### Tribhuvan University (TU) Logo PNG Vector

**Tribhuvan University**

**Faculty of Humanities and Social Science**

**A PROJECT REPORT ON**

**Online Book Store**

**Submitted to**

**Department of Computer Application**

**Everest College**

**In partial fulfillment of the requirements for the Bachelor’s in Computer Application**

**Submitted by**

Anil Shrestha (6-2-355-4-2021)

Sachin Magar (6-2-355-22-2021)

August, 2025

Under the Supervision of

Sunny Kumar Sah



**Tribhuvan University**

**Faculty of Humanities and Social Sciences**

**Everest College**

# SUPERVISOR’S RECOMMENDATION

I hereby recommend that this project prepared under my supervision by “**Anil Shrestha**” and “**Sachin Magar**”, entitled “**PustakBindu**” in partial fulfillment of the requirements for the degree of Bachelor of Computer Application’s is recommended for the final evaluation.

**Signature of the Supervisor**

**Sunny Kumar Sah**

Everest College, Thapathali, Kathmandu



**Tribhuvan University**

**Faculty of Humanities and Social Sciences**

**Everest College**

**LETTER OF APPROVAL**

This is to certify that this project prepared by “**Anil Shrestha**” and “**Sachin Magar**” entitled “**PustakBindu**” in partial fulfillment of the requirements for the degree of Bachelor in Computer Application has been evaluated. In our opinion it is satisfactory in terms of scope and quality as a project for the required degree.

|  |  |
| --- | --- |
| ………………………………  Sunny Kumar Sah  Supervisor  Thapathali, Kathmandu | ---------------------------------  Dharma Raj Poudel  Coordinator  Thapathali, Kathmandu |
| -------------------------------  Internal Examiner | ------------------------------  External Examiner |

# ABSTRACT

**PustakBindu** is an online bookstore developed using the MERN stack (MongoDB, Express.js, React, and Node.js) to provide a smooth and user-friendly platform for browsing and purchasing books. The system allows users to search and filter books based on categories and price ranges, add selected books to a shopping cart, and proceed through a secure checkout process. It includes user account management for registration and login, as well as an admin panel for managing books, categories, and orders. The backend, built with Express.js and Node.js, handles API requests and business logic, while MongoDB stores all book, user, and order information. The frontend, built with React, offers a responsive and interactive user interface accessible across devices. Designed for simplicity, performance, and scalability, PustakBindu is aimed at helping bookstores establish an effective online presence. While it does not yet include AI-based personalization or advanced recommendation features, it provides a strong, extensible foundation for future enhancements. This project demonstrates the practical application of the MERN stack in building a functional e-commerce platform.

**Keywords:** PustakBindu, MERN stack, Online bookstore, E-commerce, Web application

# ACKNOWLEDGEMENT

We would like to express our sincere gratitude to everyone who supported and contributed

to the development of the PustakBindu project. First and foremost, we would like to thank our teachers and mentors for their invaluable guidance and encouragement throughout this project. We are also deeply grateful to our supervisor for their continuous support and direction. Your expertise and feedback helped steer this project towards success. Lastly, we would like to acknowledge the online communities and resources that provided the technical knowledge and inspiration necessary for this project. Its contributions to the broader field of educational technology are deeply appreciated. Thank you all for your support and encouragement. This project would not have been possible without you.

Anil Shrestha (6-2-355-4-2021)

Sachin Magar (6-2-355-22-2021)

Table of Contents

[SUPERVISOR’S RECOMMENDATION ii](#_Toc206102227)

[ABSTRACT iv](#_Toc206102228)

[ACKNOWLEDGEMENT v](#_Toc206102229)

[LIST OF ABBREVIATIONS x](#_Toc206102230)

[CHAPTER 1: INTRODUCTION 1](#_Toc206102231)

[1.1 Introduction 1](#_Toc206102232)

[1.2 Problem Statement: 1](#_Toc206102233)

[1.3 Objectives 2](#_Toc206102234)

[1.4. Scope and Limitations 2](#_Toc206102235)

[1.4.1. Scope 2](#_Toc206102236)

[1.4.2. Limitations 2](#_Toc206102237)

[1.5. Development Methodology 3](#_Toc206102238)

[1.6. Report Organization 3](#_Toc206102239)

[CHAPTER 2 : BACKGROUND STUDY AND LITERATURE REVIEW 5](#_Toc206102240)

[2.1 Background Study 5](#_Toc206102241)

[2.2. Literature Review 5](#_Toc206102242)

[2.2.1. Review of Similar Systems 6](#_Toc206102243)

[CHAPTER 3: SYSTEM ANALYSIS AND DESIGN 7](#_Toc206102244)

[3.1. System Analysis 7](#_Toc206102245)

[3.1.1 Requirement Analysis 7](#_Toc206102246)

[3.1.2. Feasibility Study 9](#_Toc206102247)

[3.1.3. Object Modelling 11](#_Toc206102248)

[3.1.4. Dynamic Modeling (State & Sequence diagram) 12](#_Toc206102249)

[3.1.5. Process modelling: Activity Diagram 15](#_Toc206102250)

[3.2. System Design 16](#_Toc206102251)

[3.2.1. Refinement of Classes and Object 16](#_Toc206102252)

[3.2.2. Component Diagram 17](#_Toc206102253)

[3.3. Algorithm 18](#_Toc206102254)

[CHAPTER 4: IMPLEMENTATION AND TESTING 19](#_Toc206102255)

[4.1. Implementation 19](#_Toc206102256)

[4.1.1. Tools Used (CASE tools, programming language, database platforms) 19](#_Toc206102257)

[4.1.2. Implementation details of modules (description of procedures/ functions/ classes/ methods) 20](#_Toc206102258)

[4.2. Testing 22](#_Toc206102259)

[4.2.1. Unit testing 22](#_Toc206102260)

[4.2.2. Test Cases for System Testing 29](#_Toc206102261)

[CHAPTER 5: CONCLUSION AND FUTURE RECOMMENDATIONS 30](#_Toc206102262)

[5.1. Conclusion 30](#_Toc206102263)

[5.2. Lesson Learnt / Outcome 30](#_Toc206102264)

[5.2. Future Recommendation 31](#_Toc206102265)

[APPENDICES 32](#_Toc206102266)

**TABLE OF FIGURES**

[Figure 1: Iterative Waterfall Model of PustakBindu 3](#_Toc206105270)

[Figure 2: USE CASE Diagram of PustakBindu 8](#_Toc206105271)

[Figure 3: Gantt Chart of PustakBindu 10](#_Toc206105272)

[Figure 4: Class Diagram of PustakBindu 11](#_Toc206105273)

[Figure 5: State Diagram for Users 12](#_Toc206105274)

[Figure 6: State Diagram for Admin 13](#_Toc206105275)

[Figure 7: Sequence Diagram 14](#_Toc206105276)

[Figure 8: Activity Diagram of user in PustakBindu 15](#_Toc206105277)

[Figure 9: Refinement of Class Diagram of PustakBindu 16](#_Toc206105278)

[Figure 10: Refinement of Object Diagram of PustakBindu 17](#_Toc206105279)

[Figure 11: Object Diagram of PustakBindu 17](#_Toc206105280)

[Figure 12: Test Case Evidence 1.1 23](#_Toc206105281)

[Figure 13: Test Case Evidence 2.1 24](#_Toc206105282)

[Figure 14: Test Case Evidence 3.1 24](#_Toc206105283)

[Figure 15: Test Case Evidence 4.1 25](#_Toc206105284)

[Figure 16: Test Case Evidence 5.1 25](#_Toc206105285)

[Figure 17: Test Case Evidence 6.1 26](#_Toc206105286)

[Figure 18: Test Case Evidence 7.1 27](#_Toc206105287)

[Figure 18: Test Case Evidence 8.1 27](#_Toc206105288)

[Figure 19: Test Case Evidence 8.2 28](#_Toc206105289)

[Figure 20: Test Case Evidence 8.3 28](#_Toc206105290)

**LIST OF TABLES**

[Table 4.1: Test Cases for Unit Testing 22](#_Toc206093927)

# LIST OF ABBREVIATIONS

|  |  |
| --- | --- |
| OOP: | Object-Oriented Programming |
| API: | Application Programming Interface |
| UML: | Unified Modeling Language |
| IDE: | Integrated Development Environment |
| GUI: | Graphical User Interface |
| DBMS: | Database Management System |
| CRUD: | Create, Read, Update, Delete |

JSON: JavaScript Object Notation

XML: eXtensible Markup Language

# CHAPTER 1: INTRODUCTION

## Introduction

PustakBindu is an online bookstore platform designed to provide a seamless and user-friendly experience for book lovers and sellers alike. In today’s fast-growing digital marketplace, accessing and managing books efficiently is essential, yet many small sellers and readers face challenges due to limited reach, complex interfaces, or lack of tailored search and filtering options.

