Online Book Store - Project Report

1. Introduction

Project Title: Online Book Store

Team Members:

1. Irshad Ahmed M (Frontend Developer): Responsible for designing and implementing the user interface using React.

2. Ganesh Ram K (Backend Developer): Manages server-side logic and API development using Node.js and Express.js.

3. Dhayanithi D(Database Manager): Handles the database schema, connections, and data management with MongoDB.

4. James David P(Tester): Ensures the quality of the application through rigorous testing and debugging.

This project aims to provide a comprehensive online platform for users to browse and purchase a variety of books.

POSTER A4
Bookstore
STORE LOGO
20% OFF
BOOK
new in town !!
STORE
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Quis ipsum suspendisse ultrices gravida.
www.website.com

Online Book Store - Project Report

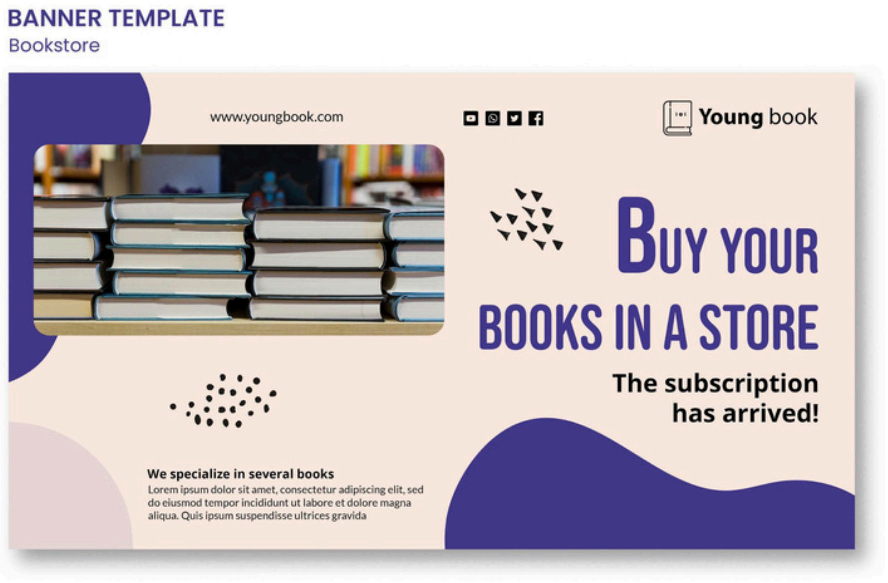
2. Project Overview

Purpose:

The Online Book Store is developed to offer an easy-to-use platform for book enthusiasts to discover, browse, and purchase books online. The application provides a seamless user experience with efficient search, filter options, and a streamlined checkout process. The project targets both individual customers and small businesses.

Key Features:

- User authentication and role-based access control (admin and customer).
- Detailed book listings with categories, authors, and rating information.
- Shopping cart with real-time updates and order summary.
- Payment integration (planned for future enhancement).
- Admin dashboard for managing inventory, users, and orders.
- Responsive design for desktop and mobile users.


BANNER TEMPLATE
Bookstore
www.youngbook.com
Young book
BUY YOUR
BOOKS IN A STORE
The subscription
has arrived!
We specialize in several books
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Quis ipsum suspendisse ultrices gravida

3. Architecture

Frontend Architecture:

The frontend is built using React, utilizing modern hooks, context API for state management, and React Router for navigation. The UI components are structured using reusable components and styled with CSS and Material-UI.

Backend Architecture:

The backend uses Node.js with Express.js to handle RESTful API requests. It includes controllers for handling business logic, routes for API endpoints, and middleware for authentication and error handling.

Database Architecture:

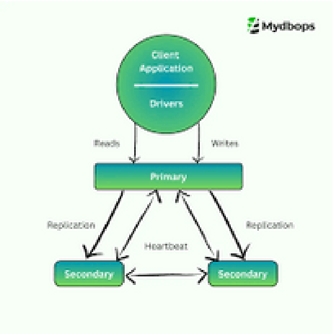
MongoDB serves as the NoSQL database, using Mongoose for schema definition and database interactions.

