Library Management System

Capstone Project-1

By Group-6

Names:

Sivabalan.S

Sai Teja.T

Sadhika.K

Sai Sahiti.M

Capstone Evaluator:

Prof. Abhishek Sengupta

Revision Sheet:

S.NO	**DATE**	**AUTHOR**	**DESCRIPTION**
1	8/7/2024	S.SIVABALAN	Back-end,Database
2	9/7/2024	K.SADHIKA	Front-End
3	9/7/2024	M.SAI SAHITHI	Front-End,PPT
4	9/7/2024	T.SAI TEJA	Documenation,Testing
5	10/7/2024	TEAM	Final Draft

Table of Contenets

1.Introduction

1.1 Purpose:

The primary purpose of the Library Management System (LMS) project is to develop an efficient, user-friendly, and comprehensive web-based application to manage and streamline the operations of a library. This system is intended to achieve the following key objectives:

- Improve Library Operations
- Enhance User Experience
- Ensure Data Integrity and Security
- Support Efficient Book Management
- Optimize Lending Processes:
- Facilitate Library Administration:
- Support Agile Development and Deployment

1.2 Scope of the project:

The scope of the Library Management System (LMS) project encompasses the development and deployment of a comprehensive web-based application that covers all aspects of library operations. The project includes the following key components and features:

- User Management
- Book Management
- Cart Management
- Security and Access Control
- Reporting and Analytics

1.3 Project Overview:

The Library Management System (LMS) is a comprehensive web-based application designed to streamline and enhance the management of library operations. The system offers a range of features, including user management, book cataloging, lending and returning of books, search functionality, and detailed reporting. It leverages modern technologies such as Python for backend development, HTML/CSS for frontend design, and SQL for data management. The LMS is built following Agile methodology and is deployed using containerization (Docker) and Kubernetes for scalability and reliability. The goal is to provide a user-friendly, secure, and efficient platform for both library staff and patrons.

2.System Requirements

2.1 Functional Requirements

2.1.1 Login with Username and Password

The system ensures that only authorized users can access the Library Management System by validating user credentials against stored records in the database.To ensure robust security, the system enforces strict password validation criteria. Passwords must be between 4 and 8 characters in length and must meet specific Regex authentication standards. This combination of measures ensures that only authorized users can access the system, while also maintaining high security for user credentials.

2.1.2 Book Management page

The Library Management System features comprehensive functionalities to enhance user experience and streamline library operations. One of the key features is the categorization of books, allowing users to browse and display books under various categories such as Fiction, Horror, and more. This organized approach helps users easily find books that match their interests.Additionally, the system enables users to select books using checkboxes, ensuring a straightforward and intuitive selection process. To maintain data integrity and user convenience, the system is designed to prevent duplicate books from being added to the cart, ensuring that users can manage their selections efficiently without encountering repetitive entries.

2.1.3 Cart Management

The Cart Page of the Library Management System is designed to display selected books along with their lending period details. For each book, the start and end dates of the lending period are shown, ensuring clarity for the users. The standard duration for borrowing books is set at 30 calendar days, excluding weekends. This system ensures that users have ample time to enjoy their borrowed books while maintaining a consistent lending policy.

2.2 Non-Functional Requirements

2.2.1 Security:

The Library Management System will prioritize security by implementing secure user authentication, ensuring that only authorized users can access the system. To protect user credentials, passwords will be encrypted, preventing unauthorized access to sensitive information. Furthermore, secure communication protocols such as HTTPS will be utilized to safeguard data transmitted between the user's browser and the server, ensuring the integrity and confidentiality of the information exchanged within the system.

2.2.2 Performance:

To ensure optimal performance and user experience, the Library Management System will focus on efficient handling of book selections and cart operations. This includes implementing robust mechanisms to manage user interactions with the book selection process and ensuring that no duplicate books are added to the cart. Additionally, the system will be designed to optimize database queries, reducing response times and enhancing the overall efficiency of the application. This optimization is crucial for maintaining a seamless and responsive user interface, even as the system scales to handle larger volumes of data and users.

2.2.3 Usability:

The Library Management System will include a user-friendly interface for book selection and cart management, simplifying the process for users to browse and choose books. Clear error messages and validation feedback will be integrated to help users understand any issues or required actions, enhancing the overall usability and ensuring a seamless experience.

2.2.4 Scalability

The system will be designed to efficiently handle an increasing number of users and books, ensuring scalability and performance even as the library's collection and user base grow.

Additionally, the architecture will support future enhancements and additional features, allowing for continuous improvement and expansion without significant overhauls.

2.2.5 Maintainability

Writing clean, modular code will be a priority to ensure easy maintenance and readability. Proper documentation will also be maintained throughout the development process, providing clear guidance and reference for future developers and users.

3.Design

3.1 Architecture

1. Client Browser

The client browser is the front-end of the application, implemented using HTML, CSS, and JavaScript. It includes the following pages:

Login Page: Allows users to enter their username and password. The password is validated using regex to ensure it is between 4 and 8 characters.

Book Management Page: Displays different categories of books (e.g., Fiction, Horror) with checkboxes for selection. Selected books are added to the cart.

Cart Page: Displays the books added to the cart, along with the start and end dates of the lending period (excluding weekends).

2. Web Server (Flask Application)

The Flask application serves as the backend of the LMS. It handles HTTP requests, interacts with the database, and renders HTML templates. Key routes include:

/login: Handles user authentication.

/book_management: Displays the book management page.

/add_to_cart: Adds selected books to the cart and calculates the lending period.

/cart: Displays the cart page.

