CHAPTER 1
INTRODUCTION

1.1 PROBLEM STATEMENT:

Library database Management System is a system that shows all the available books and their count and also books taken by people, the date on which they took that particular book, expected date of return, late due fees, membership details, and so on. Everything will be crystal clear. There will be no ambiguity. It will be beneficial for both students and librarians.

This library database management is very efficient and also cost-effective. It saves a lot of time for both librarians and also students. With this, manual work is reduced, requiring less staff and maintenance. This system is user-friendly and also very easy to use.

1.2 OBJECTIVES :

The project aims and objectives that will be achieved after completion of this project are discussed in this subchapter. The aims and objectives are as follows:

- To develop a database that stores user details and book details.
- To create an easy to understand and user-friendly interface for library patrons.
- To provide reliable search facilities for users.
- To develop a security system with separate logins for administrators and students.
- To keep track of the information of books, such as title, name, author, and subjects available in the library.
- To keep track of transaction information, such as issue and return of books, fines, and reservations.
- To enable library staff to manage the entire lifecycle of library resources, from acquisition to disposal.
- To enable library patrons to access library resources and services 24/7, from the comfort of their homes or on-the-go.

1.3 SQL COMMANDS :

1.3.1 Admin:

```
CREATE TABLE Admin (
  ID INT PRIMARY KEY,
  A_name VARCHAR(10) UNIQUE,
 User_name VARCHAR(10),
  Password VARCHAR(10),
 );
```

1.3.2 Student :

```
CREATE TABLE Student (
 Student_ID INT PRIMARY KEY,
 ID INT(10),
  Student_name VARCHAR(10) ,
  Email VARCHAR(20),
  Password VARCHAR(10),
  MobileNo VARCHAR(15),
  Reg_date DATE,
```

```
 FOREIGN KEY (ID) REFERENCES Admins(ID)

 FOREIGN KEY (Dname) REFERENCES Departments(Dept_name));
```

1.3.3 Books

```
CREATE TABLE Books (

 Book_ID INT PRIMARY KEY,

 ID INT (10)

 Author_ID INT (10),

 Price INT(20) ,

 FOREIGN KEY (ID) REFERENCES Admin(ID)

 FOREIGN KEY (Author_ID) REFERENCES Authors(Author_ID)

);
```

1.3.4 Author

```
CREATE TABLE Author (

Author_ID INT PRIMARY KEY,

Author_name VARCHAR(10),

Creation_date DATETIME NOT NULL,

 Updation_date DATETIME NOT NULL

);
```

1.3.5 Issued_Books

```
CREATE TABLE Issued_Books (

 Issue_ID INT PRIMARY KEY,

 Student_ID INT(10),

 Book_ID INT(10),

 Issue_date DATE NOT NULL,

 Return_date DATE NOT NULL,

 Fine INT(20),

 FOREIGN KEY (Student_ID) REFERENCES students(Student_ID),

 FOREIGN KEY (Book_ID) REFERENCES Books(Book_ID)

);
```

1.3.6 Department

```
CREATE TABLE Department (

 Dept_name VARCHAR(10) ,

 Dept_num INT (10)

);
```

CHAPTER 2
METHODOLOGY

2.1 ABOUT BACKEND CONNECTION

- **Designing the Database Schema:** Begin by designing the database schema to store information such as products, users, orders, etc. For a footwear website, you might have tables for products, categories, users, orders, etc.
- **Setting up the Backend with Node.js:** Use Node.js along with a framework like Express.js to create the backend server. Express.js makes it easier to handle HTTP requests, route requests to appropriate handlers, etc.
- **Implementing Authentication and Authorization:** Implement user authentication using techniques like JWT (JSON Web Tokens). This involves creating endpoints for user registration, login, and logout. Additionally, implement authorization to restrict access to certain endpoints based on user roles or permissions.
- **Handling Payments:** Integrate a payment gateway like Stripe or PayPal to handle payments securely. This involves creating endpoints to initiate and process payments. Error Handling and Validation: Implement error handling and validation to ensure data integrity and provide meaningful error messages to the client.
- **Setting Up React Frontend:** Create a React application for the frontend of the website. Use components to structure the UI and manage state using tools like Redux or React Context API.
- **Consuming Backend APIs:** Use Axios or Fetch API to make HTTP requests from the React frontend to the backend APIs you've created. Update the UI based on the data received from the backend.
- **Dependencies:** The code imports necessary modules such as Express, MySQL, Cors, Multer (for handling file uploads), and Path.
- **Middleware Setup:** Middleware like CORS, JSON body parser, and serving static files are set up using Express.

2.2 INTRODUCTION TO SERVER :

- **MySQL:** MySQL is an open-source relational database management system renowned for its performance, reliability, and versatility. It organizes data into tables with rows and columns, facilitating efficient data storage and retrieval. Its support for SQL. enables developers to execute a wide range of database operations seamlessly, MySQL is highly scalable, capable of handling large datasets and accommodating both vertical and horizontal scaling strategies. It offers features like replication and clustering for data redundancy, fault tolerance, and high. availability. The diverse storage engines cater to different use cases, ensuring flexibility and optimization. Supported by a vibrant community and backed by commercial support options, MySQL. enjoys widespread adoption across various platforms and programming languages, Its open-source nature fosters innovation and collaboration, making it a preferred choice for businesses ranging from startups to large enterprises.

- **Web Chrome Server:** The "Web Server for Chrome" extension, which allows users to serve local files and folders via HTTP, essentially turning their computer into a local web server. This extension enables developers to test their web applications locally without needing to deploy them to a remote server.
  The "Web Server for Chrome extension provides a simple user interface within the browser, allowing users to select the directory they want to serve and configure basic settings such as port number and access control. Once the server is running, users can access their local files through a URL. generated by the extension, typically in the format "http://localhost:port or "http://127.0.0.1:port", where "port" is the port number specified during setup. This local web server capability is invaluable for web development tasks such as testing frontend code, debugging JavaScript applications, or experimenting with server-side scripting languages like PHP or Nodejs. While primarily intended for development purposes, it's important to note that local web servers set up through Chrome extensions like "Web Server for Chrome" are not suitable for hosting production websites, as they lack the security, scalability, and reliability features necessary for live deployments.

CHAPTER 3
SYSTEM REQUIREMENTS AND SPECIFICATIONS

3.1 ABOUT SOFTWARE REQUIREMENTS:

FrontEnd:

User Interface Design: HTML, CSS, JS, Material UI
Web Browser: Google Chrome

Back End:

Coding Language: Node.js
Database: MySQL
Software: XAMPP 3.3 (primarily for local server setup which includes Apache, MySQL, PHP)
Operating Systems: Windows 11 (provides the environment for running the server and other backend processes)

3.2 ABOUT HARDWARE REQUIREMENTS:

- Processor: Dual-core or higher for handling server operation efficiently.
- RAM: At least 4GB, though 8GB or more would be beneficial for smoother performance.
- Storage: 512 SSD storage for faster read/write operations.
- Network: Stable internet connection for serving requests to client devices.

