

**COLLEGE CODE : 8201**

**COLLEGE NAME : A.R.J COLLEGE OF ENGG. TECH.**

**DEPARTMENT : B.TECH.AI&DS**

**STUDENT NM-ID :**

**fe5233754019fb3ac43460bf056f9e3f**

**ROLL NO : 820123243013**

**DATE : 08-10-2025**

**Completed the project named as Phase : 5**

**TECHNOLOGY PROJECT NAME : IBM-NJ-Library Book Management**

**SUBMITTED BY**

**NAME: MOHANA.S**

**MOBILE NO: 6374439762**

**PHASE 5 – PROJECT DEMONSTRATION & DOCUMENTATION**

( Library book management)

**5.1 Final Demo Walkthrough**

**Overview**

The final demonstration phase is focused on showcasing the complete working model of the **Library Book Management System**.  
It includes the integration of all features developed in previous phases — from authentication to book management, issue/return processes, and report generation.

The purpose of this phase is to verify that the project meets the user requirements, functions smoothly, and is ready for deployment. The demonstration is conducted to both technical evaluators and end users to validate usability and performance.

**Demonstration Steps**

1. **Login Module Demonstration**
   * The user navigates to the login page.
   * Admin or student credentials are entered.
   * Based on the role, the user is redirected to the respective dashboard.
2. **Admin Dashboard**
   * Admin can add, update, or delete books.
   * Can view the availability status of each book.
   * Has access to the complete list of registered users.
3. **Book Search Module**
   * Users can search for books using the title, author, or ISBN.
   * The search results display book availability and details.
4. **Issue & Return Module**
   * Admin issues a book to a student.
   * The system records the issue date, return date, and borrower details.
   * When the book is returned, the system automatically updates availability.
5. **Report Generation**
   * Admin can generate reports on issued, returned, and overdue books.
   * Reports can be downloaded in PDF format.

6.**Deployed System Demo**

* + The application is hosted on **IBM Cloud** (backend) and **Netlify/Vercel** (frontend).
  + The live system is demonstrated through the browser, showing complete functionality and responsiveness.

**5.2 Project Report**

**Structure of the Project Report**

The project report is an essential documentation that explains the entire lifecycle of the system — from problem identification to deployment. It is designed to serve as a reference for future improvements and evaluations.

**Report Contents**

1. **Title Page**
   * Project Title
   * Team Members & Roll Numbers
   * Institution & Department Name
   * Mentor/Supervisor Details
2. **Abstract**
   * A concise summary highlighting the problem statement, objectives, and major outcomes of the project.
   * Example:  
     “The Library Book Management System is designed to automate the process of managing books, issuing, and returning them efficiently. It reduces manual errors and improves accessibility for both students and librarians.”
3. **Introduction**
   * Describes the need for automation in library systems.
   * Discusses traditional challenges such as manual record keeping and lack of real-time tracking.
4. **Problem Statement**
   * Manual management of library books leads to inefficiency, loss of records, and time consumption.
   * The system aims to automate these processes using a web-based application.
5. **Requirements and Design Phases**
   * Include Phase 1 and Phase 2 details (Problem Understanding, Architecture, and Diagrams).
6. **Implementation Details**
   * Explain the coding part, database connectivity, and the working modules (from Phase 3 and Phase 4).
7. **Testing & Results**
   * Provide screenshots and test outcomes to show that all features function as intended.
8. **Conclusion & Future Scope**
   * Mention project outcomes and how it can be extended with new features (e.g., barcode scanning, AI-based recommendations).