3. Application Logic

The application logic is implemented in the Flask application. It includes**:**

User Authentication and Authorization: Ensures users are logged in before accessing the book management and cart pages.

Book Management: Lists books by category and allows users to select books.

Cart Management: Manages the cart by adding selected books, ensuring no duplicate books are added, and calculating the lending period excluding weekends.

4. Database

The database stores all the necessary data for the LMS. The schema includes:

Users Table: Stores user information (e.g., id, username, password).

Books Table: Stores book information (e.g., id, title, category).

Cart Table: Stores cart entries (e.g., id, user_id, book_id, start_date, end_date).

5. Authentication & Authorization (Flask-Login)

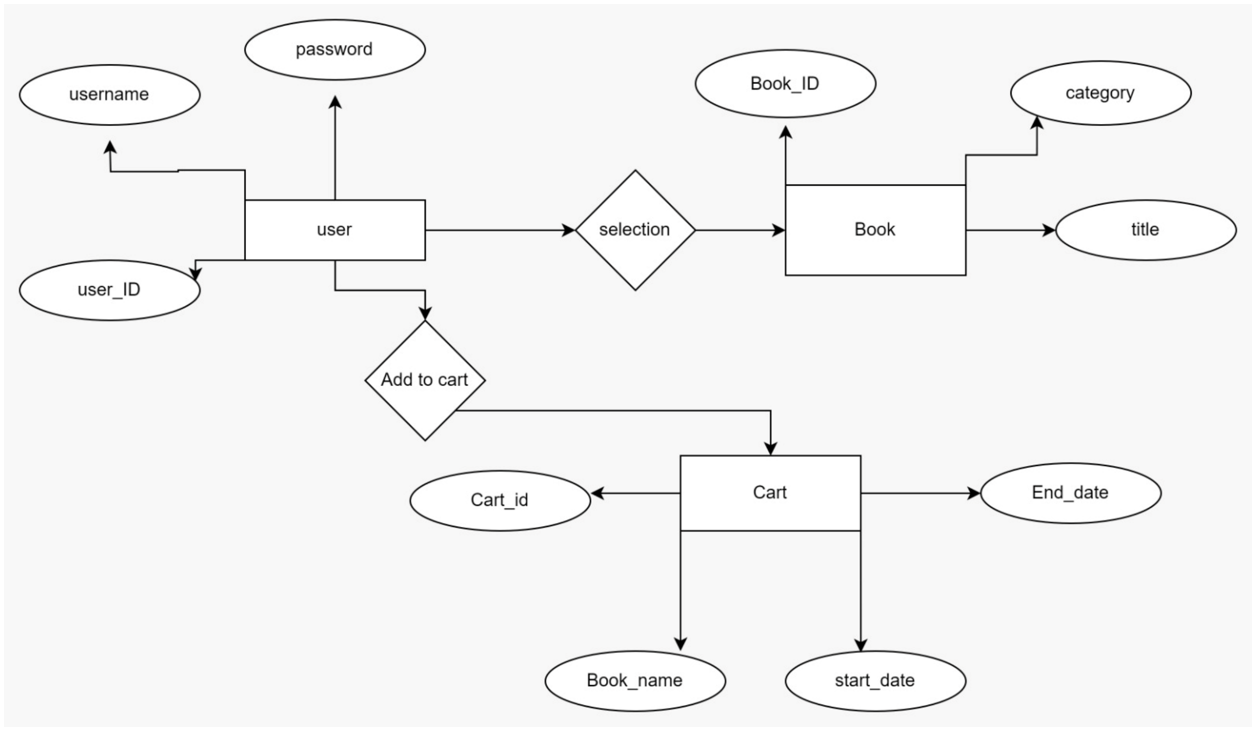
Flask-Login is used to manage user sessions. It includes:

User Login: Validates user credentials and creates a session for the logged-in user.

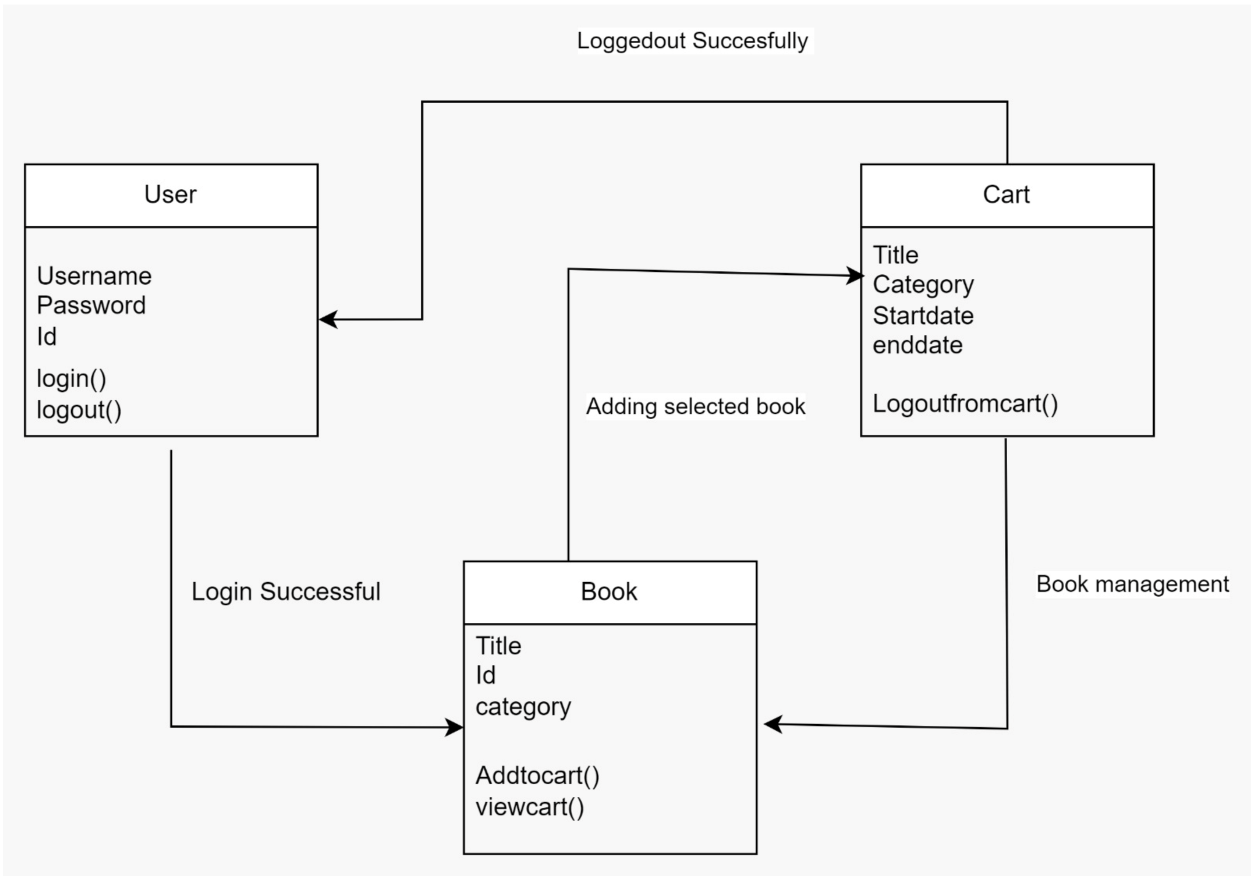
User Logout: Ends the user session.

