

# **Project Documentation**

## **Team Members:**

- **Mohamed Abdo Mostafa**
- **Zeyad Mohamed Sherif Gamal Eldin**
- **Nada Hassan Helmy**
- **Nada Kamal Elsayed**
- **Shrouk Nasser Elsayed**

## • Project Planning & Management

### Project Proposal: Online Bookstore

#### 1. Overview

The **Online Bookstore** is a web-based platform designed to provide users with a seamless experience for browsing, purchasing, and managing books online. It will cater to book lovers, students, and professionals, offering both physical and digital books across various genres. The platform will feature an easy-to-use interface, secure payment methods, and personalized recommendations.

#### 2. Objectives

- Develop a user-friendly and responsive web application for book purchasing.
- Provide an intuitive search and filtering system to help users find books easily.
- Implement a secure payment gateway for hassle-free transactions.
- Enable users to create accounts, save favorites, and track orders.
- Build an admin panel for managing books, orders, and users efficiently.
- Ensure scalability and security for future growth.

#### 3. Scope

In-Scope:

- User authentication and profiles
- Book catalog with categories, search, and filters
- Shopping cart and checkout process
- Payment gateway integration (PayPal, Stripe, etc.)
- Order history and tracking
- Reviews and ratings system
- Admin panel for book and order management

Out-of-Scope (for this version):

- Audiobooks and subscriptions
- AI-driven recommendations
- multi-language support

### Project Plan: 8 Weeks

## 1. Gantt Chart (8 Weeks)

Phase	Tasks	Week1	Week2	Week3	Week4	Week5	Week6	Week7	Week8
Planning & Design	Requirements, UI/UX design								
Backend Development	Database setup, API development								
Frontend Development	UI Implementation, API Integration								
Feature Integration	Cart, Checkout, Payment Integration								
Admin Panel	CRUD Operations for Books & Orders								
Testing	Unit Testing, Bug Fixing, Security Tests								
Deployment	Final Testing, Lunch								

## 2. Milestones & Deliverables

Milestone	Expected Completion	Deliverables
Project Planning Completed	Week 2	Finalized requirements, UI/UX wireframes
Backend Development Completed	Week 4	Functional API with database integration
Frontend Prototype Ready	Week 5	UI with basic navigation & static pages
Core Features Implemented	Week 6	Functional cart, checkout, payment system
Admin Panel Completed	Week 7	CRUD operations for books, orders, users
Testing Phase Completed	Week 8	Bug-free, security-tested app
Deployment & Go-Live	Week 8	Fully deployed online bookstore

## 3. Resource Allocation

Resources	Role	Task Assigned
Project Manager	Oversees project	Planning, team coordination, deadlines
Frontend Developer	UI/UX & frontend	UI implementation, API integration
Backend Developer	API & database	Backend logic, database setup, authentication
QA Engineer	Testing & bug fixes	Unit, integration, and security testing
DevOps Engineer	Deployment & hosting	CI/CD setup, server deployment, monitoring

#### 4. Risk Assessment Table & Mitigation Plan

Risk Category	Risk Description	Likelihood	Impact	Mitigation Strategy
Technical Risks	Payment gateway integration failures	Medium	High	Use well-documented APIs (PayPal, Stripe); conduct early-stage testing.
	Database performance issues	Low	High	Optimize queries, use indexing, and conduct load testing.
	Security vulnerabilities (data leaks, cyber threats)	High	High	Implement SSL, encryption, secure authentication (OAuth, JWT). Conduct regular security audits.
Operational Risks	Delayed feature implementation	Medium	Medium	Use Agile methodology with sprint planning and prioritizing critical features.
	Lack of skilled developers	Medium	High	Allocate time for training and knowledge sharing. Hire additional resources if needed.
User & Market Risks	Poor user adoption	Low	High	Conduct user research, usability testing, and refine UI/UX based on feedback.
Financial Risks	Budget overruns	Medium	High	Track expenses using project management tools

				and maintaining buffer funds.
<b>Legal &amp; Compliance Risks</b>	Non-compliance with data protection laws (GDPR, PCI DSS)	Medium	High	Follow regulatory guidelines, ensure proper user data handling, and provide privacy policies.

Mitigation Strategy	Action Steps
Regular Testing	Conduct unit, integration, and security tests before deployment.
Security Measures	Use HTTPS, data encryption, and role-based access control (RBAC).
Scalable Architecture	Use cloud-based hosting and database scaling to handle traffic.
Agile Development	Implement Agile sprints to quickly address issues and iterate.
Backup & Recovery Plan	Schedule daily database backups and disaster recovery tests.

## 5. KPIs (Key Performance Indicators)

Category	KPI	Target	Measurement Method
<b>Performance</b>	System Response Time	≤2 seconds for critical pages	Monitoring tools (e.g., New Relic, GTmetrix)
	System Uptime	≥98% during peak hours	Server logs & uptime monitors (e.g., UptimeRobot)
<b>User Engagement</b>	User Adoption Rate	≥70% within first month post-launch	Analytics tools (e.g., Google Analytics)
	User Satisfaction Score (NPS)	≥8/10	Post-launch surveys & feedback forms
<b>Development Efficiency</b>	Sprint Velocity	Completion of 90% of sprint tasks	Agile tools (e.g., Jira, Trello)
	Bug Resolution Time	≤48 hours for critical bugs	Issue-tracking systems (e.g., GitHub Issues)
<b>Security &amp; Compliance</b>	Security Incident Frequency	0 critical vulnerabilities post-test	Regular security audits & penetration tests
	Compliance Rate (GDPR/PCI DSS)	100% adherence	Internal audits & third-party certifications

<b>Financial</b>	Budget Adherence	≤5% deviation from allocated budget	Financial tracking tools (e.g., QuickBooks)
<b>Operational Success</b>	Deployment Success Rate	100% error-free deployment	CI/CD pipeline logs (e.g., Jenkins, GitHub Actions)
	Feature Completion Rate	100% of core features by Week 8	Gantt chart progress & milestone reviews

