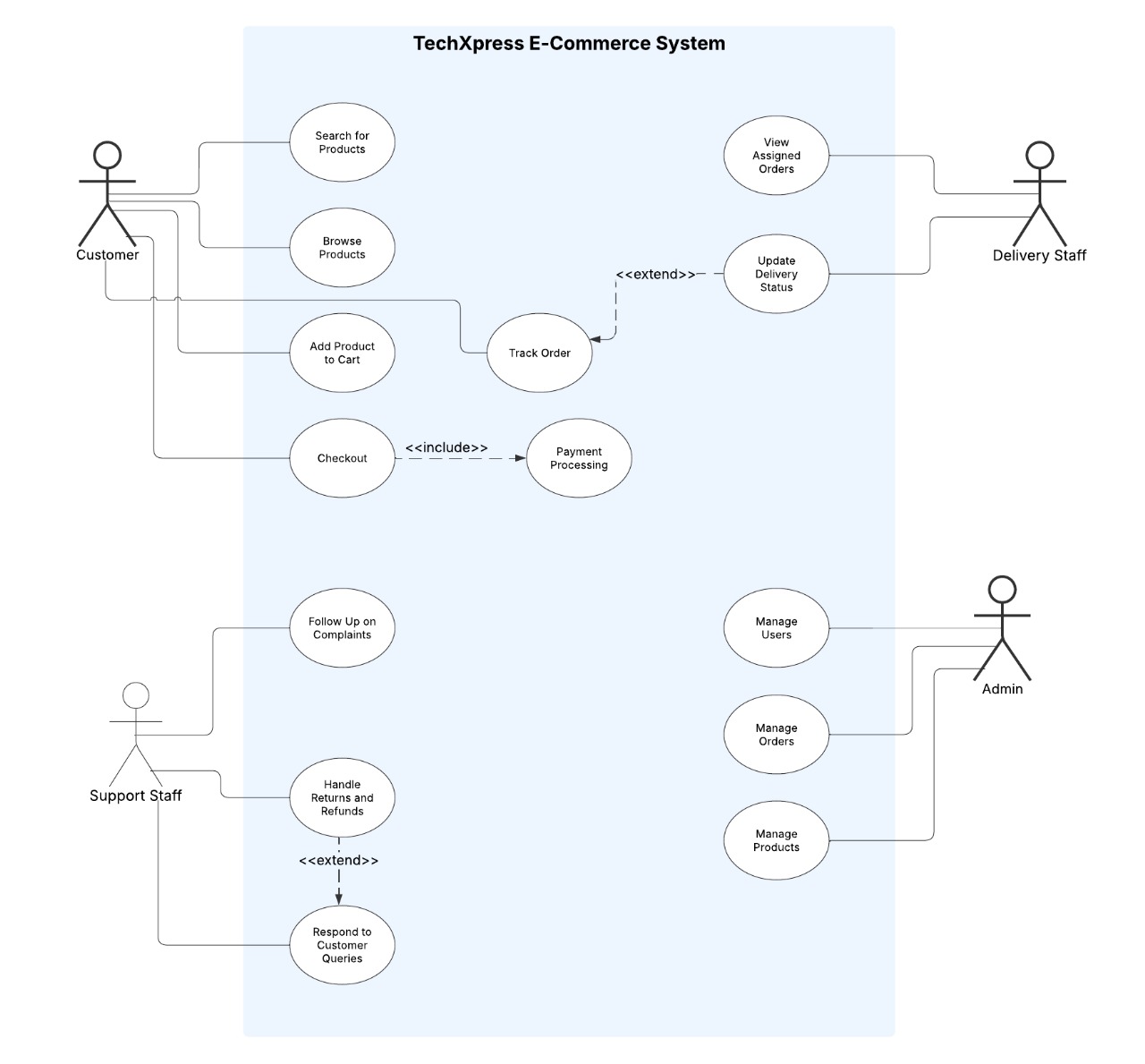
Search products, Add to cart, Checkout, Track order, Manage products, Handle refunds.

1. Functional Requirements:

Authentication, Shopping cart, Order processing, Admin dashboard, Payment integration.

1. Non-Functional Requirements:

Performance (<2s response time), Security (SSL/TLS, hashed passwords), Scalability (500+ users), Mobile-friendly UI

* **System Analysis & Design**
  + **Problem Statement & Objectives:**
    - **Use Cases Diagram:  
        
        
      **
    - **Use Cases Descriptions:**
      * **Use Case: Checkout**
        + **Actor:** Customer
        + **Description:**The system shall allow customers to finalize their purchase by selecting a payment method and completing the transaction.
        + **Preconditions:**

The customer must have items in their shopping cart.

The customer must be logged in.

* + - * + **Main Flow**:

The customer proceeds to checkout.

The system displays the order summary, including items, total price, and shipping details.

The customer selects a payment method (e.g., credit card, PayPal, Stripe).

The system processes the payment.

If successful, the system confirms the order and updates stock.

The customer receives an order confirmation.

* + - * + **Alternate Flow:**

If payment fails, the system prompts the customer to retry or select a different payment method.

* + - * + **Postconditions:**

The order is placed successfully, and the payment is recorded.

* **Use Case: Track Order:**
  + **Actor:** Customer
  + **Description:**

The system shall allow customers to monitor the status of their orders in real-time.

* + **Preconditions:**
    - The customer must have placed at least one order.
  + **Main Flow:**

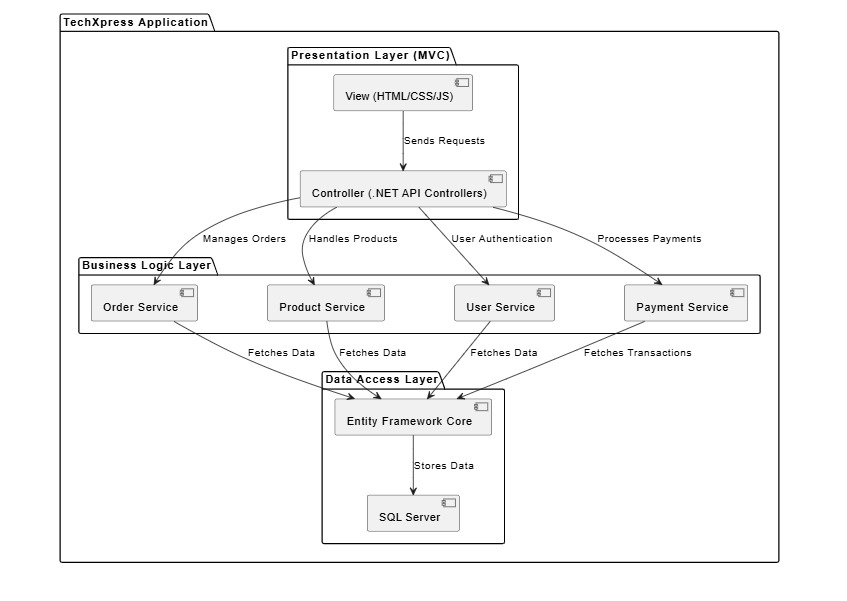
1. The customer navigates to the "My Orders" section.
2. The system retrieves and displays the current status of all orders.
3. The customer selects an order to view detailed tracking information.
4. The system displays the latest order updates (e.g., processing, shipped, out for delivery, delivered).  
   * **Alternate Flow:**
     + If no orders are found, the system displays a message: "You have no orders yet."
   * **Postconditions:**
     + The customer is informed about the current status of their order.