Access Control: Ensures only authenticated users can access certain routes.

3.2 ER Diagram



Class Diagram:



4.Implementation

4.1 Login Page

4.1.1 Username and Password:

Username Validation

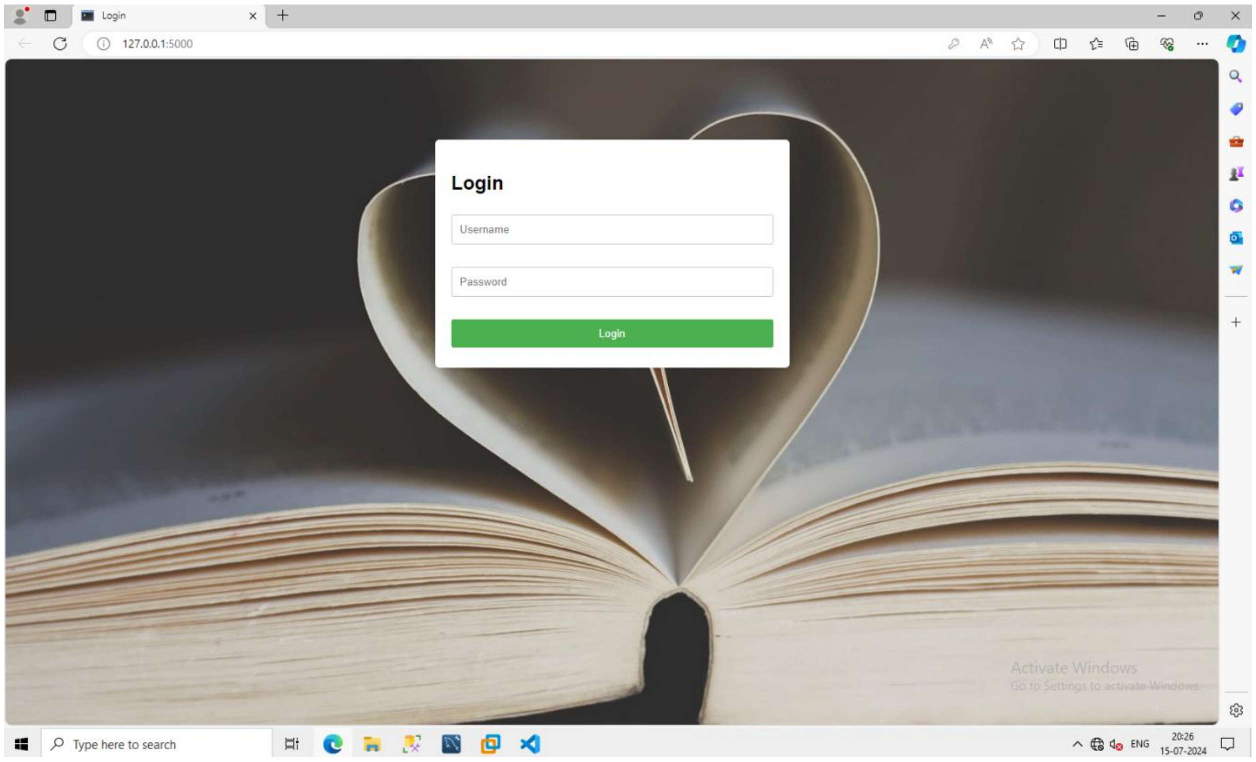
Usernames are validated to ensure they meet the system's requirements for uniqueness and format. Typically, usernames:

- Must be unique within the system to avoid duplication.
- Should adhere to specific character limitations, often alphanumeric characters with possible special characters like underscores or hyphens.
- Are validated against existing records in the database to confirm availability during registration or login processes.

