BookNest : MERN Bookstore Application -Report

# Introduction

Project Title: BookNest : MERN Bookstore Application -Report

## Team Members:

* + Abinaya shree D
  + Anto jerin joshva A
  + Kesava moorthi A
  + Gokul kannan K

# Project Overview

## Purpose:

## The Book Store Project is a comprehensive e-commerce solution designed to streamline book discovery, purchase, and management. It offers a user-friendly interface with features like advanced search, filtering options, personalized recommendations, secure payment integration, and real-time order tracking. Administrators benefit from robust backend tools, including inventory management, sales analytics, and operational optimization, enabling data-driven decision-making. With scalable architecture, multi-platform compatibility, and stringent security protocols, the platform caters to diverse customer needs while driving business growth and expanding market reach.

## Features:

**1.User Features:**

AdvancedSearch books by title, author, genre, or price.

Personalized Recommendations**,** AI-driven suggestions based on preferences

**2. Admin Features:**

Inventory Management likeAdd, update, and track book stock.

Reports and Analytics Insights into sales trends and customer behavior

**3. Technical Features:**

Secure Payment Integration Supports cards, UPI, and wallets.

Responsive Design Optimized for web, tablet, and mobile devices.

**4.Security:**

Data Encryption Protects user and payment data.

Authentication Role-based access with two-factor authentication.

# Architecture

## Frontend:

* + Developed using **React.js** for its component-based architecture and state management capabilities.
  + Features dynamic components like:
    - **Product Listings**: Displays products with sorting and filtering options.
    - **Cart Management**: Allows users to add, update, or remove books in their cart.
    - **User Authentication**: Login and registration pages with secure validation.
    - **Admin Panel**: Provides sellers with tools to manage inventory and view analytics.

## Backend:

* + Built with **Node.js** and **Express.js**, ensuring scalability and high performance.
  + Features include:

## API Endpoints:

* + - * /books for fetching book data.
      * /orders for processing customer orders.
      * /users for managing user authentication.
    - **Middleware** for error handling and authentication using **JWT**.

## Database:

* + **MongoDB** serves as the database, storing collections for:
    - **Users**: Authentication credentials, profiles, and purchase history.
    - **Products**: Information on inventory, prices, categories, and descriptions.
    - **Orders**: Details about placed orders, delivery status, and payment.

This architecture ensures modularity, scalability, and efficient data management.

# Setup Instructions

## Prerequisites:

* + **Node.js**: v14 or later
  + **MongoDB**: Installed locally or set up using a cloud provider like MongoDB Atlas
  + **npm**: Package manager for installing dependencies

## Installation:

**Clone the repository:**

<https://github.com/ABINAYASHREE2204/bookstorenm>

## Navigate to the project directory:

cd bookstore

## Install dependencies for the backend and frontend:

Backend:

cd frontend

npm install Frontend:

cd backend npm install

## Run the servers:

Backend:

node index.js

Frontend:

npm start:dev

# Folder Structure

## Client:

* + src/components: Contains reusable components like **Navbar**, **ProductCard**, **CartItem**, etc.
  + src/pages: Holds page components such as **Home**, **Cart**, **Checkout**, and

## AdminPanel.

* + src/redux: Implements state management for cart items, user authentication, and order status.

## Server:

* + routes: Defines all RESTful API routes for users, products, and orders.
  + controllers: Contains the logic for handling backend requests.
  + models: Defines database schemas for **Users**, **Products**, and **Orders**.
  + middleware: Handles authentication (JWT) and error management.

# Running the Application

## Frontend:

Navigate to the frontend:

cd frontend

Start the React application:

npm start

## Backend:

Navigate to the backend:

cd backend

Start the Node.js server:

node index.js

# API Documentation

## Endpoints:

1. **GET /books**
   * Fetches the list of all products.
   * Parameters: Filters like author,genre , price.