This project addresses these challenges by offering a comprehensive, easy-to-use web application where users can browse, search, and filter books by categories and price. Buyers can easily list their books, and reach a wider audience without needing extensive technical expertise.

PustakBindu emphasizes three key principles: simplicity, responsiveness, and personalization. The platform offers fast, dynamic search and filtering capabilities, helping users find exactly what they want in real-time. Its responsive design ensures a smooth experience across devices, while personalized recommendations and user ratings enhance discovery and trust.

Built using the MERN stack — MongoDB for database management, Express.js and Node.js for backend APIs, and React.js for a dynamic frontend — PustakBindu is a scalable and robust solution tailored for the modern online book market. This report details the project’s goals, system design, development process, testing outcomes, and future enhancement plans.

## Problem Statement:

In today’s digital marketplace, book buyers often struggle to find a convenient and efficient platform that offers a wide range of books with easy search and filtering options. Existing online bookstores may have complex interfaces or lack personalized features, making it difficult for users to discover and purchase books quickly. Additionally, many platforms do not provide real-time responsiveness or clear categorization, resulting in a suboptimal shopping experience. There is a need for a user-friendly, responsive, and scalable online bookstore where users can easily browse, search, and filter books to find exactly what they want.

## Objectives

PustakBindu is a stateless, real-time online bookstore platform designed to provide users with quick and efficient access to a wide range of books. It aims to implement a responsive system that delivers fast search and filtering results based on user queries and preferences, ensuring a seamless browsing and purchasing experience without maintaining session state on the server.

## 1.4. Scope and Limitations

## 1.4.1. Scope

This project focuses on building a lightweight, user-friendly online bookstore platform designed for book buyers to browse, search, filter, and purchase books efficiently. The system manages book inventory and user interactions without maintaining server-side sessions, providing a fast and responsive experience.

It includes:

* A backend API for managing book data, user requests, and order processing
* Dynamic search and filtering functionality based on categories and price
* A React frontend that delivers a smooth and responsive user interface
* Testing to ensure reliable performance across various devices and use cases

The project does not cover seller-managed listings or marketplace features.

### 1.4.2. Limitations

While PustakBindu is designed to be effective and practical, it has the following limitations:

* The platform does not support multiple sellers or user-generated listings; all books are managed and sold directly by the store.
* Inventory management features are basic and may not handle complex stock scenarios like backorders or multiple warehouses.
* The system assumes users have a stable internet connection for real-time search and filtering; offline access is not supported.
* The project does not include features such as user accounts with extensive profiles or social features like reviews and comments.

## 1.5. Development Methodology

The Iterative Waterfall Model is a hybrid software development lifecycle (SDLC) approach that combines the structured, sequential phases of the traditional Waterfall Model with iterative elements to allow for flexibility and incremental improvements.

A diagram of a process

AI-generated content may be incorrect.

Figure 1: Iterative Waterfall Model of PustakBindu

## 1.6. Report Organization

**Chapter 1**: Provides an overview of the online bookstore domain and introduces PustakBindu as a MERN-based platform for browsing, searching, and purchasing books. Defines project objectives, establishes the scope of store capabilities, and acknowledges system limitations.

**Chapter 2:** Examines existing online bookstore solutions, comparing search and filtering systems across platforms. Reviews current trends in digital book sales and analyzes gaps in fast, responsive, and user-friendly solutions for buyers.

**Chapter 3:** Presents the technical architecture including Express API design, MongoDB database schema, and React-based user interface. Details the implementation of core modules: authentication system, book catalog management, search and filtering, payment integration, and order processing.

**Chapter 4:** Documents unit testing of individual components, integration testing of the complete system, and performance benchmarks. Presents functional testing results using sample browsing and purchase scenarios.

**Chapter 5:** Summarizes project achievements against stated objectives, discusses practical applications for online book retail, and identifies areas for enhancement including advanced recommendation features and expanded payment options.

# CHAPTER 2 : BACKGROUND STUDY AND LITERATURE REVIEW

## 2.1 Background Study

An online bookstore platform refers to a web-based system that enables users to browse, search, and purchase books conveniently. In e-commerce for books, the primary challenges involve ensuring fast search results, smooth navigation, secure payment processing, and accurate inventory updates. Poor design or slow responses can lead to user dissatisfaction and abandoned purchases.

This project uses a dynamic, stateless design model, where each request is processed independently, and all necessary data is included in the request. It avoids server-side session storage and instead uses JSON Web Tokens (JWT) for secure, logic-driven authentication and authorization.

Key concepts and terminologies involved include:

* **Book Catalog Management** – maintaining book details including title, author, category, price.
* **Dynamic Search** – returning relevant results instantly as the user types or filters
* **Category and Price Filtering** – narrowing search results based on selected criteria
* **Shopping Cart Management** – adding, removing, and updating books before checkout
* **Order Processing** – recording completed purchases and updating stock accordingly
* **Responsive Design** – ensuring the platform adapts smoothly to different screen sizes and devices

These elements are processed in real time using backend logic in Express, with MongoDB for data storage, and delivered to users through a React-based frontend.

## 2.2. Literature Review

Online bookstore platforms in Nepal have emerged in various forms, often targeting specific market segments or prioritizing certain features. **BooksMandala.com** [1] offers a wide range of books with home delivery and occasional promotional discounts, but its search and filtering features are limited, making it harder for users to quickly find specific titles. **Thuprai.com** [2] provides both physical and e-book options along with a clean interface, yet it lacks real-time filtering and has limited payment method options. **BiblioNepal.com** [3] focuses on curated book collections and local delivery, but it does not provide advanced categorization or dynamic browsing capabilities.

From the reviewed platforms, it is evident that there is a gap in offering a lightweight, highly responsive bookstore system that combines efficient real-time search, secure payment integration, and a smooth, mobile-friendly interface. Existing solutions often compromise on either speed, filtering options, or integration flexibility.

PustakBindu addresses this gap by providing a MERN-based online bookstore that supports dynamic search and filtering by category and price, secure JWT-based authentication, and a responsive design that adapts seamlessly to various devices. Unlike existing platforms, it emphasizes both performance and user experience, making it practical and scalable.

### 2.2.1. Review of Similar Systems

**BooksMandala.com (2025)** [1]:

BooksMandala offers a wide collection of books with home delivery services and occasional promotional discounts. The platform has a clean layout but provides limited filtering options, which can make it harder for users to quickly find specific titles. Search results are sometimes slow to update when applying multiple filters.

**Thuprai.com (2025)** [2]:

Thuprai provides both printed books and e-books along with a subscription-based reading option. The platform has a user-friendly interface but lacks advanced filtering features and real-time search capabilities. Payment methods are also limited compared to modern e-commerce platforms.

# CHAPTER 3: SYSTEM ANALYSIS AND DESIGN

## 3.1. System Analysis

### 3.1.1 Requirement Analysis

Requirement analysis is a critical step in determining the success of a system or software project. It involves identifying and documenting the essential needs and expectations that the project must fulfill during its development. These requirements serve as the foundation for designing and implementing the system effectively.

