**SYSTEM REQUIREMENT SPECIFICATION DOCUMENT FOR E-BOOK WEB APPLICATION**

**Abstract:**

The eBook Web Application is a user-friendly platform designed to provide a diverse collection of digital books to readers. With an intuitive interface, users can easily explore various genres, find books using advanced search options, and receive personalized recommendations. The application supports features such as progress tracking, bookmarking, and interactive reading tools. It aims to create a vibrant reading community by offering discounts, user profiles, and responsive design for seamless reading across devices. The goal is to make reading enjoyable, accessible, and engaging for users of all interests and preferences.

**Functional Requirements:**

**1. User Module Functional Requirements:**

Search Books: Users can search for books based on criteria such as title, author, genre, or keywords. Results should be accurate, relevant, and displayed promptly.

View Book Details: Users can view detailed information about each book, including descriptions, author details, cover images, genre, publication date, and ratings.

User Account Management: Users can create new accounts or log in with existing credentials. Registered users have access to their reading history, saved books, bookmarks, and preferences.

Add to Wishlist: Users can add books to a wishlist for future reference. The wishlist allows users to save and manage desired books separately from their reading list.

Add to Reading List: Users can add books to their reading list, marking them for current or future reading. The reading list helps users keep track of books they plan to read.

Purchase eBooks: Users can purchase eBooks directly from the application. The system provides secure payment gateways and ensures a seamless checkout process.

**2. Administrator Module Functional Requirements:**

Manage eBook Database: Admins should be able to add new eBooks to the system, edit existing book details such as title, author, description, price, and availability, and remove eBooks from the platform.

User Management: Admins can manage user accounts, including creating new accounts, editing user profiles, resetting passwords, and viewing user activity such as reading history and purchases.

Generate Reports: The system allows administrators to generate reports on various aspects, such as popular books, sales trends, user demographics, and inventory levels.

Manage Orders and Notifications: Admins can view, update, and track all eBook orders. They can change order statuses, send order confirmations, shipping details, and delivery notifications to users via email or in-app notifications.

**Non-Functional Requirements :**

Usability: The application should have an intuitive and user-friendly interface, clear navigation, and easily accessible book details.

Security: Implement encryption for user data, secure payment gateways, user authentication, and access controls to prevent unauthorized access to eBooks and user information.

Performance: Ensure fast loading times for book searches, page navigation, and reading experiences. The application should handle concurrent users and large eBook files efficiently.

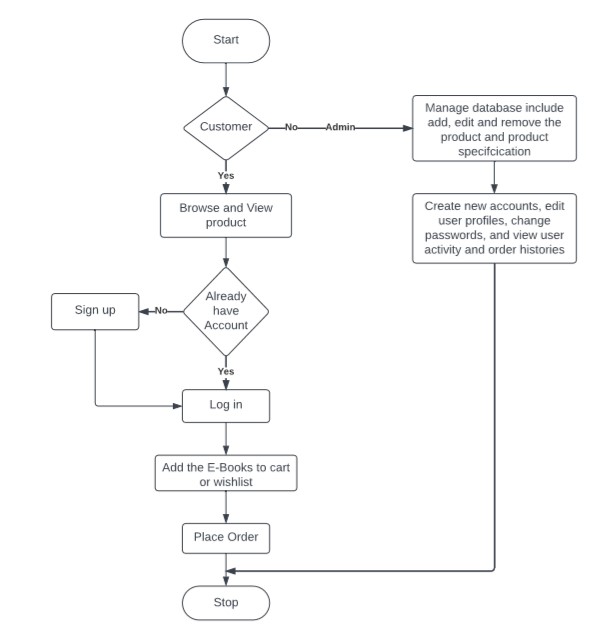
Maintainability: The system should be easy to maintain, update, and scale. Code should be well-documented, and updates should not disrupt user experience.

Scalability: The application should be scalable to accommodate a growing number of users, books, and transactions. It should support horizontal and vertical scaling as needed without compromising performance.

These requirements ensure that the eBook application meets user needs for discovering, purchasing, and reading eBooks while providing administrators with tools to manage the platform effectively.

**Flow Chart :**

1. Login: The first step is login into the system based on whether the user is customer or admin.
2. Customer Login: Customers can either log in with their registered credentials to access their accounts directly or sign up as new customers by providing their email and creating a password.
3. Admin Login: Admin gain access to the admin portal upon login, enabling them to manage the database, user accounts, and product listings.
4. Customer Actions: Once logged in, customers can browse products, view details, images, and prices. They can add items to their cart for future purchase, proceed to checkout, review selections, and make payments using various methods like credit cards or digital wallets.
5. Admin Actions: Admin can add new products, update existing listings, and delete products. They create customer accounts, manage user profiles, reset passwords, and oversee user activity, including login times and order history.
6. Order History and Activity: Customers can view past orders, check statuses, and track deliveries. Admin have a broader view, analyzing system-wide order history to track sales trends and popular items, aiding in inventory management and customer service.

