Order Processing System

Database Final Project

27-12-2025

8963 Mazen Azhary Omar Abdelmawgoud 8776

8873 Omar Elsenousy Youssef Mohamed 8821

Introduction

This document presents a comprehensive overview of the Order Management System, a full-stack web application designed to manage book inventory, customer orders, and administrative operations for an online bookstore. The system provides distinct interfaces for administrators and customers, each with tailored functionality to meet their respective needs.

Implemented Features

Authentication System

- **User Registration**: New users can create accounts with username, password, email, phone, and shipping address with validations for each input.
- **User Login**: Secure authentication with JWT token-based authorization.
- **Role-Based Access Control**: Differentiation between admin and customer roles with appropriate permissions.
- **Session Management**: Token storage in local storage for persistent authentication.
- **Auto-redirect**: Automatic redirection based on user role upon successful login.

Customer Features

- **Customer Dashboard**: Personalized welcome screen with navigation to all customer features.
- **Book Search and Browse**:
  - Search books by title, author, or ISBN
  - Filter books by genre/category
  - View detailed book information including cover images
- **Shopping Cart**:
  - Add books to cart
  - Update quantities
  - Remove items
  - View total price calculation
- **Order Management**:
  - Place orders from cart
  - View order history
  - Track order status
- **Profile Management**:
  - View and edit personal information
  - Update shipping address
  - Change password

Administrator Features

- **Admin Dashboard**: Central hub for administrative operations with enhanced security.
- **Book Management**:
  - Add new books with ISBN, title, author, publisher, price, genre, publication year
  - Upload book cover images
  - Set stock thresholds
  - Update existing book information
  - Delete books from inventory
- **Order Processing**:
  - View pending customer orders
  - Order details including customer information, book details, and order date
  - Confirm orders with visual status changes
  - Persistent order history (confirmed orders remain visible)
- **System Reports**:
  - **Previous Month Sales**: Total sales, books sold, and number of orders
  - **Sales by Date**: Filter sales data by specific date
  - **Top 5 Customers**: Ranked by purchase amount (last 3 months)
  - **Top 10 Selling Books**: Ranked by copies sold (last 3 months)
  - **Book Order History**: Track replenishment orders for specific books
- **Access Control**: Admin-only pages with authentication checks and automatic redirection for unauthorized users.

Technical Features

- **Responsive Design**: Mobile-friendly interface that adapts to different screen sizes.
- **RESTful API**: Backend API with Spring Boot for database operations.
- **Docker Containerization**: Complete system runs in Docker containers (MySQL, Backend, Frontend) using a single docker compose file.
- **Form Validation**: Client-side and server-side validation for data integrity.
- **Session Storage**: Form data persistence to prevent data loss during navigation.
- **Modern UI/UX**: Gradient designs, smooth animations, and intuitive navigation.

API Manual

Examples from our API manual :

1. User Signup

http
POST /api/auth/signup
```
{
 "username": "john_doe",
 "password": "password123",
 "firstName": "John",
 "lastName": "Doe",
 "email": "john@example.com",
 "phoneNumber": "+1-555-0123",
 "shippingAddress": "123 Main St, New York, NY"
}
```

2. User Login

http
POST /api/auth/login
```
{
 "username": "john_doe",
 "password": "password123"
}
```

Response:
```
{
 "accessToken": "eyJhbGciOiJIUzUxMiJ9...",
 "refreshToken": "eyJhbGciOiJIUzUxMiJ9...",
 "tokenType": "Bearer",
 "username": "john_doe",
 "role": "CUSTOMER"
}
```

3. Refresh Access Token

http
POST /api/auth/refresh
```
{
 "refreshToken": "eyJhbGciOiJIUzUxMiJ9..."
}
```

All endpoints return consistent error responses:

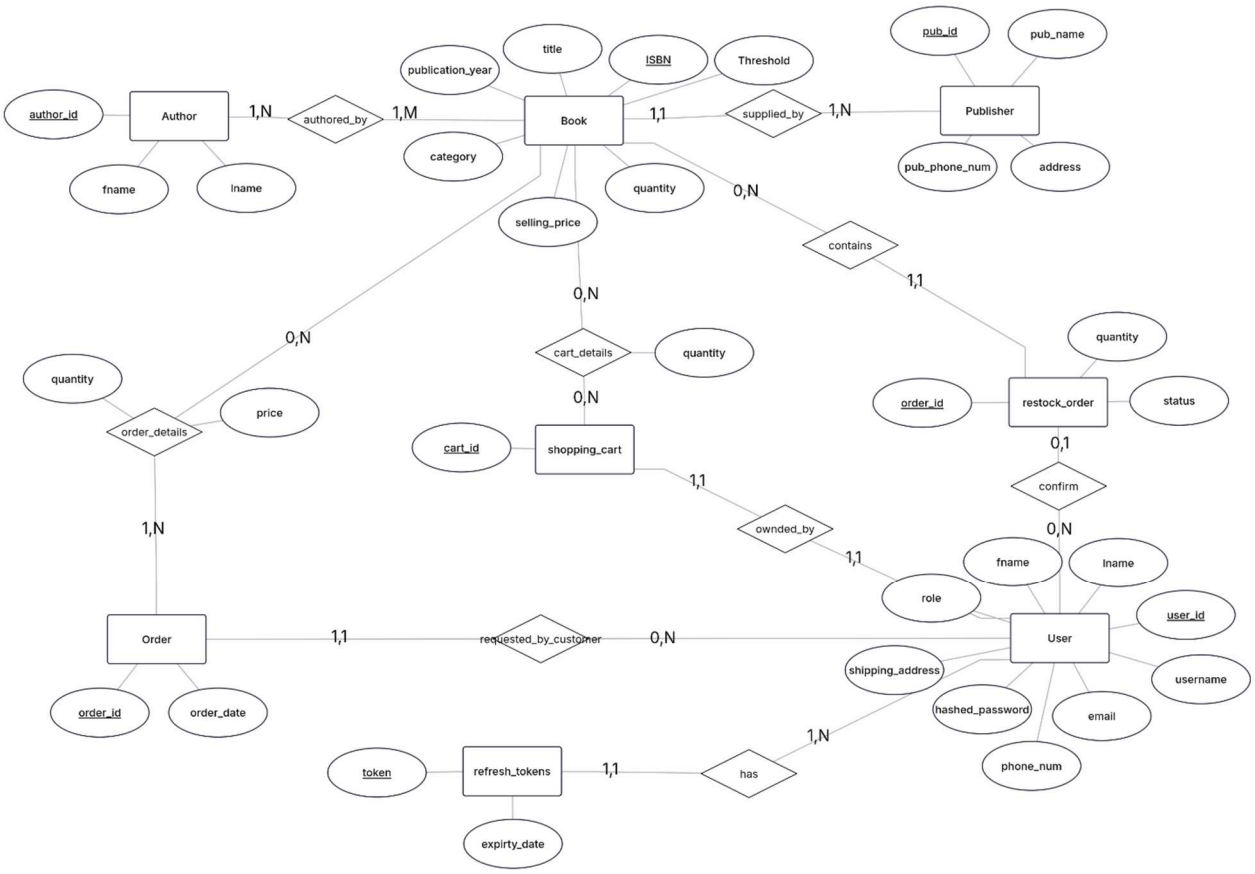
json
```
{
 "success": false,
 "message": "Error description",
 "data": null
}
```
Status codes