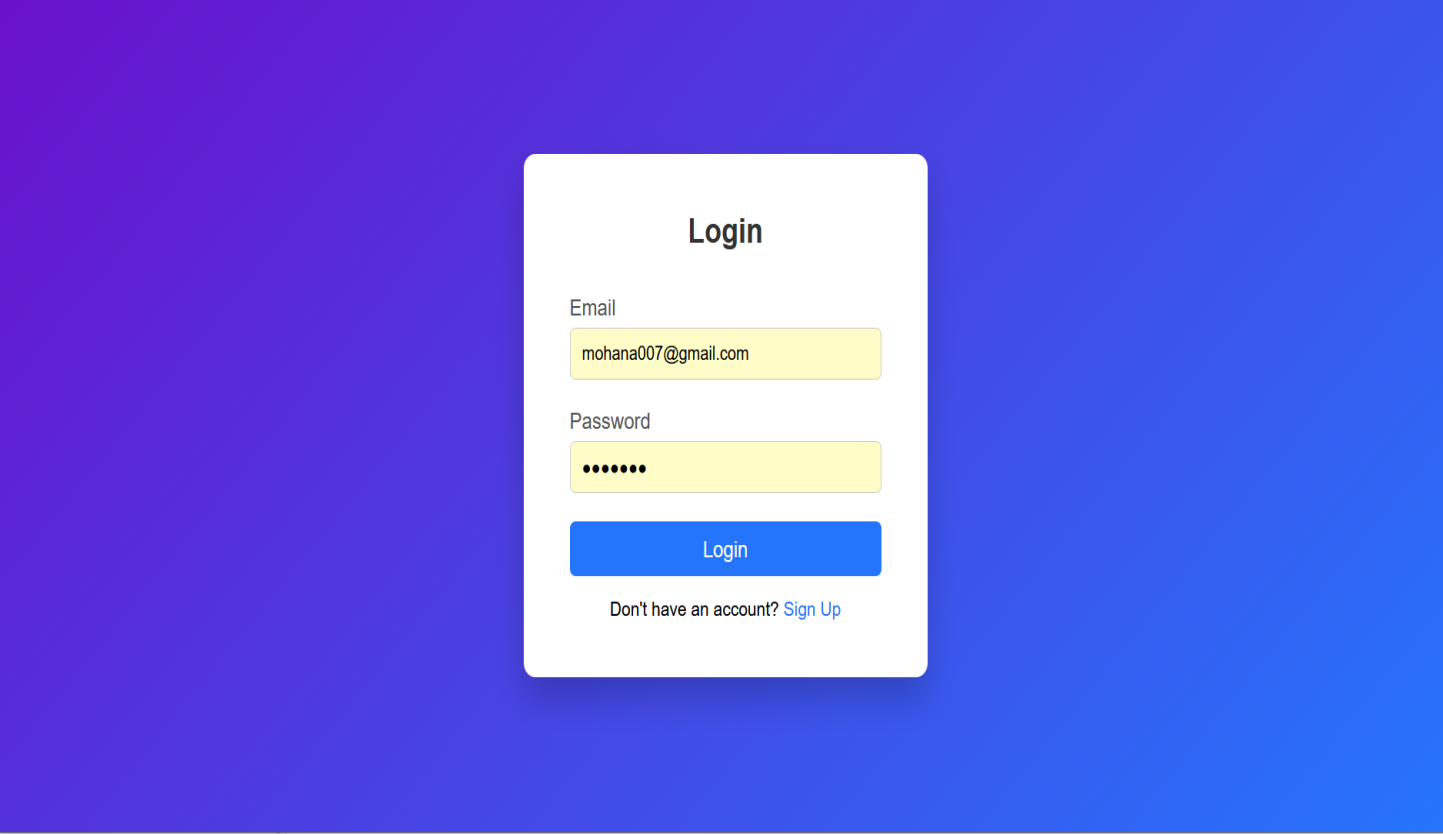
| **Technology** | **Purpose / Usage** |
| --- | --- |
| **HTML, CSS, JavaScript** | Build and style the web pages |
| **Bootstrap** | Make the UI responsive and user-friendly |
| **Node.js + Express.js** | Backend server and API handling |
| **MongoDB** | Store books, users, and transaction data |
| **JWT (JSON Web Tokens)** | User authentication and role management |
| **Nodemailer** | Send email notifications and overdue reminders |
| **Git / GitHub** | Version control and project management |
| **Postman** | Test backend APIs |
| **Chart.js** | Display reports and statistics like most issued books |
| **Heroku / AWS** | Deploy the web application online |