#### Functional Requirements

* **User Registration and Login** – Users can create an account and log in using JWT-based authentication.
* **Book Catalog Display** – The system should list all available books with details such as title, author, category, description, price, and availability.
* **Search Functionality** – Users should be able to search for books by title or author.
* **Filtering** – Users should be able to filter books by category and price range.
* **Shopping Cart** – Users can add, update, and remove books from their cart.
* **Order Placement** – Users can place orders for books in their cart.
* **Payment Integration** – The platform should support secure payment processing for purchases.
* **Order Management** – The system should store order details and update inventory upon successful purchase.
* **Admin Panel** – Administrators can add, edit, and remove books from the catalog.

**Use Case Diagram**

Our system, PustakBindu, consists of three actors: Admin, Customers, and the Payment Gateway. Both customers and admins can log in to the system. Customers can browse, search, and filter books, add books to their cart, place orders, and make payments. On the other hand, admins can manage the book catalog by adding, editing, or removing books and overseeing orders. The payment gateway processes payments securely during checkout. The use case diagram is shown below:

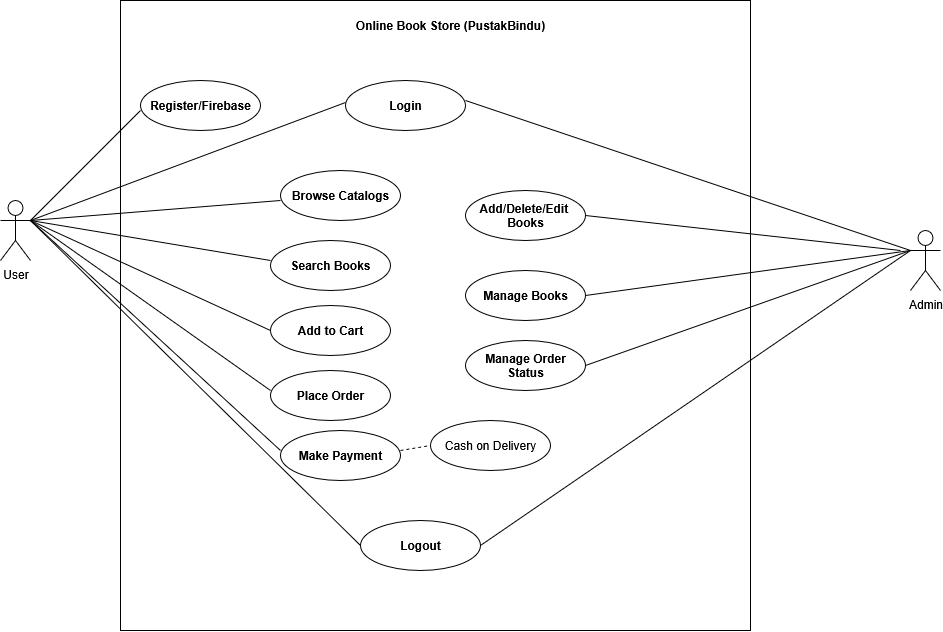


Figure 2: USE CASE Diagram of PustakBindu

#### ii. Non-functional Requirements

**Non-functional requirements** specify the criteria that judge the operation of the system, rather than specific behaviors or functions. These requirements define the quality attributes, performance, and constraints that the **Online Book Store** must satisfy to be effective and user-friendly.

* **Performance:** The system shall respond to user actions, such as searching for books or adding items to the cart, within **2 seconds** under normal network conditions.
* **Usability:** The user interface shall be intuitive and easy to navigate for users of all technical levels, with clear categories and a responsive design.
* **Security:** The system shall enforce user authentication and authorization to protect sensitive customer and transaction data.
* **Reliability:** The system shall maintain at least **99% uptime**, ensuring availability during peak shopping hours.
* **Scalability:** The system architecture shall support future scaling to accommodate increased users and a growing book catalog.
* **Maintainability:** The system shall be designed to allow easy updates, feature additions, and bug fixes with minimal downtime.
* **Compatibility:** The website shall be responsive and support access on major browsers and devices, including desktops, tablets, and smartphones.

### 3.1.2. Feasibility Study

Feasibility refers to the practicality or possibility of a proposed plan, project, or system being successful and effective. Following feasibilities were studied while building the system to ensure the system be built with exact requirements in specified time.

#### 3.1.2.1. Technical Feasibility

The system is technically feasible using **React.js**, **Node.js**, and **MongoDB**. All required tools and technologies are open-source, widely supported, and have been tested during development to ensure compatibility and performance.

#### 3.1.2.2. Operational Feasibility

The system is operationally feasible as it meets the functional requirements of the Online Book Store. Users can easily search, view, and purchase books through an intuitive interface, while admins can manage books and categories efficiently. The system is designed to integrate smoothly into existing workflows with minimal training.

#### 3.1.2.3. Economic Feasibility

The project was built entirely using free, open-source tools from the MERN stack, requiring no licensing or paid software. Hosting and backend requirements are minimal, which keeps the overall costs low and makes the project cost-effective.

In a feasibility study, three main aspects are considered: technical, operational, and economic feasibility. Among these, a Gantt chart is most closely related to technical and operational feasibility, as it helps in planning, scheduling, and monitoring the progress of project tasks like frontend development, backend API creation, and database management. While the Gantt chart does not directly measure economic feasibility, it indirectly supports cost management by identifying potential delays or resource issues that could impact the project timeline and budget.

**Gantt Chart :**

A Gantt chart is a project management tool that visually displays the schedule of tasks as a bar chart, showing their durations and sequence. It provides a clear overview of the project timeline and helps track progress efficiently.

For the online bookstore project, the total duration is 101 days, starting from April 21, 2025, and ending on July 31, 2025. There are six main phases in the project: Planning, Requirement, Design, Development, Testing, and Documentation. The documentation phase has already started and will last till the end.