Response: jjson

{

"\_id": 1,

      "title": "How to Grow Your Online Store",

      "description": "Learn the best strategies to grow your online store in today's competitive market.",

      "category": "business",

      "trending": true,

      "coverImage": "book-1.png",

      "oldPrice": 29.99,

      "newPrice": 19.99

    }

## POST /orders

* + Places a new order.
  + Parameters: book ID, book details, and category.

Response: json

{

const getAllBooks =  async (req, res) => {

    try {

        const books = await Book.find().sort({ createdAt: -1});

        res.status(200).send(books)

    } catch (error) {

        console.error("Error fetching books", error);

        res.status(500).send({message: "Failed to fetch books"})

    }

}

# Authentication

## JWT-based Authentication:

* + **Login**: Issues a JWT token upon successful authentication.
  + **Token Validation**: Protects private routes like /orders and /admin.
  + **Logout**: Invalidates the token on the client side.

# User Interface

## Screens:

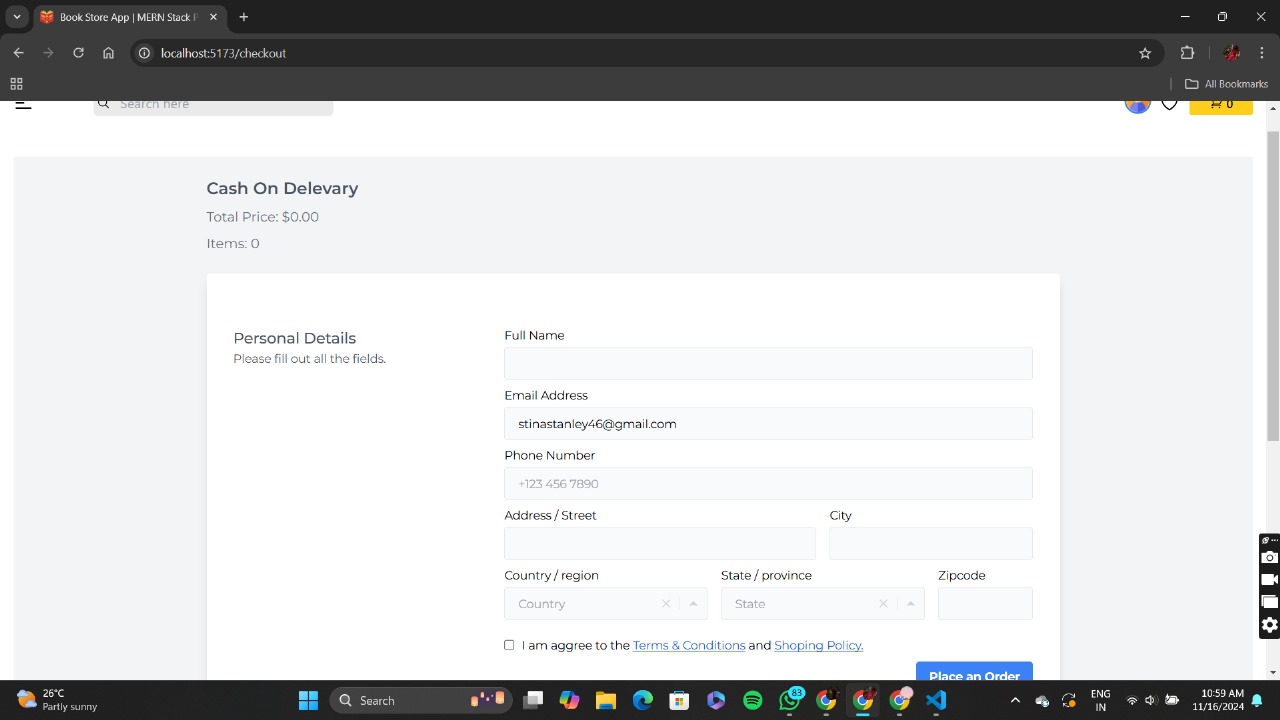
* + **Home:** Displays trending books and categories.
  + **Book Details:** Shows detailed information about a selected book.
  + **Admin Dashboard:** Offers order status updates and analytics for sellers.