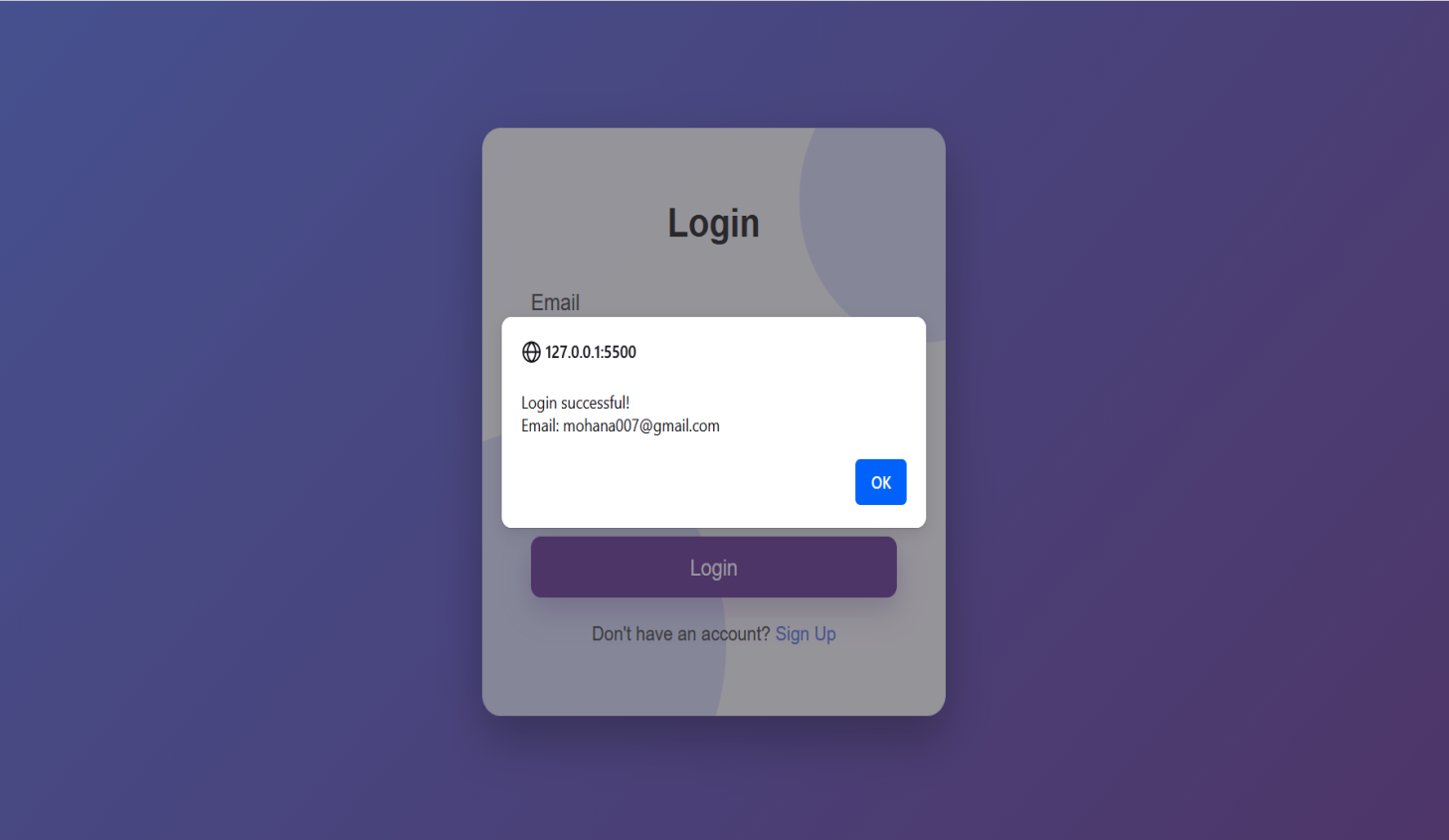
**5.3 Screenshots and API Documentation**

