LIBRARY MANAGEMENT SYSTEM

REPORT OF
MAIN PROJECT

BACHELOR OF COMPUTER APPLICATIONS

SUBMITTED BY
SHUBHAM MISHRA
Batch Year -2022-25
Enrollment No.-M2246020

PROJECT GUIDE -Mr. Anurag Singh

C.M.P Degree College
(Constituent of University of Allahabad,Prayagraj)

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to my department “Department of Computer Application” and my college “C.M.P Degree college” for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I would like to express my deepest gratitude to all those who have supported and contributed to the successful completion of the Library Management System Website project.

First and foremost, I extend my heartfelt thanks to my mentor “Anurag sir” for their invaluable guidance, encouragement, and insightful feedback throughout the development of this project. Their expertise and support have been instrumental in shaping the direction and success of this endeavour.

Finally, I would like to acknowledge the support of my Family and Friends, whose unwavering encouragement and understanding have been a source of motivation throughout this journey.

Thank you all for your contributions and support.

-

Thanking you,

Shubham Mishra

DECLARATION

I Shubham Mishra, a student of Bachelor of Computer Application, hereby declare that the report titled " Library Management System " submitted to my department “Department of Computer Application” is a genuine work done by me under the guidance of “Anurag Singh”.

I certify that all the information and data presented in this synopsis are true and original to the best of my knowledge. Any external sources used have been duly acknowledged.

I affirm that the content and functionalities of this website are developed with the intention of enhancing learning and promoting a deeper understanding of codes, and I take full responsibility for the integrity and accuracy of the information provided.

Date:

Certificate

This is to certify that the project titled "Library Management System" submitted by Shubham Mishra, bearing Enrollment No. M2246020, to CMP Degree College, is a record of the candidate's original work carried out under my supervision. The project has been reviewed and is considered satisfactory in terms of its objectives, execution, and compliance with academic standards.

I further certify that:

1. The project is the result of the student's own efforts and has not been submitted elsewhere for any academic credit or qualification.
2. All external sources and references used by the candidate have been duly acknowledged.
3. The work adheres to the ethical norms and academic regulations established by CMP Degree College.

Anurag Singh
(Project Supervisor)
CMP Degree College, Prayagraj

Table of Contents

Preface

In an era where information is paramount, libraries continue to serve as vital hubs for knowledge, education, and research. Managing vast collections of books, journals, and digital resources, however, demands efficiency and precision. Traditional manual systems often struggle to keep pace with the growing needs of modern libraries. This has led to the development of digital solutions that not only streamline operations but also enhance the overall experience for both administrators and users.

This project, Library Management System, is designed to simplify and automate the various activities involved in managing a library. From tracking book inventories and handling member registrations to managing issue/return operations and maintaining records, the system aims to cover all essential functions with ease and reliability. The system also introduces features like search functionality, overdue notifications, and reporting tools to improve accessibility and transparency.

The purpose of this project is to create a user-friendly, secure, and scalable platform that supports both library staff and patrons in managing and accessing resources efficiently. This work is a step towards integrating technology with traditional library services, making information management faster, more organized, and future-ready.

I sincerely hope this system will contribute positively to library administration and will continue to evolve with future needs and technological advancements

INTRODUCTION

Libraries have always been essential institutions for education, research, and personal development. As the volume of books and other resources has increased, so has the complexity of managing them efficiently. A Library Management System (LMS) is a software solution specifically designed to handle the day-to-day operations of a library in an organized and systematic manner.

The primary objective of a Library Management System is to support the management of books, journals, digital media, and user accounts with minimal manual effort. The system automates key library processes such as cataloging, book issuance, returns, membership management, and fine calculation for overdue items. It also facilitates quick searches, real-time updates, and easy record-keeping, greatly improving both administrative efficiency and user satisfaction.

By implementing a Library Management System, libraries can ensure better control over their inventories, reduce human error, and offer a more convenient experience for users. In addition, such a system provides valuable data and insights that can be used for reporting, planning, and enhancing library services. This project aims to design a flexible, reliable, and user-friendly system that meets the evolving needs of modern libraries and their communities

OBJECTIVES

🕐 **Automate Library Operations:** To automate everyday tasks like book cataloging, issuing, returning, and reserving books to minimize manual work and errors.

🕐 **Efficient Inventory Management:** To maintain an up-to-date and organized record of all books, journals, and digital materials available in the library.

🕐 **User Management:** To manage library members' registrations, profiles, borrowing history, and privileges effectively.

🕐 **Quick Information Retrieval:** To enable users and librarians to search and locate books or resources quickly through an efficient search system.

🕐 **Fine and Penalty Calculation:** To automatically track due dates and calculate fines for late returns to ensure discipline in borrowing and returning books.

🕐 **Enhanced User Experience:** To offer a user-friendly interface that provides seamless navigation for both administrators and users.

🕐 **Data Security and Integrity:** To protect sensitive user data and library records against unauthorized access and ensure data consistency.

🕐 **Reporting and Analytics:** To generate detailed reports related to issued books, returned books, fines collected, and user activities for administrative use.