* Use Case: Handle Returns and Refunds
  + **Actor:** Support Staff
  + **Description:**  
    The system shall allow support staff to manage customer requests for product returns and refunds.
  + **Preconditions:**
    - A customer must have previously placed an order.
    - The request for a return or refund must be submitted within the allowed time frame.
  + **Main Flow:**

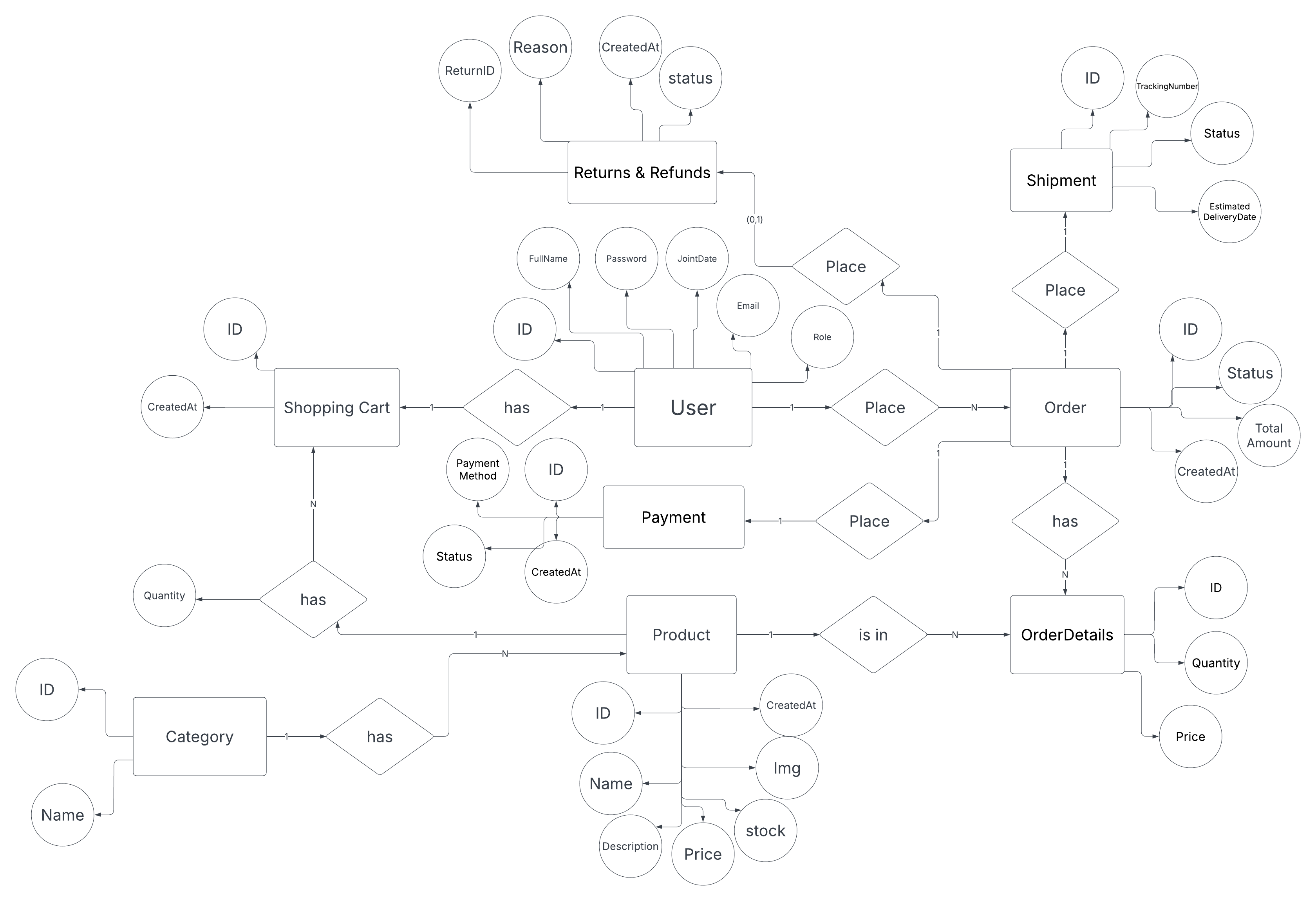
1. The support staff navigates to the "Returns & Refunds" section.
2. The system displays a list of return requests submitted by customers.
3. The support staff reviews the request and verifies eligibility based on the return policy.
4. The system updates the request status to "Approved" or "Rejected" based on the evaluation.
5. If approved, the system initiates the refund process and notifies the customer.
6. The refund is processed, and the order status is updated accordingly.  
   * **Alternate Flow:**
     + If the return request is rejected, the system notifies the customer with a reason.
   * **Postconditions:**
     + The customer's return request is resolved, and necessary actions are taken.

* **Functional & Non-Functional Requirements**
* **Functional Requirements:**
  + The system shall allow admins to add, edit, and delete products.
  + The system shall support product categorization.
  + Customers shall be able to browse and search for products.
  + The system shall allow users to register, log in, and manage their profiles.
  + Customers shall be able to add, remove, and modify products in their shopping cart.
  + The system shall calculate the total price dynamically.
  + Customers shall be able to complete purchases through an integrated Stripe payment gateway.
  + The system shall process orders and update inventory accordingly.
  + Customers shall receive order confirmation notifications.
  + Admins shall be able to view and manage orders.
  + The system shall provide an admin dashboard with product and order management functionalities.
  + Admins shall be able to view sales analytics and manage users.
* **Non-Functional Requirements**
  + The system shall respond to user actions within 2 seconds under normal load.
  + It shall support at least 500 concurrent users without degradation.
  + All user passwords shall be securely hashed and stored.
  + The system shall encrypt sensitive transactions using SSL/TLS.
  + Role-based access control (RBAC) shall restrict unauthorized access.
  + The system shall follow NTier Architecture for separation of concerns.
  + Future enhancements shall be possible without major restructuring.
  + The UI shall be responsive and mobile-friendly using Bootstrap.
  + Navigation shall be intuitive, ensuring ease of use for customers and admins.
  + It shall maintain 99.9% uptime for service availability.