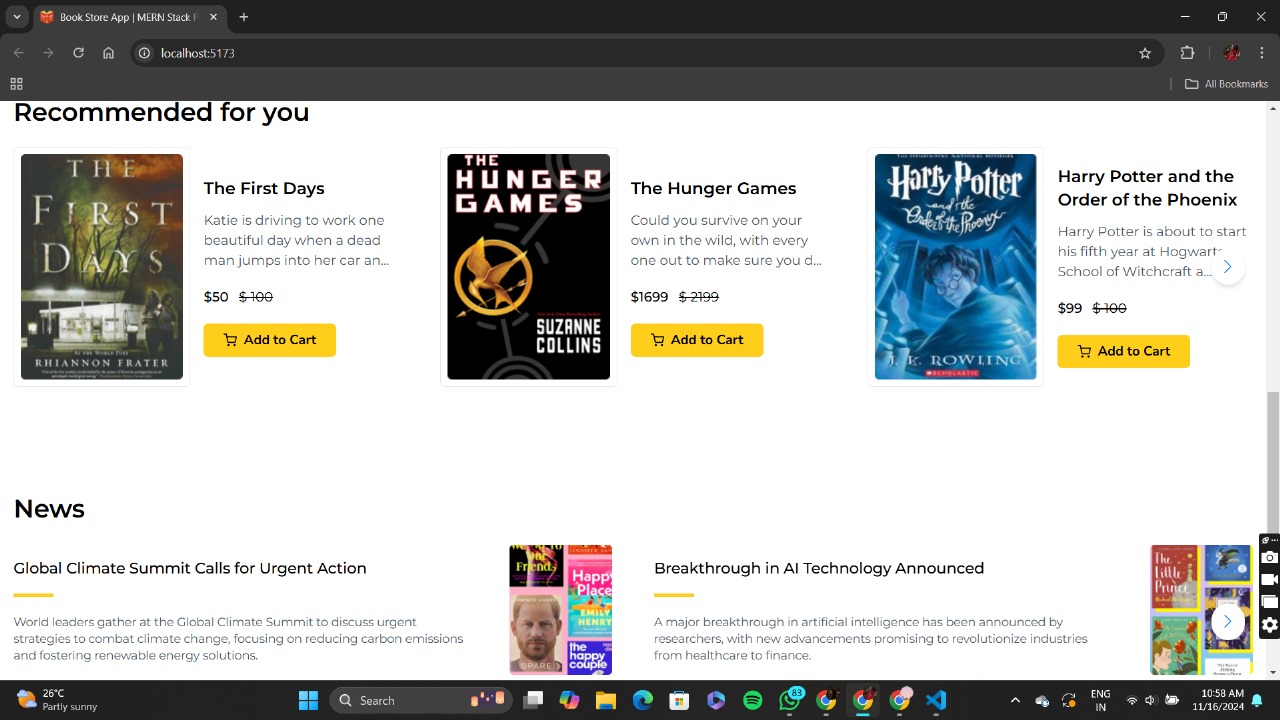
# Testing

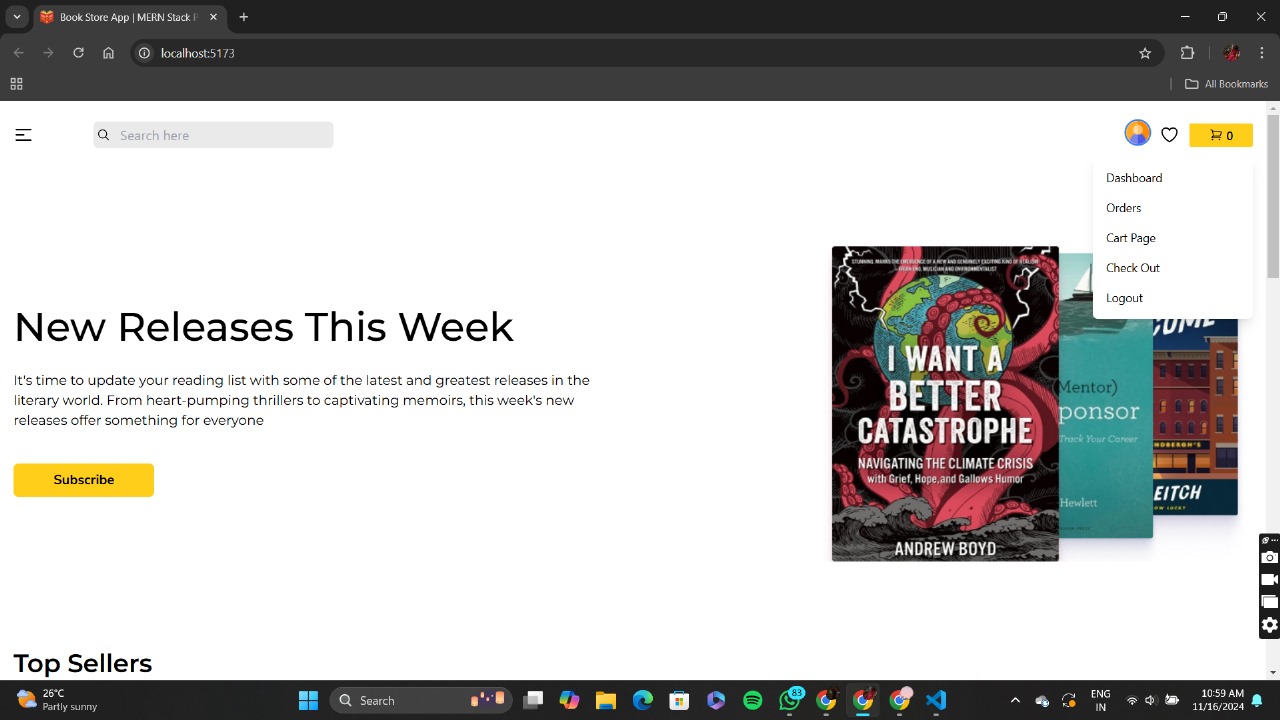
* + **Unit Testing:** Conducted using **Jest** for components and backend logic.
  + **API Testing:** Performed using **Postman** to validate endpoints.