🕐 **Reservation and Renewal System:** To allow users to reserve books in advance and renew borrowed items without visiting the library physically.

🕐 **Support for Digital Resources:** To manage access to digital content such as e-books, research papers, and online journals alongside physical books.

BENEFITS

🕐 **Time-Saving:** Automates repetitive tasks like issuing, returning, and cataloging, saving valuable time for both librarians and users.

🕐 **Improved Accuracy:** Reduces human errors in record-keeping, fine calculation, and inventory management.

🕐 **Easy Access to Information:** Allows quick and efficient searching for books, member details, and transaction histories.

🕐 **Better Organization:** Keeps the library collection well-organized, making it easier to track borrowed, returned, reserved, or lost items.

🕐 **Enhanced User Experience:** Provides a smooth and convenient interface for users to search, reserve, and renew books online.

🕐 **Inventory Management:** Helps libraries monitor stock levels, identify missing or damaged items, and plan acquisitions more effectively.

🕐 **Reduced Administrative Workload:** Minimizes manual paperwork and administrative burdens, allowing staff to focus on improving library services.

🕐 **Real-Time Updates:** Keeps records updated instantly whenever any transaction like issuing, returning, or reserving a book takes place.

🕐 **Cost-Effective:** Reduces the need for manual processes and large staff for management, lowering operational costs over time.

🕐 **Customizable Reports:** Generates various types of reports to help in decision-making, auditing, and future planning.

🕐 **Enhanced Security:** Protects user data and library assets with proper access control and backup features.

HARDWARE AND SOFTWARE REQUIREMENTS

This proposed software runs effectively on a computing system that has the minimum requirements. Undertaking all the equipment necessities are not satisfied but rather exist in their systems administration between the user's machines already. So, the main need is to introduce appropriate equipment for the product. The requirements are split into two categories, namely:

SOFTWARE REQUIREMENTS

The basic software requirements to run the program are:

- Windows 7
- Visual Studio Code
- HTML, CSS, JavaScript, Bootstrap
- Browser example Mozilla Firefox, Safari, Chrome, etc.

HARDWARE REQUIREMENTS

The basic hardware required to run the program are:

- Hard disk of 5 GB
- System memory (RAM) of 512 MB
- I3 processor-based computer or higher.

METHODOLOGY

1. System Requirements Analysis

In the first phase, the functional and non-functional requirements of the Library Management System were carefully gathered and analyzed. Interviews with library staff and users were conducted to understand their needs and challenges. Key requirements included user registration, book cataloging, issuing and returning functionality, search features, fine management, and report generation. System performance, security, and scalability requirements were also considered during this phase.

2. System Design

Based on the requirements, the system architecture and database design were planned. This phase included creating detailed design documents, flowcharts, entity-relationship diagrams (ERDs), and user interface mockups. The system was designed to follow a modular structure, ensuring that different components like user management, inventory management, and transaction processing are logically separated for easy development and maintenance.

3. Development

During the development phase, the actual coding of the Library Management System took place. The system was built using appropriate programming languages, frameworks, and databases according to the design specifications. Development was carried out in modules, and each module was tested individually to ensure its functionality. Agile practices were followed, allowing for iterative development and continuous feedback from initial users.

4. Implementation

Once development and module testing were completed, the system was integrated and deployed for real-world use. Data migration from old manual or semi-automated systems was performed, and thorough system testing (functional, performance, and security testing) was conducted. Training sessions were organized for library staff and users to familiarize them with the new system. Implementation included configuring the system in the live environment and monitoring its initial usage.

5. Maintenance and Support

After implementation, regular maintenance and support were provided to ensure smooth operation. This included fixing any bugs, making small enhancements based on user feedback, updating the system to remain compatible with new technologies, and ensuring data backups and security measures were consistently applied. Ongoing technical support was provided to assist users and resolve any operational issues promptly.

Testing Strategy

1. Unit Testing: Each module and function of the system (such as book issue, return, user registration, search, and fine calculation) was tested individually. Developers performed unit testing during the development phase to catch errors early and verify that each component worked as intended.

2. Integration Testing: After unit testing, modules were combined and tested together to ensure they interacted correctly. For example, testing how the book issuance module works with the user account and inventory modules. This helped identify interface errors between modules.

3. System Testing: The complete system was tested as a whole to verify that it met the specified requirements. Functional testing (verifying features work) and non-functional testing (performance, usability, security) were conducted at this stage.

4. User Acceptance Testing (UAT): Real users, including library staff and selected members, tested the system in a controlled environment. Their feedback was collected to ensure that the system was user-friendly, practical, and met the expectations set during the requirements analysis phase.

5. Performance Testing: The system was tested under different loads (e.g., multiple users searching or issuing books simultaneously) to verify its speed, responsiveness, and stability.

6. Security Testing: Security checks were conducted to protect sensitive data like user information and transaction records. This included testing for unauthorized access, data breaches, and system vulnerabilities.