Password Validation

Passwords undergo strict validation to enhance system security and user account protection. Key validation criteria include:

- **Length Requirement**: Passwords must typically be between 4 to 8 characters long to balance security and usability.
- **Character Composition**: They should include a mix of alphanumeric characters (letters and numbers) to strengthen complexity.
- **Regex Pattern**: A regex (regular expression) pattern is employed to enforce these criteria programmatically, ensuring that each password meets specified security standards.
- **Error Messaging**: Clear error messages inform users of any password requirements not met, guiding them towards creating stronger, compliant passwords.



4.2 Book Management Page

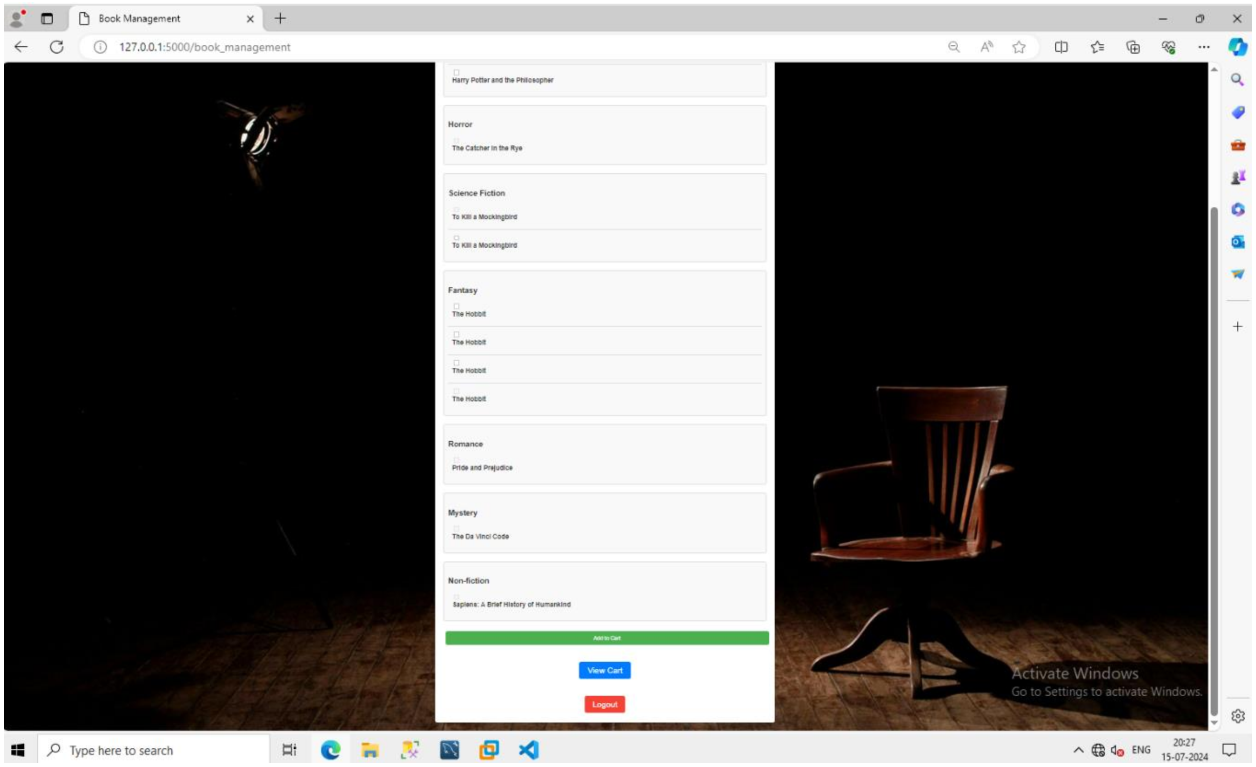
4.2.1 Categories Display

The Book Management Page categorizes books to facilitate organized browsing for users. Each category, such as Fiction, Horror, Science Fiction, etc., is displayed with relevant books listed under them. This categorization aids users in quickly locating books of interest without needing to browse through an extensive list. Categories are dynamically populated from the database, ensuring that any new categories added or removed reflect immediately on the interface.

4.2.2 Selection Handling

Users can interact with books using intuitive selection features on the Book Management Page. This includes functionalities such as:

- **Checkbox Selection**: Each book is accompanied by a checkbox that users can select to indicate their interest or intention to borrow.

- **Duplicate Prevention**: Robust backend logic ensures that users cannot inadvertently add the same book multiple times to their cart, maintaining clarity and preventing accidental duplicates.
- **Feedback Mechanism**: Clear visual cues and feedback messages inform users about their selections, ensuring transparency and reducing errors during the selection process.
- 