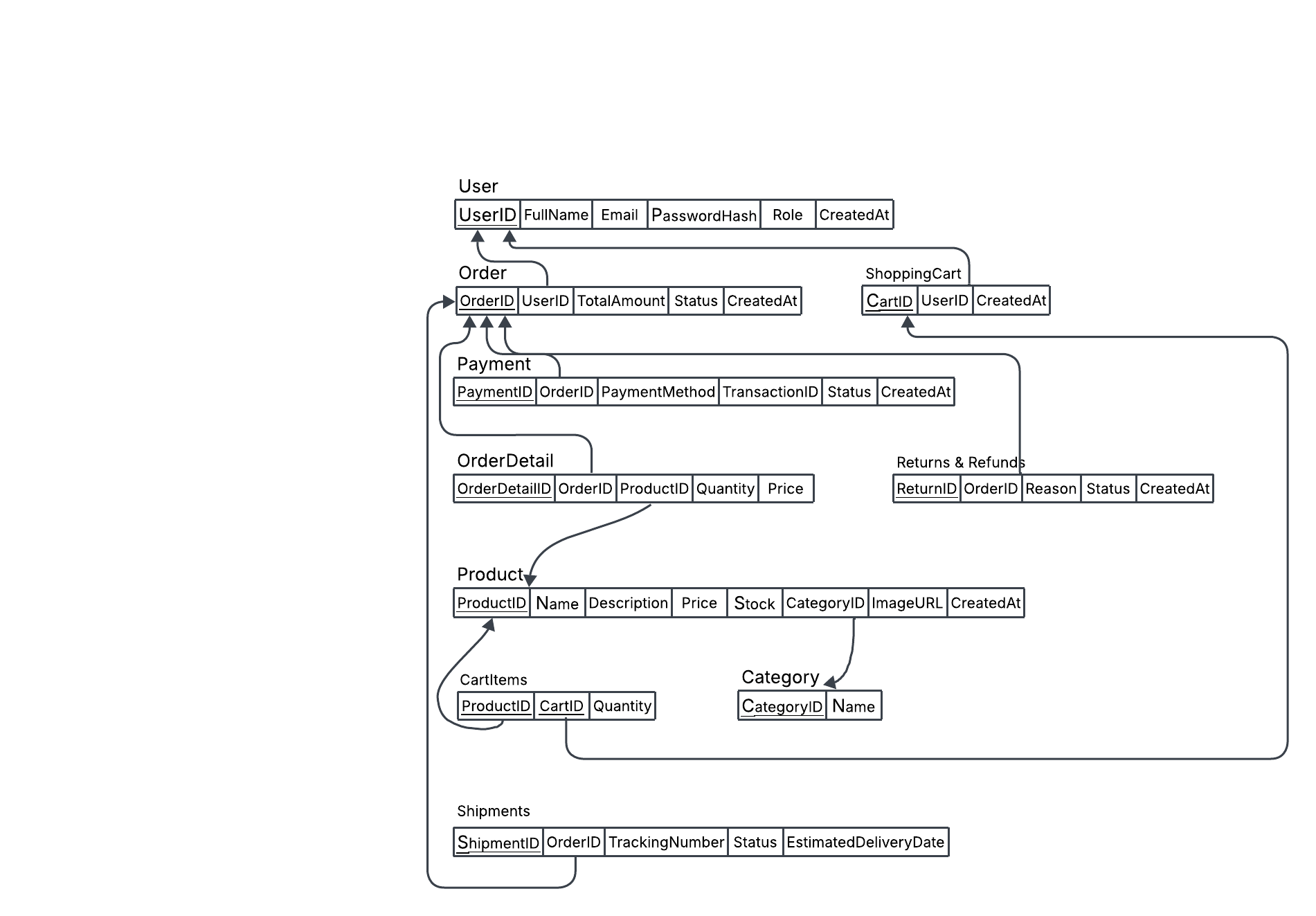
**• Software Architecture:**

****

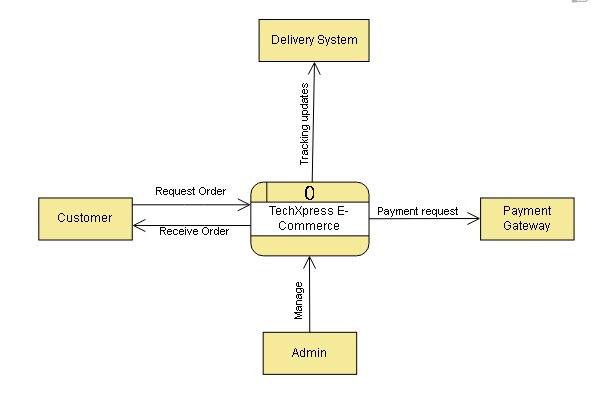
* **Database Design & Data Modeling:**
  + **ERD (Entity-Relationship Diagram)**