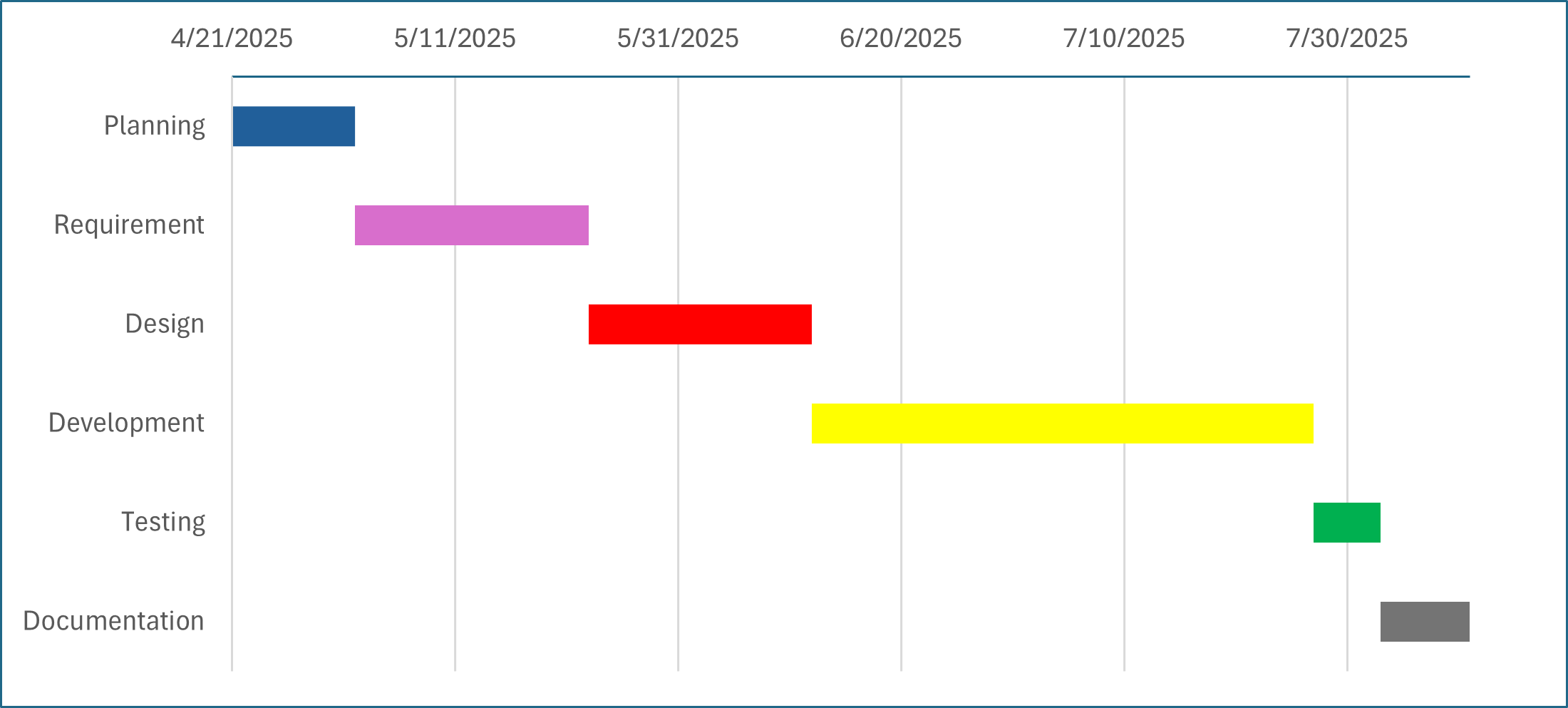


Figure 3: Gantt Chart of PustakBindu

### 3.1.3. Object Modelling

Object modelling is used to represent the static structure of the system. It includes class diagrams that define the classes, their attributes, methods, and relationships. Alongside, object diagrams show actual instances of those classes with real data. This helps in understanding how different parts of the system are connected and interact at the data level.

A black screen with white text

AI-generated content may be incorrect.

Figure 4: Class Diagram of PustakBindu

### 3.1.4. Dynamic Modeling (State & Sequence diagram)

A state diagram shows how an object changes its state based on events. It helps to track object behavior from one state to another, like Login → Dashboard → Logout. This is useful to model lifecycle and control flow in the system.

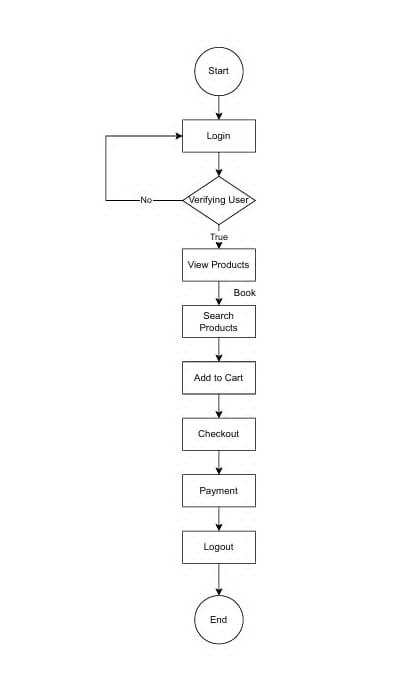


Figure 5: State Diagram for Users

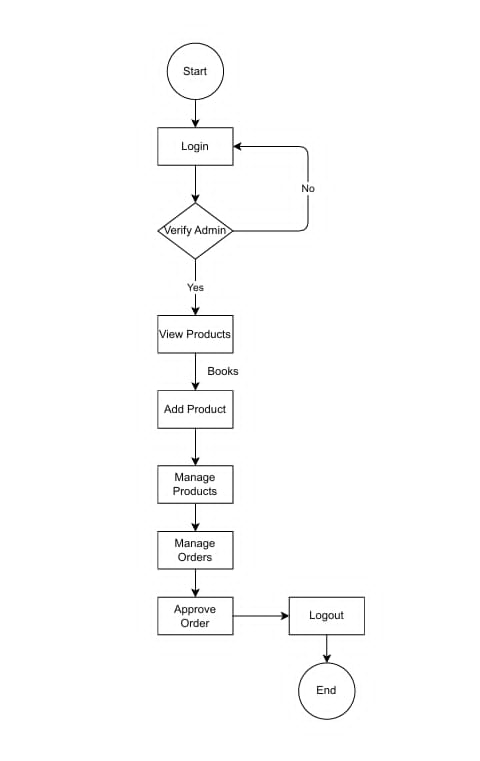


Figure 6: State Diagram for Admin

**Sequence Diagram**

A sequence diagram shows how objects interact with each other in a specific order. It displays the flow of messages between objects over time. This helps to understand the step-by-step process of a use case. In addition, sequence diagrams are useful for visualizing the logic of complex operations, clarifying the roles of different components, and ensuring that the sequence of events is correct before implementation. They are widely used in software design to improve communication between team members and to provide a clear reference during development and testing.

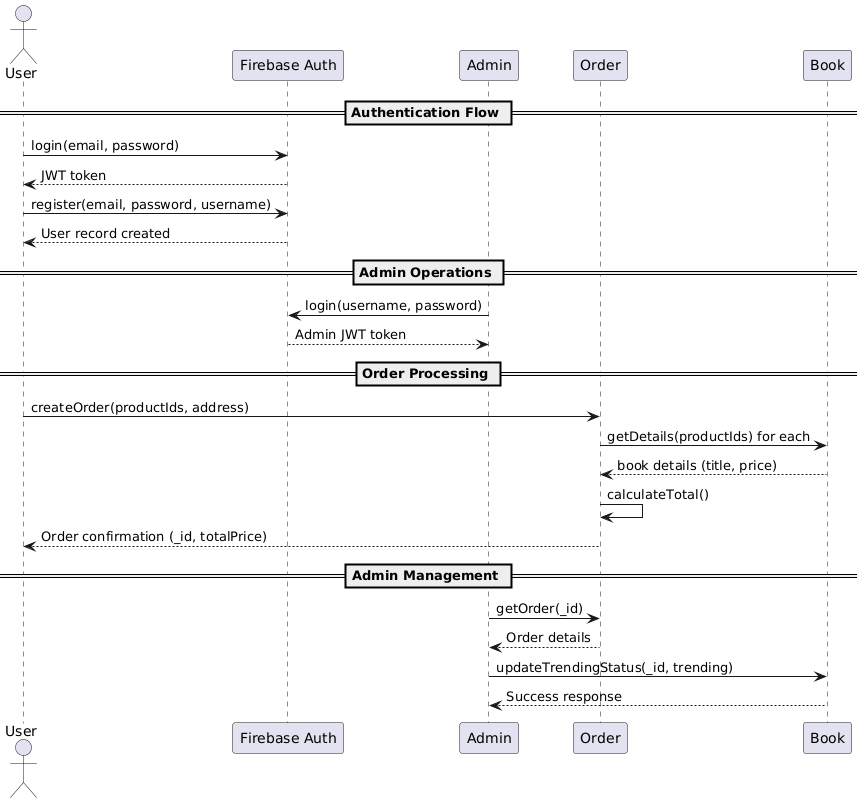


Figure 7: Sequence Diagram

### 3.1.5. Process modelling: Activity Diagram

An activity diagram shows the flow of actions or processes in the system. It’s like a flowchart that represents different steps from start to end. This helps us to understand how the system behaves during different operations.

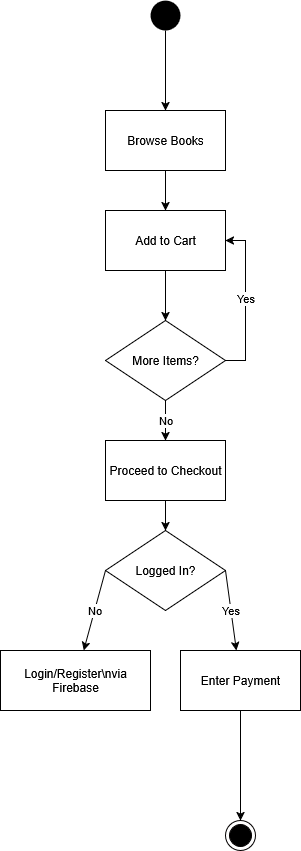


Figure 8: Activity Diagram of user in PustakBindu

## 3.2. System Design

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. It is a critical step in software engineering and is used to plan how a system should be structured and how different parts of the system will interact. It involves making strategic decisions about architecture, data flow, components, and technologies to ensure the system is efficient, scalable, and maintainable.