- 200: Success
- 400: Bad Request (validation errors)
- 401: Unauthorized (invalid credentials/token)
- 403: Forbidden (insufficient permissions)
- 404: Not Found
- 500: Internal Server Error

Database Design

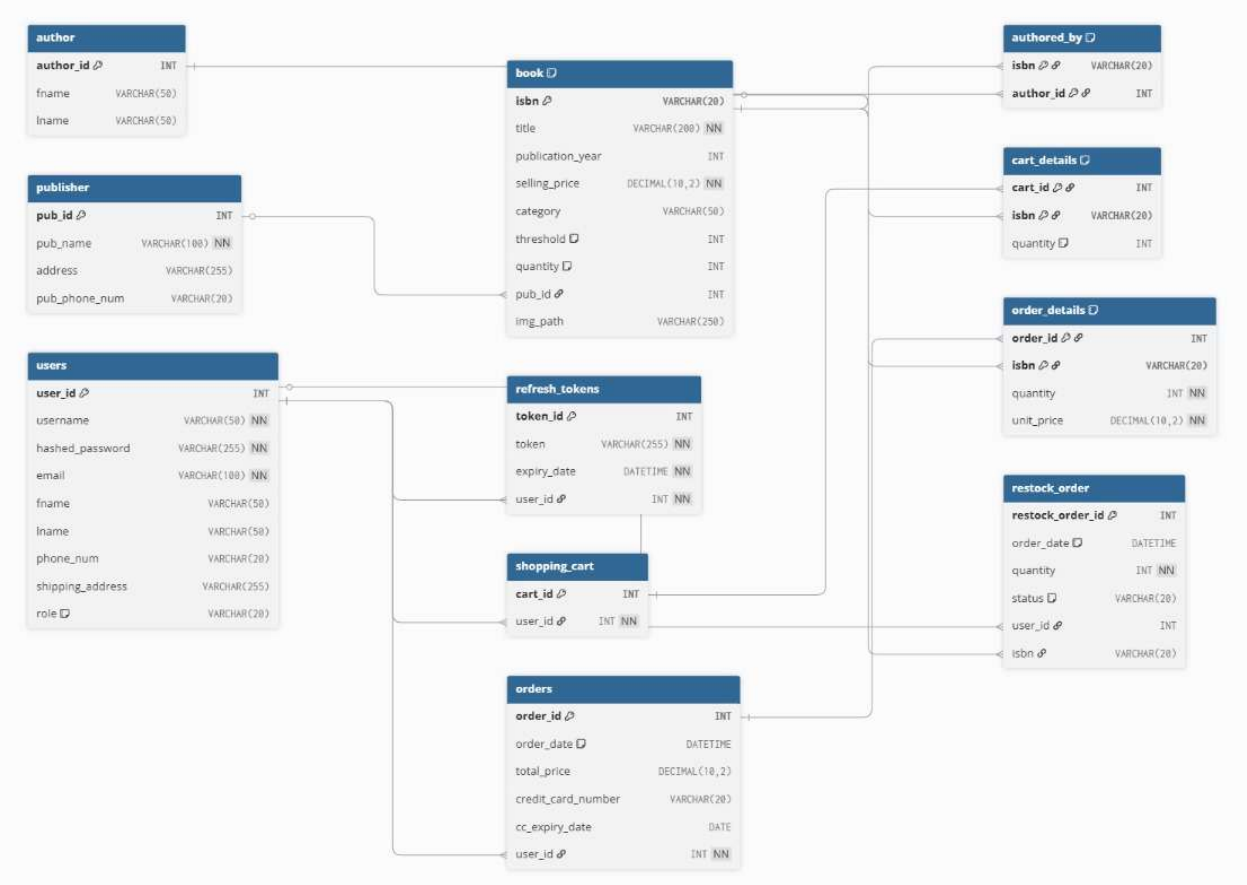
Entity Relationship Diagram (ERD)

The ERD illustrates the relationships between the main entities in the system: Users, Books, Orders, Shopping Cart, Publishers, and Authors.