## 4. System Analysis & Design

- **Problem Statement & Objectives**

- **Use Case Diagram & Descriptions**

- **The system has three main actors: Customer, Admin, and System:**

- **Customer:**

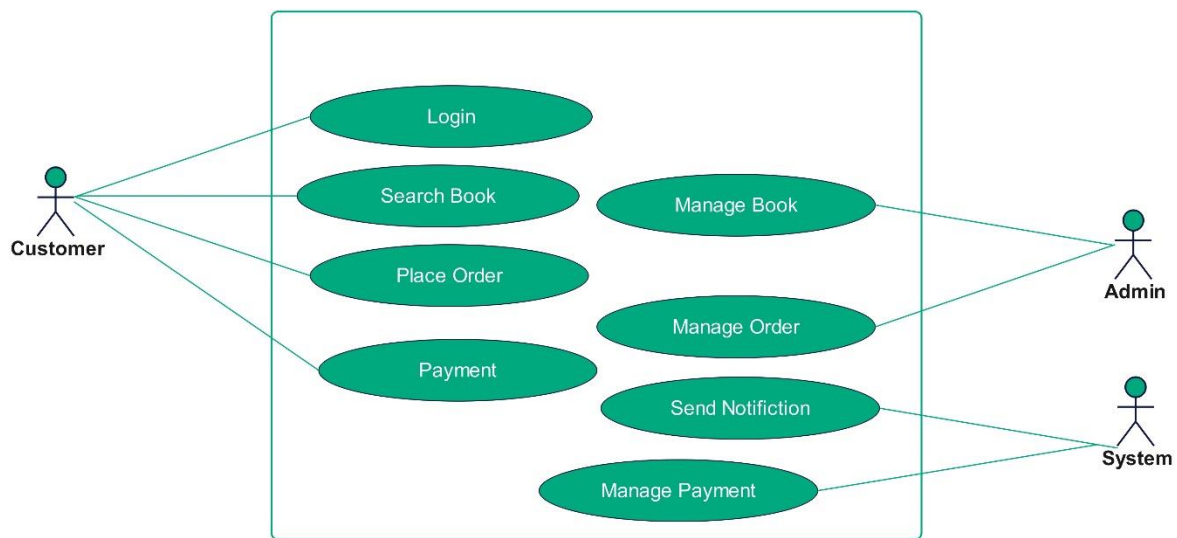
- Register/Login
      - Browse/Search Books
      - View Book Details
      - Add to Cart
      - Making Payment
      - View Order History
      - Review and Rate Books

- **Admin:**

- Manage Books (Add, Update, Delete)
      - Manage Orders
      - View Sales Reports
      - Manage User Accounts
      - Manage Book Reviews

- **System:**

- Send Notifications
      - Manage Payment Processing
      - Generate Sales Reports



## ○ Functional & Non-Functional Requirements

### Functional Requirements:

**User Authentication:** Users must be able to register and log in securely.

**Book Browsing and Searching:** Users can search books by title, author, genre, or keyword.

**Shopping Cart:** Users can add books to their cart and view/edit the cart before checkout.

**Payment Processing:** Secure payment gateway integration (e.g., PayPal, Stripe).

**Order Management:** Users can view their past orders and track current orders.

**Book Management:** Admin can add, update, or remove books from the inventory.

**User Management:** Admin can manage user profiles and privileges.

**Review and Rating System:** Users can rate, and review purchased books.

**Notification System:** Users receive notifications on order status and promotional offers.

**Reporting:** Admin can view sales reports and track business performance.

### Non-Functional Requirements:

**Performance:** The system should handle at least 10,000 concurrent users.

**Reliability:** 99.9% uptime with automatic failover and backup.

**Security:** SSL/TLS encryption for data transfer and strong password policies.

**Usability:** User-friendly and intuitive interface for both customers and admins.

**Scalability:** Support for scaling both horizontally and vertically.

**Maintainability:** Modular codebase with clear documentation for easy maintenance.

- **Software Architecture – High-level design outlining system components, interactions, and architecture style**

### **Software Architecture:**

#### **Architecture Style:** MVC (Model-View-Controller)

The system architecture is designed following the MVC pattern, which promotes a clear separation of concerns and supports scalability.

### **High-Level Design:**

#### **Frontend (View):**

**Technologies:** HTML, CSS, JavaScript.

**Features:** User interface for browsing, purchasing, and managing books.

#### **Backend (Controller):**

**Technologies:** C#, .NET Stack.

**Features:** Business logic, user authentication, and payment processing.

#### **Database (Model):**

**Technologies:** MySQL

**Features:** Storing user data, book inventory, order history, and reviews.

### **Payment Gateway:**

**Integrations:** PayPal, Stripe

**Features:** Secure payment processing.

### **Recommendation Engine:**



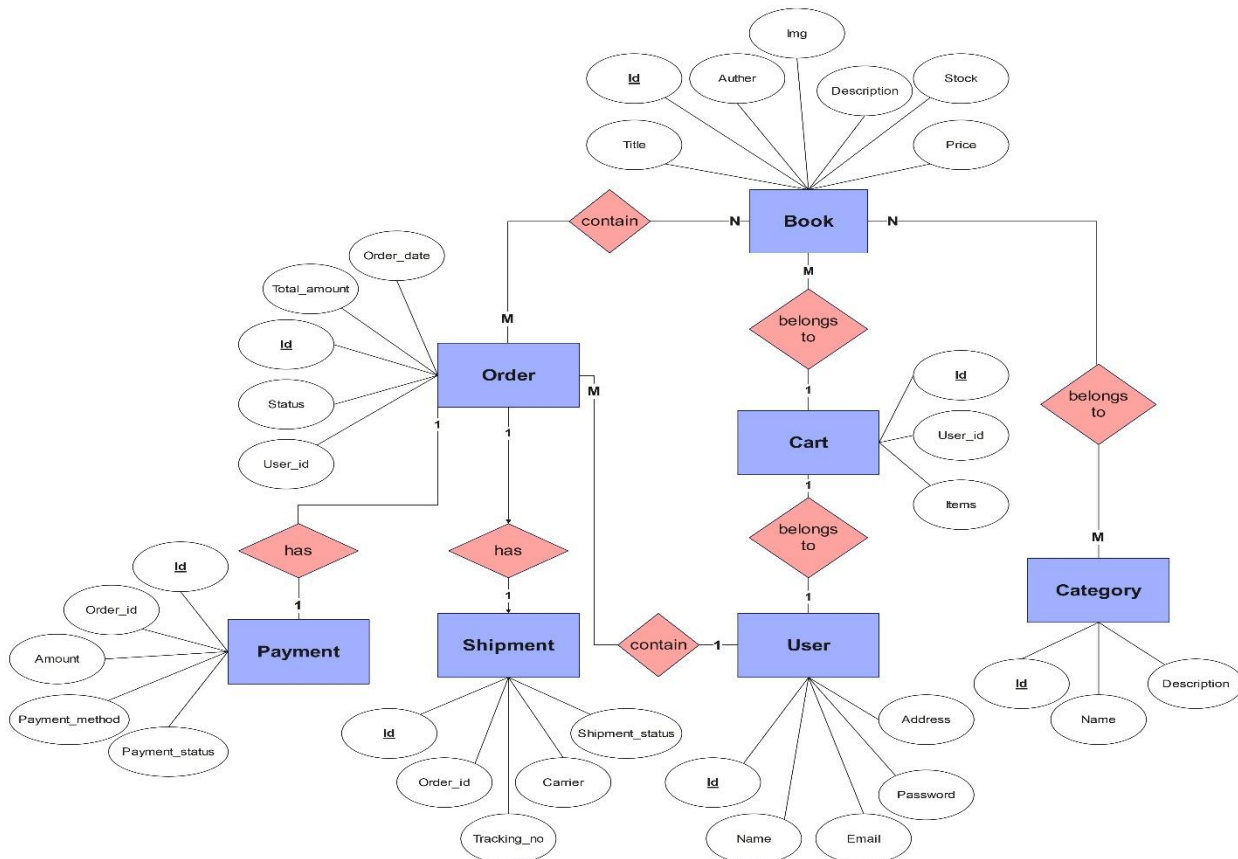
The front end, integrated with backend APIs.