CHAPTER 4
SYSTEM DESIGN

4.1 ER DIAGRAM :

The Entity Relational Model is a model for identifying entities to be represented in the database and representation of how those entities are related. The ER data model specifies an enterprise schema that represents the overall logical structure of a database graphically.The Entity Relationship Diagram explains the relationship among the entities present in the database. ER models are used to model real-world objects like a person, a car, or a company and the relation between these real-world objects. In short, the ER Diagram is the structural format of the database.

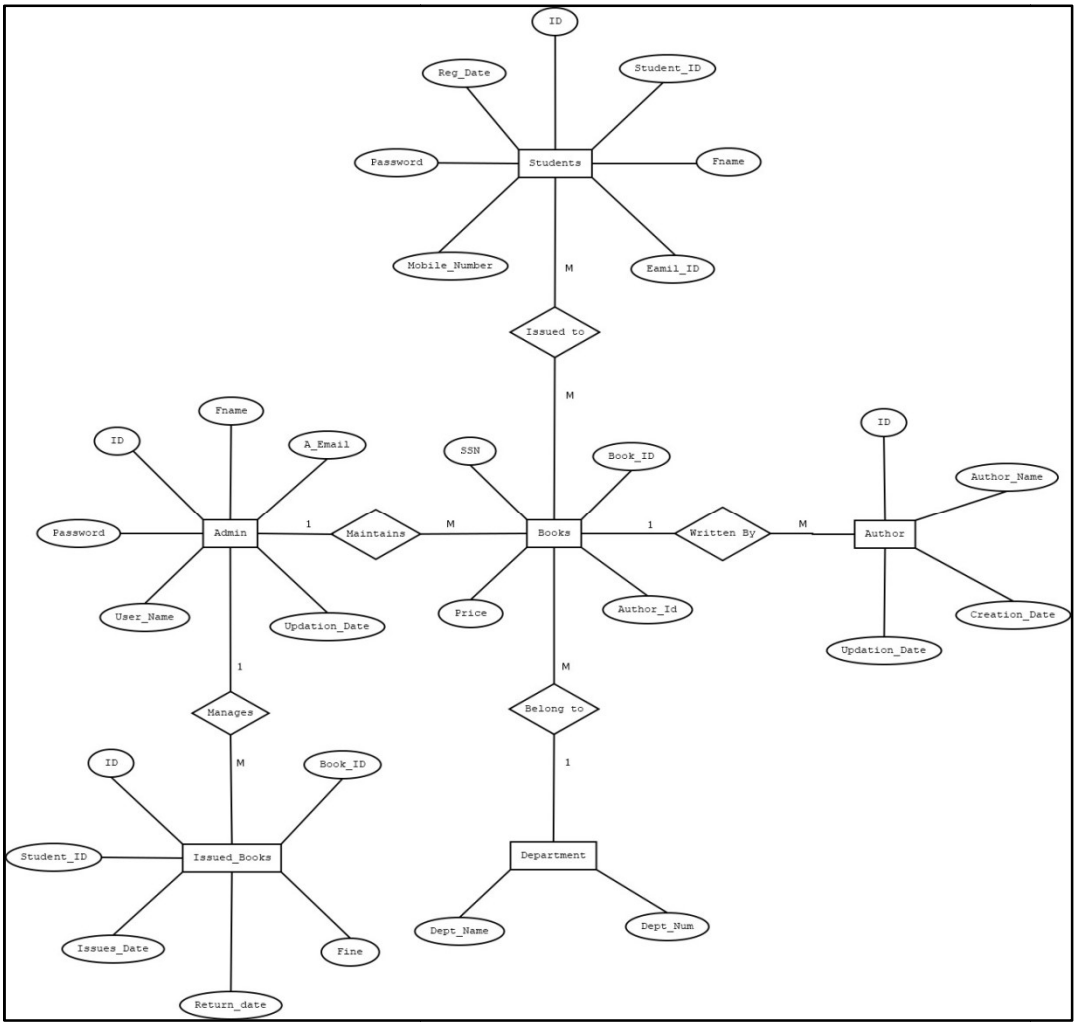

Figure 4.1: ER Diagram

4.2 SCHEMA DIAGRAM:

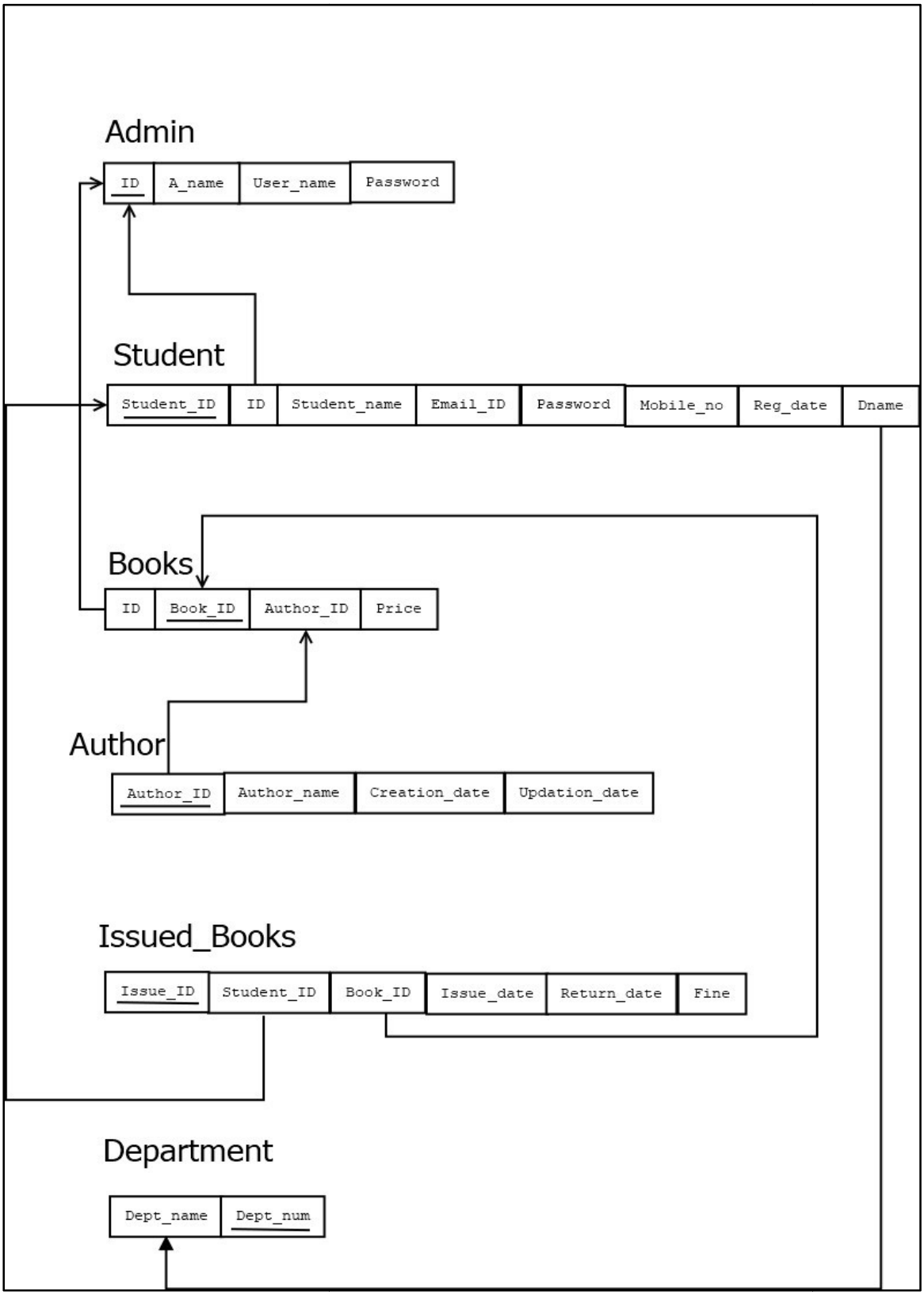

Figure 4.2: Schema Diagram

- **Admin:-**
  Primary Key: ID
- **Student:-**
  Primary Key: Student_ID
  Foreign Key: ID(refers to Admin's ID)
  Foreign Key: Dname(refers to Department's Dept_name)
- **Book:-**
  Primary Key:Book_ID
  Foreign Key:Author_ID(refers to Author's Author_ID)
  Foreign Key: ID(refers to Admin's ID)
- **Author:-**
  Primary Key:Author_ID
- **Issued_Books:-**
  Primary Key:Issue_ID
  Foreign Key:Student_ID(refers to Students's Student_ID)
  Foreign Key: Book_ID(refers to Book's Book_ID)
- **Department:-**
  Primary Key:Dept_name

Relationship and Cardinality Ratio:

Entity 1	Relationship	Entity 2	Cardinality Ratio
Admin	Maintains	Books	1:M
Admin	Manages	Issued_Book	1:M
Books	Written By	Author	1:M
Books	Issued to	Students	M:M
Books	Belongs to	Department	M:1

CHAPTER 5
IMPLEMENTATION

BACKEND:

The code appears to be a Node.js server application using Express.js framework for handling HTTP requests and interacting with a MySQL database. Here's a breakdown of the major components and functionalities:

- **Dependencies:**
  - o `express`: For building the web server and API endpoints.
  - o `mysql`: For connecting to and interacting with the MySQL database.
  - o `cors`: Middleware for enabling Cross-Origin Resource Sharing (CORS).
  - o `multer`: For handling file uploads.
  - o `path`: For working with file and directory paths.

- Server Setup:
  - o Express app is created.
  - o CORS is enabled to allow cross-origin requests.
  - o JSON body parsing middleware is added.
  - o Static file serving middleware is set up to serve images from the '/images directory.

- Database Connection:
  - o Connection to a MySQL database named 'Librarywebsite' is established.

- Middleware:
  - o Multer middleware is used for handling file uploads when adding a new product(`/addproduct`).

- Server Listening:
  - o The server is set to listen on port 8081.

Overall, this code provides a comprehensive backend for an e-commerce website, handling authentication, user management, product management, order management, and more, using Express and MySQL.

DATABASE TABLES:

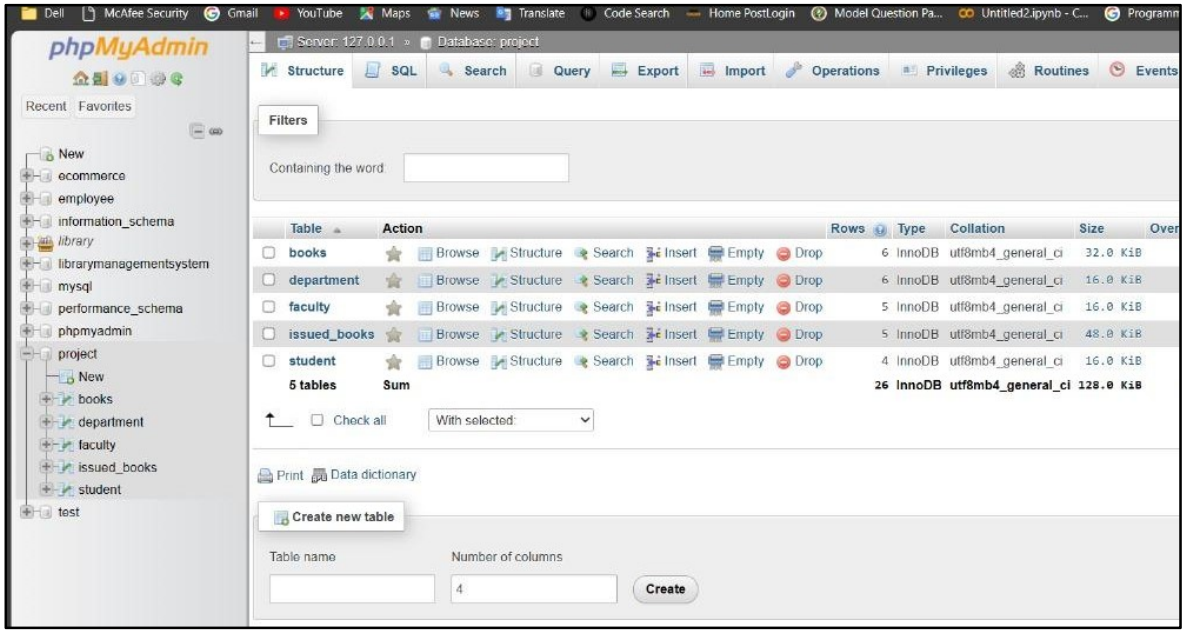

Figure 5.1:Tables

Books Table

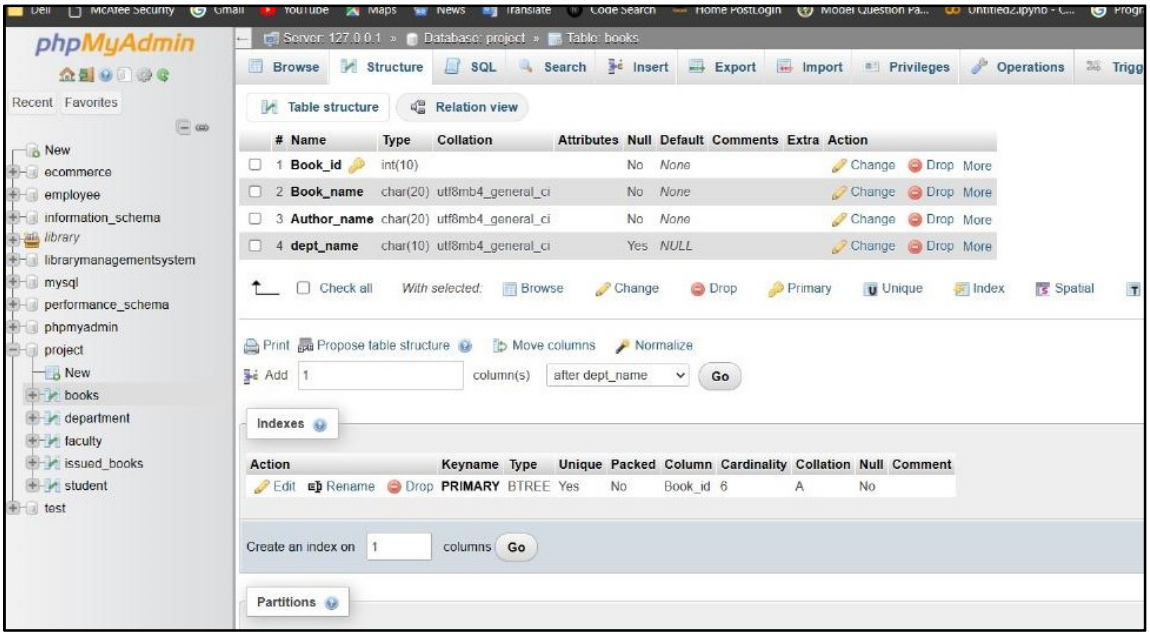

Figure 5.2:Books Table

Department Table