Relational Schema

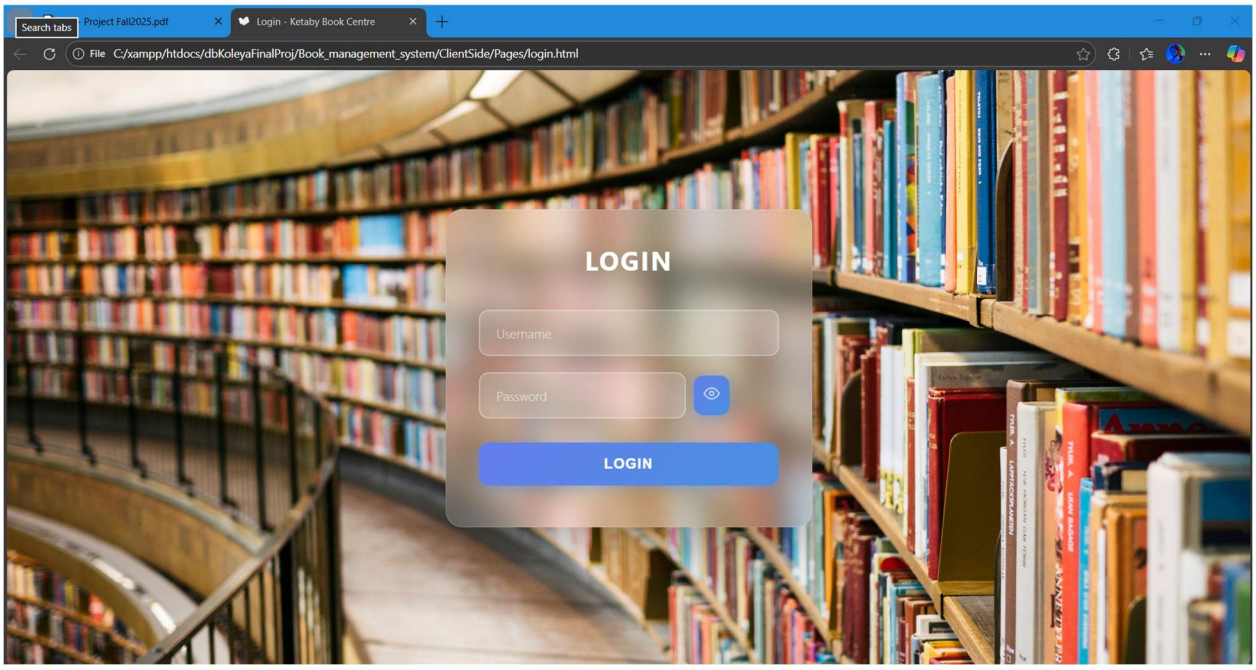
The relational schema shows the database tables with their attributes, primary keys, and foreign key relationships.



User Interface Screens

Authentication Screens

Login Page



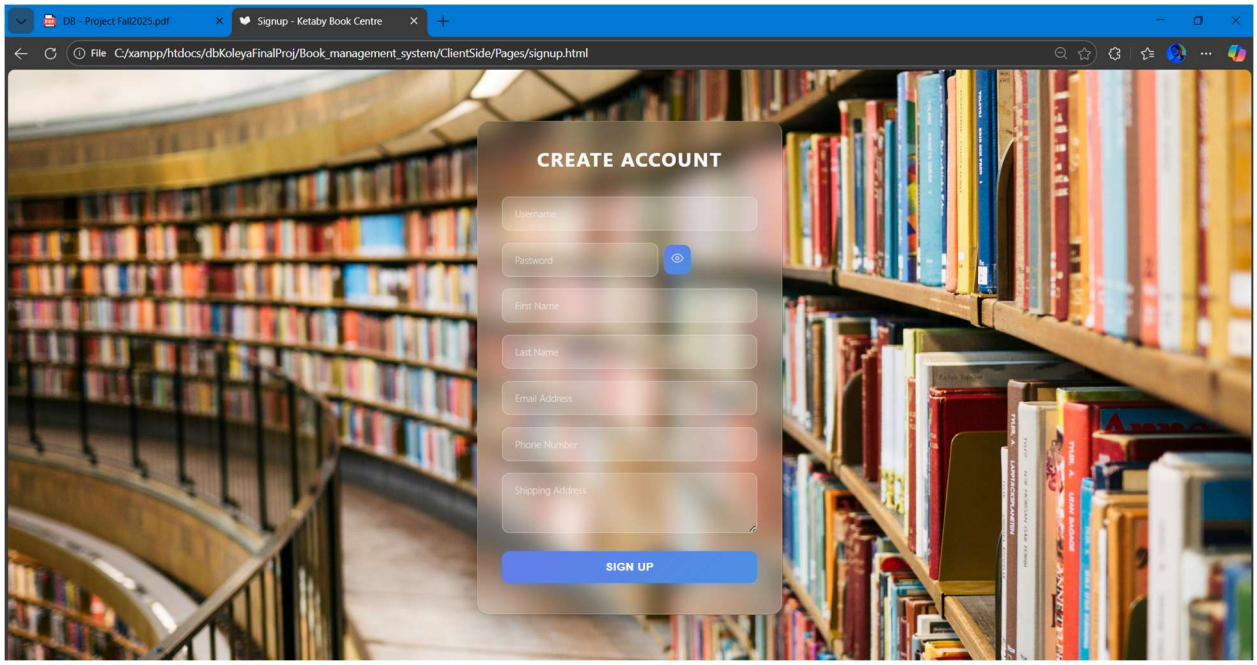
Purpose: Authenticate users and grant access to the system.