****

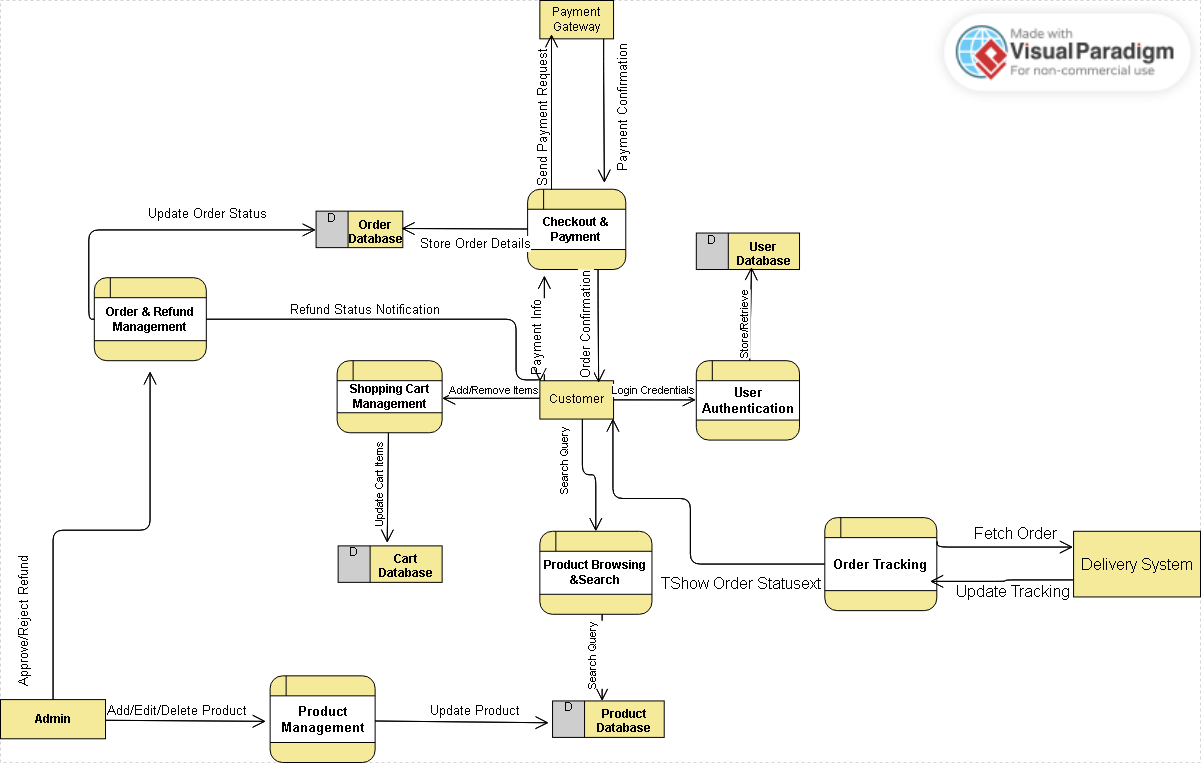
* **Logical & Physical Schema**

****

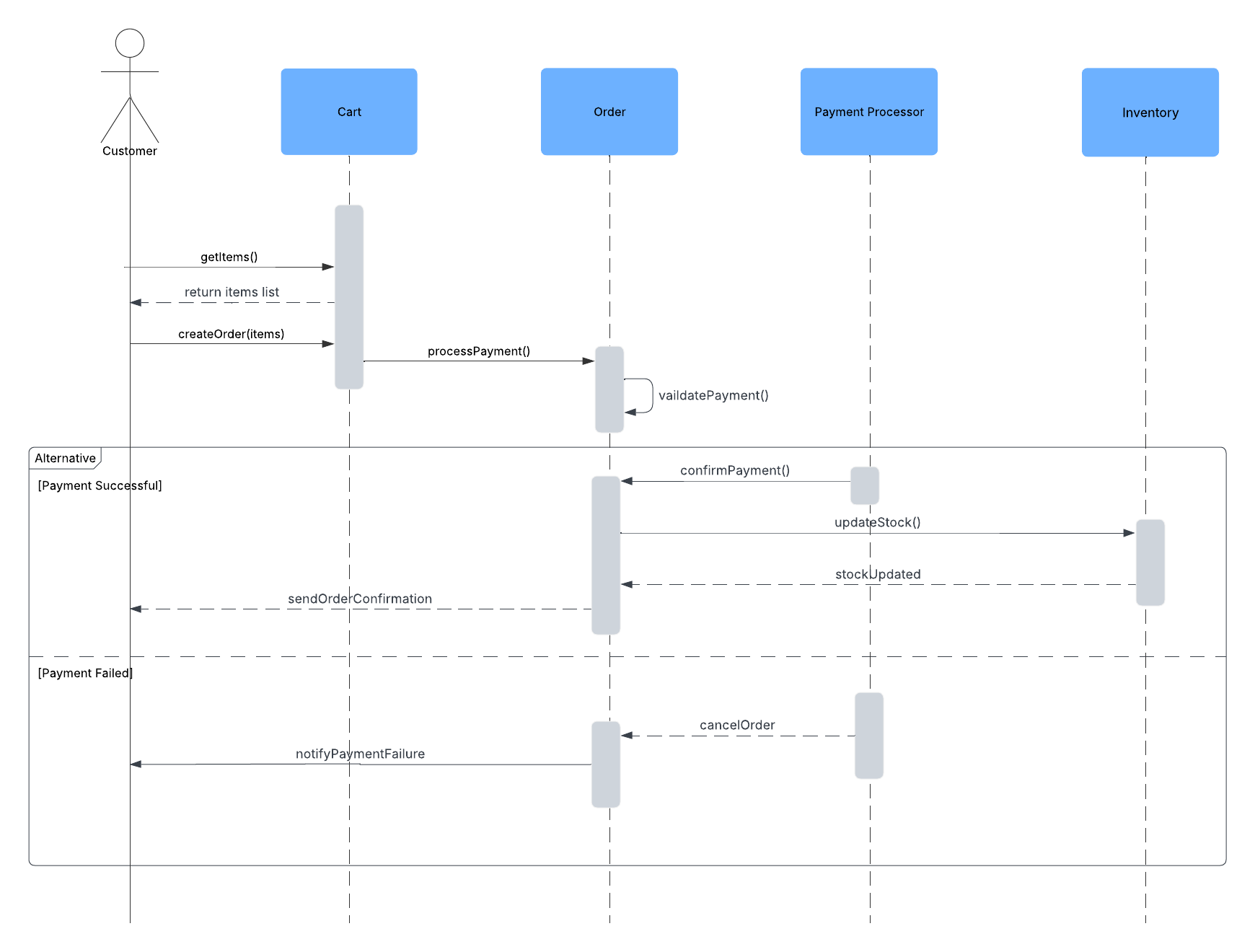
* **Data Flow & System Behavior:**
  + **DFD (Data Flow Diagram)**
    - **Context Level:**