Collections include:

- Users: Stores user credentials, roles, and session data.

- Books: Contains book details, categories, and stock information.

- Orders: Tracks user orders, including items, quantities, and status updates.



4. Setup Instructions

Prerequisites:

- Node.js (v14 or higher)

- MongoDB (local instance or MongoDB Atlas)

- Git for version control

Installation Steps:

1. Clone the repository: git clone [repository-url]

2. Install dependencies:

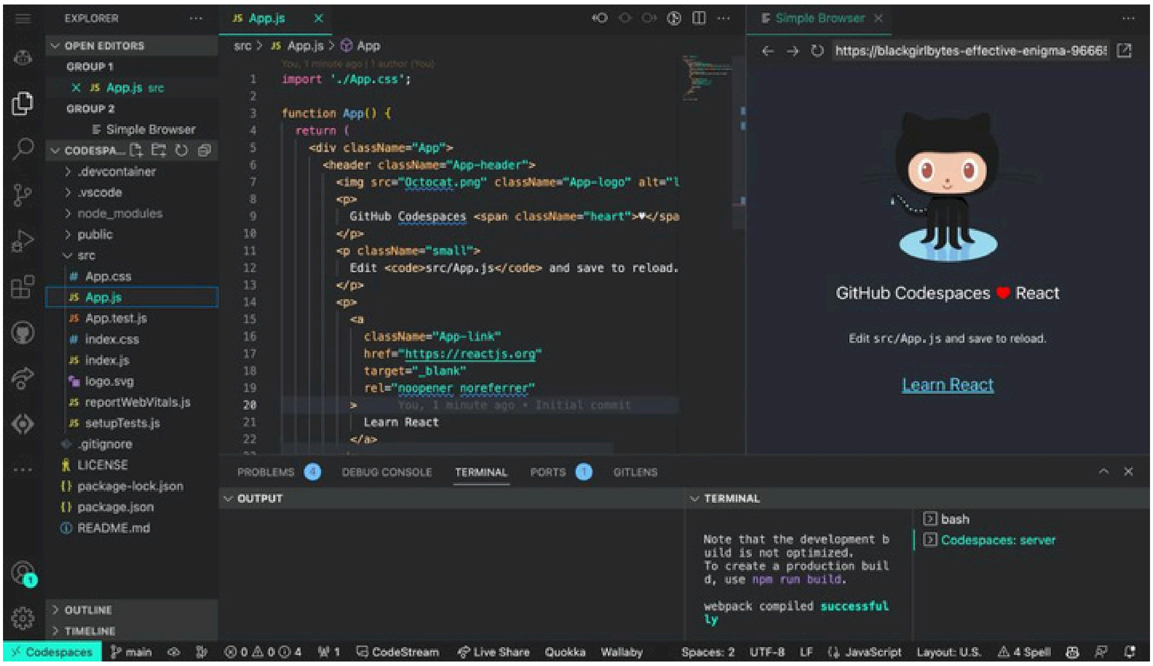
- Navigate to the client directory and run: npm install

- Navigate to the server directory and run: npm install

3. Configure environment variables:

- Create a .env file in both client and server directories.

- Set variables for database URI, API keys, and JWT secrets.



```
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src="Octocat.png" className="App-logo" alt="l
        <p>
          GitHub Codespaces <span className="heart">♥</spa
        </p>
        <p className="small">
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <p>
          <a
            className="App-link"
            href="https://reactjs.org"
            target="_blank"
            rel="noopener noreferrer"
          >
            Learn React
          </a>
```

```
Note that the development b
uild is not optimized.
To create a production buil
d, use npm run build.

webpack compiled successful
ly
```

GitHub Codespaces ♥ React

Edit src/App.js and save to reload.