**Features:** Manage books, orders, users, and view reports.

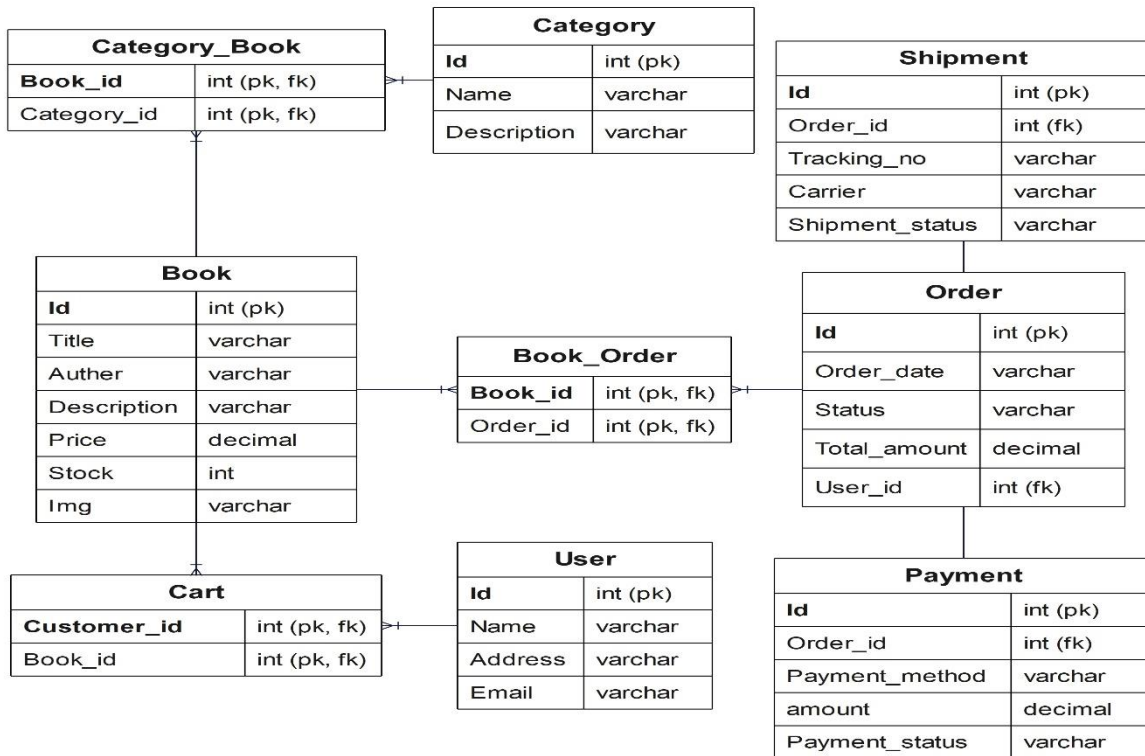
## 2. Database Design & Data Modeling

- ER Diagram (Entity-Relationship Diagram):

### *BookStore ERD*



- **Logical & Physical Schema:**



### 3. Data Flow Diagram (DFD):

#### DFD Level 0 – Context Diagram

This diagram provides a high-level overview of the **Online Bookstore System**. It shows how data flows between the main system and external entities:

- **Customers** place orders and provide payment details.
- The **Online Bookstore System** processes these orders.
- The system communicates with the **Warehouse** to check stock availability.
- Payments are processed through a **Payment Gateway**.
- Once the transaction is complete, customers receive confirmation and order details.



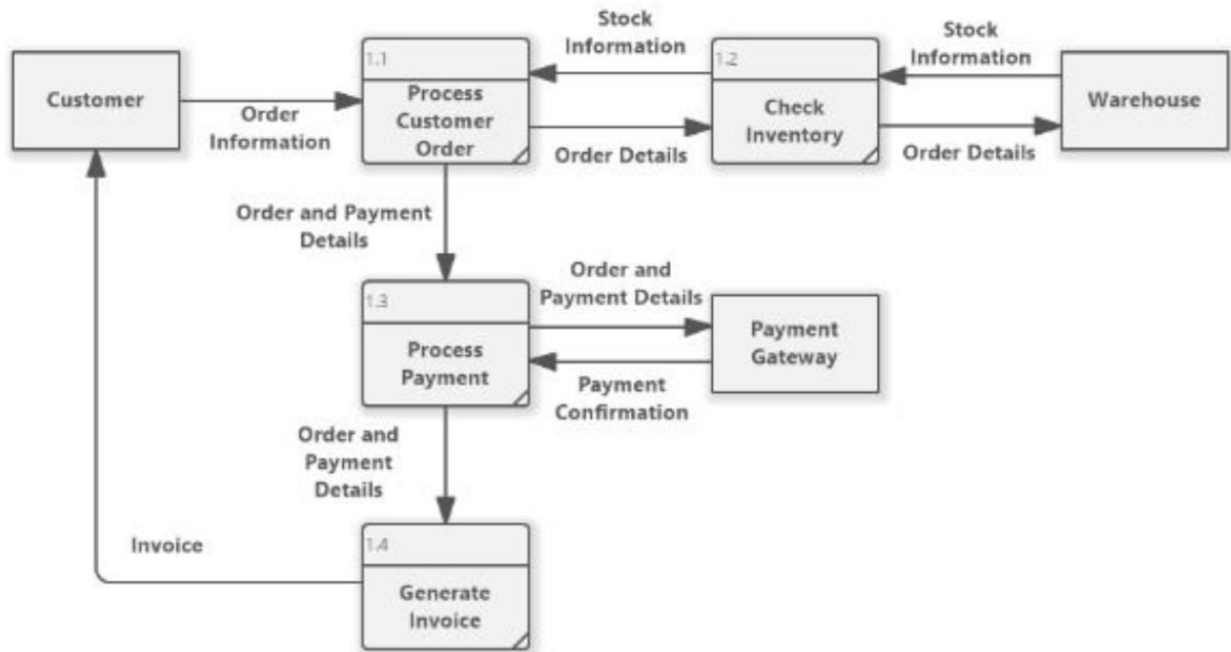
#### DFD Level 1 – Detailed Process Breakdown

DFD Level 1 expands the main system into four sub-processes:

- Process Customer Order → Receives order details from the customer.
- Check Inventory → Confirms book availability from the warehouse.
- Process Payment → Sends payment details to the Payment Gateway for processing.
- Generate Invoice → Issues an invoice after successful payment.