Tools Used:

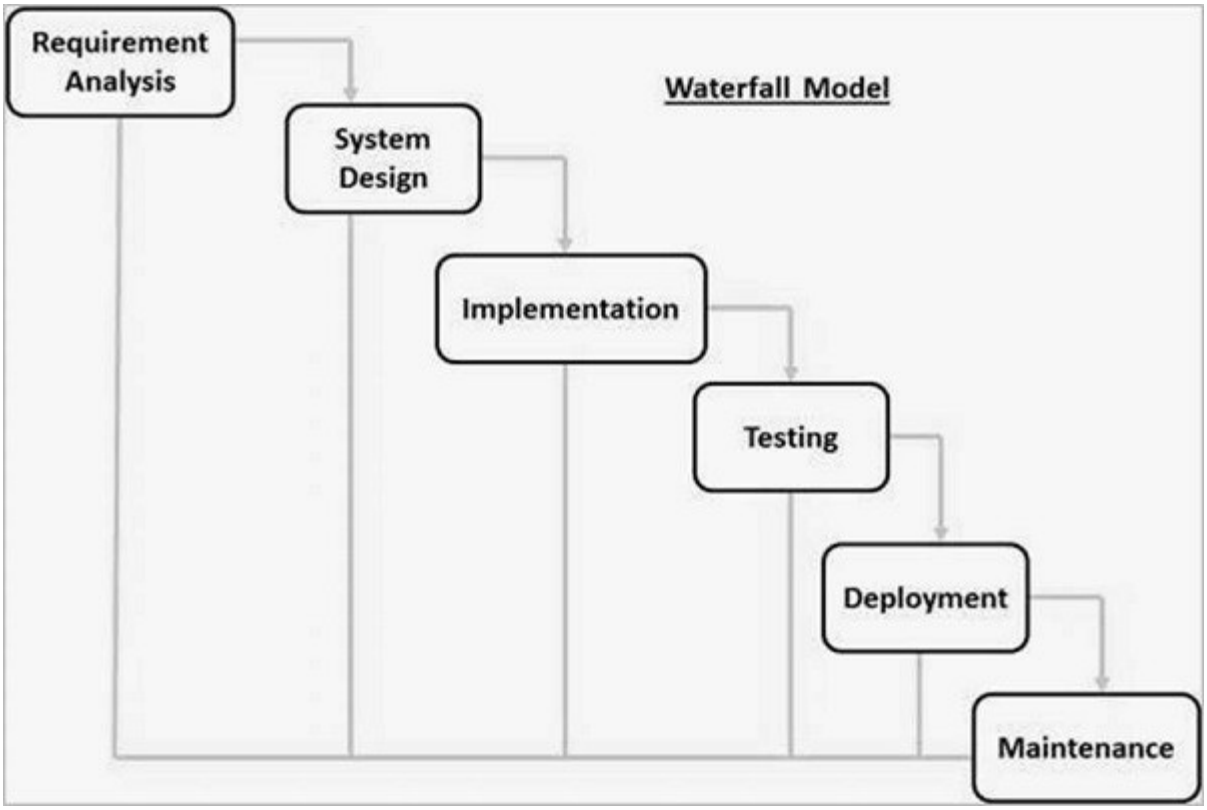
- Manual testing (for functional and UI tests)
- Database queries for backend validation
- Automated scripts (optional, if applicable, for regression testing)

Key Outcomes:

- Defects were identified and resolved early.
- The system's stability, security, and performance were validated.
- Users confirmed that the system met their needs and expectations

Software Process Model

The Waterfall Model is a linear and sequential approach to software development, where each phase must be completed before the next one begins. Here's how it can be applied to an Online Parking System Project:



1. Requirements Analysis

In this phase, all requirements of the online parking system are gathered and documented in detail. This includes:

Functional requirements: User registration and login.
Parking slot availability checking.
Slot booking and cancellation.
Admin panel for managing parking slots.

Non-functional requirements: System performance.
Scalability for increasing users.
Data security (e.g., user data).

Constraints: Operating systems, browser support, etc.

Deliverable: Software Requirement Specification (SRS) document.

2. **System Design**

Here, the architecture of the system is planned, including:

High-Level Design (HLD): Design of modules: User, Admin, Parking Slot.
Low-Level Design (LLD):
Database schema for user profiles, booking records, etc.

3. **Implementation**

In this phase, coding begins based on the design specifications:

Frontend:
Create user interface components for registration, login, booking, etc.
Backend:
Develop APIs for booking, and admin management.

Database:
Implement the database structure.
Deliverable: Fully developed system.

4. Deployment

The tested system is deployed in the production environment:

Hosting the system on a server or cloud platform.
Setting up the domain for online access.
Configuring SSL for secure communication.
Deliverable: Fully deployed and accessible online parking system.

5. Maintenance
Post-deployment, the system is monitored and updated:

Fixing bugs reported by users.
Adding new features (e.g., loyalty programs).
Ensuring compatibility with new technologies (browsers, devices).
Deliverable: Regular updates and maintenance logs.