Harry Potter and the Philosopher

Horror
The Catcher in the Rye

Science Fiction
To Kill a Mockingbird
To Kill a Mockingbird

Fantasy
The Hobbit
The Hobbit
The Hobbit
The Hobbit

Romance
Pride and Prejudice

Mystery
The Da Vinci Code

Non-fiction
Sapiens: A Brief History of Humankind

4.3 Cart Page

4.3.1 Display selected books.

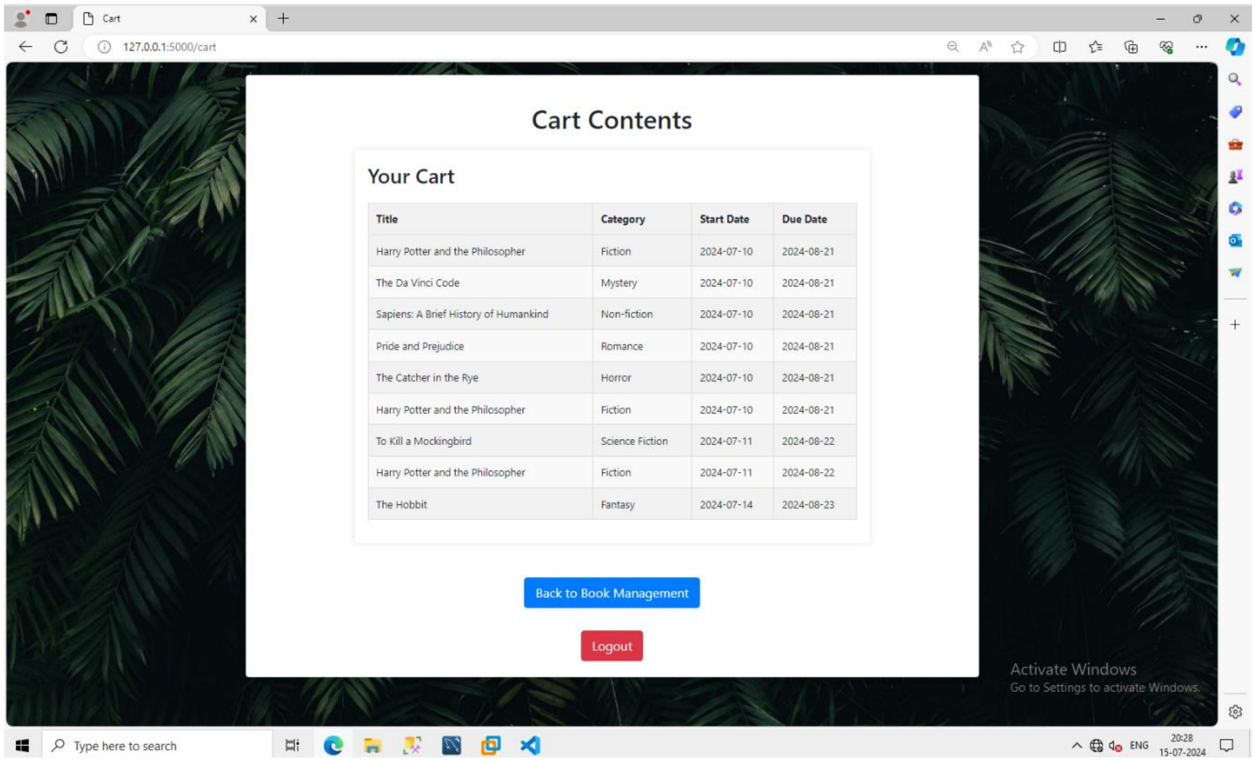
The Cart Page presents a comprehensive view of all books that the user has selected for lending. Each book is listed with relevant details such as title, author, and category. This display ensures users can easily review their selections before proceeding to finalize their borrowing.

4.3.2 Allow setting start and end dates for lending.

Users can specify the start and end dates for lending each selected book directly from the Cart Page. The system calculates the lending period, typically 30 calendar days excluding weekends, ensuring clarity and adherence to borrowing policies. Date pickers or input fields facilitate seamless date selection, enhancing user control over the borrowing period.

4.3.3 Prevent adding the same book multiple times.

To maintain accuracy and prevent confusion, the system employs checks to prevent users from adding the same book multiple times to their cart. This validation ensures that each book appears only once in the cart, promoting a clear and manageable borrowing experience.



Cart Contents

Your Cart

Title	Category	Start Date	Due Date
Harry Potter and the Philosopher	Fiction	2024-07-10	2024-08-21
The Da Vinci Code	Mystery	2024-07-10	2024-08-21
Sapiens: A Brief History of Humankind	Non-fiction	2024-07-10	2024-08-21
Pride and Prejudice	Romance	2024-07-10	2024-08-21
The Catcher in the Rye	Horror	2024-07-10	2024-08-21
Harry Potter and the Philosopher	Fiction	2024-07-10	2024-08-21
To Kill a Mockingbird	Science Fiction	2024-07-11	2024-08-22
Harry Potter and the Philosopher	Fiction	2024-07-11	2024-08-22
The Hobbit	Fantasy	2024-07-14	2024-08-23

5.Testing

5.1 Unit Testing

Unit testing ensures that individual components or units of code function correctly in isolation. For the Library Management System, unit tests will validate functions such as user authentication, book selection algorithms, and database interactions. Test cases will cover various scenarios to verify the expected behavior of each unit.

5.2 Integration Testing

Integration testing checks how different modules or components work together as a whole. In the context of the system, integration tests will focus on testing interactions between the frontend and backend systems, ensuring smooth communication and data flow. Tests will cover scenarios such as login processes, book category displays, and cart management functionalities.