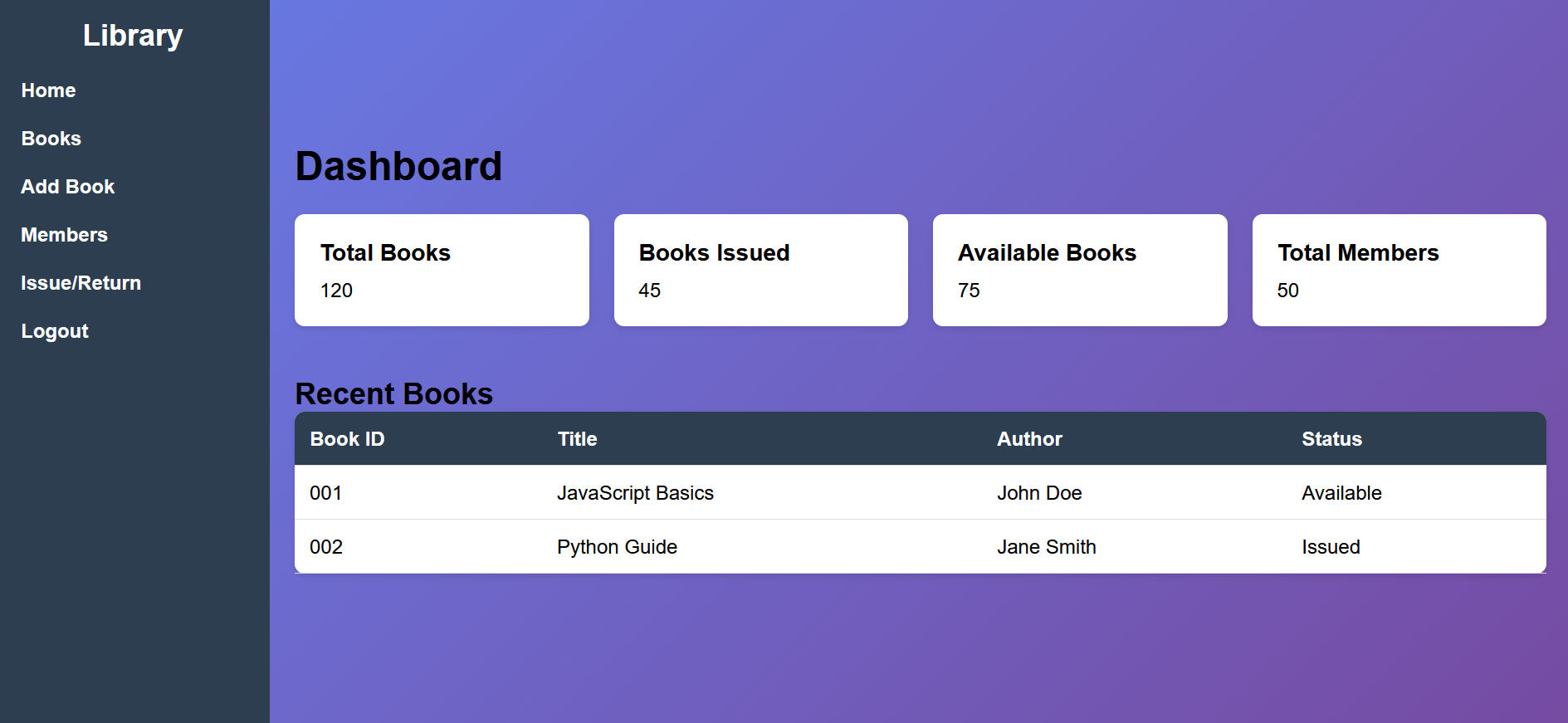
**Screenshots to Include**

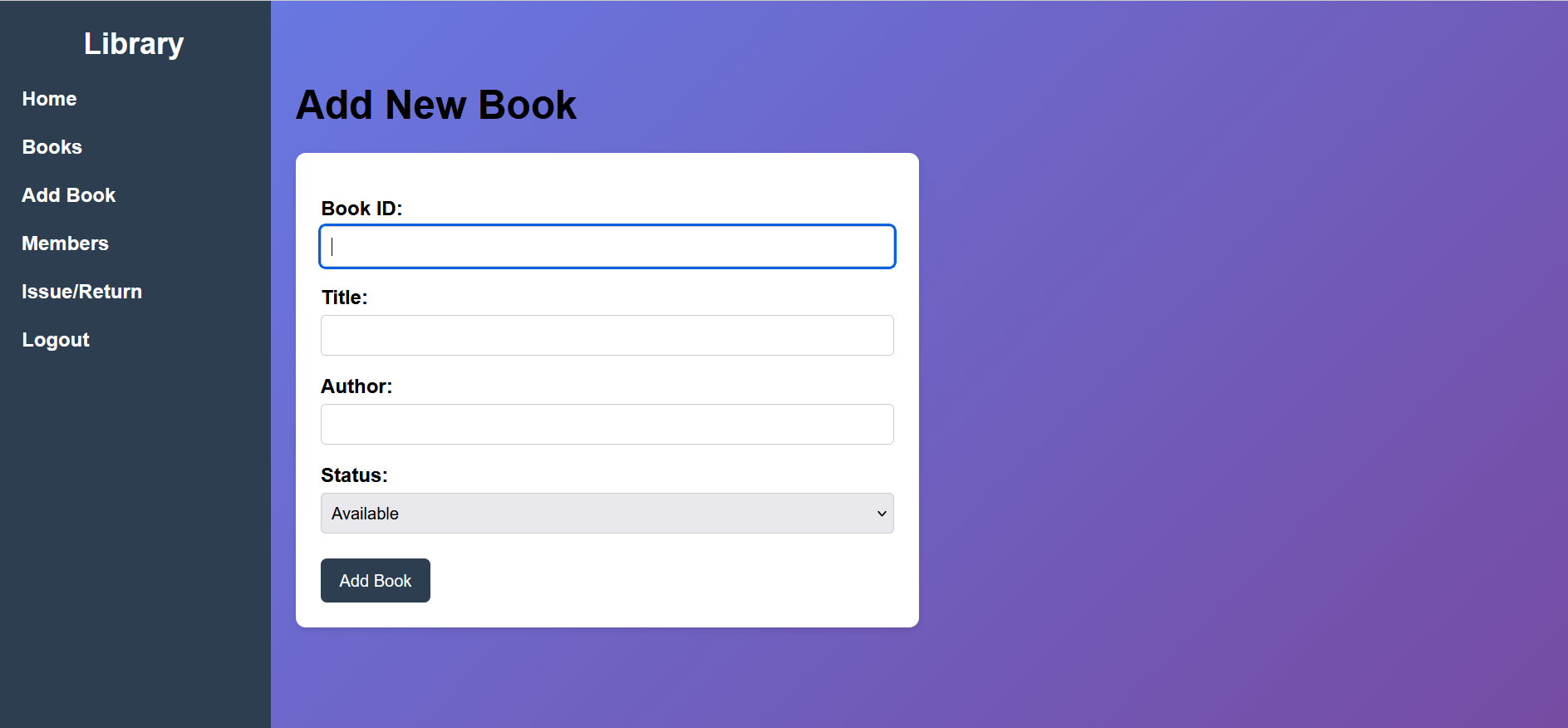
Add clear screenshots from your