Advantages of the Waterfall Model

Clear Structure: The Waterfall model is straightforward and easy to understand, with well-defined phases.
Easy Management: Project management is simplified due to the linear nature of the model.
Well-suited for Smaller Projects: For small projects with well-defined requirements, the Waterfall model can be efficient.
Systematic Approach: It provides a structured approach to software development, ensuring that each phase is completed before moving to the next.
Clear Milestones: Each phase has specific deliverables and deadlines, making it easier to track progress.

Disadvantages of the Waterfall Model

Rigid Approach: Once a phase is completed, it's difficult to go back and make significant changes, limiting flexibility.
Risk of Errors: Errors may not be detected until later stages, leading to costly fixes.
Limited Customer Involvement: Customer feedback is often limited to the initial requirements gathering phase.
Longer Development Time: The sequential nature of the model can lead to longer development cycles.
Less Adaptable to Change: The Waterfall model struggles to accommodate changing requirements or unforeseen challenges.

LANGUAGES USED FOR FRONT-END

HTML*: -*

HTML, or Hypertext Markup Language, is the standard language used to create and design web pages and web applications. It provides the basic structure for web content by defining elements such as headings, paragraphs, links, images, and other types of media. HTML is a cornerstone technology of the World Wide Web, alongside CSS (Cascading Style Sheets) and JavaScript.

USES OF HTML: -

- **Creating Web Page Structure:** HTML provides the fundamental structure of a web page, defining elements such as headings, paragraphs, links, images, tables, and forms. It lays out the content and its organization.

- **Embedding Media:** HTML allows for the inclusion of various media types, including images, videos, and audio. Tags like <img>, <video>, and <audio> are used to incorporate these elements.

- **SEO (Search Engine Optimization):** HTML tags such as <title>, <meta>, and semantic tags (e.g., <header>, <footer>, <article>) help in improving a website's search engine ranking by providing meaningful content structure.

CSS*: -*

CSS (Cascading Style Sheets) is a style sheet language used to control the presentation and layout of HTML elements on a web page. It separates the content of a web page (written in HTML) from its visual and aural layout, allowing developers to apply styles to multiple pages simultaneously and maintain a consistent look and feel across a website.

Uses of CSS

- **Styling Web Pages**: CSS is used to apply colors, fonts, spacing, and layout designs, enhancing the visual appeal of web pages.
- **Responsive Design**: Enables websites to adapt to different screen sizes and devices using media queries.
- **Animations and Effects**: Adds animations, transitions, and visual effects to elements for interactive user experiences.
- **Customizing UI Components**: Styles buttons, forms, menus, and other interface elements.
- **Separation of Content and Design**: Keeps HTML structure clean by handling design elements separately in a CSS file.

JavaScript (JS): -

JavaScript is a high-level, versatile programming language primarily used for creating dynamic and interactive content on websites. It is an essential technology of web development, alongside HTML and CSS. While HTML provides the structure and CSS the style, JavaScript adds behaviour and functionality, enabling web pages to respond to user interactions and provide a more engaging user experience.

Uses of JavaScript

- **Interactivity**: Adds dynamic behavior, such as toggling menus, form validation, and modals, making web pages interactive.
- **DOM Manipulation**: Modifies HTML and CSS dynamically, such as updating content without refreshing the page.
- **Event Handling**: Responds to user actions like clicks, scrolls, and key presses.
- **Asynchronous Operations**: Manages tasks like data fetching from servers using APIs or AJAX without reloading the page.
- **Game Development**: Creates simple web-based games or interactive visualizations by combining logic and graphics.

“By combining CSS, JavaScript, and React, you can create rich, interactive web applications with well-organized, maintainable code.”

LANGUAGES USED FOR BACK-END

Express.js: -

Express.js, commonly known as Express, is a fast, minimalist web application framework for Node.js. It is designed to simplify the development of server-side applications and APIs by providing a robust set of features for building web and mobile applications. Express.js is one of the most popular frameworks for Node.js and is widely used in web development due to its simplicity, flexibility, and extensive ecosystem.

MongoDB*:* -

MongoDB is a popular open-source NoSQL database designed for storing, retrieving, and managing large volumes of unstructured or semi-structured data. Unlike traditional relational databases, which use structured tables with predefined schemas, MongoDB uses a flexible, document-oriented approach to data storage. It is known for its scalability, high performance, and ease of use, making it suitable for modern applications that require dynamic and diverse data handling.

“By combining **HTML** for structure, **CSS** for styling, **JavaScript** for interactivity, **React.js** for dynamic components, and **Express.js** with **MongoDB** for backend and data management, we can build a powerful and engaging website.”

Tool Used

Visual Studio is an Integrated Development Environment(IDE) developed by Microsoft to develop Desktop applications, GUI(Graphical User Interface), console, web applications, mobile applications, cloud, and web services, etc. With the help of this IDE, you can create managed code as well as native code. It uses the various platforms of Microsoft software development software like Windows store, Microsoft Silverlight, and Windows API, etc. It is not a language-specific IDE as you can use this to write code in C#, C++, VB(Visual Basic), Python, JavaScript, and many more languages. It provides support for 36 different programming languages. It is available for Windows as well as for macOS.