### 3.2.1. Refinement of Classes and Object

A black screen with white text

AI-generated content may be incorrect.

Figure 9: Refinement of Class Diagram of PustakBindu

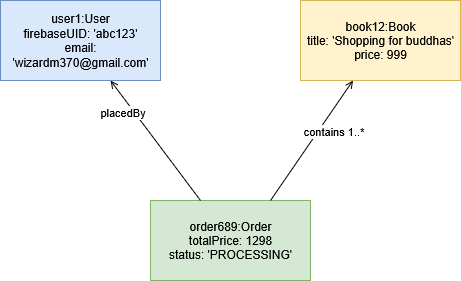


Figure 10: Refinement of Object Diagram of PustakBindu

### 3.2.2. Component Diagram

A component diagram shows the main parts (components) of the system and how they interact. It represents modules like frontend, backend, database, or authentication as separate blocks. This helps in visualizing the system’s structure at a higher level.

A diagram of a computer

AI-generated content may be incorrect.

Figure 11: Component Diagram of PustakBindu

## 3.3. Algorithm

The **Order-Based Popular Books Algorithm** ranks and retrieves books according to how many times they have been ordered. This popularity measure comes from the orderCount field, which is updated every time a purchase occurs. By sorting books in descending order of orderCount, the algorithm identifies the most demanded titles, making it easier to recommend top-selling books to users and manage inventory effectively.

### Key Characteristics:

1. Popularity is determined from cumulative orderCount values.
2. The orderCount is updated whenever an order is placed.
3. Books are retrieved in descending order of popularity.
4. Results are limited to a specified number (e.g., top 10).

**Algorithm Steps:**

1. Query the books collection.
2. Sort books by orderCount in descending order.
3. Limit the number of results to N (e.g., top 10).
4. Return the sorted and limited list.

**Pseudo Code:**

function getOrderBasedPopularBooks(limit):

books = getAllBooksFromDatabase()

sort books by orderCount in descending order

popularBooks = take first 'limit' books

return popularBooks

**Time Complexity:**

* Let **b** = total books.
* Sorting → **O(b log b)**.
* Limiting → **O(1)** (small constant).
* **Total:** **O(b log b)**.

# CHAPTER 4: IMPLEMENTATION AND TESTING

## 4.1. Implementation

During the implementation phase of the **PustakBindu** Online Bookstore project, a range of modern technologies were utilized to build a reliable and user-friendly application. **MongoDB** was used as the primary database for flexible and scalable data storage. The backend was developed using **Node.js** and **Express.js**, providing a robust server environment and efficient API handling. For the frontend, **React.js** combined with **Tailwind CSS** was employed to create an intuitive and responsive user interface. **Visual Studio Code** served as the main code editor, facilitating smooth development, debugging, and testing processes. This technology stack ensured that PustakBindu is both performant and easy to use, delivering a seamless experience for customers to browse, search, and purchase books online.

### 4.1.1. Tools Used (CASE tools, programming language, database platforms)

For the development of the project “PustakBindu”, the following technologies have been used:

1. **CASE Tools:**

Following CASE tools were used during the development of the system.

* **Diagramming Tools**

1. **draw.io:** Draw.io, also known as diagrams.net, was used to create the activity diagram, sequence diagram, class diagram, use case diagram, deployment diagram, and component diagram for the Online Book Store project.

* **Version Control Tools**

1. **GitHub:** GitHub was used as version control tool for the development of the project.

* **Integrated Development Environment (IDE)**

1. **Visual Studio Code**: Visual Studio Code was used as the integrated development environment (IDE) for the PustakBindu project.
2. **Programming languages**

Following programming languages were used during the development of the system.

* **Frontend**

1. **HTML:** It is used to create a structure and content of web pages.
2. **React.js:** It is a JavaScript library used for building user interface in the project.
3. **Tailwind CSS:** It is a utility-first CSS framework which is used to quickly style HTML elements.

* **Backend**

1. **Node.js:** A JavaScript runtime environment used to build the backend server.
2. **Express.js:** A web framework for Node.js used to handle routing and API requests in the project.
3. **Database Platforms**

Following database platform was used during the development of the system.

1. **MongoDB:** It was used to store system data such as user profiles, novel details, reviews, and bookmarks, providing flexibility and scalability for the application.

### 4.1.2. Implementation details of modules (description of procedures/ functions/ classes/ methods)

* **Authentication Module**

The Authentication Module of PustakBindu ensures secure access for users. For customers, Firebase Authentication is used to manage registration and login, eliminating the need to store sensitive credentials on the server. Firebase handles token-based authentication, maintaining secure sessions and validating user identity. For administrators, a custom MongoDB schema stores credentials securely, with passwords hashed using bcrypt before saving. The role field ensures that only users with administrative privileges can perform certain actions, providing role-based access control.

* **Book Management Module**

The Book Management Module stores and manages all book-related information in the MongoDB Book schema. It includes fields for title, description, author, category, trending status, cover image, and price. Administrators can perform CRUD operations through dedicated API endpoints. A search functionality using MongoDB’s $regex operator allows users to find books by title or author, while a trending filter highlights popular books on the homepage. This module ensures data consistency and easy management of the bookstore’s inventory.

* **Order Management Module**

The Order Management Module handles all purchase-related activities. Customer orders are stored in the Order schema, which contains details such as name, email, phone number, address, ordered product IDs, total price, and order status. Orders are created with a default status of “pending” and can be updated to “approved” by administrators. Each order references books via MongoDB ObjectIDs, maintaining a clear relationship between products and orders. This module ensures accurate tracking and management of customer purchases.

* **Search Module**

The Search Module helps users quickly find books of interest. It performs a case-insensitive search on both the title and author fields using MongoDB’s $regex feature. To optimize performance, search results are limited to 50 entries per query. This module improves user experience by providing efficient and relevant search results without compromising system speed or performance.

* **Frontend Module**

The Frontend Module is developed using React.js and styled with Tailwind CSS to provide a responsive and user-friendly interface. Customers can browse books, view detailed descriptions, add items to their cart, and place orders. The administrator dashboard enables efficient book and order management through an intuitive interface. Components are modular and reusable, allowing clean, maintainable code and a consistent visual design across the platform.

* **API Module**

The API Module, implemented using Node.js and Express.js, acts as the communication layer between the frontend and backend. It handles CRUD operations for books and orders, processes authentication for administrators, and manages customer orders. Middleware functions are used for authentication and validation, while centralized error handling ensures consistent and meaningful API responses, providing a reliable backend infrastructure for the online bookstore.