Logic Flow:

1. User enters username and password
2. Form data is saved to session storage as user types (prevents data loss)
3. On submission, credentials are sent to backend API endpoint `/api/auth/login`
4. Backend validates credentials and returns JWT tokens (access & refresh) along with user role
5. Tokens and user information stored in local storage
6. User redirected based on role:
   - Admin → Admin Dashboard
   - Customer → Customer Dashboard
7. If backend unavailable, system falls back to local storage validation
8. Password visibility toggle available for user convenience

Signup Page

Purpose: Allow new users to create accounts.

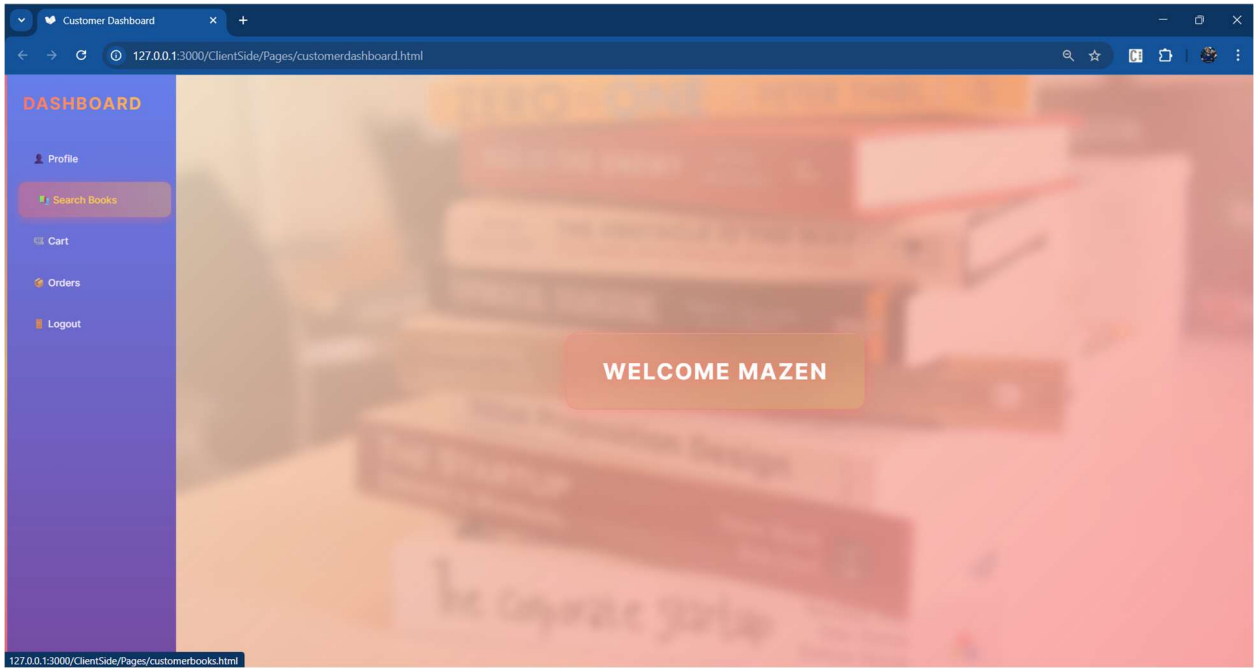


Logic Flow:

1. User fills registration form with username, email, password, phone, and shipping address
2. Form data persisted in session storage during input
3. Client-side validation ensures all required fields are completed
4. Password strength validation (minimum length, special characters)
5. On submission, data sent to `/api/auth/register` endpoint
6. Backend validates uniqueness of username and email
7. Account created in database with encrypted password
8. Success message displayed
9. User redirected to login page
10. Session storage cleared after successful registration

Customer Interface

Customer Dashboard



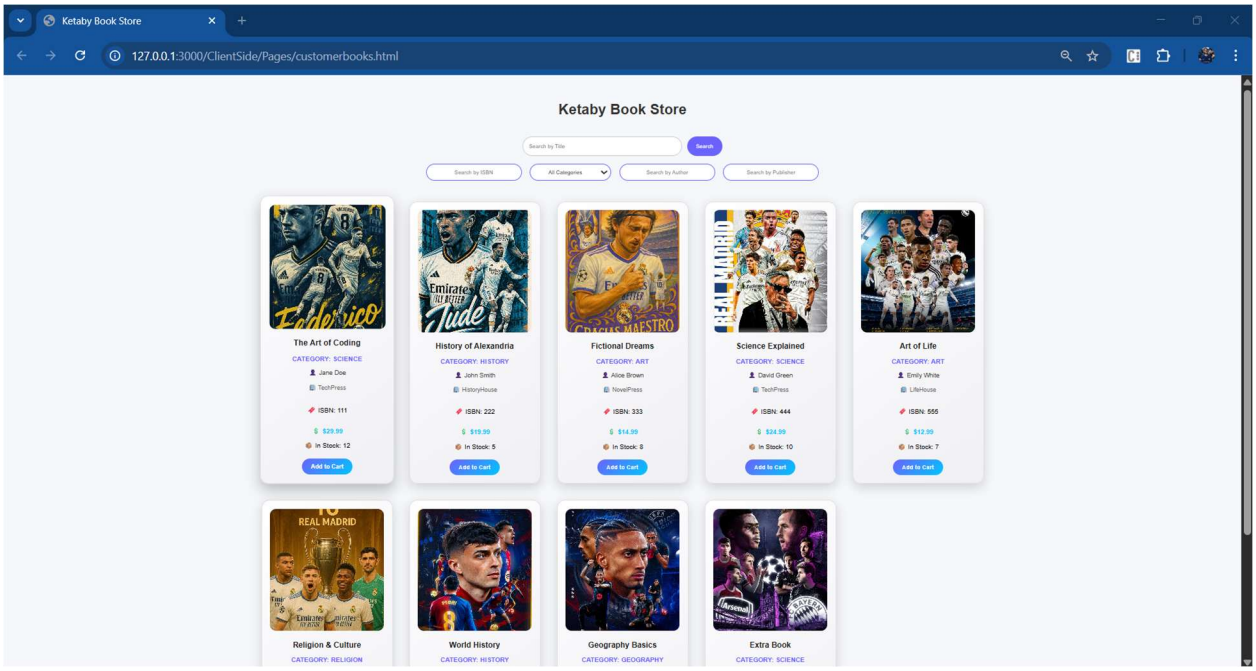
Purpose: Main navigation hub for customer operations.