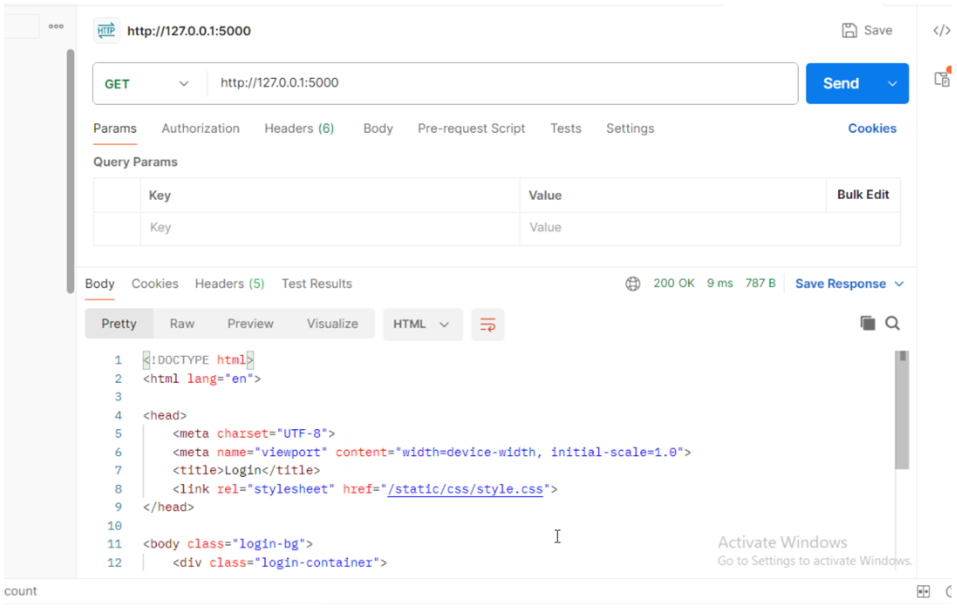
5.3 System Testing

System testing evaluates the complete system against specified requirements. It ensures that the Library Management System meets functional and non-functional requirements as a cohesive unit. Tests will validate end-to-end workflows, including user journeys from login to book borrowing and cart management. Performance, security, and usability aspects will also be assessed during system testing.

5.4 Acceptance Testing

Acceptance testing validates whether the system meets business requirements and user expectations. It involves testing the system with real users or stakeholders to ensure it fulfills its intended purpose effectively. For the Library Management System, acceptance tests will confirm that features like book categorization, cart operations, and lending period management align with user needs and organizational goals.

Testing By Using Postman



GET http://127.0.0.1:5000

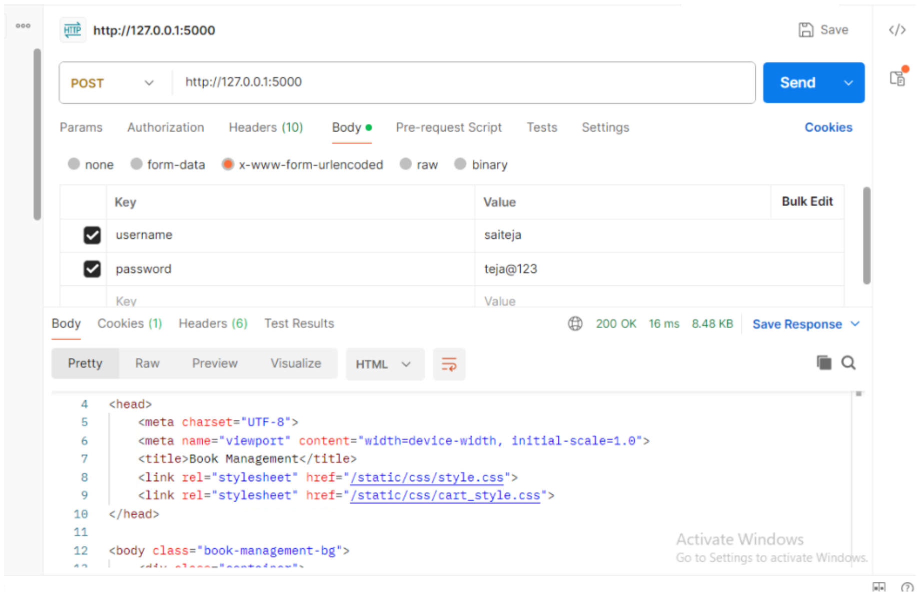
Key	Value
Key	Value

200 OK 9 ms 787 B

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <link rel="stylesheet" href="/static/css/style.css">
</head>

<body class="login-bg">
    <div class="login-container">
```



POST http://127.0.0.1:5000

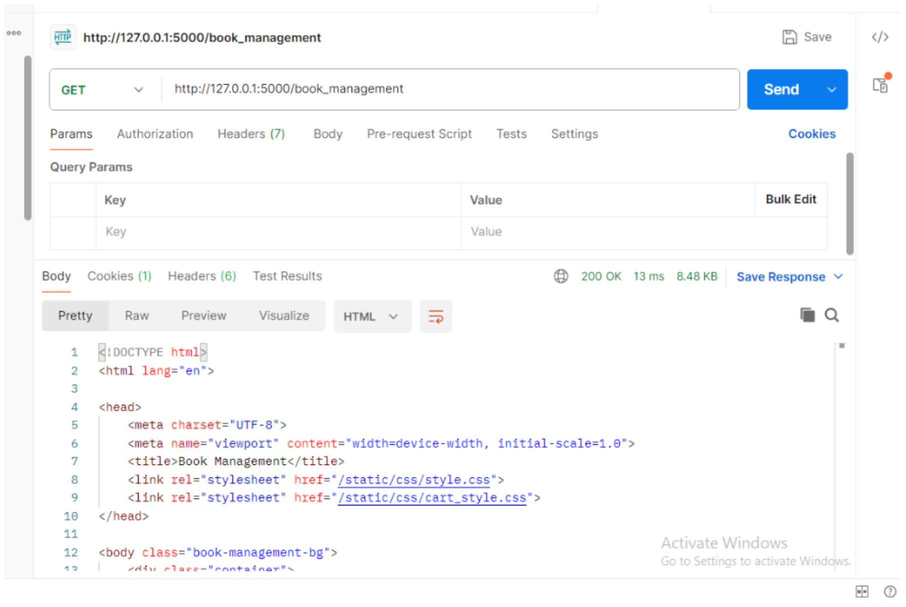
none form-data [✓ x-www-form-urlencoded] raw binary

	Key	Value
✓	username	saiteja
✓	password	teja@123
	Key	Value

200 OK 16 ms 8.48 KB

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Book Management</title>
    <link rel="stylesheet" href="/static/css/style.css">
    <link rel="stylesheet" href="/static/css/cart_style.css">
</head>

<body class="book-management-bg">
```