project interface:

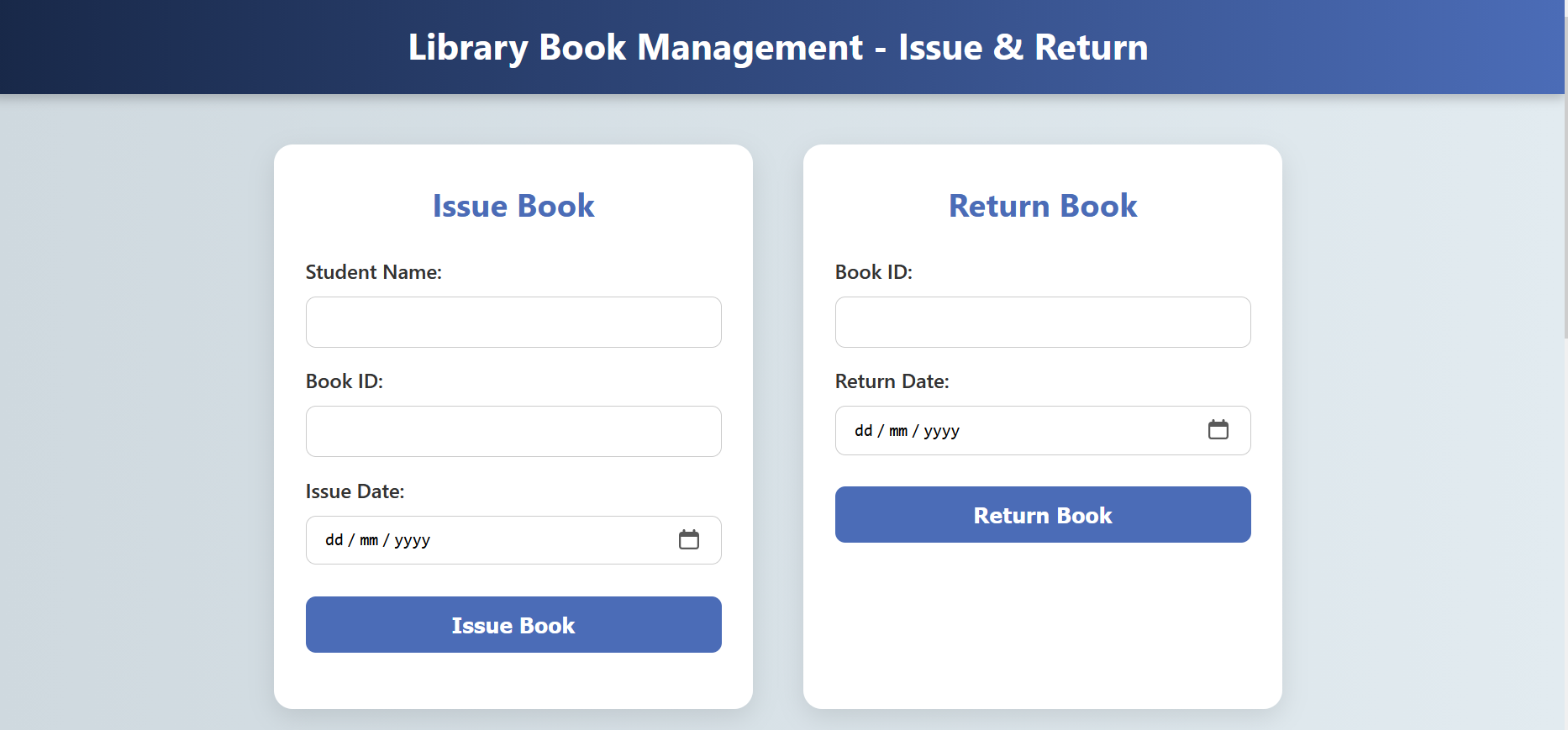
1. **Login Page (Admin/Student)**
2. **Admin Dashboard**
3. **Add Book / Edit Book Interface**
4. **Search Book Module**
5. **Issue/Return Screen**
6. **Reports Page**
7. **Deployed Website Screenshot (Browser View)**