Logic Flow:

1. Authentication check on page load (redirect to index if not logged in)
2. Display personalized welcome message with username
3. Sidebar navigation with links to:
   - Profile management
   - Book search
   - Shopping cart
   - Order history
   - Logout
4. Background image with overlay for visual appeal
5. Responsive design collapses sidebar on mobile devices

Book Search Page

Purpose: Browse and search for books in the catalog.



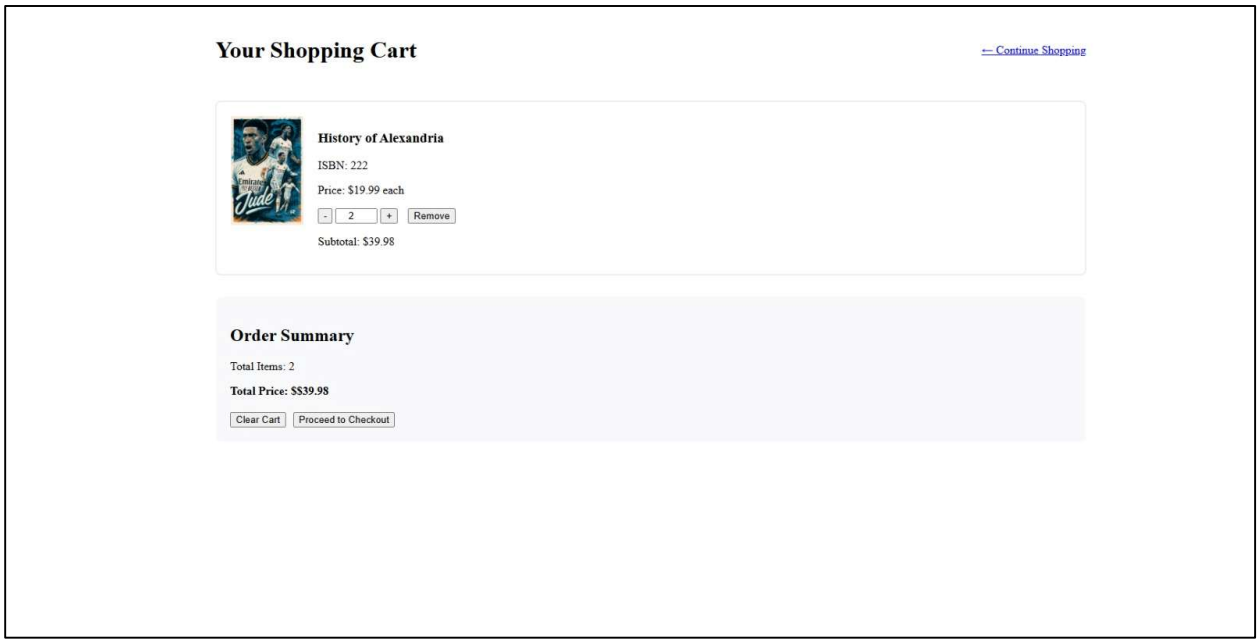
Logic Flow:

1. Page loads with all available books from `/api/books` endpoint
2. Search bar filters results by title, author, or ISBN (real-time filtering)
3. Genre dropdown filters books by category
4. Each book card displays:
   - Cover image
   - Title and author
   - Price
   - Publication year
   - Stock status
   - "Add to Cart" button
5. Clicking "Add to Cart" sends request to backend with book ID and user ID
6. Cart icon updates with item count

7. Confirmation message displayed on successful addition

Shopping Cart

Purpose: Review selected items before purchase.



Logic Flow:

1. Fetch cart items from `/api/cart/{userId}` endpoint
2. Display each cart item with:
   - Book cover image
   - Title and author
   - Unit price
   - Quantity selector (increment/decrement buttons)
   - Subtotal calculation
   - Remove button
3. Quantity changes trigger PUT request to update cart in database
4. Remove button sends DELETE request to `/api/cart/{itemId}`
5. Total price calculated dynamically as items change
6. "Checkout" button creates order from cart items
7. Empty cart message displayed when no items present