http://127.0.0.1:5000/book_management

GET http://127.0.0.1:5000/book_management

Query Params

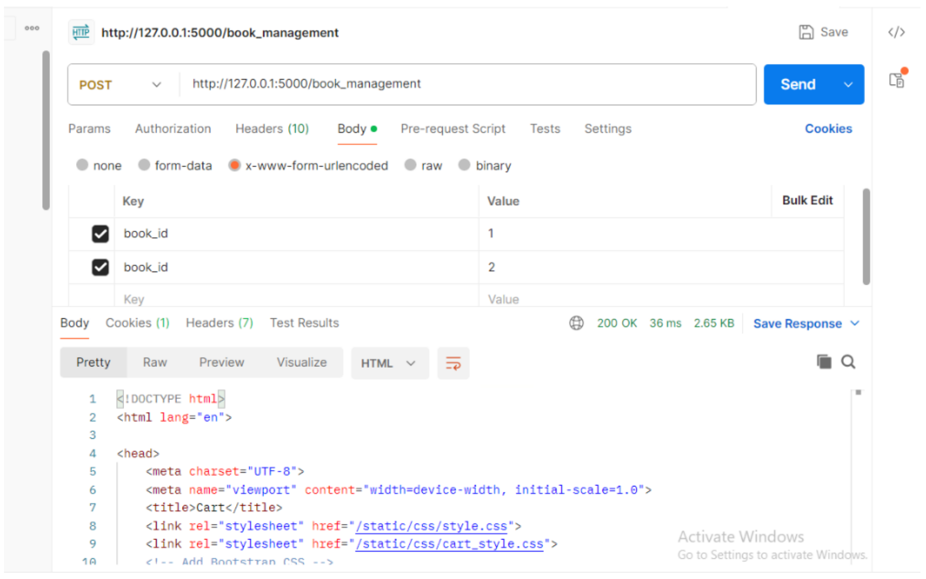
Key	Value
Key	Value

200 OK 13 ms 8.48 KB

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Book Management</title>
    <link rel="stylesheet" href="/static/css/style.css">
    <link rel="stylesheet" href="/static/css/cart_style.css">
</head>

<body class="book-management-bg">
    <div class="container">
```


http://127.0.0.1:5000/book_management

POST http://127.0.0.1:5000/book_management

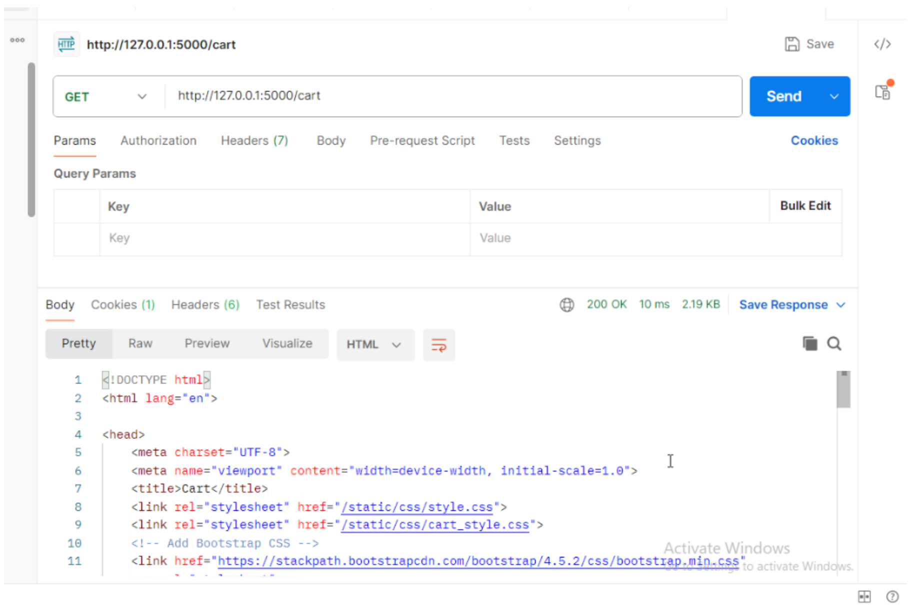
Body: x-www-form-urlencoded

	Key	Value
✓	book_id	1
✓	book_id	2
	Key	Value

200 OK 36 ms 2.65 KB

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cart</title>
    <link rel="stylesheet" href="/static/css/style.css">
    <link rel="stylesheet" href="/static/css/cart_style.css">
    <!-- Add Bootstrap CSS -->
```