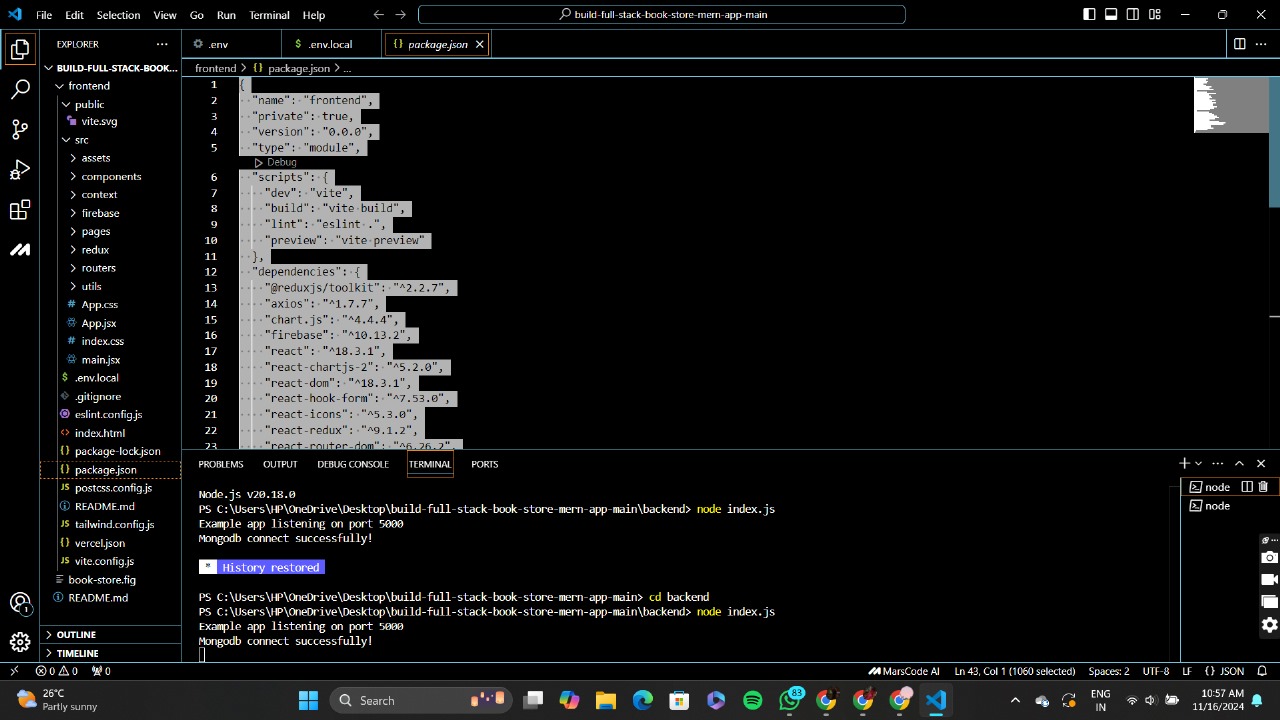
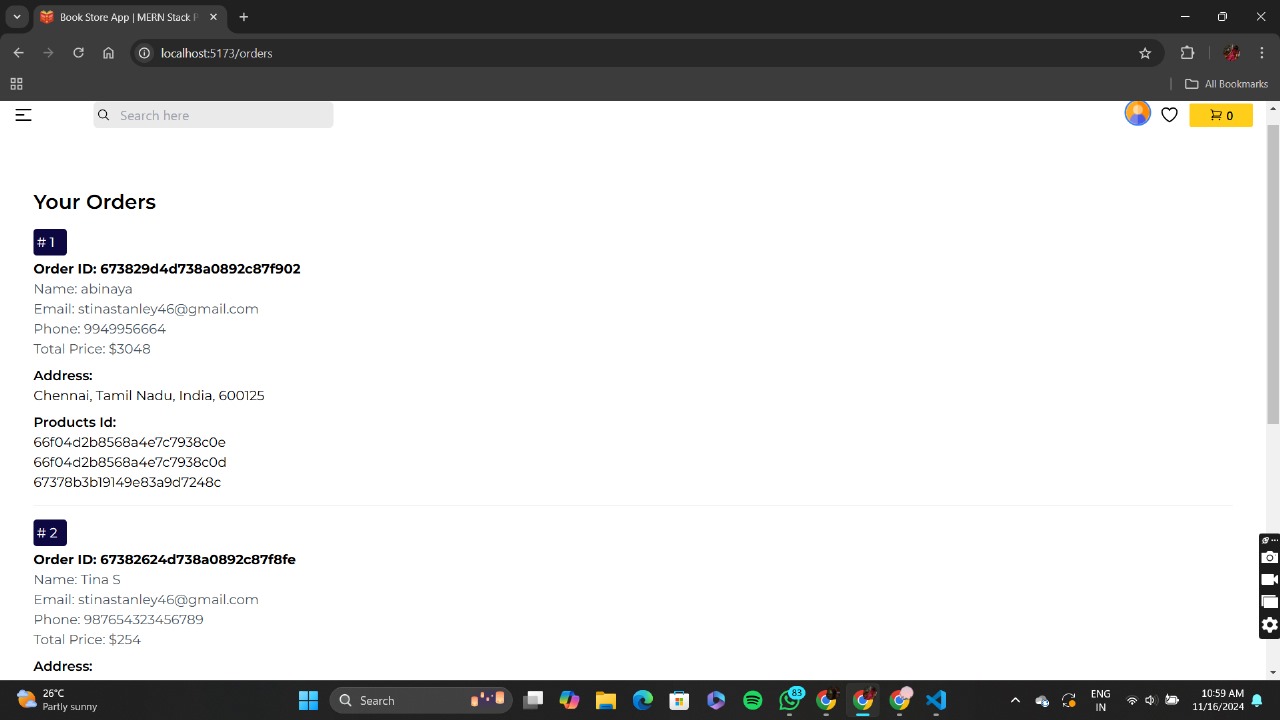
# Screenshots or Demo