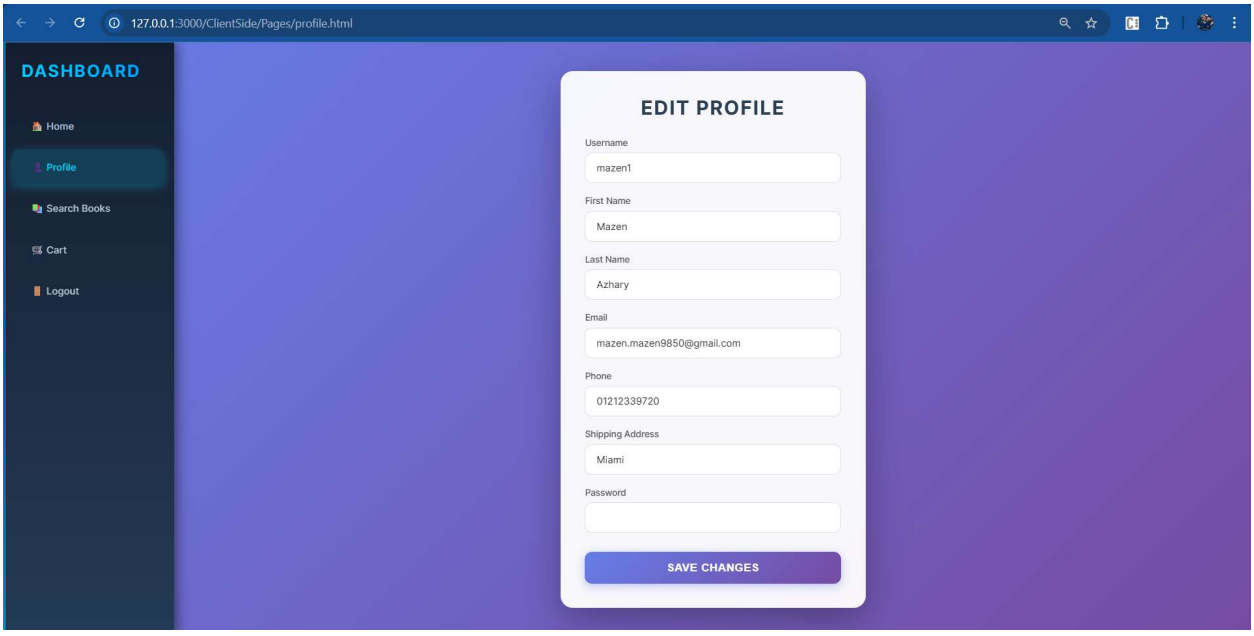
Order History

Purpose: View past and pending orders.

Logic Flow:

1. Fetch orders from `/api/orders/user/{userId}` endpoint
2. Display orders in chronological order (most recent first)
3. Each order card shows:
   - Order number and date
   - Order status (Pending, Confirmed, Shipped, Delivered)
   - List of books with quantities
   - Total amount
4. Color-coded status indicators
5. Filter options by status (All, Pending, Confirmed, etc.)
6. Click order for detailed view

Profile Management



Purpose: Update personal information and account settings.

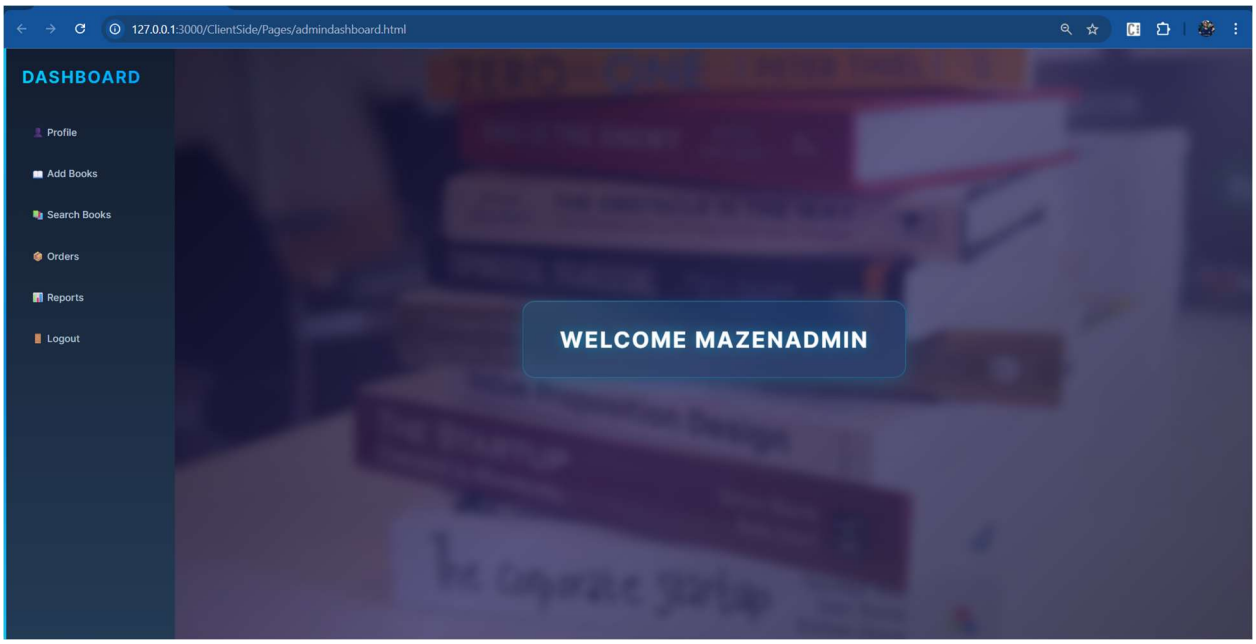
Logic Flow:

1. Load current user data from `/api/users/{userId}` endpoint

2. Pre-fill form with existing information
3. Editable fields:
    - Username (read-only for security)
    - First name and last name
    - Email
    - Phone number
    - Shipping address
    - Password (optional update)
4. Form validation on input (email format, phone format)
5. "Save Changes" sends PUT request to `/api/users/{userId}`
6. Backend validates and updates database
7. Success/error message displayed
8. Local storage updated with new information
9. Adaptive sidebar shows/hides admin options based on role