****

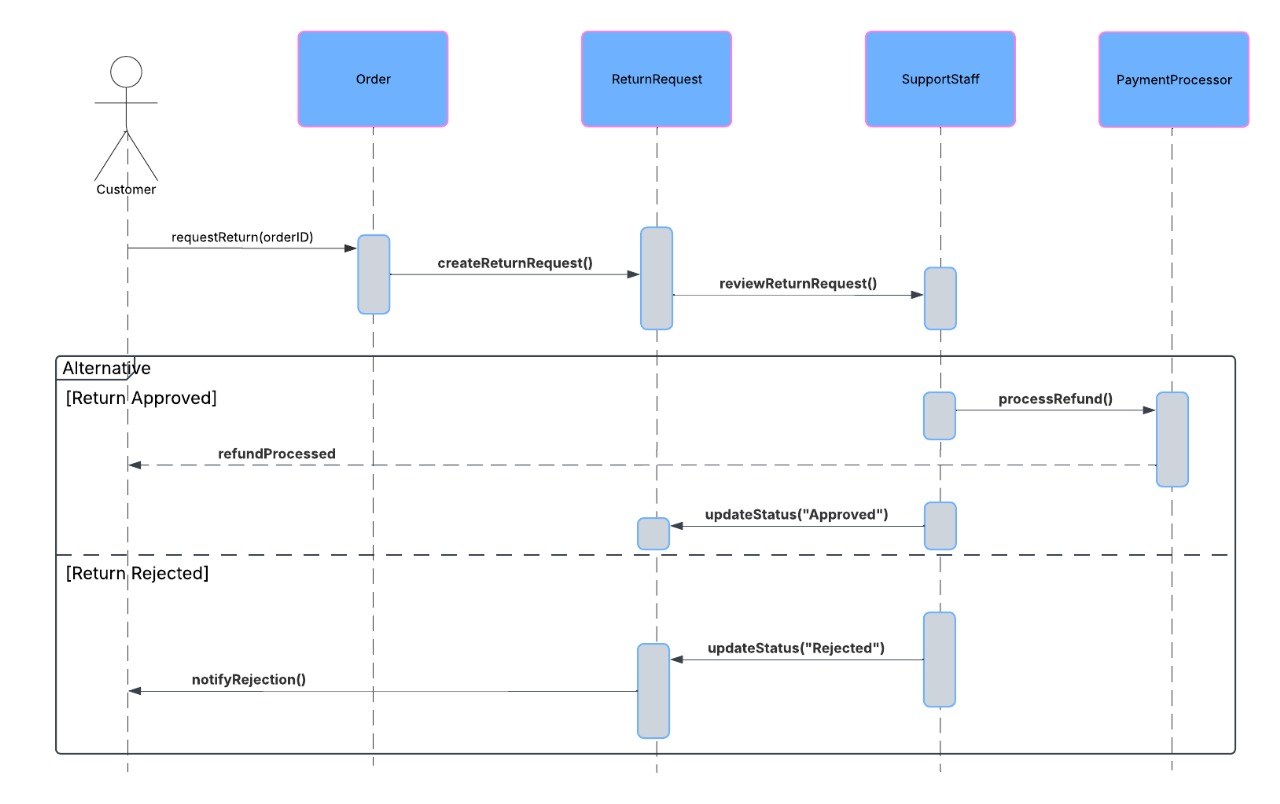
* **Level - 0:**

****

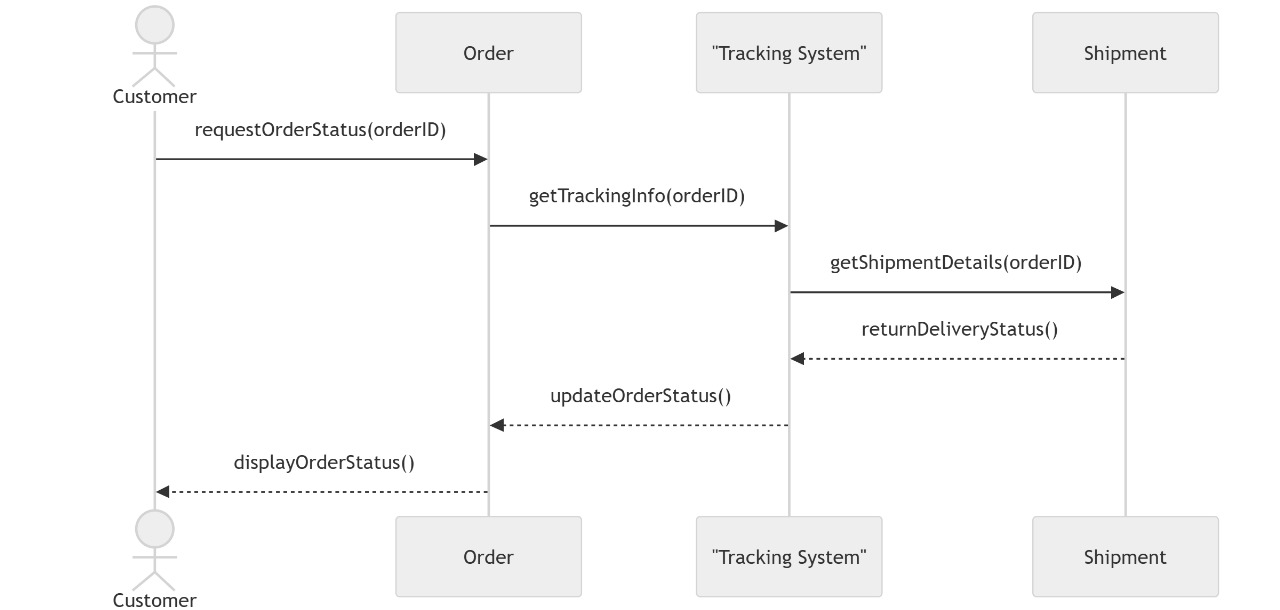
* **Sequence Diagrams:**
  + **Checkout Diagram:**