#### Data Flow:

- The system retrieves stock information before confirming an order.
- Payments are processed and verified.
- Once the transaction is complete, the system generates an invoice for the customer.

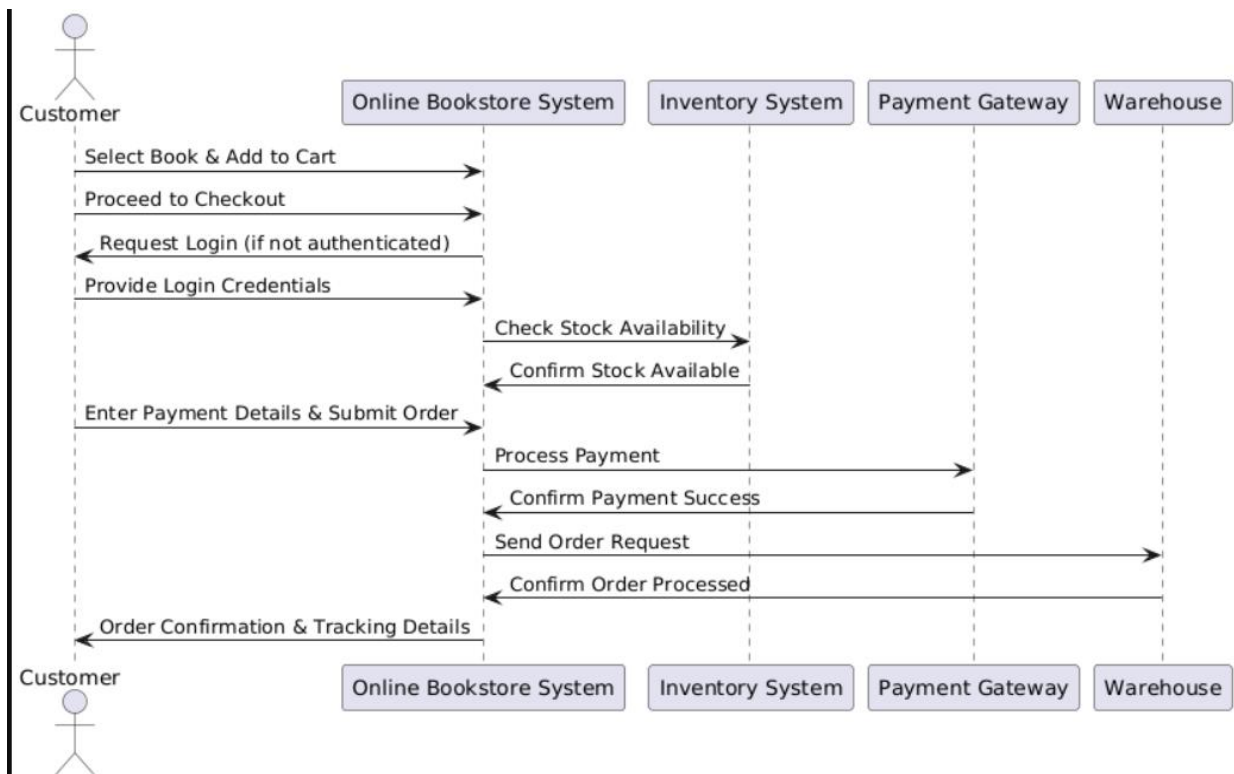


## 2. Sequence Diagram – Placing an Order more details:

This Sequence Diagram represents the step-by-step process when a customer places an order on an online bookstore.

Steps:

- The customer selects a book and adds it to the cart.
- Customer proceeds to check out.
- If not logged in, the system asks for login credentials.
- If available, the customer enters payment details and submits the order.
- The system sends the payment request to the Payment Gateway.
- Payment is confirmed, and the system notifies the warehouse.
- The warehouse processes the order and prepares it for shipment.
- The customer receives order confirmation and tracking details.



### 3. Activity Diagram – Book Purchase Flow:

The activity diagram represents the flow of actions involved in purchasing a book from an online bookstore. It captures the interactions between the customer, the online bookstore system, the inventory system, the payment gateway, and the warehouse.

