***Document for Shoe Store Web Application using .NET Core***

**1. Problem Definition**

**1.1. Problem Abstraction**

**Managing a shoe store involves handling a wide range of activities, including inventory management, customer orders, payment processing, and customer engagement. Each of these areas plays a critical role in ensuring the smooth operation of the business and meeting customer expectations. However, traditional methods, such as manual record-keeping or using isolated software systems, often result in significant challenges.**

**Key Challenges of Traditional Methods:**

1. **Inefficiencies: Manual processes are time-consuming and prone to bottlenecks. For instance, manually updating inventory records after each sale can delay stock replenishment and lead to missed sales opportunities.**
2. **Inaccuracies: Human error in data entry can lead to discrepancies in inventory levels, miscommunication with customers, and incorrect financial records.**
3. **Lack of Scalability: As the business grows, managing increasing volumes of data and transactions manually becomes impractical, limiting the ability to expand operations effectively.**
4. **Limited Integration: Isolated systems for sales, inventory, and customer management result in a lack of centralized data, making it difficult to generate accurate reports or gain insights into business performance.**
5. **Customer Dissatisfaction: Without accurate stock information or a streamlined ordering process, customers may face delays or disappointment, negatively impacting their shopping experience and loyalty.**

**These challenges collectively hinder the growth and operational efficiency of the store, making it difficult to compete in an increasingly digital and customer-focused marketplace. To address these issues, a comprehensive and integrated solution is required that automates core processes, reduces errors, and enhances the overall customer experience.**

**The proposed web-based application aims to modernize and streamline shoe store operations. By leveraging advanced technologies and a user-friendly interface, the system will support efficient inventory management, simplify customer orders, facilitate secure payment processing, and improve customer engagement through personalized features and real-time updates.**

**1.2. The Current System**

**Currently, the store relies on manual processes or basic software for managing inventory and orders. Some of the limitations include:**

1. **Manual Errors in Inventory Tracking: Manual updates to inventory records are prone to mistakes, leading to discrepancies between actual stock levels and recorded data. These errors can result in overstocking or understocking, directly impacting sales and operational efficiency.**
2. **Difficulty in Providing Accurate Stock Updates: Without a real-time inventory system, staff may struggle to provide customers with accurate information about product availability, leading to frustration and lost sales opportunities.**
3. **Limited Integration Between Systems: Existing solutions often function in isolation, meaning there is little to no synchronization between sales, inventory, and customer management systems. This lack of integration hinders the ability to make informed decisions and slows down operational workflows.**
4. **Restricted Customer Base: The absence of an online presence limits the store's reach to local foot traffic, significantly reducing revenue potential. Moreover, customers increasingly prefer the convenience of browsing and shopping online, putting the store at a disadvantage in the digital age.**

**These shortcomings highlight the pressing need for a more robust and interconnected system that can handle the complexities of modern retail operations while enhancing customer satisfaction and driving business growth.**

**1.3. The Proposed System**

**The proposed system is a web-based shoe store application designed to address the challenges of traditional retail operations. By automating inventory management, streamlining customer orders, and supporting administrative functions, the system aims to enhance both operational efficiency and the customer experience.**

**1.3.1. Boundaries of the System**

**Included Features:**

1. **Inventory Management:**
   * **Real-time stock updates.**
   * **Product categorization and search functionality.**
2. **Customer Registration and Login:**
   * **Secure user authentication.**
   * **Profile management.**
3. **Online Shopping Cart and Checkout:**
   * **Add/remove products from cart.**
   * **View order summary.**
   * **Multiple payment options.**
4. **Order and Payment Management:**
   * **Track order status.**
   * **Manage returns and refunds.**
5. **Basic Analytics and Reporting:**
   * **Generate sales reports.**
   * **View inventory performance metrics.**

**Excluded Features:**

1. **Advanced Analytics:**
   * **Predictive sales forecasting.**
   * **Customer behavior analysis.**
2. **Third-Party Integrations:**
   * **Shipping APIs.**
   * **External payment gateways.**

**1.3.2. Hardware and Software Requirements**

**Hardware Requirements:**

1. **Server:**
   * **Minimum 4-core CPU.**
   * **8 GB RAM.**
   * **100 GB SSD storage.**
2. **Client:**
   * **Devices with a modern web browser (e.g., Chrome, Edge, Safari).**