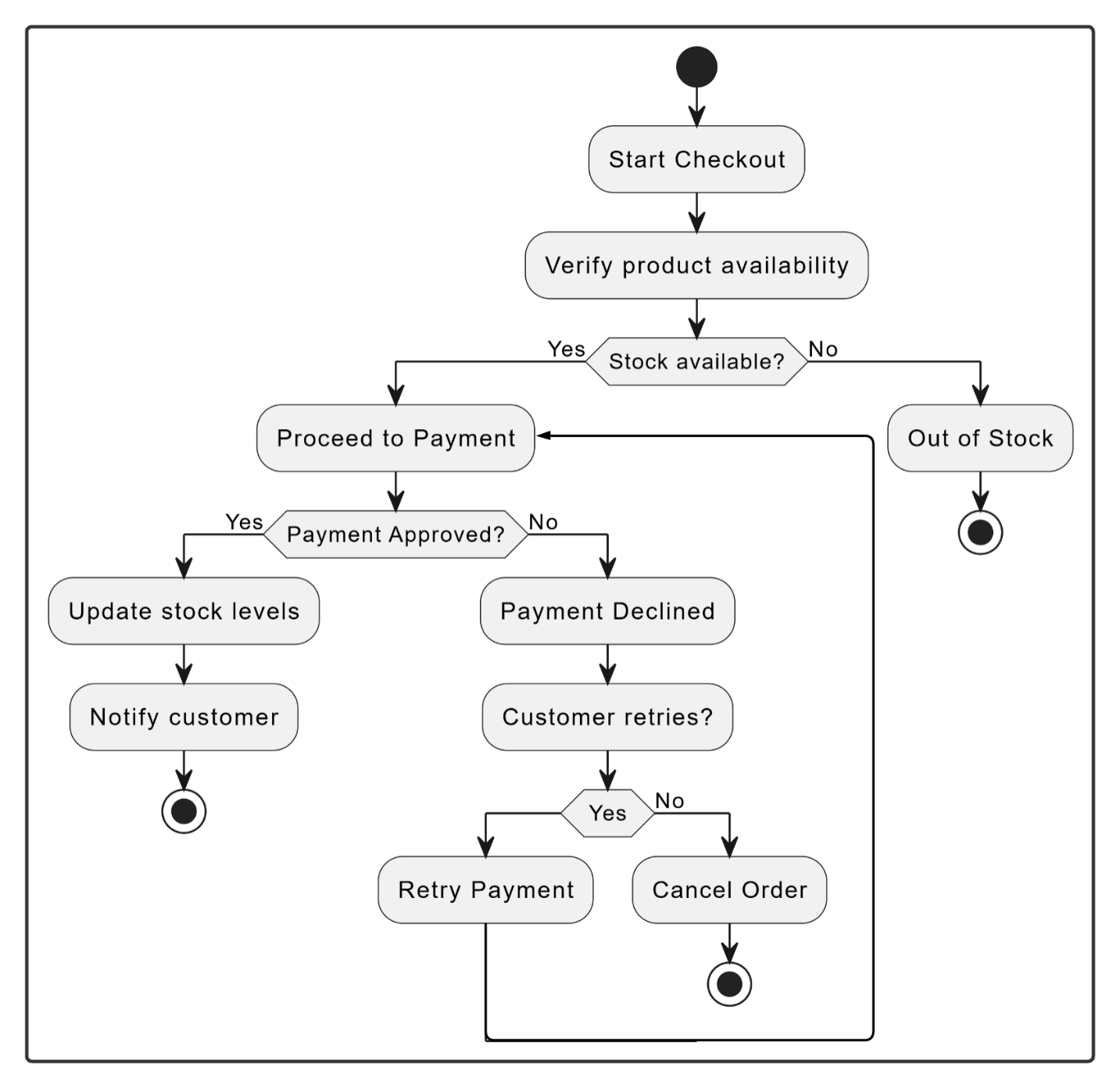
* + **Handle Return Diagram:**

****

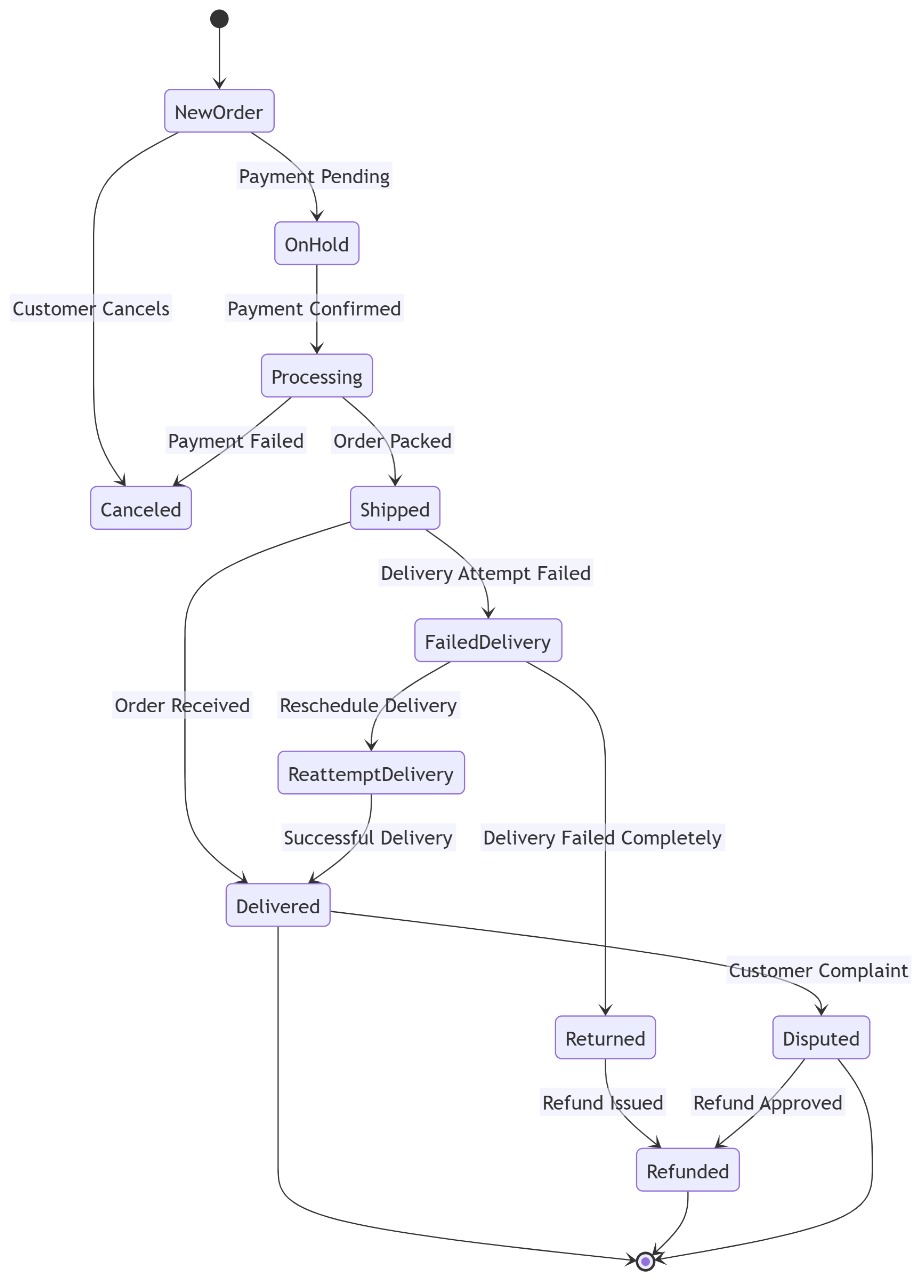
* + **Track Order:**

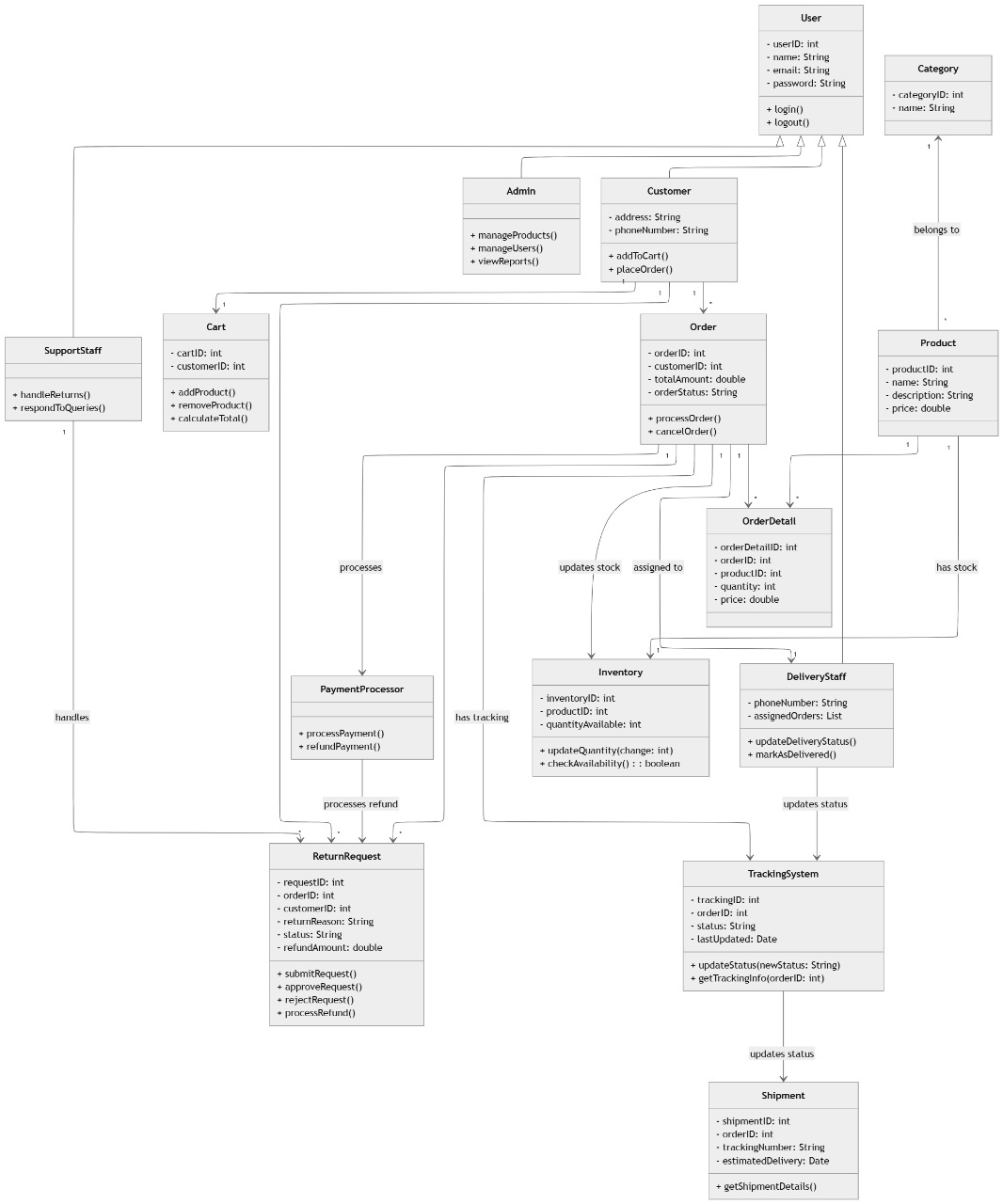
****

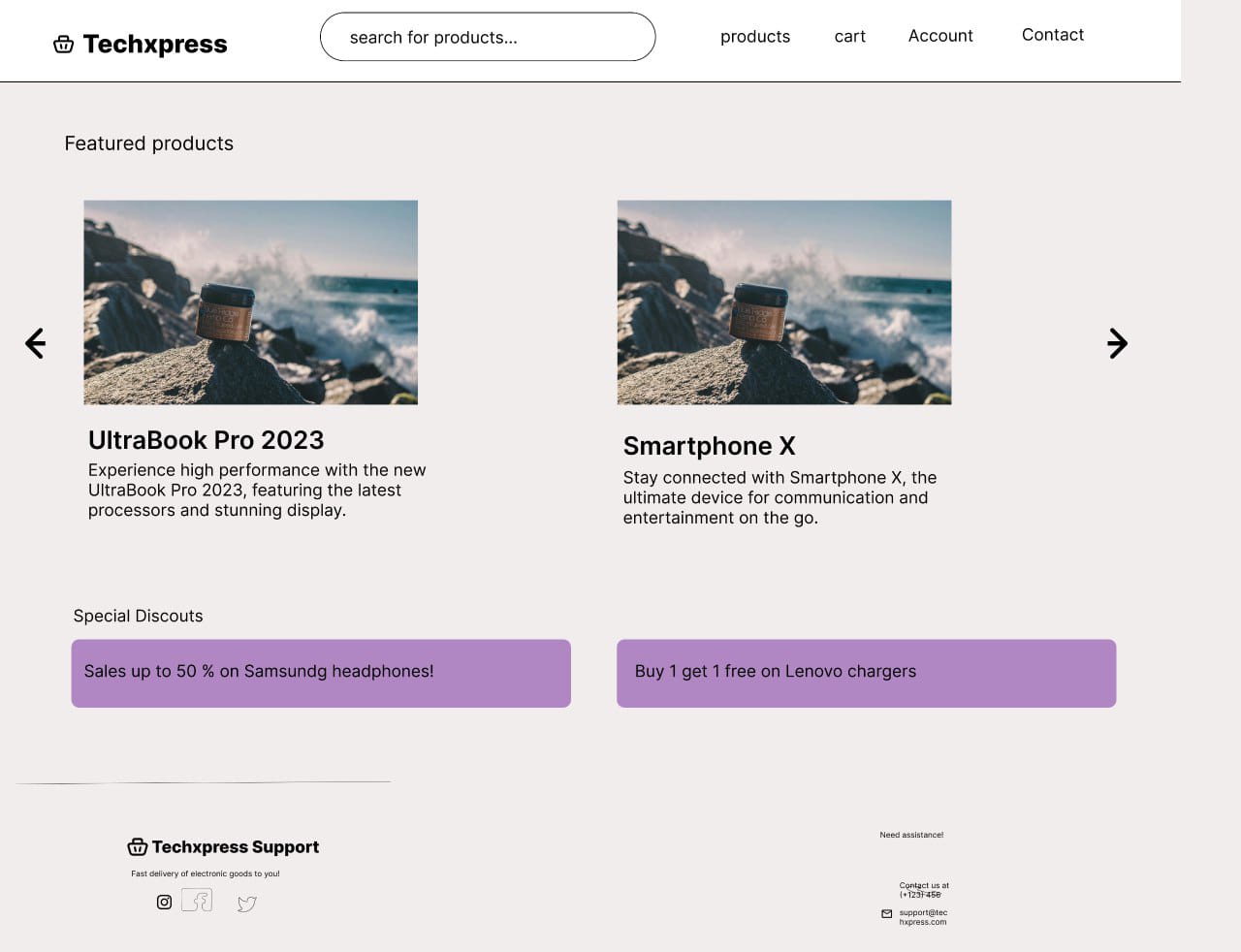
* **Checkout Activity Diagram:**

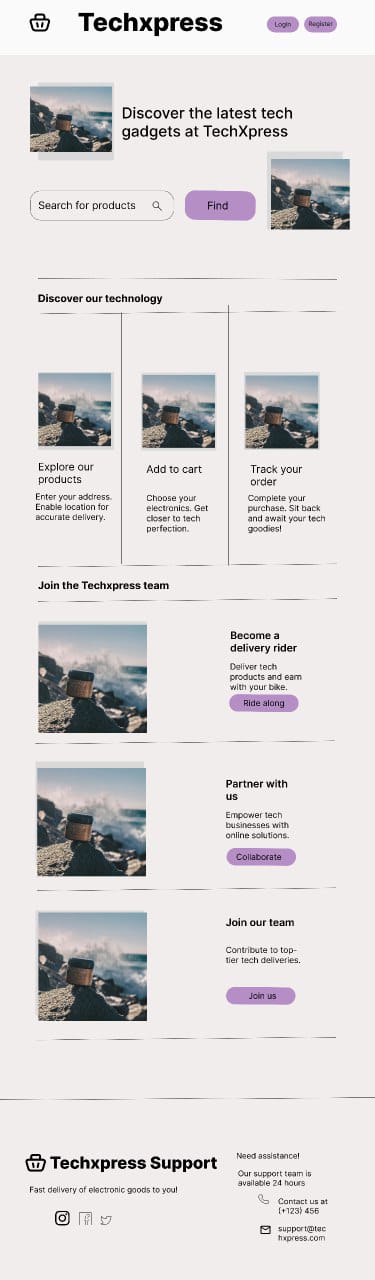
****

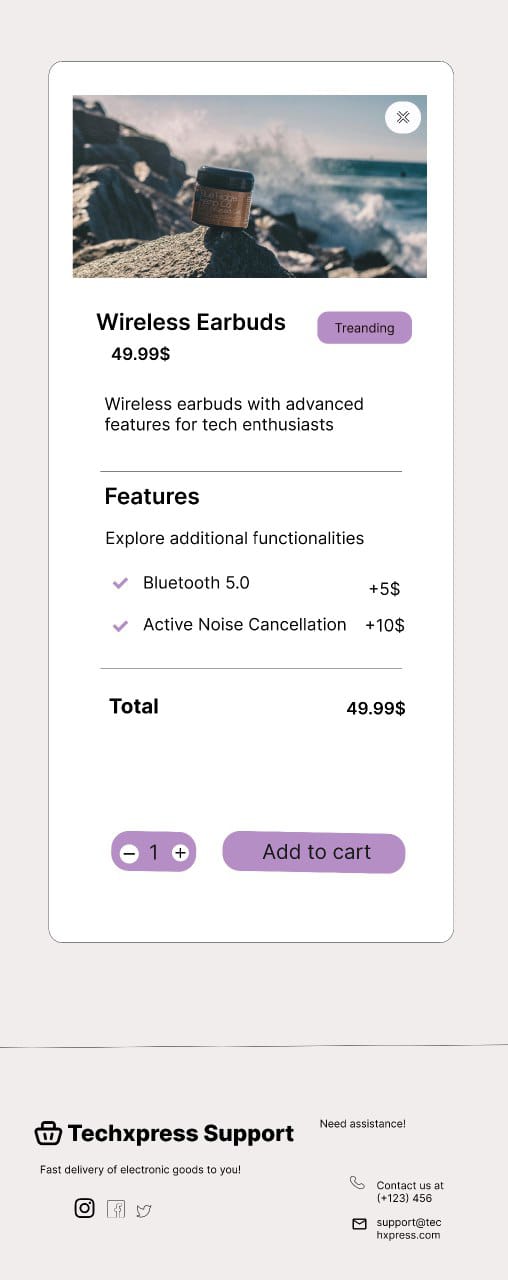
* **State Diagram:**