Learn React

5. Folder Structure

Client Folder Structure:

- /public: Static files and assets
- /src/components: Reusable UI components
- /src/pages: Different pages of the application (Home, Catalog, Checkout, etc.)
- /src/context: State management using Context API
- /src/services: API call functions

Server Folder Structure:

- /controllers: Business logic for handling API requests
- /models: MongoDB schemas for users, books, and orders
- /routes: Defines API endpoints for authentication, books, and orders
- /middleware: Includes authentication and error handling middleware

6. Running the Application

To run the application locally:

1. Start MongoDB service (if using local MongoDB): mongod

2. Start the backend server:

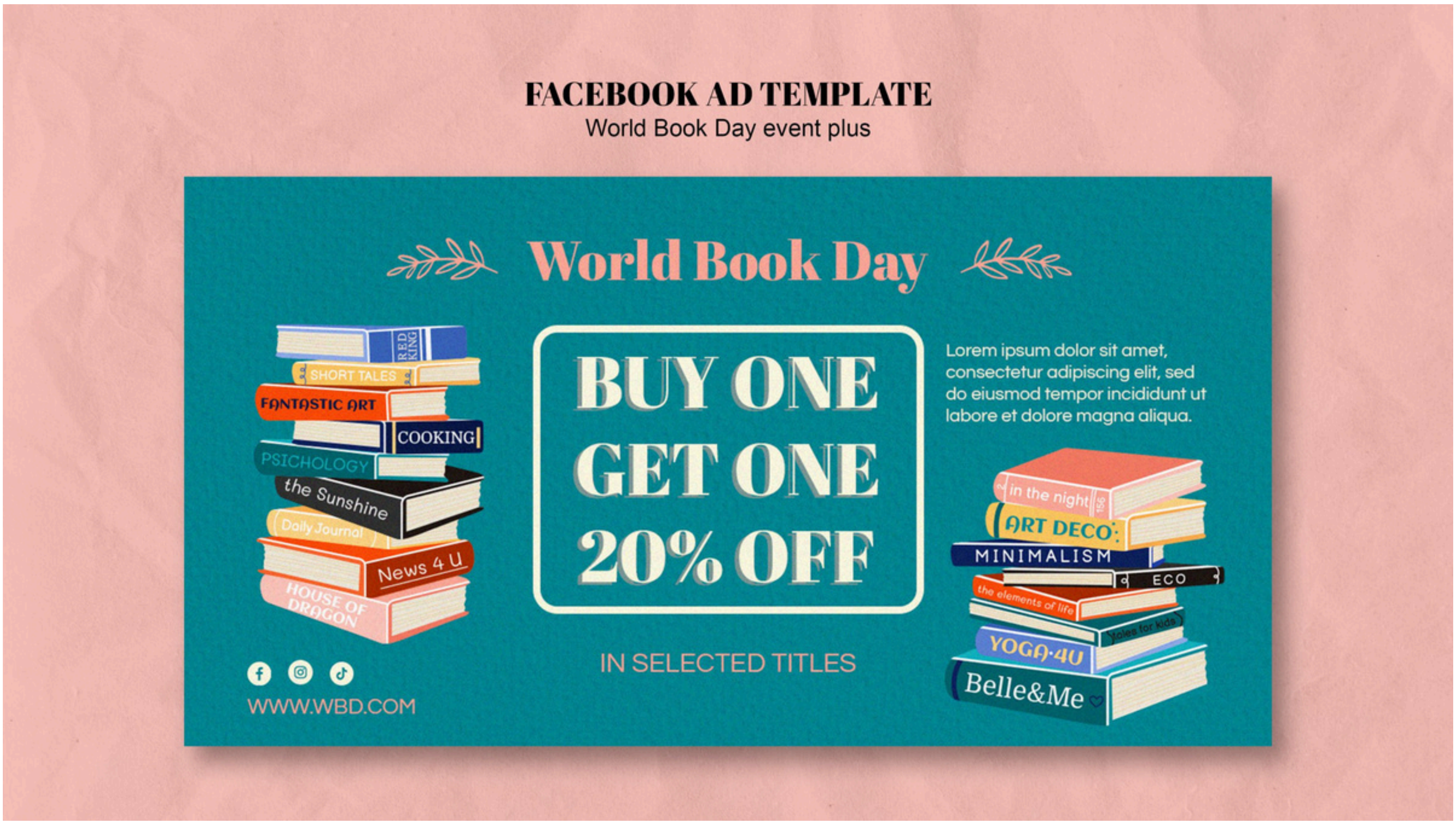
 - Navigate to the server directory and run: npm start

3. Start the frontend server:

 - Navigate to the client directory and run: npm start

4. Access the application at http://localhost:3000.

Make sure both the frontend and backend servers are running simultaneously.