## 4.2. Testing

### 4.2.1. Unit testing

Following tests for separate modules were done for this system:

Table 4.1: Test Cases for Unit Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.N** | **Unit** | **Test** | **Expected**  **Result** | **Test Outcome** | **Evidence** |
| 1. | Log In | Used invalid Log In Credentials to check login functionality | Invalid username or password | Invalid username or password | Test 1.1 |
| 2. | Log In | Used valid Log In Credentials to check login functionality | Redirected to user profile | Redirected to user profile | Test 2.1 |
| 3. | Registration | Registration to website | Get registered to website | Get registered to website | Test 3.1 |
| 4. | Browse Books | Open homepage to see all books | All books listed with title, author, price and image | Books displayed correctly | Test 4.1 |
| 5. | Search Books | Search for a book by title/author | Display only books matching the query | Search works correctly | Test 5.1 |
| 6. | Place Order | Add books to cart and submit order | Order saved with pending status and correct price | Order placed successfully | Test 6.1 |
| 7. | Order Approval  (Admin) | Approve pending orders | Status changes from pending to approved | Status updated successfully | Test 7.1 |
| 8. | Admin Book Management | CRUD operations on books | Books are added, updated, or deleted in database | CRUD works correctly | Test 8.1  Test 8.2  Test 8.3 |