Administrator Interface

Admin Dashboard

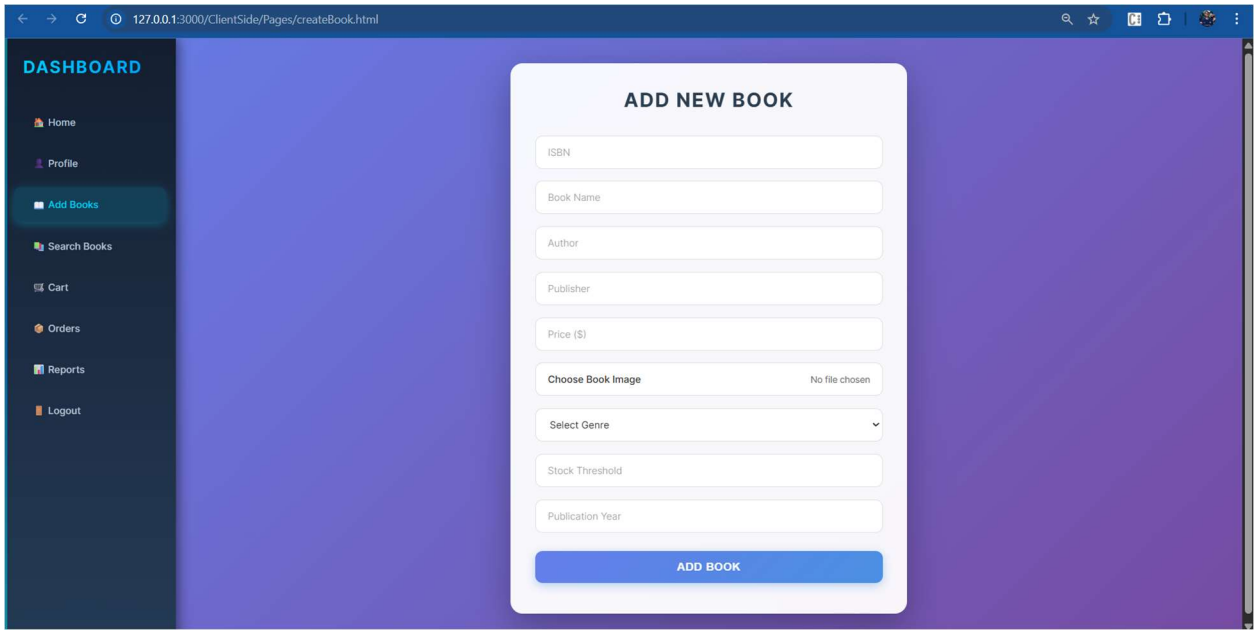


Purpose: Central hub for administrative operations.

Logic Flow:

1. Strict authentication check (redirect non-admins to index)
2. Role validation from local storage
3. Enhanced sidebar with admin-specific options:
   - Add Books
   - Manage Orders
   - System Reports
   - Profile
   - Search Books
4. Modern gradient design with glowing border effects
5. Hover animations on navigation items
6. Background overlay with welcome message

Add Book Page



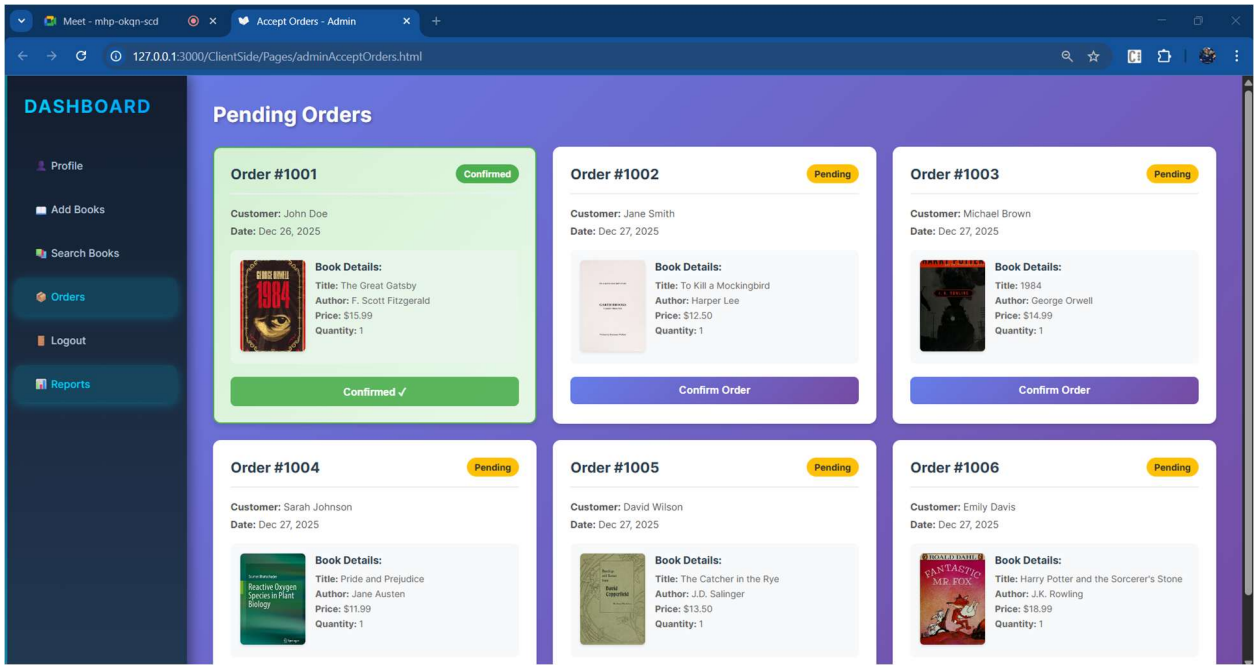
Purpose: Add new books to inventory.

Logic Flow:

1. Admin-only access with authentication check

2. Form fields for book details:
    - ISBN (10-13 characters, validated)
    - Book title
    - Author(s) (comma-separated for multiple)
    - Publisher (dropdown or text input)
    - Price (must be positive)
    - Genre (dropdown with predefined categories)
    - Publication year (1890 to current year)
    - Stock threshold (for inventory alerts)
    - Book cover image (JPG, JPEG, PNG only, max 5MB)
3. Form data persisted in session storage
4. Image preview on file selection
5. Client-side validation before submission
6. POST request to `/api/admin/books` with authorization token
7. Backend validates and adds to database
8. Success confirmation clears form and session storage
9. Error handling for duplicate ISBN or validation failures

Accept Orders Page



Purpose: Process customer orders.