Advantages Of Visual Studio:

• A full-featured programming platform for several operating systems, the web, and the cloud, Visual Studio IDE is available. Users can easily browse the UI so they can write their code quickly and precisely.

• To help developers quickly identify potential errors in the code, Visual Studio offers a robust debugging tool.

• Developers can host their application on the server with confidence because they have eliminated anything that could lead to performance issues.

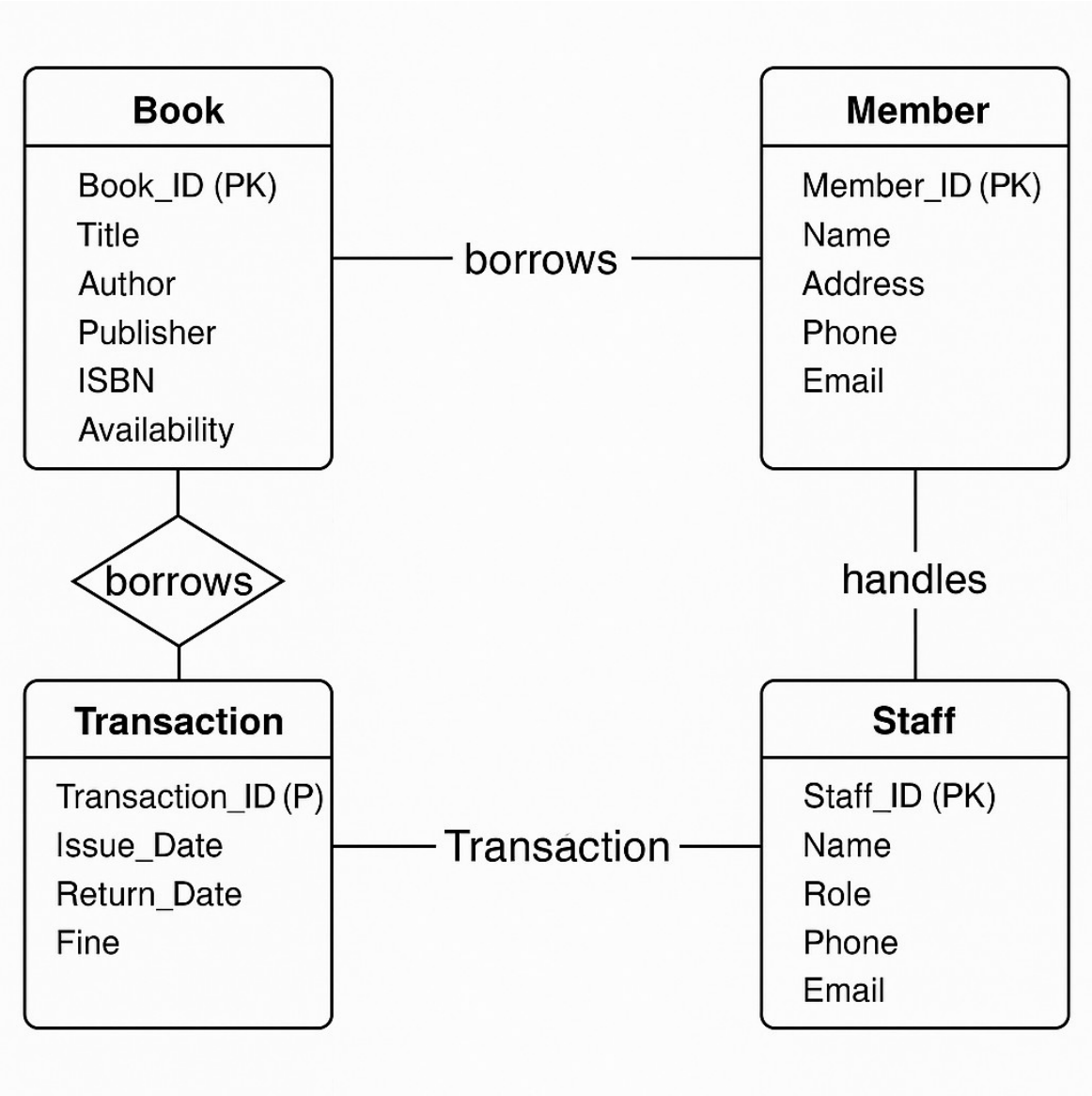
• No matter what programming language developers are using, users of Visual Studio can get live coding support. For faster development, the Platform offers an autocomplete option. The built-in intelligent system offers descriptions and tips for APIs.

• Through Visual Studio IDE you can easily collab with your teammates in a same project. This IDE helps the developers to share, push and pull their code with their teammates.

• Every user of Visual Studio has the ability to customize it. They have the option to add features based on their needs. For example, they can download add-ons and install extensions in their IDE. Even programmers can submit their own extensions.