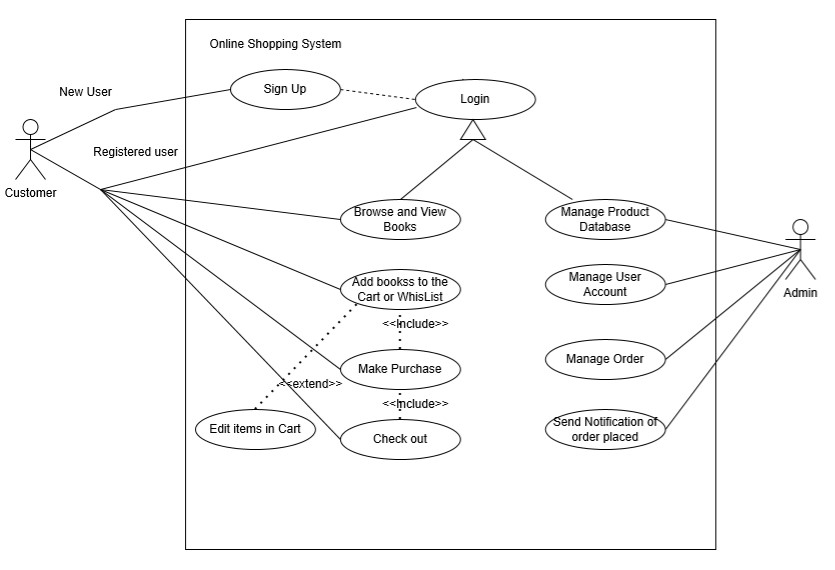


**UML (Unified Modeling Language) Diagram:**

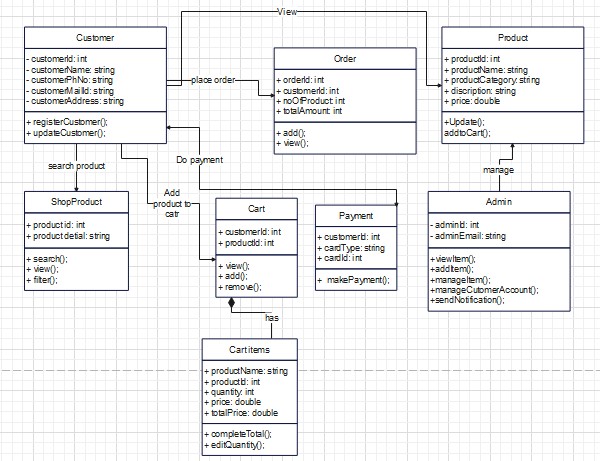
**Use case Diagram:**

Use case diagram is the type of Unified Modeling language diagram. It is a vital tool for system. It is a behavioral diagram that provides the visual representation of how users with the system. It serves as a blueprint for understanding the functionality of the system from the user’s perspective. It has three main components or part. Those are actor, use case and system boundary.

Actors are like external entities that interact with the system. These include users, other software and external hardware devices. Proper identification of actors are crucial for the proper modeling of the system. Use case are like scenes in the play. They represent the all specific function that the system does. System Boundary is a visual representation of scope or limit of the system. It defines what is inside the system and what is external to the system.



**Class Diagram:**

****

1.Customer Class:

* Attributes: customerID, name, email, address.
* Functions: viewProfile(), updateProfile(), placeOrder(), viewOrders().

2.ShopProduct Class:

* Attributes: productID, name, category, description, price.
* Functions: searchProduct(), viewProductDetails(), filterProducts().

3.Cart Class:

* Attributes: customerID, cartItems.
* Functions: viewCart(), addItem(), removeItem(), calculateTotal().

4.CartItem Class:

* Attributes: productID, quantity
* Functions: getProductDetails(), editQuantity(), calculateItemTotal().

5.Order Class:

* Attributes: orderID, customerID, productID, orderStatus
* Functions: addOrder(), viewOrderDetails(), updateOrderStatus().

6.Payment Class:

* Attributes: customerID, cardType, cardID, amount.
* Functions: makePayment(), verifyPayment(), generateReceipt().

7.Product Class:

* Attributes: productID, name, category, description, price.
* Functions: updateProductDetails(), addToCart(), viewDetails().

8.Admin Class:

* Attributes: adminID, email
* Functions: viewItems(), manageItems(), manageCustomerAccounts(), sendOrderNotification().

This diagram represents the structure of the classes in the system and their relationships. Each class encapsulates attributes and functions relevant to its responsibilities within the application.

**Sequence Diagram:**

A sequence diagram illustrates how objects interact in a particular sequence to achieve a specific functionality or use case within a system.