Logic Flow:

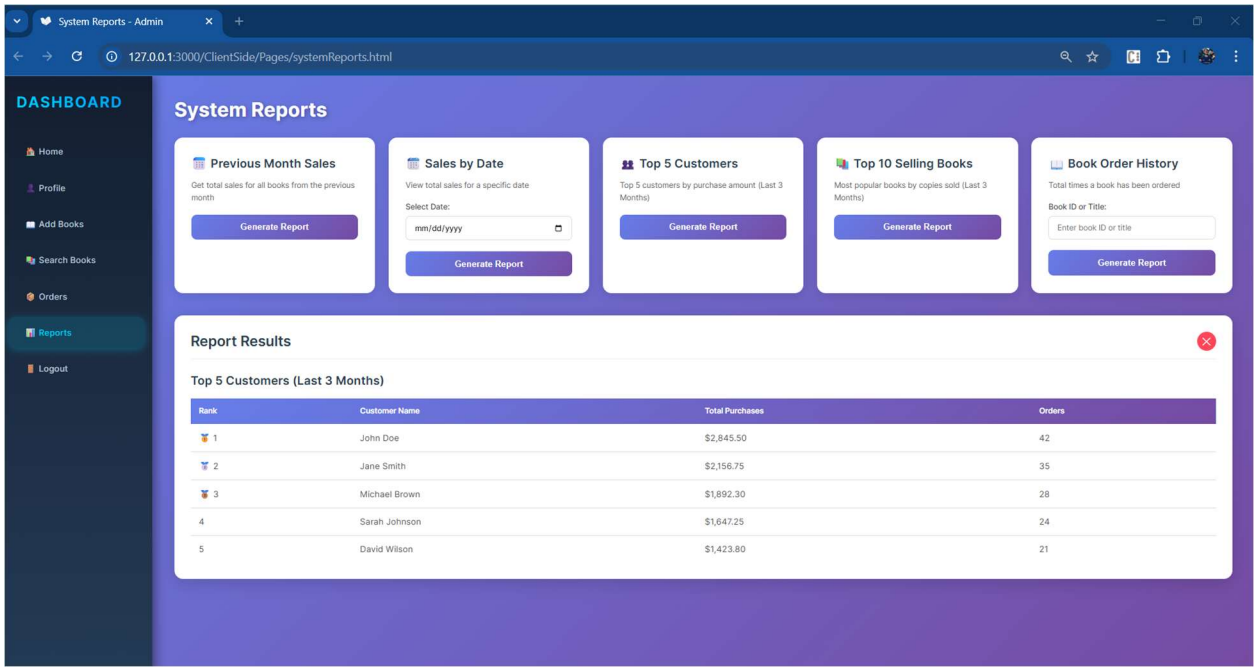
1. Fetch pending orders from `/api/admin/orders` endpoint
2. Display orders in grid layout
3. Each order card contains:
   - Order number and status badge
   - Customer name and order date
   - Book cover image
   - Book details (title, author, price, quantity)
   - "Confirm Order" button
4. Clicking confirm:
   - Sends PUT request to `/api/admin/orders/{orderId}/confirm`
   - Updates order status in database

- Changes card background to green gradient
- Status badge updates to "Confirmed"
- Button becomes disabled and shows "Confirmed ✓"
- Order remains visible (not removed from page)

5. Responsive grid adjusts to screen size
6. Real-time visual feedback without page reload

System Reports Page

Purpose: Generate business intelligence reports.



Logic Flow:

1. Admin authentication verification
2. Five report cards displayed:

   Previous Month Sales:

   - Button click sends GET request to `/api/reports/previous-month`
   - Backend calculates total sales, books sold, and orders
   - Results displayed in stat boxes

   Sales by Date:

- Date picker for user input
- Validation ensures date is selected
- GET request to `/api/reports/sales-by-date?date={date}`
- Shows sales metrics for specific date

Top 5 Customers:

- GET request to `/api/reports/top-customers?period=3months`
- Backend queries aggregated purchase data
- Results displayed in ranked table with medals (🥇 🥈 🥉)

Top 10 Selling Books:

- GET request to `/api/reports/top-books?period=3months`
- Shows books ranked by copies sold
- Table format with rank, title, author, quantity

Book Order History:

- Text input for book ID or title
- Validation ensures input is provided
- GET request to `/api/reports/book-orders?bookId={id}`
- Displays total times ordered, total copies, last order date

3. Results appear in modal-style container with close button
4. Loading animation during data fetch
5. Error handling for invalid inputs or API failures
6. Export functionality (future enhancement)

Technical Architecture

Frontend

- **Technology**: HTML5, CSS3, Vanilla JavaScript
- **Styling**: Custom CSS with gradients, animations, and responsive design
- **Deployment**: Nginx web server in Docker container

- **Port**: 3000

Backend

- **Framework**: Spring Boot (Java)
- **Architecture**: RESTful API with service-repository pattern
- **Authentication**: JWT (JSON Web Tokens)
- **Security**: Spring Security with role-based access control
- **Port**: 8080

Database

- **DBMS**: MySQL 8.0
- **Port**: 3306
- **Initialization**: SQL script automatically executed on container start

Deployment

- **Orchestration**: Docker Compose
- **Services**: mysql, backend, frontend
- **Networking**: Custom bridge network for inter-container communication
- **Volumes**: Persistent MySQL data storage
- **Health Checks**: MySQL health monitoring for dependent services
- Dark mode theme