7. API Documentation

API Endpoints:

- GET /api/books: Retrieves a list of all books.

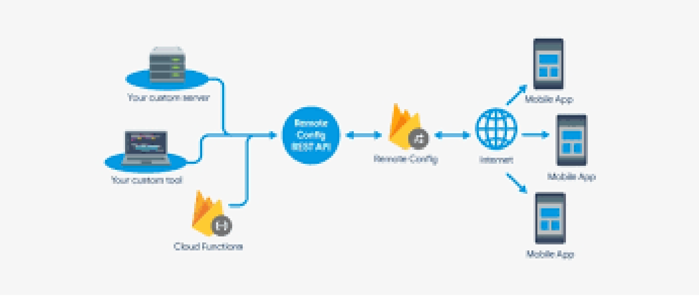
- GET /api/books/:id: Fetches details of a specific book by its ID.

- POST /api/auth/register: Registers a new user.

- POST /api/auth/login: Authenticates a user and provides a JWT.

- GET /api/orders: Retrieves all orders for an authenticated user.

- POST /api/orders: Creates a new order.



8. Authentication

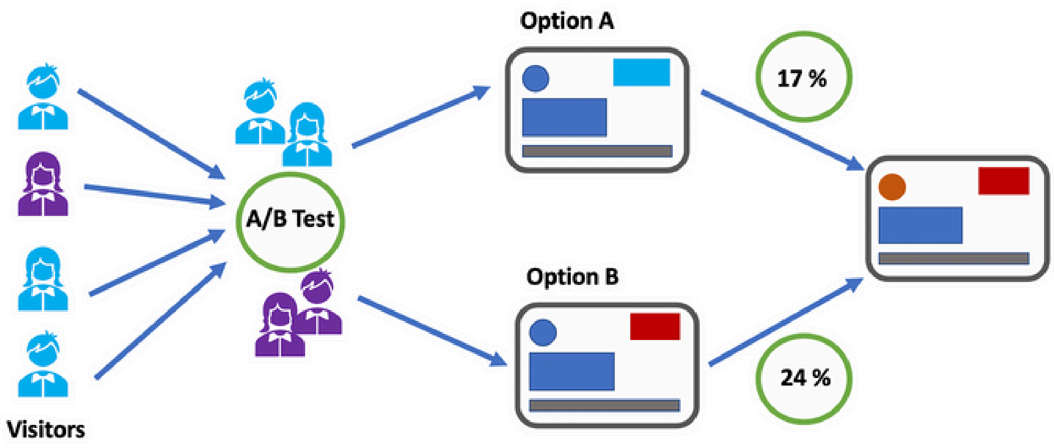
Authentication and Security:

The application uses JSON Web Tokens (JWT) for user authentication. Tokens are generated upon successful login and stored in the client-side local storage. Protected routes are secured with middleware that verifies the token before granting access. The system uses bcrypt for hashing passwords.

9. Testing

Testing Strategy:

Frontend testing is conducted using Jest and React Testing Library to verify component behavior and UI elements. Backend testing employs Mocha, Chai, and Supertest for API endpoint testing, covering positive and negative test cases. Continuous testing is facilitated through GitHub Actions for CI/CD integration.



10. Known Issues & Future Enhancements

Known Issues:

- UI may have alignment issues on smaller devices.

- The search functionality may not handle special characters correctly.

- Error messages could be more descriptive for API failures.

Future Enhancements:

- Payment gateway integration for seamless transactions.

- Wishlist feature for users to save books for later purchase.

- Integration with a book recommendation engine based on user preferences.

- Enhanced admin analytics dashboard with detailed sales reports.