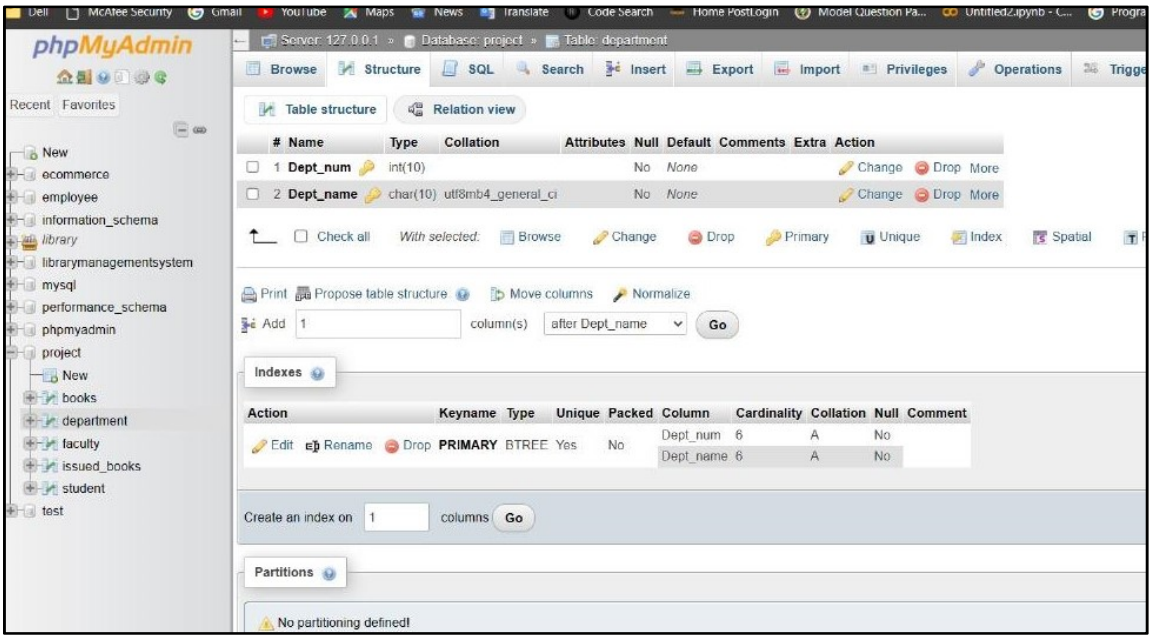

Figure 5.3:Department Table

Issued Book Table:

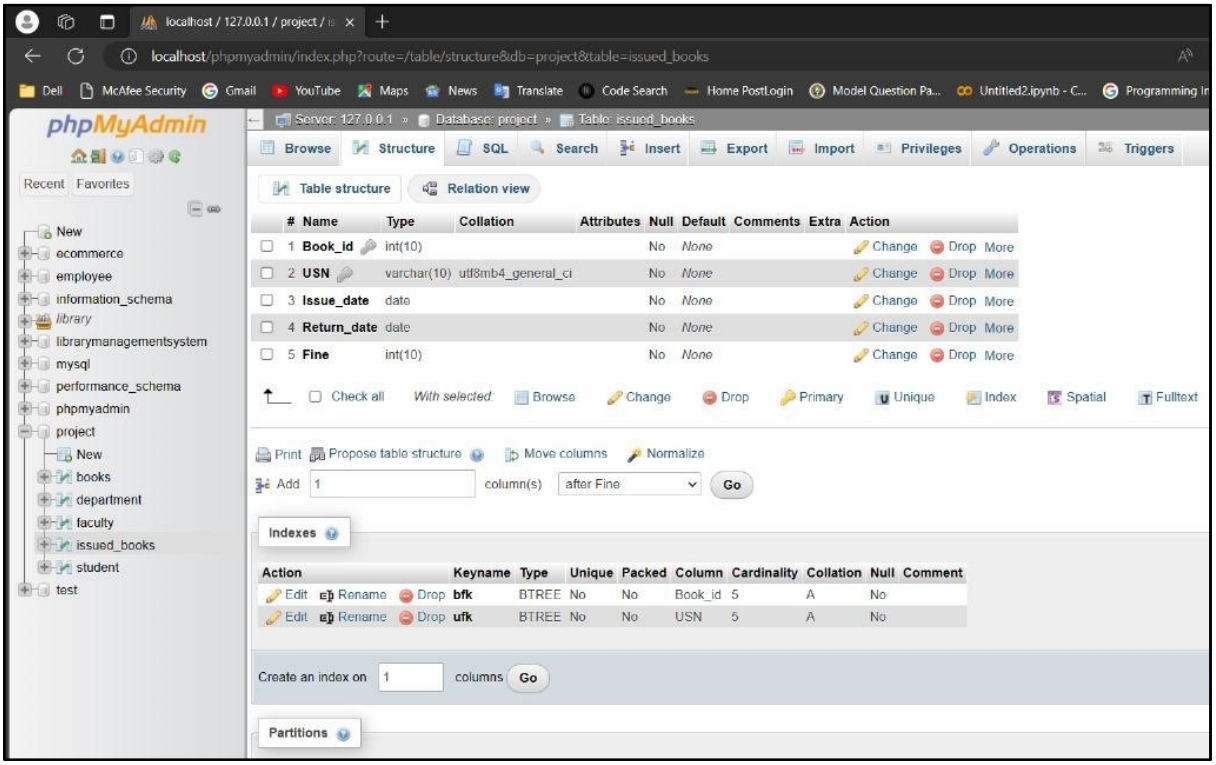

Figure 5.4:Issued Book Table

Students Table:

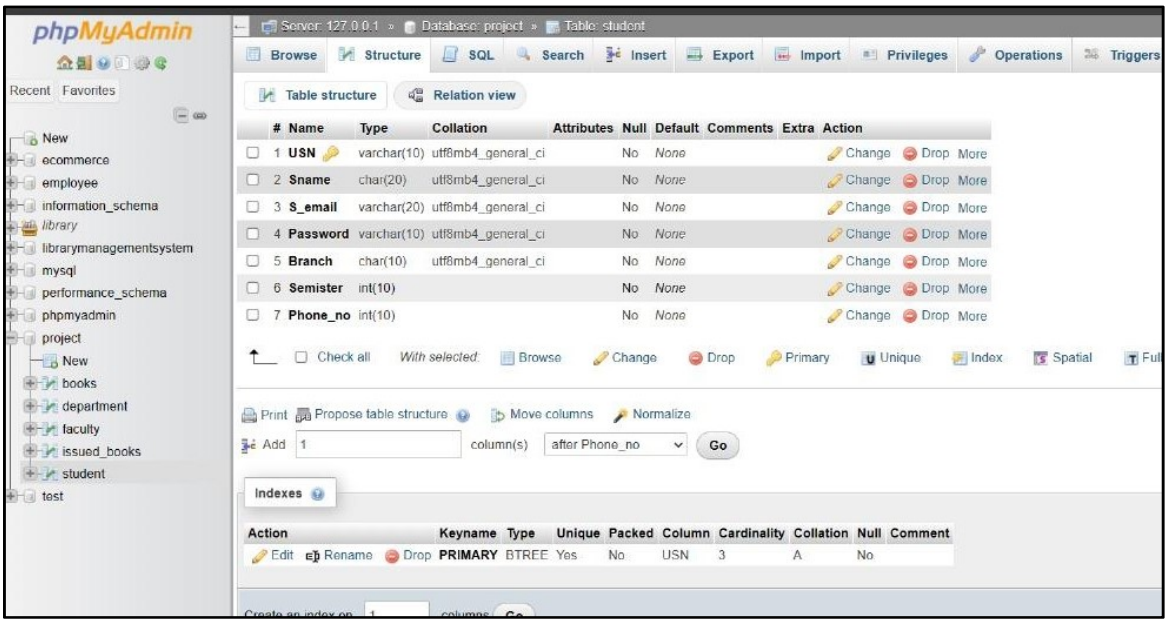

Figure 5.5:Student Table

CHAPTER 6
RESULT

The below snapshot illustrates the project entitled “Library Management System”

FRONTEND USER PANEL :

HOME:

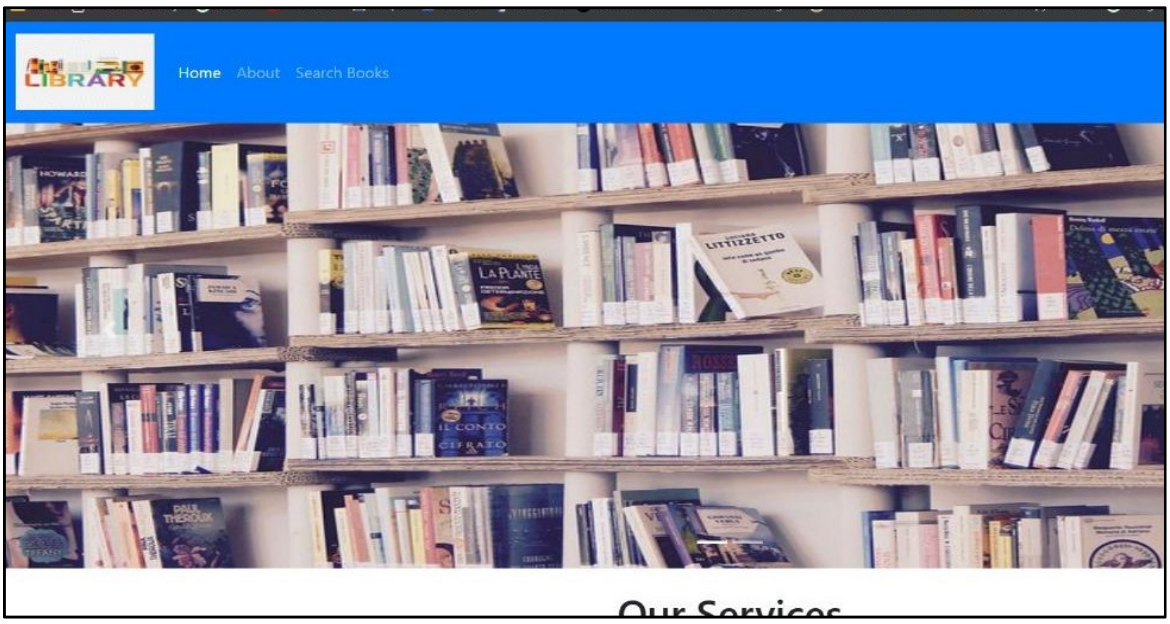

Figure 6.1: Home Page

ISSUANCE OF BOOK:

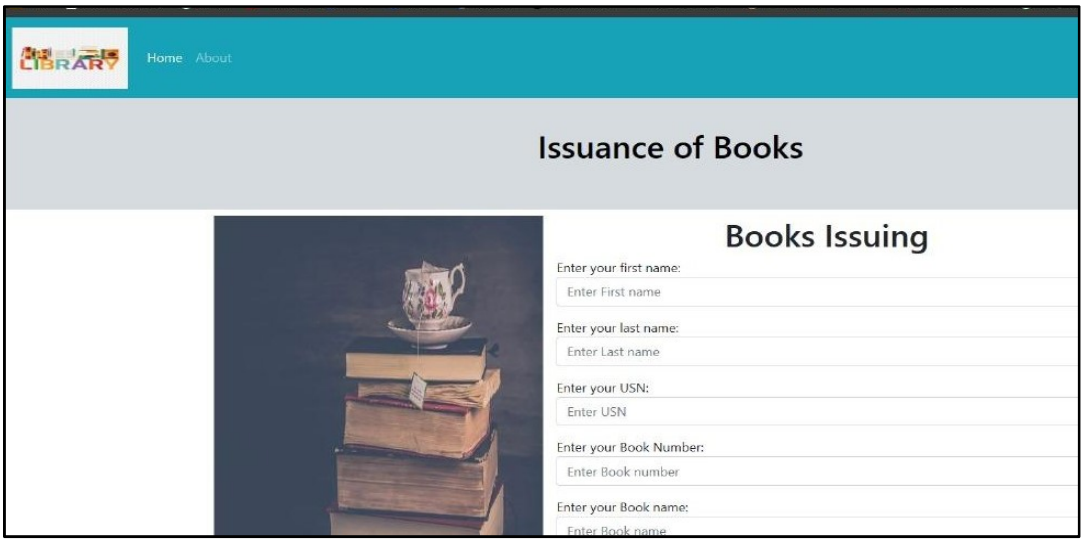

Figure 6.2: Issuance of Book Page

ABOUT THE LIBRARY:

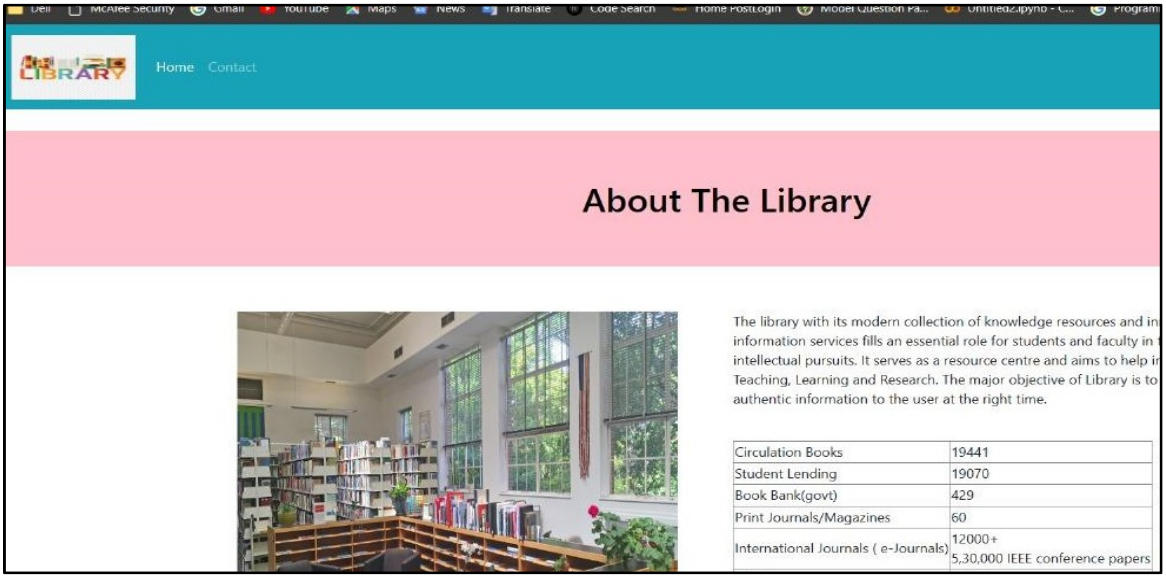

Figure 6.3: About Page

CONTACT DETAILS:

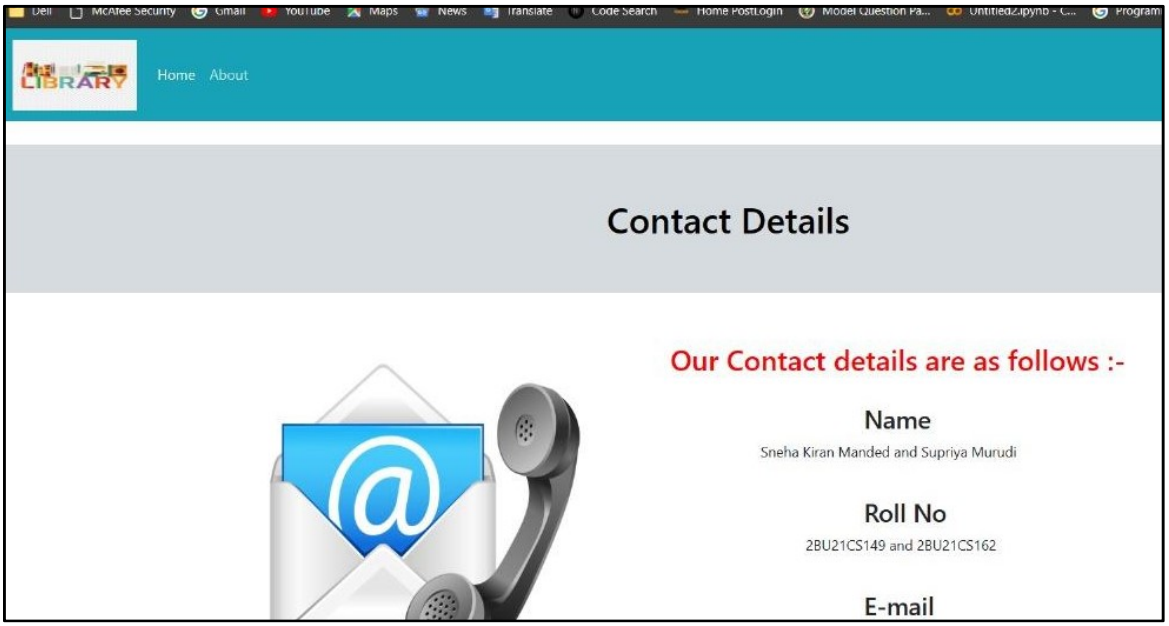

Figure 6.4: Contact Page

BOOK COLLECTION:

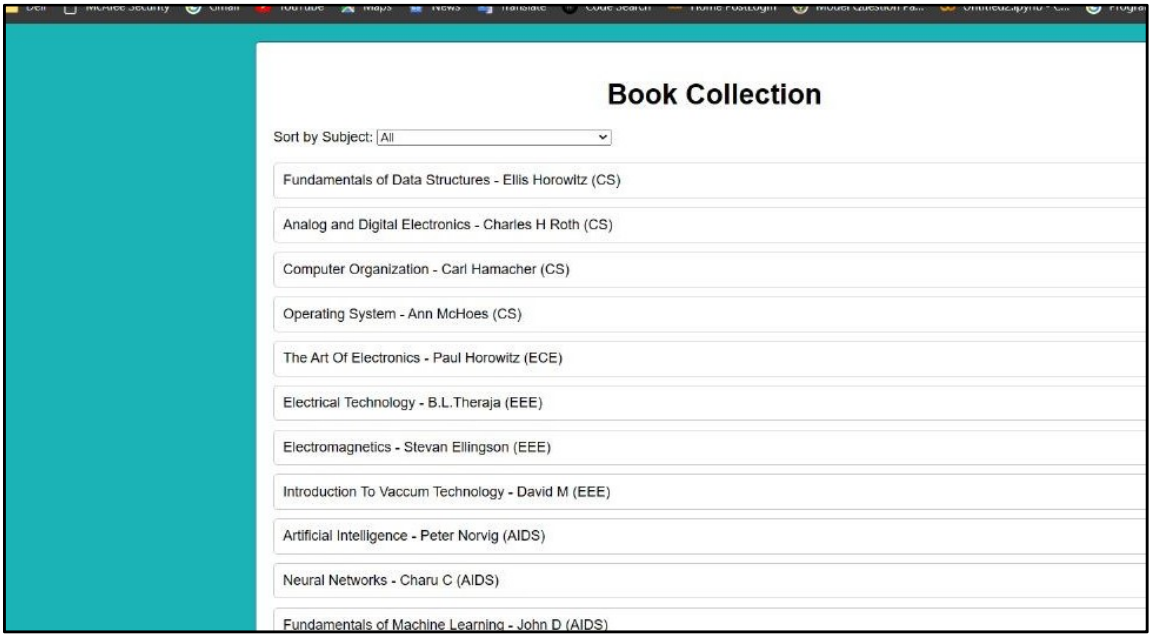

Figure 6.5: Book Collection Page

E_BOOK LIBRARY:

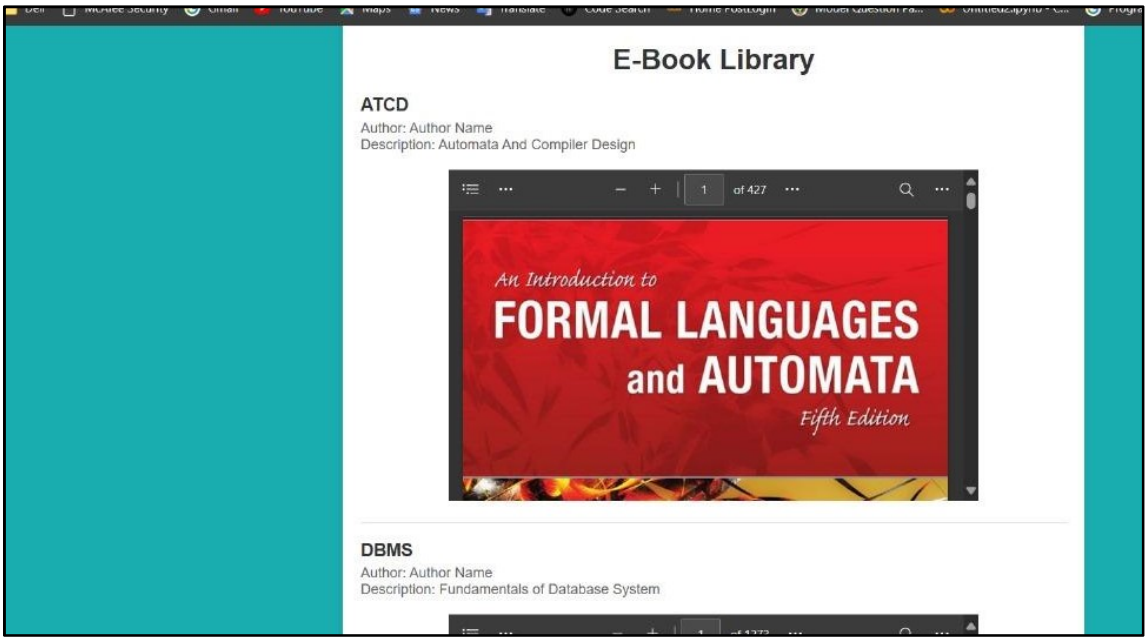

Figure 6.6: E_Book Page

STUDENT CORNER:

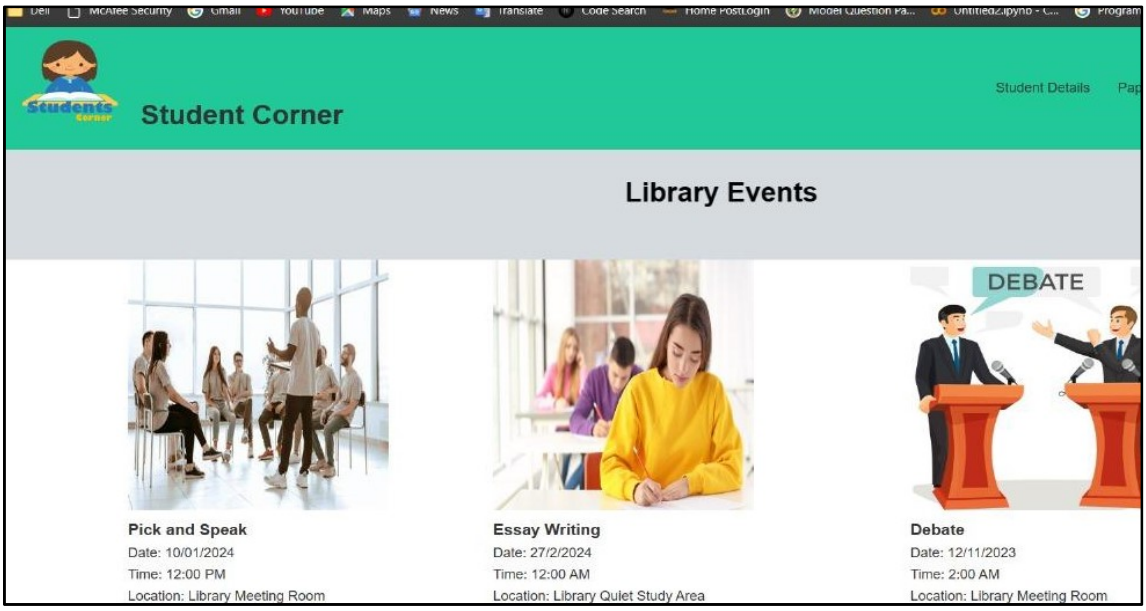

Figure 6.7: Student Corner Page

STUDENT REGISTRATION:

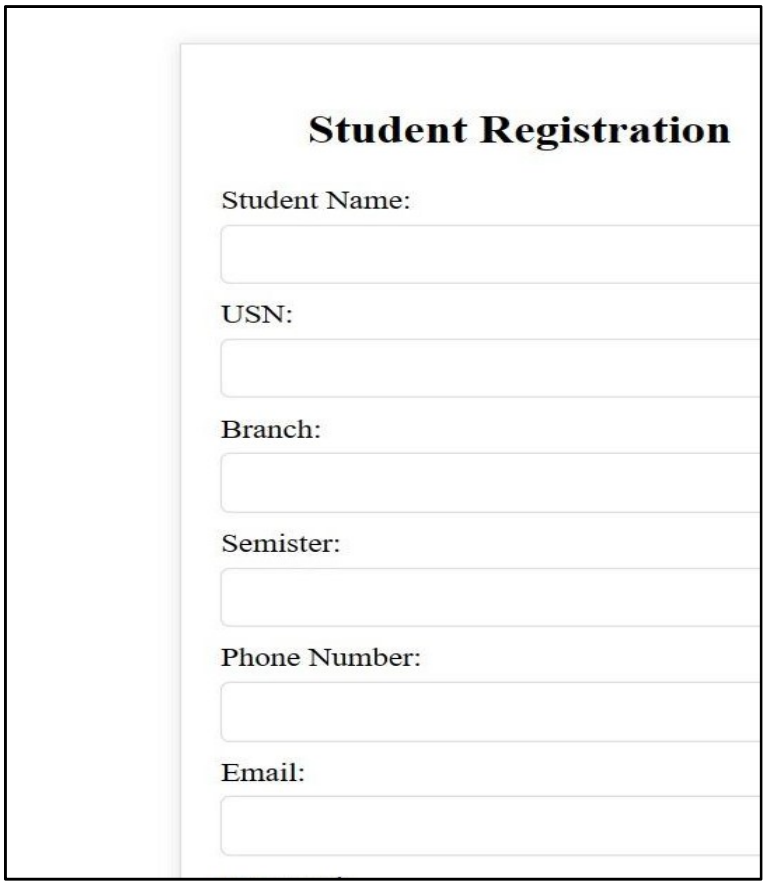

Figure 6.8: Student Registration Page

LOGIN :

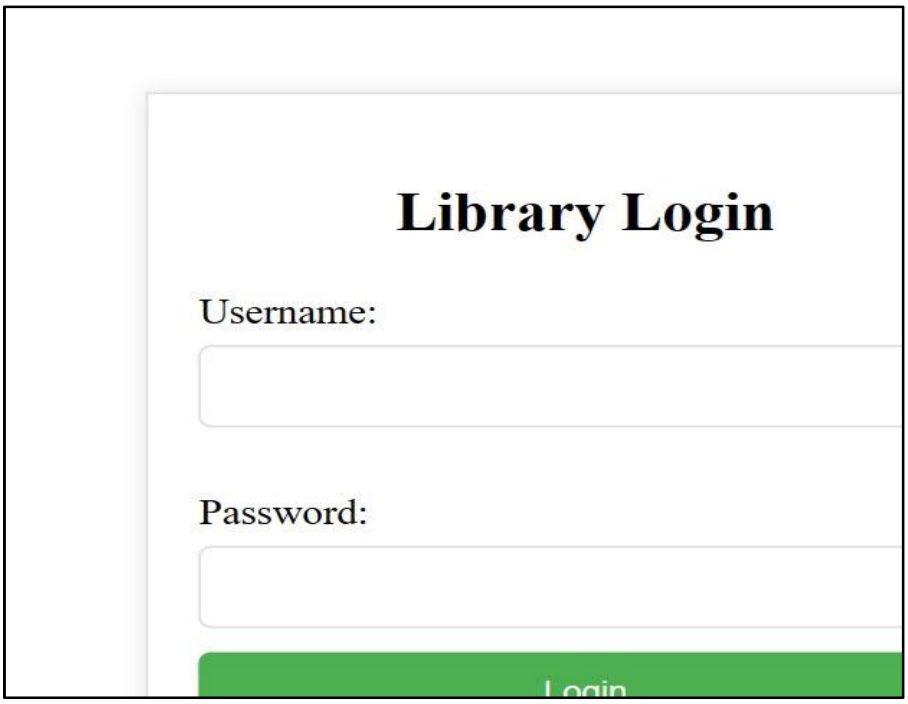

Figure 6.9: Login Page

FACULTY REGISTRATION:

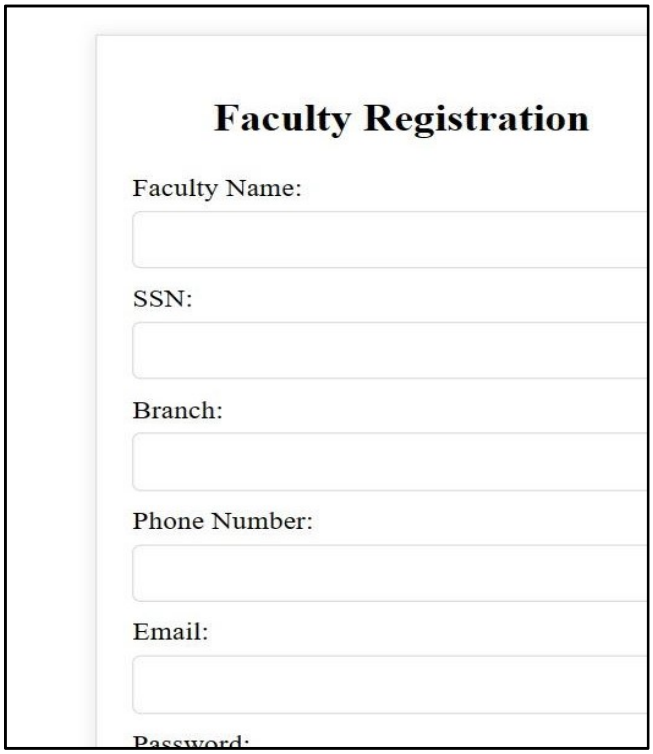

Figure 6.10: Faculty Registration Page

CHAPTER 7
CONCLUSION

A library management system is a crucial tool for educational institutions, streamlining library operations and enhancing accessibility for students and staff. It simplifies tasks such as data management, book search, catalog management, issuance, and returns. The system also offers notifications, penalty calculations, barcode integration, and mobile application integration, contributing to timely completion of work and increased productivity. By automating various library operations, the system boosts productivity, allowing librarians to allocate more resources to priority activities. Overall, the library management system is a cost-effective and practical solution for modern libraries, helping them adapt to the digital age while maintaining the integrity of their collections and services.

REFERENCE

[1] https://mui.com/material-ui/
[2] https://www.npmjs.com/package/react-router-dom
[3] https://www.npmjs.com/package/react-toastify
[4] https://www.npmjs.com/package/sweetalert2
[5] https://www.npmjs.com/package/react-countup
[6] https://www.npmjs.com/package/nanoid
[7] https://chat.openai.com/
[8] https://www.react-google-charts.com/
[9] https://www.w3schools.com/nodejs/nodejs_mysql.asp