http://127.0.0.1:5000/cart

GET http://127.0.0.1:5000/cart

200 OK 10 ms 2.19 KB

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cart</title>
    <link rel="stylesheet" href="/static/css/style.css">
    <link rel="stylesheet" href="/static/css/cart_style.css">
    <!-- Add Bootstrap CSS -->
    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
```

6.Deployment

6.1 Environment Setup

Setting up the deployment environment involves preparing the infrastructure and software components necessary to run the Library Management System. This includes:

- **Infrastructure Configuration:** Provisioning servers or cloud resources to host the backend services, frontend application, and database.
- **Software Installation:** Installing required dependencies such as Python, Flask, SQL database server (e.g., PostgreSQL), and any third-party libraries used in the system.
- **Configuration Management:** Managing environment-specific configurations for development, testing, and production environments to ensure consistency and security.
- **Database Setup:** Setting up the database schema, indexes, and initial data migration scripts using tools like SQLAlchemy for Python-based applications.

6.2 Deployment Strategy

Deploying the Library Management System involves a strategic approach to ensure smooth transitions between development stages and production release. Key aspects include:

- **Continuous Integration/Continuous Deployment (CI/CD):** Implementing CI/CD pipelines to automate the build, test, and deployment processes. This includes using tools like Jenkins, GitLab CI/CD, or GitHub Actions to streamline deployment workflows.
- **Deployment Pipeline:** Establishing deployment pipelines for different environments (development, testing, production) to maintain code consistency and stability across releases.
- **Containerization:** Utilizing container technologies such as Docker to package the application and its dependencies into standardized units, ensuring consistency between development and production environments.
- **Orchestration:** Using container orchestration platforms like Kubernetes or Docker Swarm to automate deployment, scaling, and management of containerized applications.
- **Monitoring and Logging:** Implementing monitoring tools (e.g., Prometheus, Grafana) and logging frameworks (e.g., ELK stack) to track application performance, detect issues, and troubleshoot in real-time.

7.Maintainance and support

7.1 Monitoring and Logging

Effective monitoring and logging are crucial for ensuring the stability, performance, and security of the Library Management System. Key activities include:

- **Performance Monitoring:** Utilizing tools like Prometheus and Grafana to monitor system metrics such as CPU usage, memory utilization, and response times. This helps identify performance bottlenecks and optimize system resources.
- **Error Monitoring:** Implementing centralized logging with tools like ELK stack (Elasticsearch, Logstash, Kibana) or Splunk to capture and analyze application logs. This enables proactive identification of errors, exceptions, and potential security incidents.
- **Alerting:** Setting up alerts and notifications for critical system events or anomalies detected during monitoring. Alerts should be configured to notify the appropriate stakeholders (e.g., developers, system administrators) for timely resolution.

7.2 Bug Fixes and Updates

Continuous improvement through bug fixes and updates is essential to maintain the reliability and functionality of the Library Management System. Activities include:

- **Bug Tracking:** Using issue tracking systems like JIRA or Bugzilla to log and prioritize reported bugs. Each bug should be assigned to the responsible team member for investigation and resolution.
- **Patch Management:** Developing a systematic approach for applying patches, updates, and security fixes to the system components (e.g., application code, libraries, dependencies). This ensures that the system remains secure against emerging threats.
- **Version Control:** Maintaining version control with Git or another VCS (Version Control System) to manage codebase changes, track revisions, and facilitate collaboration among developers.
- **Regression Testing:** Conducting regression testing after bug fixes or updates to verify that previously implemented features and functionalities continue to work as expected without unintended side effects.

8.Security Considerations

8.1 Authentication and Authorization

Effective authentication and authorization mechanisms are essential to ensure secure access to the Library Management System:

- **Authentication:** Implementing strong user authentication using hashed passwords stored securely in the database. Utilizing industry-standard protocols like OAuth or OpenID Connect for single sign-on (SSO) capabilities if necessary.
- **Authorization:** Enforcing role-based access control (RBAC) to restrict user permissions based on their roles within the system. Implementing least privilege principles to limit access to sensitive functionalities and data.

8.2 Data Protection

Ensuring the confidentiality, integrity, and availability of data within the Library Management System:

- **Encryption:** Encrypting sensitive data both in transit and at rest using strong encryption algorithms (e.g., AES-256). Utilizing HTTPS/TLS protocols for secure communication between clients and servers.
- **Data Masking and Anonymization:** Applying data masking techniques to obfuscate sensitive information in non-production environments. Anonymizing data where possible to protect user privacy.
- **Backup and Recovery:** Implementing regular data backups with appropriate retention policies. Testing backup integrity and establishing procedures for data recovery in case of data loss or corruption incidents.

9.User Documentation

9.1 User Guide

The User Guide provides comprehensive instructions for users on how to effectively utilize the Library Management System. It aims to empower users with the knowledge needed to navigate through various features and functionalities seamlessly.

- **Introduction to the System**
  - Brief overview of the Library Management System.
  - Purpose and benefits of using the system.
- **Getting Started**
  - Accessing the Login Page:
    - Instructions on how to log in using username and password.
    - Password validation criteria.
- **Navigating the System**
  - Exploring the Book Management Page:
    - Viewing different book categories and selections.
    - Handling book selections and adding them to the cart.

    - Managing the Cart:
        - Viewing selected books in the cart.
        - Setting lending start and end dates.
        - Preventing duplicate book entries in the cart.
- **User Interface**
    - Interface overview:
        - Navigation menu and key features.
        - Using checkboxes for book selections.
        - Clear error messages and validation feedback.
- **Additional Features**
    - Exploring additional functionalities:
        - Customizing user settings if applicable.
        - Accessing help and support resources.
- **Security Guidelines**
    - Best practices for maintaining account security:
        - Managing passwords securely.
        - Logging out and session management tips.
- **Troubleshooting**
    - Common issues and resolutions:
        - Troubleshooting login problems.
        - Handling errors during book selection or cart management.
- **Contact Support**
    - Contact information for technical support.
    - Procedure for reporting issues or providing feedback.

10.Conclusion

In conclusion, the Library Management System aims to streamline and enhance the process of managing books and user interactions within your library. By providing a user-friendly interface, robust security measures, and efficient management of book selections and carts, this system is designed to meet the needs of both library administrators and users alike.

Throughout this documentation, we have outlined the system's architecture, implementation details, testing strategies, deployment considerations, and ongoing maintenance plans. Each component has been meticulously designed to ensure scalability, reliability, and security.

As we move forward, continuous improvement and updates will be prioritized to adapt to changing user needs and technological advancements. We invite feedback from users and stakeholders to further enhance the system's functionality and usability.