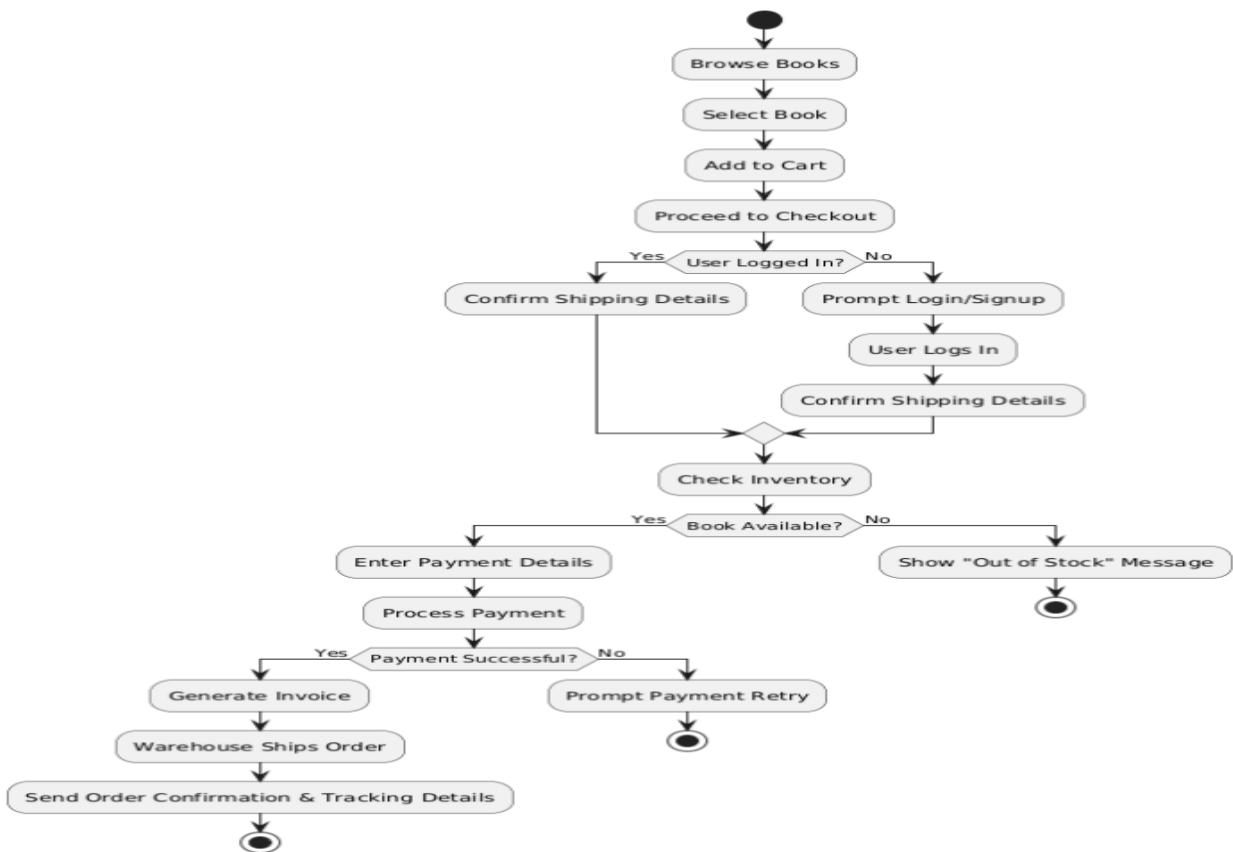
Process Overview:

#### 1. Book Selection & Checkout:

- The customer browses books, selects the desired book, and adds it to the cart.
- At checkout, the system verifies if the user is logged in. If not, it prompts for login or signup.
- Once logged in, the customer confirms the shipping details.

#### 2. Inventory Check:

- The system checks the inventory to determine book availability.
- If the book is in stock, the process continues to payment.
- If out of stock, the system notifies the customer, and the process ends.



### 3. Payment Processing:

- The customer enters payment details, and the system sends the request to the payment gateway.
- If the payment is successful, an invoice is generated, and the order is forwarded to the warehouse.
- If the payment fails, the system prompts the customer to retry with a different payment method.

### 4. Order Fulfillment & Shipping:

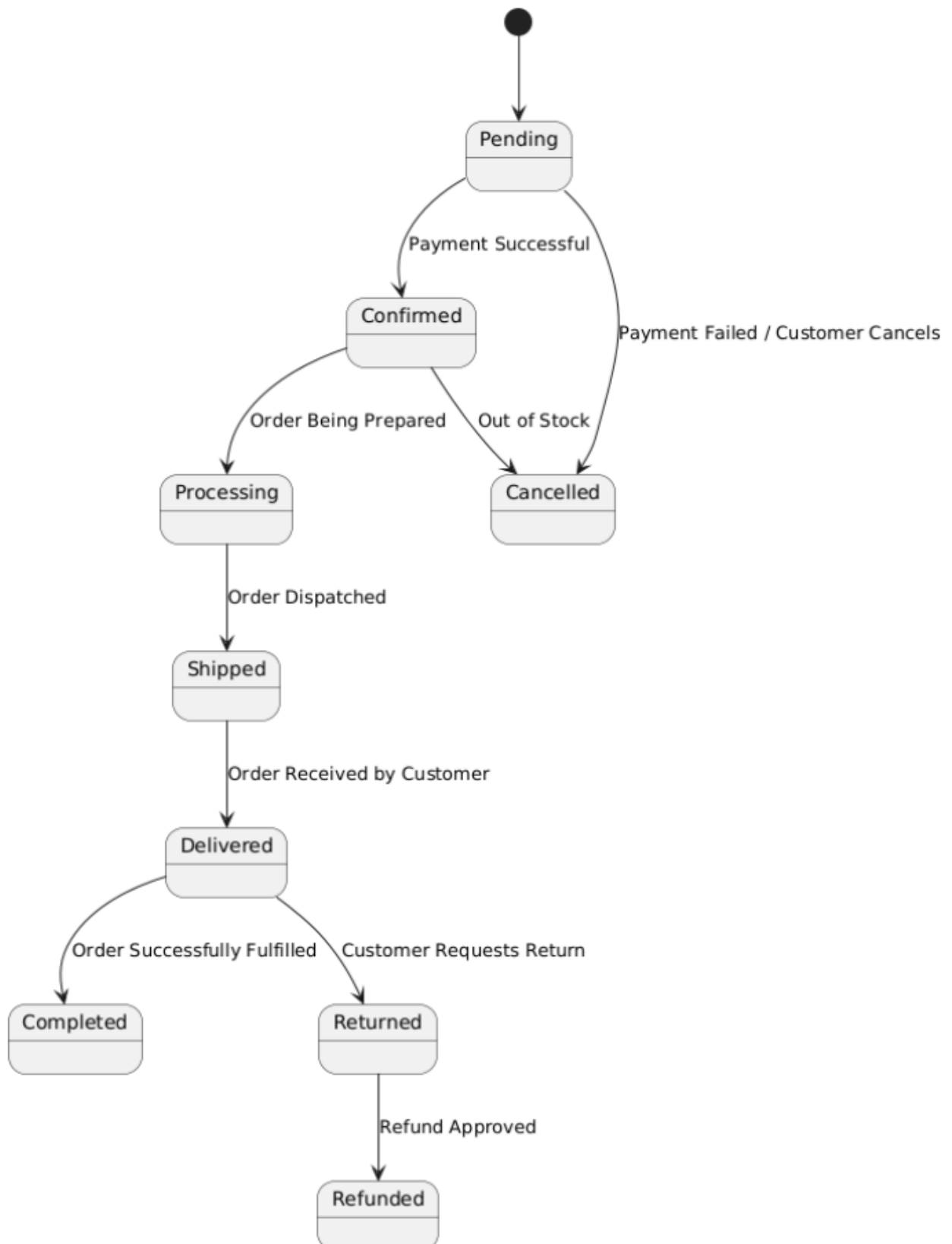
- The warehouse receives the order details, processes the shipment, and updates the tracking information.
- The system sends order confirmation and shipping details to the customer.