**Software Requirements:**

1. **Backend:**
   * **.NET Core 8.0 for API and business logic.**
2. **Frontend:**
   * **Razor Pages with Bootstrap for responsive UI design.**
3. **Database:**
   * **SQL Server for data storage and management.**
4. **Development Tools:**
   * **Visual Studio 2022 as the primary Integrated Development Environment (IDE).**
5. **Version Control:**
   * **Git for collaborative development and source control.**

**These specifications ensure that the system is robust, scalable, and capable of supporting both current operations and future growth. By implementing these technologies, the proposed solution aims to provide a seamless experience for users and administrators alike.**

**2. Customer Requirements Specification**

**2.1. Users of the System**

**- Customers:**

1. **Account Management:**
   * **Register and log in to their accounts securely.**
   * **Update personal details such as name, email, and password.**
2. **Product Browsing:**
   * **Browse and search for available products by categories, sizes, and price ranges.**
   * **View detailed product descriptions, including images and reviews.**
3. **Shopping Cart and Checkout:**
   * **Add items to a shopping cart and adjust quantities.**
   * **Proceed to checkout with options for delivery or pickup.**
   * **Choose from multiple payment options, including credit cards and digital wallets.**
4. **Order History and Tracking:**
   * **View past orders with detailed breakdowns of items and costs.**
   * **Track the status of current orders in real-time.**

**- Admins:**

1. **Inventory Management:**
   * **Add, update, or delete product details, including prices, sizes, and availability.**
   * **Manage product categories and featured items.**
2. **Order Processing:**
   * **Monitor customer orders and update their statuses (e.g., "Shipped," "Delivered").**
   * **Handle customer inquiries regarding order issues or returns.**
3. **Reporting and Analytics:**
   * **Generate basic sales reports, including revenue and popular products.**
   * **Monitor inventory levels to identify trends and prevent stockouts.**
4. **Customer Management:**
   * **View and update customer details as needed.**
   * **Address customer queries and feedback efficiently.**

**2.2. System Functions**

**- Customer Features:**

1. **Registration and Login:**
   * **Create a new account with secure password hashing.**
   * **Log in with multi-factor authentication for added security.**
2. **Profile Management:**
   * **Update account information, including personal details and passwords.**
   * **Manage saved addresses for quicker checkouts.**
3. **Product Browsing:**
   * **View a comprehensive catalog of products.**
   * **Filter and sort items by various attributes (e.g., size, price, popularity).**
4. **Shopping and Checkout:**
   * **Add items to a shopping cart, with an option to save for later.**
   * **Complete purchases with a seamless checkout process.**
   * **Receive order confirmation and payment receipts via email.**
5. **Order Tracking:**
   * **Monitor the status of orders from processing to delivery.**
   * **Request returns or refunds directly through the platform.**

**- Admin Features:**

1. **Product Management:**
   * **Perform CRUD (Create, Read, Update, Delete) operations for inventory.**
   * **Add promotional offers and discounts to specific products.**
2. **Order Management:**
   * **Review and approve or reject pending orders.**
   * **Update order statuses with automatic customer notifications.**
3. **Analytics and Reports:**
   * **Generate reports for daily, weekly, or monthly sales.**
   * **Identify best-selling products and categories.**
4. **Customer Support:**
   * **Access a dashboard for managing customer queries.**
   * **Resolve issues related to orders, accounts, or payments.**

**By implementing these features, the system will cater to the needs of both customers and administrators, ensuring a comprehensive and user-friendly experience.**

**3. System Designs**

**3.1. Entity Relationship Diagram (ERD)**

**The ERD illustrates the relationships between the main entities in the system, such as Users, Products, Orders, and OrderDetails. (Detailed diagram will be created using tools like draw.io or Lucidchart.)**