ER DIAGRAM

-


Book
Book_ID (PK)
Title
Author
Publisher
ISBN
Availability
borrows
Member
Member_ID (PK)
Name
Address
Phone
Email
borrows
handles
Transaction
Transaction_ID (P)
Issue_Date
Return_Date
Fine
Transaction
Staff
Staff_ID (PK)
Name
Role
Phone
Email

Coding

FRONTEND

Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="A web app that manages all the functions that are done in a library"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See
https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
```

```
    work correctly both with client-side routing and a non-root public URL.
    Learn how to configure a non-root public URL by running `npm run build`.
  -->
  <title>Library Management System</title>
 </head>
 <body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
  <!--
    This HTML file is a template.
    If you open it directly in the browser, you will see an empty page.

  -->
 </body>
</html>
```

About.js

```
import React from 'react'
import './About.css'

function About() {
  return (
    <div className='about-box'>
      <h2 className="about-title">About the Library</h2>
      <div className="about-data">
        <div className="about-img">
          <img src="https://images.unsplash.com/photo-1583468982228-
19f19164aee2?
ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&ixlib=rb-1.2
.1&auto=format&fit=crop&w=913&q=80" alt="" />
```

```
            </div>
            <div>
               <p className="about-text">
                when an unknown printer took a galley of type
                  and scrambled it to make a type specimen book. It has survived not
                  only five centuries.<br/>
                  <br/>
                  Contrary to popular belief, Lorem Ipsum is not simply random text.
                  It has roots in a piece of classical Latin literature from 45 BC,
                  making it over 2000 years old. Richard McClintock, a Latin
professor
                  at Hampden-Sydney College in Virginia, looked up one of the more
obscure
                  Latin words, consectetur, from a Lorem Ipsum passage.<br/>
                  <br/>
                  There are many variations of passages of Lorem Ipsum available,
                  but the majority have suffered alteration in some form, by injected
                  humour<br/>
                  <br/>
                  Your suggestions for improvement are always welcome!
               </p>
            </div>
         </div>
      </div>
   )
}

export default About
```

Footer.js

```
import React from 'react'
import './Footer.css'

import TwitterIcon from '@material-ui/icons/Twitter';
import LinkedInIcon from '@material-ui/icons/LinkedIn';
import TelegramIcon from '@material-ui/icons/Telegram';
import InstagramIcon from '@material-ui/icons/Instagram';

function Footer() {
   return (
      <div className='footer'>
         <div>
            <div className='footer-data'>
               <div className="contact-details">
                  <h1>Contact Us</h1>
                  <p>Librarian</p>
                  <p>Government School</p>
                  <p>Visakhapatnam-530041</p>
                  <p>Andhra Pradesh</p>
                  <p>India</p>
                  <p><b>Email:</b>example@gmail.com</p>
               </div>
               <div className='usefull-links'>
                  <h1>Usefull Links</h1>
                  <a href='#home'>Link-1</a>
                  <a href='#home'>Link-1</a>
                  <a href='#home'>Link-1</a>
                  <a href='#home'>Link-1</a>
               </div>
               <div className='librarian-details'>
```

```
                <h1>Librarian</h1>
                <p>Name</p>
                <p>Education</p>
                <p>Contact: +91 9123456787</p>
            </div>
        </div>
        <div className="contact-social" >
            <a href='#home' className='social-icon'><TwitterIcon
style={{ fontSize: 40,color:"rgb(283,83,75)"}} /></a>
            <a href='#home' className='social-icon'><LinkedInIcon
style={{ fontSize: 40,color:"rgb(283,83,75)"}} /></a>
            <a href='#home' className='social-icon'><TelegramIcon
style={{ fontSize: 40,color:"rgb(283,83,75)"}} /></a>
            <a href='#home' className='social-icon'><InstagramIcon
style={{ fontSize: 40,color:"rgb(283,83,75)"}} /></a>
        </div>
    </div>
  )
}

export default Footer
```

Header.js

```
import { React, useState } from 'react'
import { Link } from 'react-router-dom'
import './Header.css'

import MenuIcon from '@material-ui/icons/Menu';
import ClearIcon from '@material-ui/icons/Clear';

function Header() {
```

```
    const [menutoggle, setMenutoggle] = useState(false)

    const Toggle = () => {
        setMenutoggle(!menutoggle)
    }

    const closeMenu = () => {
        setMenutoggle(false)
    }

    return (
        <div className="header">
            <div className="logo-nav">
            <Link to='/'>
                <a href="#home">LIBRARY</a>
            </Link>
            </div>
            <div className='nav-right'>
                <input className='search-input' type='text' placeholder='Search a
Book'/>
                <ul className={menutoggle ? "nav-options active" : "nav-options"}>
                    <li className="option" onClick={() => { closeMenu() }}>
                        <Link to='/'>
                            <a href="#home">Home</a>
                        </Link>
                    </li>
                    <li className="option" onClick={() => { closeMenu() }}>
                        <Link to='/books'>
                        <a href="#books">Books</a>
                        </Link>
```

```
            </li>
            <li className="option" onClick={() => { closeMenu() }}>
              <Link to='/signin'>
              <a href='signin'>SignIn</a>
              </Link>
            </li>
          </ul>
        </div>