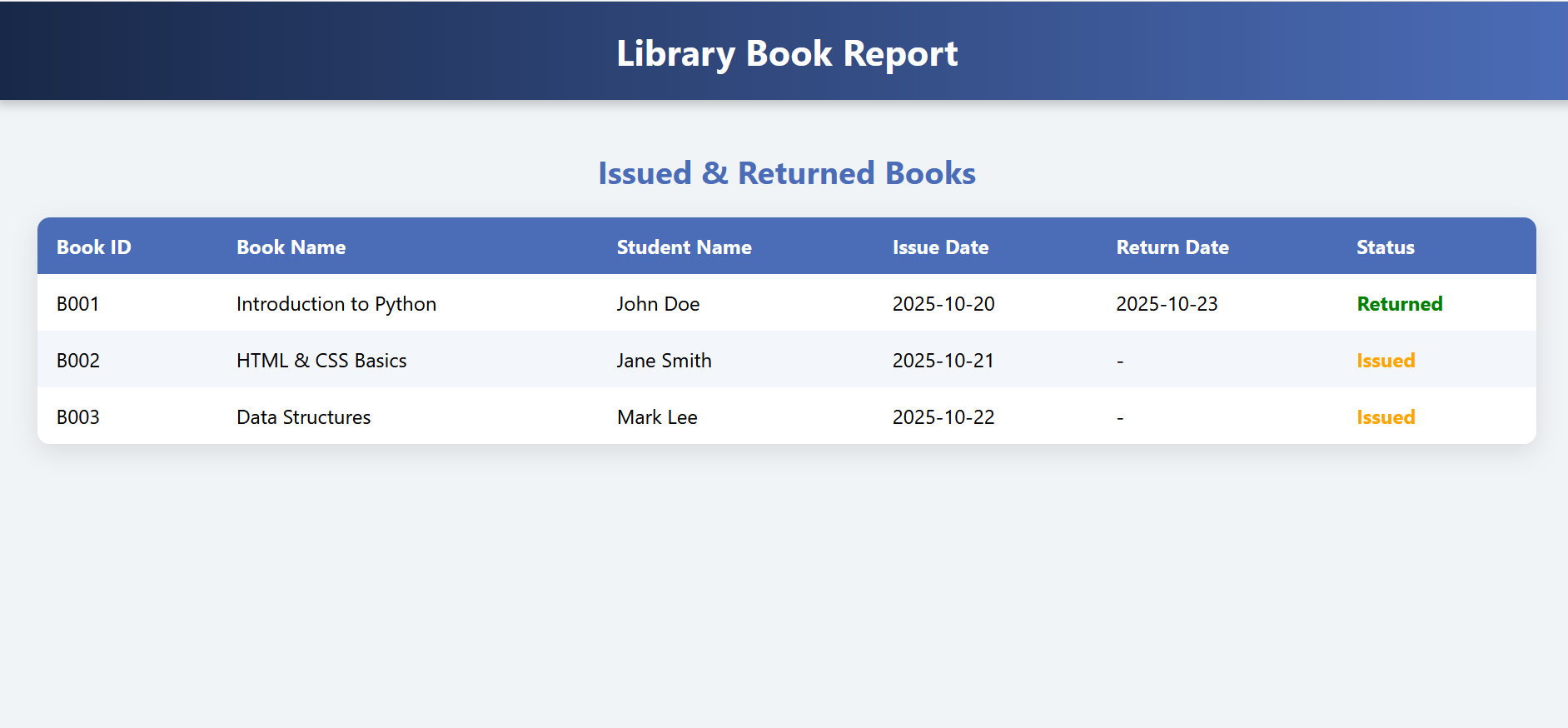


****

****

****

****

****

**API Documentation Example**

| **Endpoint** | **Method** | **Description** |
| --- | --- | --- |
| /api/login | POST | Authenticates users (Admin/Student) |
| /api/register | POST | Registers new user accounts |
| /api/books | GET | Fetches list of available books |
| /api/books/:id | PUT | Updates book details |
| /api/issue | POST | Issues a book to a student |
| /api/return | POST | Returns a book |
| /api/reports | GET | Generates issue/return report |

All APIs were tested using **Postman**, and responses were verified with the live database connection.

**5.4 Challenges and Solutions**

| **Challenge** | **Description** | **Solution Implemented** |
| --- | --- | --- |
| Database Connectivity | Issues connecting IBM Db2 to the Flask backend | Used ibm\_db driver and tested using dummy data |
| Session Management | Maintaining user login sessions | Implemented JWT (JSON Web Token) authentication |
| Responsive Design | UI not scaling properly on mobile | Applied Tailwind CSS and Flexbox for layout adjustments |
| Deployment Issues | Backend deployment errors on IBM Cloud | Adjusted configurations and environment variables |
| Data Consistency | Duplicate book entries in database | Added unique ISBN validation logic |

These challenges helped improve debugging skills and provided a deeper understanding of full-stack development and cloud hosting.

**5.5 GitHub README and Setup Guide**

Our **GitHub repository** should include:

* Project title and description
* Tech stack used (Frontend, Backend, Database)
* Setup instructions
* Folder structure
* Deployment links
* API documentation section

**Sample README.md Structure**

**Library Book Management System**

**Overview**

A full-stack web-based library management system that allows users to manage, issue, and return books efficiently.

**Tech Stack**

- Frontend: HTML, CSS, JavaScript, React.js

- Backend: Python Flask

- Database: IBM Db2 Cloud

- Hosting: Netlify (Frontend) & IBM Cloud (Backend)

**Installation**

```bash

git clone https://github.com/username/library-management.git

cd library-management

pip install -r requirements.txt

python app.py

**Deployment**

* Frontend: https://library-management.netlify.app
* Backend: https://library-api.ibmcloud.app

**5.6 Final Submission (Repository + Deployed Link)**

**Deliverables**

1. **GitHub Repository Link**

- Must include all source files, documentation, and commit history.

- Example:

`https://github.com/book-nm/library-management`

2. **Deployed Application Link**

- **Frontend**: Netlify/Vercel

- **Backend:** IBM Cloud

- **Database:** IBM Db2 Instance

Example:

- Live Project: [https://library-system.netlify.app](https://library-system.netlify.app)

- Backend API: [https://library-api.ibmcloud.app](https://library-api.ibmcloud.app)

3. **Documentation Files**

- Project Report (PDF)

- Screenshots Folder

- README & Setup Guide

- Optional: Demo Video Link (YouTube/Drive)

**Conclusion**

The Library Book Management System successfully demonstrates the implementation of modern web technologies to solve real-world problems in library operations.

By integrating authentication, CRUD operations, reporting, and cloud deployment, the project achieves the goal of providing an efficient, user-friendly, and scalable solution for library automation.