#### **4. State Diagram – Order Lifecycle:**

The State Diagram illustrates the different states an order undergoes in an online bookstore, along with the transitions triggered by system events or user actions. It defines how the order progresses from placement to fulfillment or cancellation.

##### **Order Lifecycle States & Transitions**

1. Pending – The order is created but awaiting payment.
  - If payment is successful, it transitions to Confirmed.
  - If payment fails or the customer cancels, it moves to Cancelled.
2. Confirmed – The order is successfully placed, and payment is verified.
  - If the book is out of stock, the order is Cancelled.
  - Otherwise, it proceeds to Processing.
3. Processing – The order is being prepared for shipment.
  - Once ready for delivery, it moves to Shipped.
4. Shipped – The order is dispatched from the warehouse.
  - When received by the customer, it transitions to Delivered.
5. Delivered – The order reaches the customer.
  - If there are no issues, it is marked as Completed.
  - If a return request is initiated, it moves to Returned.
6. Returned (Optional State) – The customer sends the order back due to issues.
  - If the return is approved, the order is updated to Refunded.



## 5. Class Diagram – System Structure:



The Class Diagram defines the core components of the online bookstore system, showing their attributes, methods, and relationships.

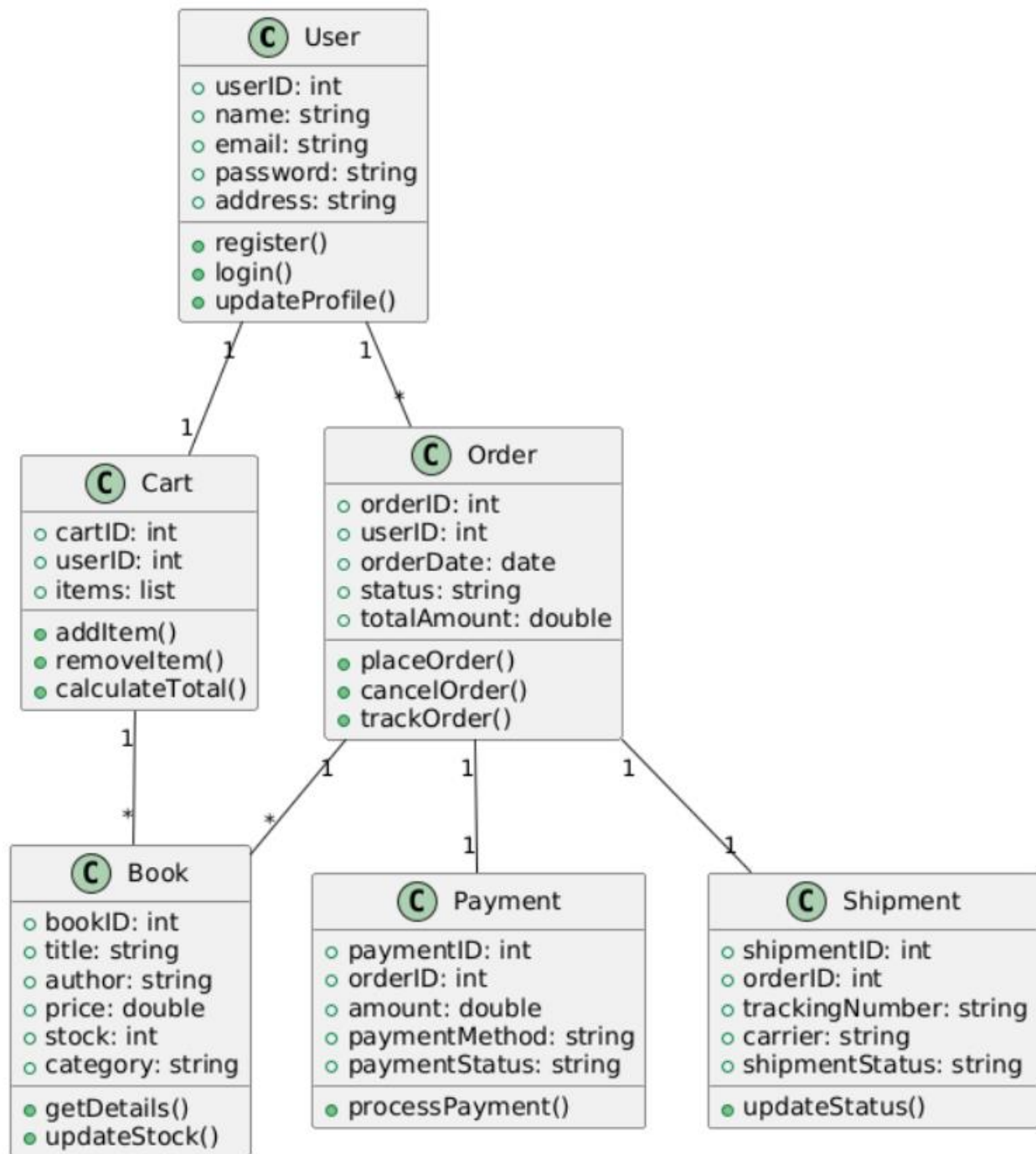
#### Classes and Their Responsibilities:

- **User:** Represents a registered customer with attributes like userID, name, email, password, and address.
  - **Methods:** register(), login(), updateProfile().
  - **Relationships:** A user has one cart and can place multiple orders.
- **Book:** Stores book-related details, including bookID, title, author, price, stock, and category.
  - **Methods:** getDetails(), updateStock().
  - **Relationships:** Books are linked to carts and orders.
- **Cart:** Contains books selected by a user before purchase.
  - **Methods:** addItem(), removeItem(), calculateTotal().
  - **Relationships:** A user has one cart, and a cart can contain multiple books.
- **Order:** Represents a purchase, containing attributes like orderID, userID, orderDate, status, and totalAmount.
  - **Methods:** placeOrder(), cancelOrder(), trackOrder().
  - **Relationships:** An order is linked to a user, books, payment, and shipment.
- **Payment:** Manages transactions with attributes such as paymentID, orderID, amount, paymentMethod, and paymentStatus.
  - **Methods:** processPayment().
  - **Relationships:** Each order is linked to one payment.
- **Shipment:** Tracks the delivery of an order with attributes like shipmentID, orderID, trackingNumber, carrier, and shipmentStatus.
  - **Methods:** updateStatus().
  - **Relationships:** Each order has one shipment.

#### Relationships:

- User ↔ **Cart (1:1)** – One user has one cart.
- User ↔ **Order (1:M)** – A user can place multiple orders.
- Cart ↔ **Book (1:M)** – A cart can contain multiple books.
- Order ↔ **Book (1:M)** – An order can include multiple books.