**3.2. Database Design**

**Tables and Their Structures**

1. **Users  
   This table stores information about customers and admins who use the system.  
   Columns:**
   * **Id (PK): Unique identifier for the user (int).**
   * **Name: Name of the user (nvarchar(100)).**
   * **Email: Email address of the user (nvarchar(100)).**
   * **PasswordHash: Encrypted password for authentication (nvarchar(255)).**
   * **Role: Defines the user's role (e.g., Admin or Customer) (nvarchar(50)).**
   * **DateCreated: The date and time the user was registered (datetime).**
2. **Products  
   This table stores details about the shoes available for purchase.  
   Columns:**
   * **Id (PK): Unique identifier for the product (int).**
   * **Name: Name of the shoe (nvarchar(100)).**
   * **Description: A brief description of the shoe (nvarchar(255)).**
   * **Price: Price of the product (decimal).**
   * **Stock: Quantity available in stock (int).**
   * **CategoryId: Foreign key referencing the Categories table (int).**
   * **ImageUrl: URL path to the product image (nvarchar(255)).**
3. **Categories  
   This table stores product category information for better organization.  
   Columns:**
   * **Id (PK): Unique identifier for the category (int).**
   * **Name: Name of the category (nvarchar(50)).**
4. **Orders  
   This table tracks customer orders and their statuses.  
   Columns:**
   * **Id (PK): Unique identifier for the order (int).**
   * **UserId (FK): Foreign key referencing the Users table (int).**
   * **OrderDate: Date and time the order was placed (datetime).**
   * **TotalAmount: Total price of the order (decimal).**
   * **OrderStatus: Current status of the order (e.g., Pending, Shipped, Delivered) (nvarchar(50)).**
5. **OrderDetails  
   This table provides a breakdown of items in each order.  
   Columns:**
   * **Id (PK): Unique identifier for the detail row (int).**
   * **OrderId (FK): Foreign key referencing the Orders table (int).**
   * **ProductId (FK): Foreign key referencing the Products table (int).**
   * **Quantity: Number of units of the product ordered (int).**
   * **UnitPrice: Price per unit of the product at the time of the order (decimal).**
6. **Carts  
   This table tracks products added to a customer's cart before purchase.  
   Columns:**
   * **Id (PK): Unique identifier for the cart entry (int).**
   * **UserId (FK): Foreign key referencing the Users table (int).**
   * **ShoeId (FK): Foreign key referencing the Products table (int).**
   * **Quantity: Number of units of the product in the cart (int).**

**Relationships Between Tables**

* **Users and Orders:  
  One user can place multiple orders, but each order belongs to one user (1:N relationship).**
* **Orders and OrderDetails:  
  Each order can have multiple order details, but each order detail belongs to one order (1:N relationship).**
* **Products and OrderDetails:  
  Each product can appear in multiple order details, and each order detail references one product (1:N relationship).**
* **Users and Carts:  
  Each user can have multiple items in their cart, but each cart item belongs to one user (1:N relationship).**
* **Products and Carts:  
  Each product can appear in multiple carts, and each cart entry references one product (1:N relationship).**
* **Categories and Products:  
  Each category can have multiple products, but each product belongs to one category (1:N relationship).**

**3.3. Sitemap**

* **Public Pages:**
  + **Home**
  + **Product Listing**
  + **Product Details**
  + **Login/Registration**
  + **Cart and Checkout**
* **Admin Pages:**
  + **Dashboard**
  + **Product Management**
  + **Order Management**
  + **Sales Reports**
  + **User Management**

**3.4. System Functions Design**

* **Login and Authentication:**
  + **Use ASP.NET Core Identity for secure login and role-based access.**
* **CRUD Operations:**
  + **Admins can add, edit, delete, and view products through a dedicated interface.**
* **Order Workflow:**
  + **Customers place orders, and admins process them, updating statuses accordingly.**

**4. Task Sheet**

* **Frontend Development:**
  + **Create responsive Razor Pages for customer-facing features.**
  + **Design admin interface with Bootstrap for consistency.**
* **Backend Development:**
  + **Implement controller logic for handling customer and admin actions.**
  + **Develop APIs for AJAX-based functionalities (e.g., product search).**
* **Database Implementation:**
  + **Set up SQL Server database and write migration scripts using Entity Framework.**
* **Testing and Validation:**
  + **Unit testing for critical business logic.**
  + **Manual testing for UI and functional workflows.**

**5. Validation Checklists**

* **Customer Features:**
  + **Ensure users can register and log in securely.**
  + **Verify that the shopping cart retains selected items.**
  + **Validate correct calculations for order totals.**
* **Admin Features:**
  + **Test CRUD operations for product inventory.**
  + **Confirm that orders can be processed and statuses updated correctly.**
* **Overall System:**
  + **Test for responsiveness across devices.**
  + **Verify error handling for invalid inputs.**
  + **Perform load testing to ensure the application can handle concurrent users.**