        <div className="mobile-menu" onClick={() => { Toggle() }}>
          {menutoggle ? (
            <ClearIcon className="menu-icon" style={{ fontSize: 40 }} />
          ) : (
            <MenuIcon className="menu-icon" style={{ fontSize: 40 }} />
          )}
        </div>
      </div>
    )
}

export default Header
```

PopularBooks.js

```
import React from "react";
import "./PopularBooks.css";

function PopularBooks() {
  return (
    <div className="popularbooks-container">
```

```
<h className="popularbooks-title">Popular Books</h>
<div className="popularbooks">
 <div className="popularbook-images">
  <img
   className="popular-book"
   src="https://encrypted-tbn0.gstatic.com/images?
q=tbn:ANd9GcRS34iIDoKVXOhKhdwsiGSLc9RJmtq_lSQDig&usqp=CAU"
   alt=""
  ></img>
  <img
   className="popular-book"
   src="https://encrypted-tbn0.gstatic.com/images?
q=tbn:ANd9GcRfRHNwRyPkTxnMOzOvv5dOK4OS_lq4-2Yugg&usqp=CAU"
   alt=""
  ></img>
  <img
   className="popular-book"
   src="https://encrypted-tbn0.gstatic.com/images?
q=tbn:ANd9GcQ7ElcNy_v2Ri1H3VhYjYP1MzR6zBUwFQWbOirCkaqcfOqJnbr
K5ZvdZNUwEfrlmJwn7pA&usqp=CAU"
   alt=""
  ></img>
  <img
   className="popular-book"
   src="https://encrypted-tbn0.gstatic.com/images?
q=tbn:ANd9GcRS34iIDoKVXOhKhdwsiGSLc9RJmtq_lSQDig&usqp=CAU"
   alt=""
  ></img>
  <img
   className="popular-book"
   src="https://encrypted-tbn0.gstatic.com/images?
q=tbn:ANd9GcRfRHNwRyPkTxnMOzOvv5dOK4OS_lq4-2Yugg&usqp=CAU"
```

```
          alt=""
        ></img>
        <img
          className="popular-book"
          src="https://encrypted-tbn0.gstatic.com/images?
q=tbn:ANd9GcQ7ElcNy_v2Ri1H3VhYjYP1MzR6zBUwFQWbOirCkaqcfOqJnbr
K5ZvdZNUwEfrlmJwn7pA&usqp=CAU"
          alt=""
        ></img>
      </div>
      <div className="popularbook-images">
        <img
          className="popular-book"
          src="https://encrypted-tbn0.gstatic.com/images?
q=tbn:ANd9GcRS34iIDoKVXOhKhdwsiGSLc9RJmtq_lSQDig&usqp=CAU"
          alt=""
        ></img>
        <img
          className="popular-book"
          src="https://encrypted-tbn0.gstatic.com/images?
q=tbn:ANd9GcRfRHNwRyPkTxnMOzOvv5dOK4OS_lq4-2Yugg&usqp=CAU"
          alt=""
        ></img>
        <img
          className="popular-book"
          src="https://encrypted-tbn0.gstatic.com/images?
q=tbn:ANd9GcQ7ElcNy_v2Ri1H3VhYjYP1MzR6zBUwFQWbOirCkaqcfOqJnbr
K5ZvdZNUwEfrlmJwn7pA&usqp=CAU"
          alt=""
        ></img>
        <img
          className="popular-book"
```

```
          src="https://encrypted-tbn0.gstatic.com/images?
q=tbn:ANd9GcRS34iIDoKVXOhKhdwsiGSLc9RJmtq_lSQDig&usqp=CAU"
          alt=""
        ></img>
        <img
          className="popular-book"
          src="https://encrypted-tbn0.gstatic.com/images?
q=tbn:ANd9GcRfRHNwRyPkTxnMOzOvv5dOK4OS_lq4-2Yugg&usqp=CAU"
          alt=""
        ></img>
        <img
          className="popular-book"
          src="https://encrypted-tbn0.gstatic.com/images?
q=tbn:ANd9GcQ7ElcNy_v2Ri1H3VhYjYP1MzR6zBUwFQWbOirCkaqcfOqJnbr
K5ZvdZNUwEfrlmJwn7pA&usqp=CAU"
          alt=""
        ></img>
      </div>
    </div>
  </div>
 );
}

export default PopularBooks;
```

RecentAddedBooks.js

```
import React from 'react'
import './RecentAddedBooks.css'