- Order ↔ **Payment (1:1)** – Each order has one payment.
- Order ↔ **Shipment (1:1)** – Each order has one shipment.




## 5. UI/UX Design & Prototyping:

### Wireframes & Mockups:

- Login/Sign-up Pages:

- Add form validation (e.g., red borders for invalid email/password).
  - Include error states (e.g., "Invalid credentials" popup).
  - Add "Forgot Password" flow (new page for resetting password).
- **Home Page:**
  - Display book carousels for "Popular" and "New Releases".
  - Add search bar functionality (auto-suggestions, filters).
- **Book Details Page:**
  - Include user reviews/ratings section.
  - Add the "Preview Book" button with a sample chapter.
- **Categories Page:**
  - Organize genres in a grid layout with thumbnail images.
  - Add dropdown filters (e.g., sort of popularity, price).
- **Profile Page:**
  - Add "Edit Profile" button and settings icon.
  - Include reading history and Wishlist sections.

Login Page



### Log In

**Email :**

**Password :**

[Login](#)

Don't have an account? [sign up](#)

[Forgot password](#)




Sign-up Page

## Welcome !

To keep connected with us please login with your personal info, or register if don't have an account.

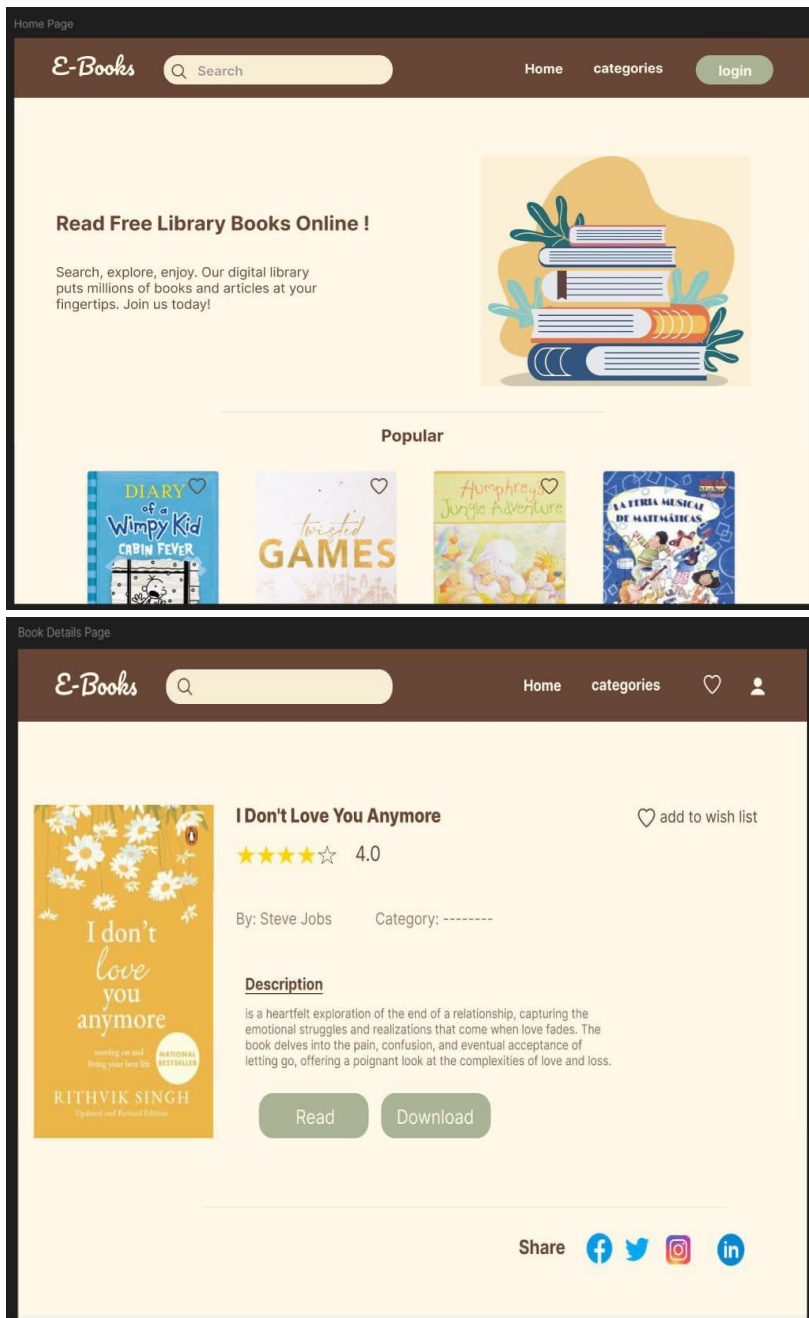
[Login](#)