**TEST EVIDENCE 1.1**

Unit Test: Log In

Test: Check Login functionality using invalid credentials

Expected Outcome: Invalid username or password

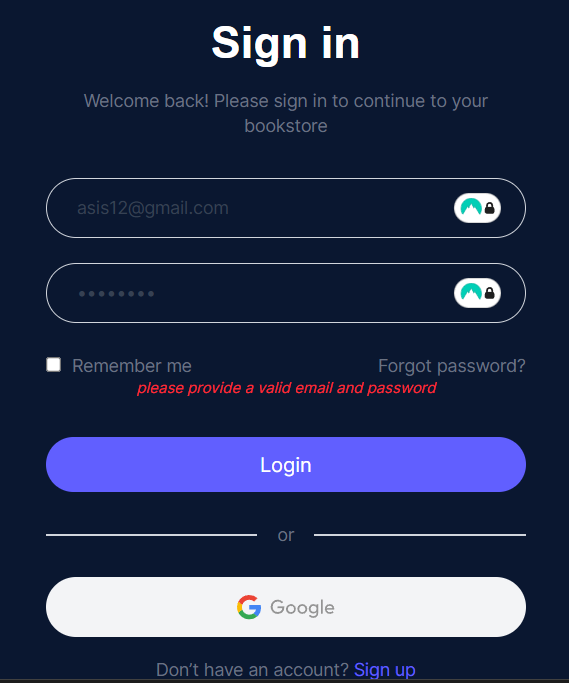
****

Figure 12: Test Case Evidence 1.1

**TEST EVIDECE 2.1**

Unit Test: Log In

Test: Check Login functionality using valid credentials

Expected Outcome: Redirected to user profile

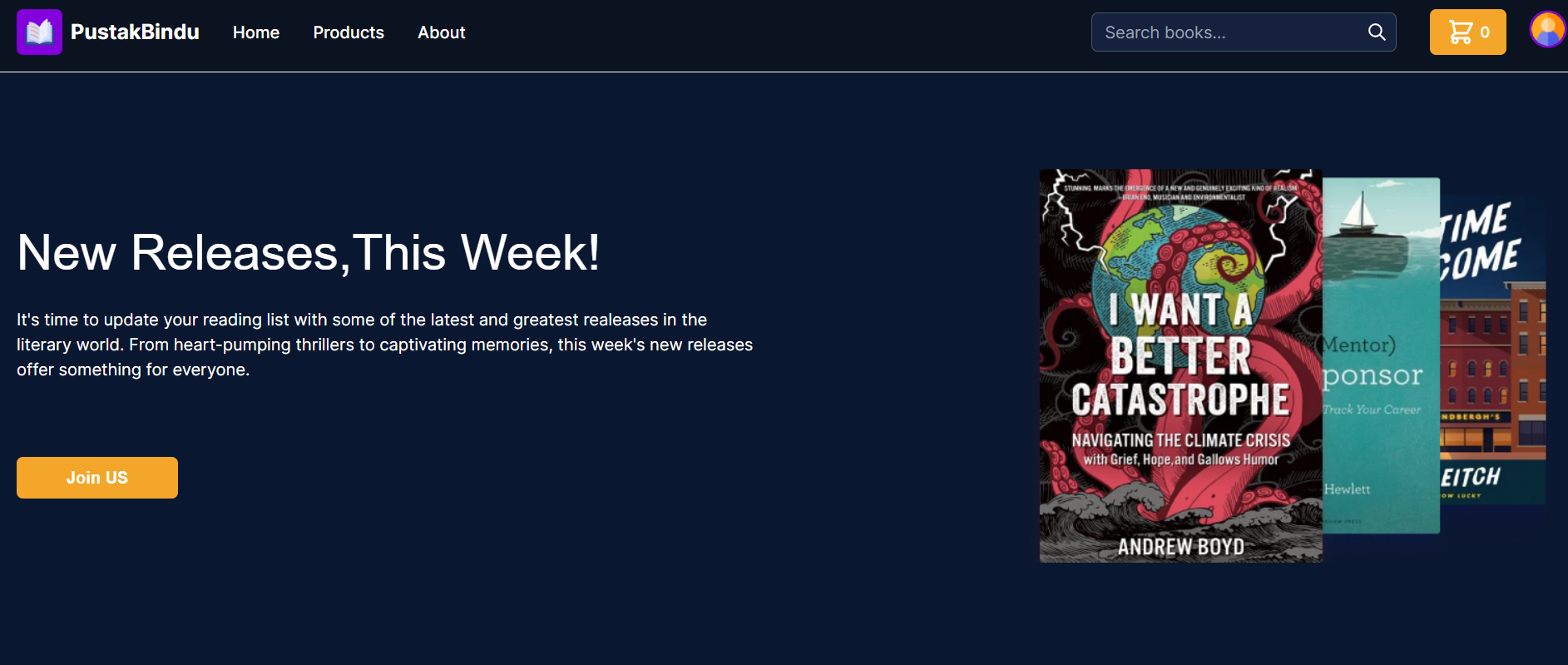
****

Figure 13: Test Case Evidence 2.1

**TEST EVIDENCE 3.1**

Unit Test: Registration

Test: Registration to website

Expected Outcome: Get registered to website



Figure 14: Test Case Evidence 3.1

**TEST EVIDENCE 4.1**

Unit Test: Browse books

Test: Open homepage to see books

Expected Outcome: All books listed with title, author, price and image

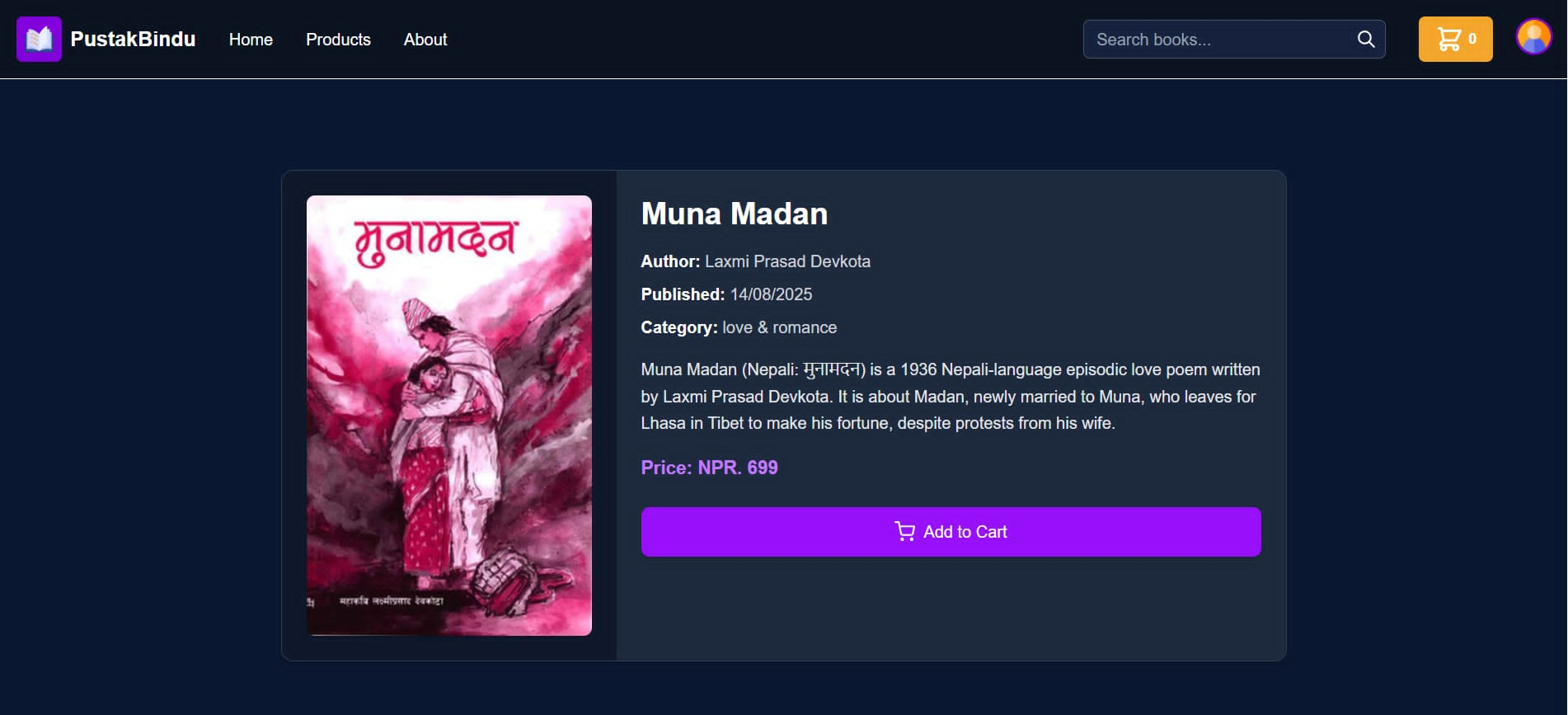


Figure 15: Test Case Evidence 4.1

**TEST EVIDENCE 5.1**

Unit Test: Search for a book by title/author

Expected Outcome: Display only books matching the query

A screen shot of a computer

AI-generated content may be incorrect.

Figure 16: Test Case Evidence 5.1

**TEST EVIDENCE 6.1**

Unit Test: Add books to cart and submit order

Expected Outcome: Order saved with pending status and correct price

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 17: Test Case Evidence 6.1

**Test Evidence 7.1**

Unit Test: Approve pending orders

Expected Outcome: Status changes from pending to approved

A screenshot of a computer

AI-generated content may be incorrect.

Figure 18: Test Case Evidence 7.1

**Test Evidence 8.1**

Unit Test: Add books to database

Expected Outcome: Books are added

A screenshot of a computer

AI-generated content may be incorrect.

Figure 18: Test Case Evidence 8.1

**Test Evidence 8.2**

Unit Test: Update books to database

Expected Outcome: Books are Updated

A screenshot of a update book

AI-generated content may be incorrect.

Figure 19: Test Case Evidence 8.2

**Test Evidence 8.3**

Unit Test: Delete books to database

Expected Outcome: Books are Deleted

A screenshot of a computer

AI-generated content may be incorrect.

Figure 20: Test Case Evidence 8.3

### 4.2.2. Test Cases for System Testing

After integrating all modules into a functional system, comprehensive system testing was conducted to verify that the application meets all functional requirements and performs as expected. The following tests were performed during the System Testing phase:

* **User Registration and Login:** Verified that new users can register successfully using valid details, and existing users can log in and log out securely. Invalid login attempts display proper error messages.
* **Book Browsing and Search:** Tested that users can browse books by category and view book details. Verified that searching by book title or author returns accurate and relevant results.
* **Cart and Order Management:** Ensured that users can add books to the cart, update quantities, and place orders. Verified that total prices are calculated correctly and order confirmation is displayed.
* **Order History:** Checked that users can view their past orders, including order details, status, and total amount.
* **Recommendation System:** Tested that the order-based recommendation algorithm suggests relevant books based on users’ previous purchases.
* **Admin Book Management:** Verified that administrators can add, update, and delete books, ensuring changes reflect immediately in the catalog.
* **Responsive User Interface:** Ensured that the website layout adjusts properly across devices, providing a seamless experience on desktops, tablets, and mobile phones.
* **Error Handling:** Simulated various errors, such as server failures and invalid form submissions, to confirm that the system provides appropriate error messages and maintains stability.

This testing phase confirmed that all system modules are functioning correctly and that users can interact with the bookstore efficiently and securely.

# CHAPTER 5: CONCLUSION AND FUTURE RECOMMENDATIONS

## 5.1. Conclusion

In conclusion, the implementation of the PustakBindu Online Bookstore has significantly enhanced the process of discovering, purchasing, and managing books compared to traditional methods. The adoption of the waterfall methodology for the project development proved effective, as the requirements were clearly defined, allowing a structured and organized approach in building the system. Users benefit from an intuitive interface that simplifies browsing books, searching by title or author, placing orders, and managing personal accounts. The order-based recommendation system provides relevant book suggestions, improving user engagement and satisfaction. Administrators can efficiently manage the book catalog, ensuring content accuracy and quality. This organized approach has not only improved the overall user experience but has also streamlined the process of book management and sales. Future improvements in personalization, recommendation algorithms, and system performance will ensure that PustakBindu continues to be a preferred platform for book lovers.

## 5.2. Lesson Learnt / Outcome

Throughout the development of the PustakBindu Online Bookstore, we gained several valuable lessons and insights that contributed to both our technical and professional growth:

* **Technology Exploration:** The project provided us with hands-on experience with modern technologies such as React.js and Tailwind CSS for creating a responsive and user-friendly interface, along with Node.js, Express.js, and MongoDB for efficient backend development and data management.
* **Practical Application of Skills:** We applied our theoretical knowledge of web development, RESTful APIs, and database operations to a real-world scenario, which strengthened our coding, problem-solving, and debugging abilities.
* **System Design and Modeling:** Designing UML diagrams, including class and sequence diagrams, helped us visualize the workflow of the bookstore system more clearly, leading to an organized and systematic development process.
* **Database Design:** Creating and implementing the MongoDB schema for storing books, orders, and user information improved our understanding of data relationships and efficient querying techniques.
* **Usage of CASE Tools:** Using CASE tools for modeling and documentation enhanced collaboration and ensured consistency throughout the project design and development.

These experiences not only improved our technical expertise but also highlighted the importance of structured project development, teamwork, and efficient workflow management. The completed system successfully met all project requirements, enabling users to search, browse, and purchase books seamlessly while providing administrators with tools to manage the platform efficiently.

## 5.2. Future Recommendation

Looking ahead, PustakBindu Online Bookstore can be enhanced to improve both usability and overall system performance. From the user perspective, upgrading the interface with a cleaner design, intuitive navigation, and personalized recommendations will enhance the browsing and purchasing experience. Implementing OAuth login options, such as Google or Facebook authentication, will improve security and convenience for users.

For administrators, adding features like advanced order management, automated email notifications, and inventory tracking will make platform management more efficient. Introducing a recommendation system based on purchase history and user preferences can increase engagement and boost sales. Additionally, optimizing backend queries and API responses will ensure faster loading times as the user base grows.

Community engagement can also be strengthened by integrating features such as user reviews, ratings, wishlists, and discussion forums for book discussions. By implementing these recommendations, PustakBindu can evolve into a more interactive, scalable, and user-friendly platform that caters effectively to modern readers and ensures a seamless online bookstore experience.

# APPENDICES

**Screenshots**

1. **Home Page**

**A blue rectangle with black border

AI-generated content may be incorrect.**

1. **Products**

A screenshot of a computer

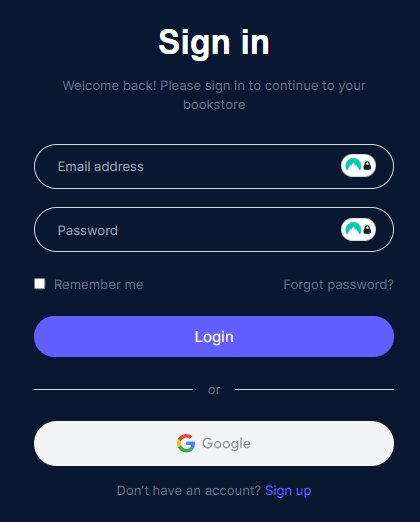
AI-generated content may be incorrect.

1. **Search bar**

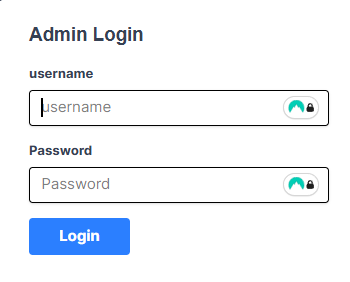
A blue rectangle with black border

AI-generated content may be incorrect.