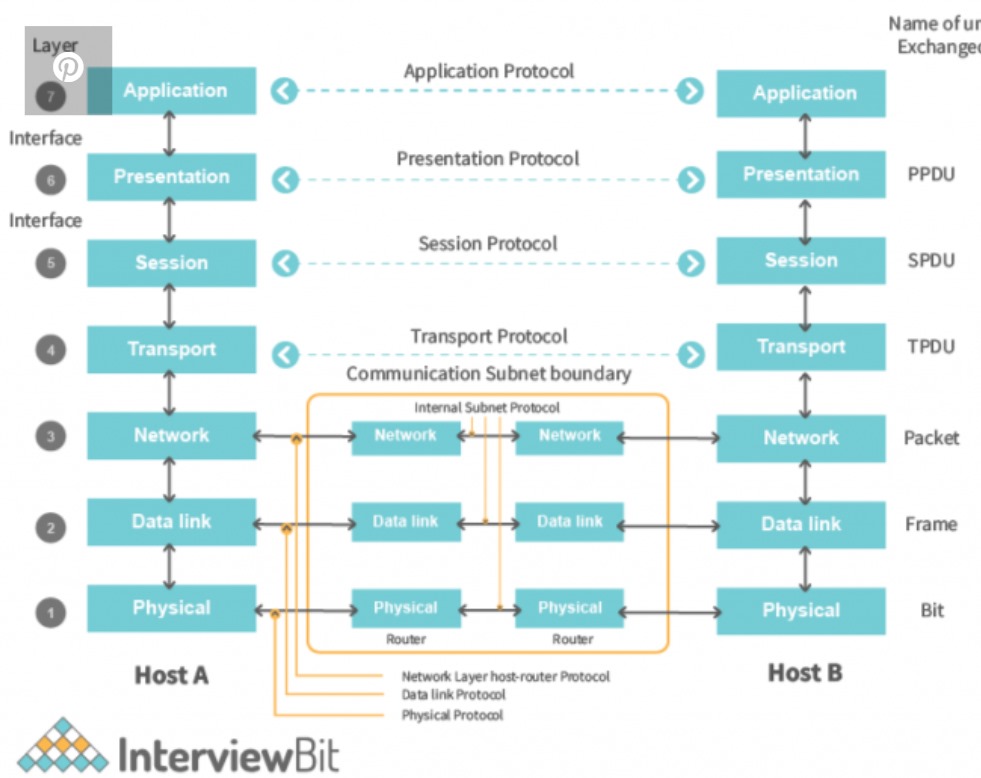
Screenshots of key UI components and admin panel.











Link to a live demo.



[**Demo Video.mp4**](https://drive.google.com/file/d/1B025A59a0S6EKPuVxcbPbKBkClTjvwuN/view?usp=sharing) **/**

https://drive.google.com/file/d/1KwnOZqA\_MH-isHI5kKAsYiUTOKjbMN7S/view?usp=sharing.

1. **Known Issues**
   * **Slow Search Performance:** Optimization is required for large product datasets.
   * **UI Bugs:** Minor alignment issues on smaller screens.

# Future Enhancements

* + **Voice Search:** Enable users to search using voice commands.
  + **Mobile App:** Create a cross-platform app using React Native.
  + **Multi-language Support:** Expand accessibility for global audiences.