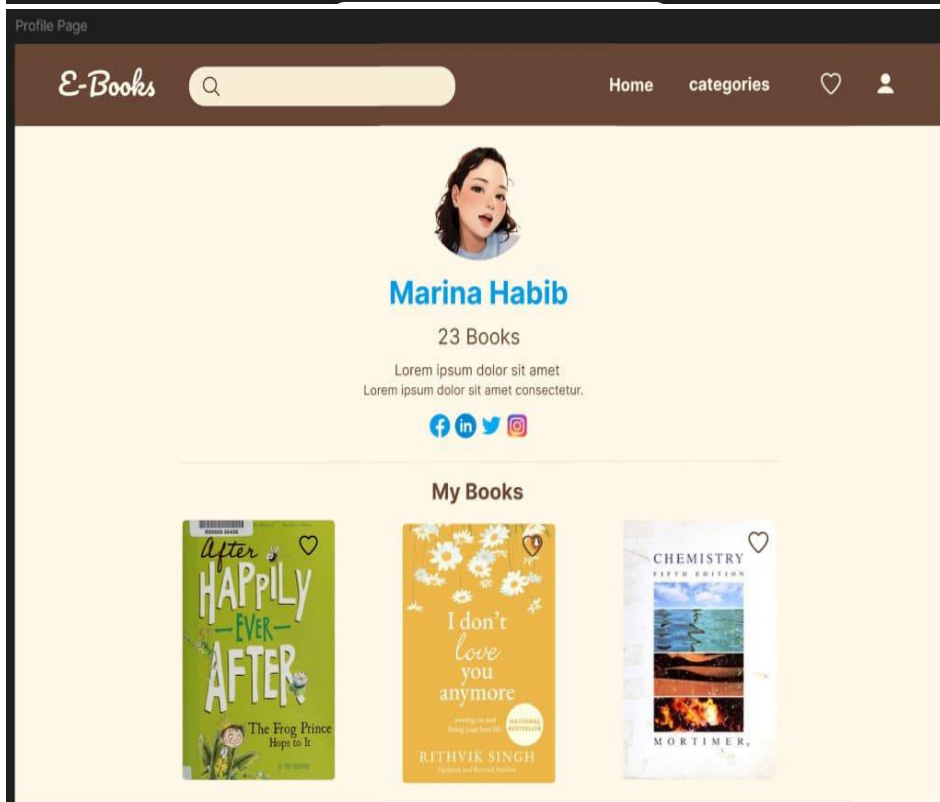
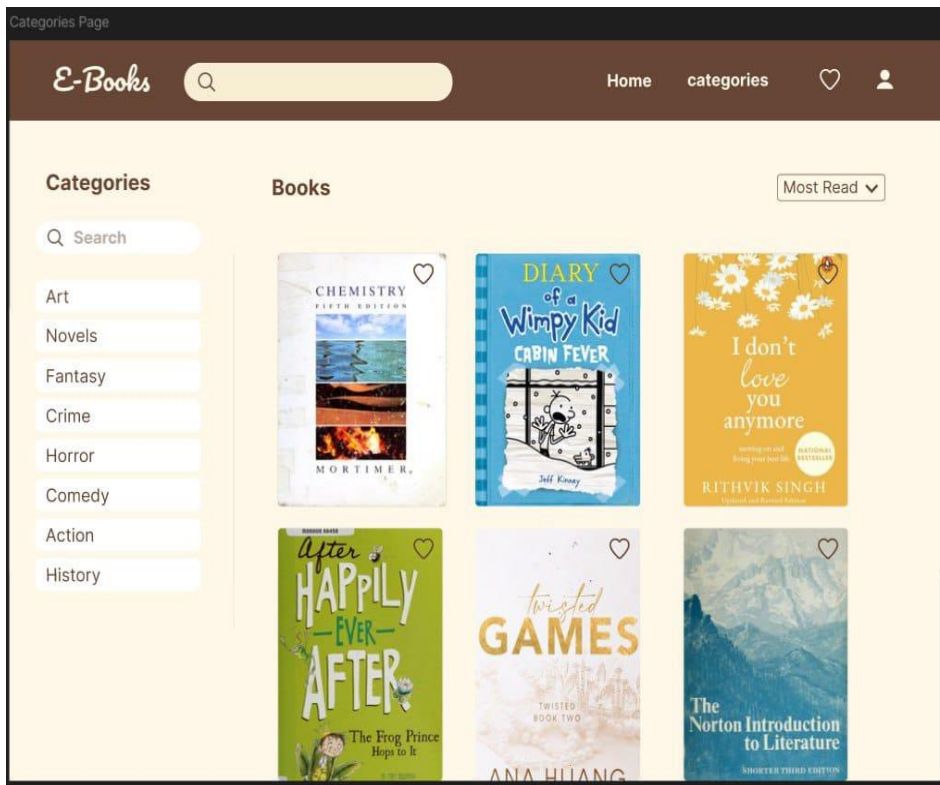
## Create Account



or use email for registration

[Sign up](#)





# UI/UX Guidelines

## 1. Design Principles:

- Minimalist & Clean: Prioritize readability with ample white space.
- User-Centric Navigation: Consistent bottom/top navigation bar with Home, Categories, Cart, and Profile icons.

## 2. Color Scheme:

- Primary: #2E5AAC (calming blue for buttons and headers).
- Secondary: #F8F9FA (light gray for backgrounds).
- Accent: #4CAF50 (green for success actions like downloads).

## 3. Typography:

- Headers: Roboto Bold (24px–32px).
- Body Text: Open Sans Regular (16px, line height 1.6).
- Buttons: Roboto Medium (14px, uppercase).

## 4. Accessibility:

- Contrast Ratio:  $\geq 4.5:1$  for text/background.
- Alt Text: For all book covers and icons.
- Keyboard Navigation: Support tabbing through form fields and buttons.

## 5. Components:

- Buttons: Rounded corners (8px), padding (12px 24px).
- Cards: Shadow effect (2px blur), hover animation (scale 1.02).
- Input Fields: Clear labels, helper text, and error messages.

## 5. System Deployment & Integration

### Technology Stack

Layer	Technology
Frontend	ASP.NET Razor Pages or Blazor (for UI), HTML5/CSS3, JavaScript (for interactivity)
Backend	ASP.NET Core (REST API/MVC), Entity Framework Core (ORM), Stripe/PayPal SDK
Database	Microsoft SQL Server (primary database), Redis (caching for sessions/cart)
DevOps	Docker (containerization), AWS EC2/IIS (hosting), Azure DevOps (CI/CD), Nginx
Security	HTTPS, ASP.NET Core Identity (authentication), bcrypt, AWS WAF
Monitoring	Application Insights (performance/errors), Prometheus + Grafana (optional)

### Deployment Diagram

#### Infrastructure Layout:

##### 1. Client Layer:

Users interact with the ASP.NET frontend (Razor Pages/Blazor) hosted on AWS EC2 (IIS server) or Azure App Service.

##### 2. Application Layer:

ASP.NET Core backend (APIs/MVC) runs in Docker containers on AWS EC2/Azure VMs.

Redis (ElastiCache/Azure Cache) for session/cart coaching.

##### 3. Database Layer:

SQL Server (AWS RDS/Azure SQL Database) with failover replicas.

##### 4. External Services:

Stripe/PayPal (payment processing), SendGrid (email notifications).

##### 5. Security:

AWS WAF/Azure Firewall, HTTPS via Let's Encrypt or Azure TLS certificates.



## Flow:

User → IIS/Azure App Service (ASP.NET App) → ASP.NET Core API → SQL Server

## Component Diagram

### Key Components & Dependencies:

#### 1. Frontend (Razor Pages/Blazor):

- Directly integrated with backend logic. Depends on **ASP.NET Core Controllers** for data.

#### 2. Backend (ASP.NET Core):

- **Controllers:**
  - **AuthController:** Handles login, registration (ASP.NET Identity).
  - **OrderController:** Manages cart, checkout, orders.
  - **PaymentController:** Integrates with Stripe/PayPal.
  - **BookController:** CRUD operations for books (Entity Framework Core).
- **Services Layer:** Business logic (e.g., CartService, PaymentService).

#### 3. Database (SQL Server):

- Stores user data, books, orders, payments. Entity Framework Core for ORM.

#### 4. Redis:

- Caches active sessions and cart data (via IDistributedCache).

### Dependencies:

- **PaymentController** → External payment gateways.
- **AuthController** → ASP.NET Identity → SQL Server.
- **Frontend** → Backend Controllers (no external API gateway needed).