****

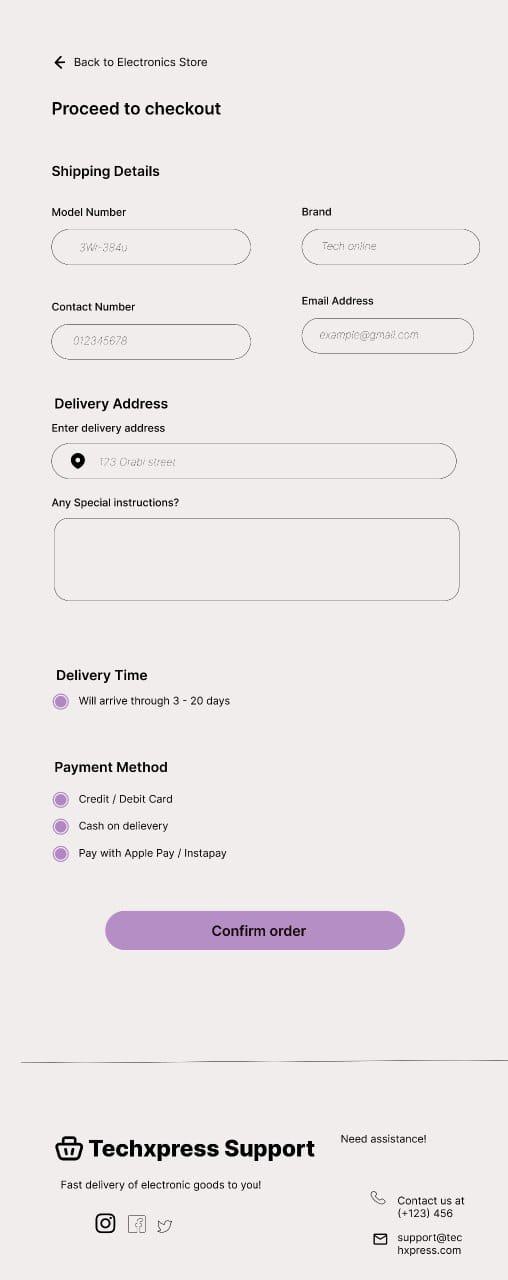
* **Class Diagram:   
  **
  + **UI/UX Design & Prototyping:**
    - **Home Page:**

****

* **Landing Page:  
  **
* **Product Details:**

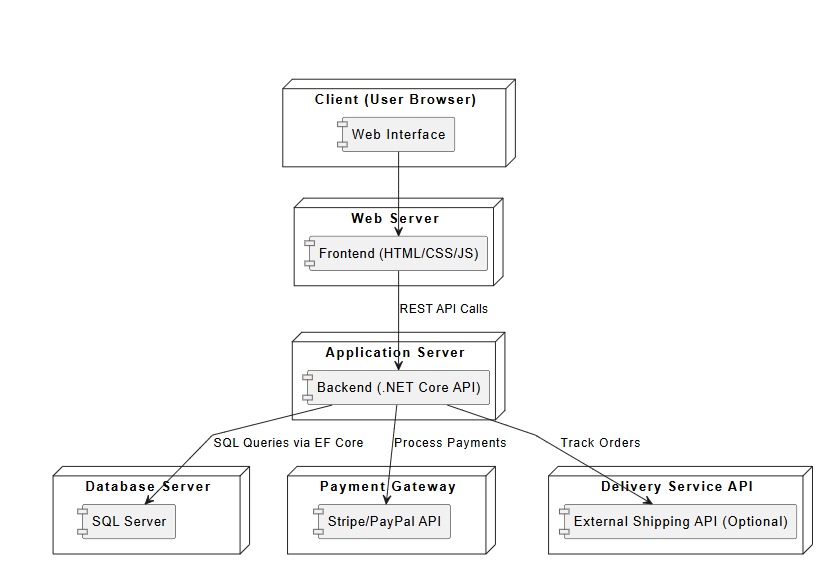
****

* **Cart and delivery page:**

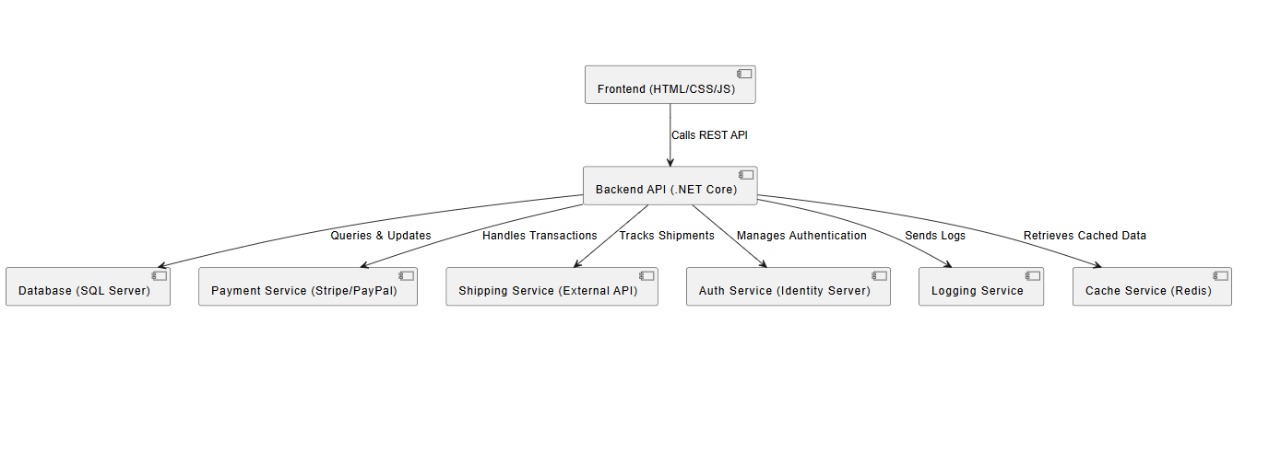
****

* **System Deployment & Integration:**
  + **Technology Stack:**
    - **Backend:**
      * **Programming language:** C#
      * **Framework:** ASP.NET CORE
      * **ORM:** Entity Framework Core
    - **Frontend:**
      * **Language:** HTML, CSS, JavaScript.
      * **CSS Frameworks:** Bootstrap
      * **UI/UX Tools:** Figma
    - **Database**:
      * **Microsoft SQL Server**.
      * **Entity Framework core migrations**.

* + **Deployment Diagram:**

****

* **Component Diagram:**

****