1. **Login Form**



1. **Admin Login**

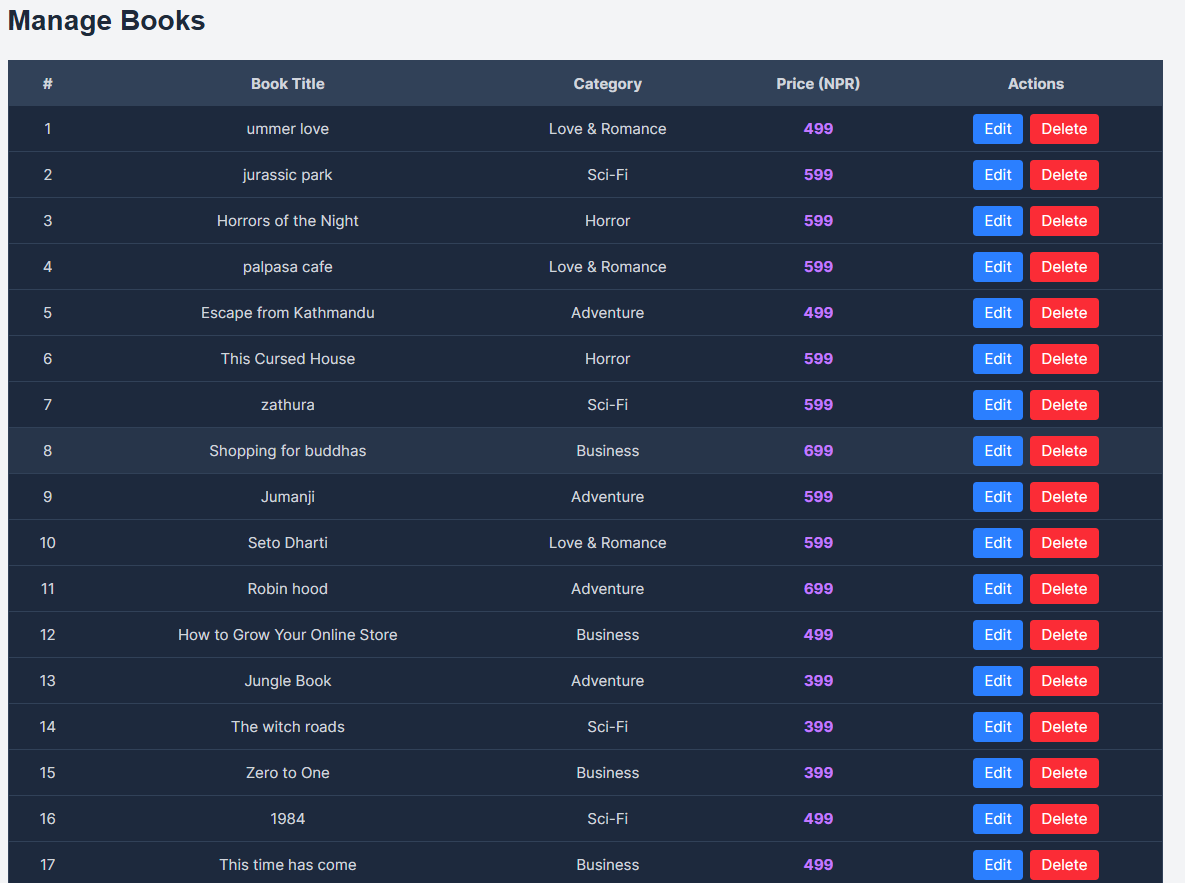


1. **Admin Dashboard**

A screenshot of a computer

AI-generated content may be incorrect.

1. **Manage Books**



1. **Add a new Book**

A screenshot of a computer

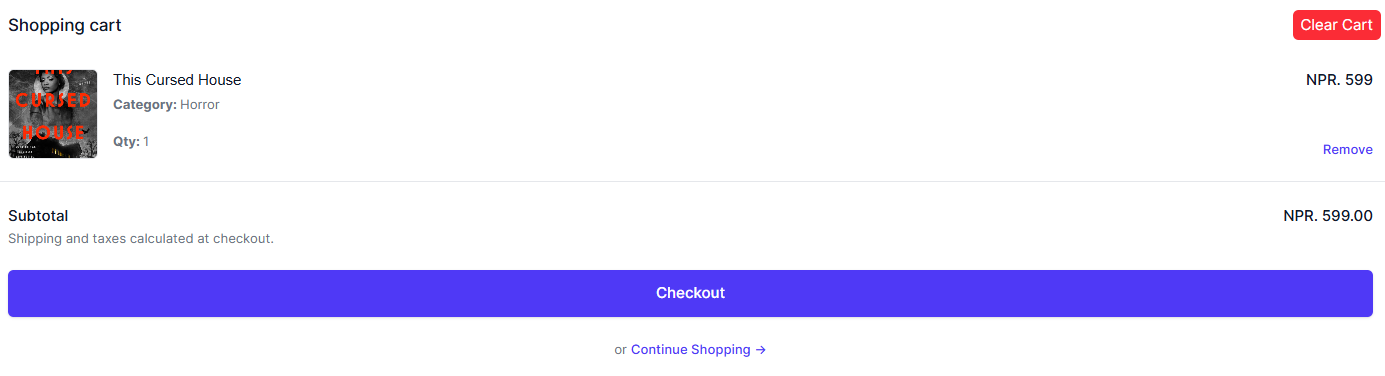
AI-generated content may be incorrect.

1. **Manage Orders**

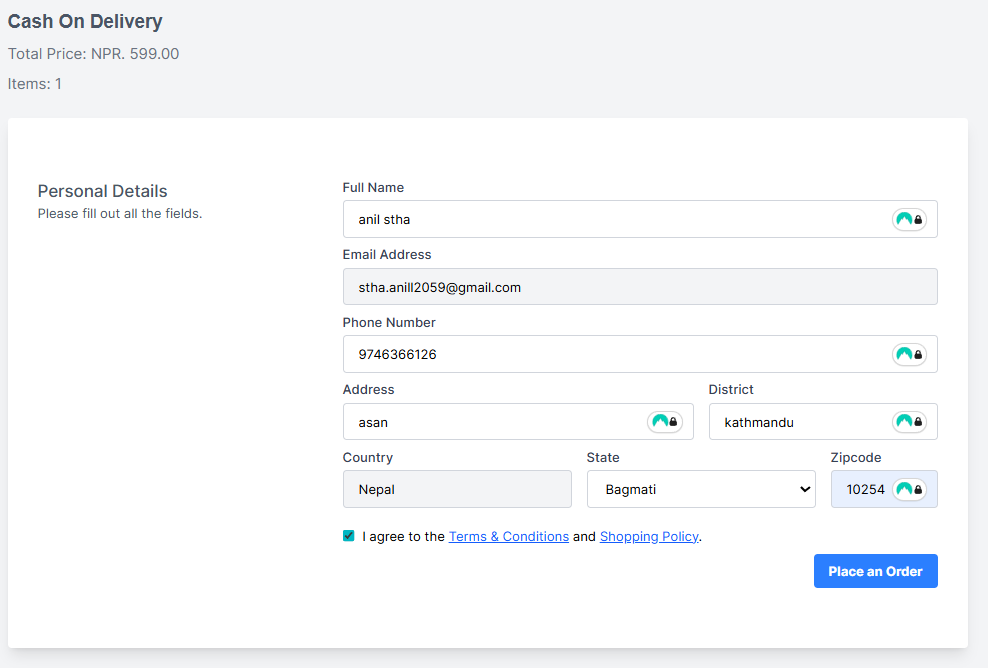
A screenshot of a computer

AI-generated content may be incorrect.

1. **Cart Page**

****

1. **User Order Details Page**

****

1. **Checkout Page**

**A screenshot of a computer

AI-generated content may be incorrect.**