function RecentAddedBooks() {
```

```
    return (
        <div className='recentaddedbooks-container'>
            <h className='recentbooks-title'>Recent Uploads</h>
            <div className='recentbooks'>
                <div className='images'>
                    <img className='recent-book'
src='https://inkinmytea.files.wordpress.com/2011/12/apj.jpg?w=640' alt=''></img>
                    <img className='recent-book' src='https://images-na.ssl-images-
amazon.com/images/I/91VokXkn8hL.jpg' alt=''></img>
                    <img className='recent-book' src='https://images-na.ssl-images-
amazon.com/images/I/81-QB7nDh4L.jpg' alt=''></img>
                    <img className='recent-book' src='https://images-na.ssl-images-
amazon.com/images/I/71m-MxdJ2WL.jpg' alt=''></img>
                    <img className='recent-book' src='https://images-na.ssl-images-
amazon.com/images/I/71t4GuxLCuL.jpg' alt=''></img>
                    <img className='recent-book'
src='https://19en282jw7pc3zpwj22pg8v0-wpengine.netdna-ssl.com/wp-content/
uploads/2021/01/Good-to-Great-Jim-Collins.jpg' alt=''></img>
                    <img className='recent-book' src='https://images-na.ssl-images-
amazon.com/images/I/81mXQdi5x+L.jpg' alt=''></img>
                    <img className='recent-book'
src='https://i.gr-assets.com/images/S/compressed.photo.goodreads.com/books/
1498813353l/34267304.jpg' alt=''></img>
                    <img className='recent-book'
src='https://d1csarkz8obe9u.cloudfront.net/posterpreviews/action-thriller-book-
cover-design-template-3675ae3e3ac7ee095fc793ab61b812cc_screen.jpg?
ts=1588152105' alt=''></img>
                </div>
                <div className='images'>
                    <img className='recent-book'
src='https://inkinmytea.files.wordpress.com/2011/12/apj.jpg?w=640' alt=''></img>
                    <img className='recent-book' src='https://images-na.ssl-images-
amazon.com/images/I/91VokXkn8hL.jpg' alt=''></img>
```

```
                <img className='recent-book' src='https://images-na.ssl-images-amazon.com/images/I/81-QB7nDh4L.jpg' alt=''></img>
                <img className='recent-book' src='https://images-na.ssl-images-amazon.com/images/I/71m-MxdJ2WL.jpg' alt=''></img>
                <img className='recent-book' src='https://images-na.ssl-images-amazon.com/images/I/71t4GuxLCuL.jpg' alt=''></img>
                <img className='recent-book' src='https://19en282jw7pc3zpwj22pg8v0-wpengine.netdna-ssl.com/wp-content/uploads/2021/01/Good-to-Great-Jim-Collins.jpg' alt=''></img>
                <img className='recent-book' src='https://images-na.ssl-images-amazon.com/images/I/81mXQdi5x+L.jpg' alt=''></img>
                <img className='recent-book' src='https://i.gr-assets.com/images/S/compressed.photo.goodreads.com/books/1498813353l/34267304.jpg' alt=''></img>
                <img className='recent-book' src='https://d1csarkz8obe9u.cloudfront.net/posterpreviews/action-thriller-book-cover-design-template-3675ae3e3ac7ee095fc793ab61b812cc_screen.jpg?ts=1588152105' alt=''></img>
            </div>
        </div>
    </div>
  )
}

export default RecentAddedBooks
```

ReservedBooks.js

```
import React from 'react'
import './ReservedBooks.css'

function ReservedBooks() {
```

```
return (
    <div className='reservedbooks-container'>
        <h className='reservedbooks-title'>Books On Hold</h>
        <table className='reservedbooks-table'>
            <tr>
                <th>Name</th>
                <th>Book</th>
                <th>Date</th>
            </tr>
            <tr>
                <td>Pranav</td>
                <td>Rich Dad Poor Dad</td>
                <td>12/7/2021</td>
            </tr>
            <tr>
                <td>Sashank</td>
                <td>The Subtle Art</td>
                <td>10/7/2021</td>
            </tr>
            <tr>
                <td>Tanishq</td>
                <td>Wings Of Fire</td>
                <td>15/9/2021</td>
            </tr>
            <tr>
                <td>Akhil</td>
                <td>The Secret</td>
                <td>02/9/2021</td>
            </tr>
            <tr>
```

```
                <td>Surya</td>
                <td>Bad Guys</td>
                <td>21/7/2021</td>
            </tr>
            <tr>
                <td>Dinesh</td>
                <td>Giovanni Rovelli</td>
                <td>02/7/2021</td>
            </tr>
        </table>
    </div>
  )
}

export default ReservedBooks
```

WelcomeBox.js

```
import React from 'react'
import './WelcomeBox.css'

function WelcomeBox() {
  return (
    <div className='welcome-box'>
      <p className='welcome-title'>WELCOME TO LIBRARY</p>
      <p className='welcome-message'>Feed Your Brain<br/>
      <span className='welcome-submessage'>Grab A Book To
Read</span></p>
    </div>
  )
}
```

```
export default WelcomeBox
```

News.js

```
import React from 'react'
import './News.css'

function News() {
   return (
      <div className='news-container'>
         <h className='news-title'>Updates for You</h>
         <div className='news-data'>
            <div className='news-empty'></div>
            <div>
               <h1 className='news-subtitle'>Competitions</h1>
               <div>
                  <div className='news-competition-event'>
                     <p>Competion-1</p>
                     <p>Lorem Ipsum is simply dummy text of the printing and
typesetting
                        industry.</p>
                  </div>
                  <div className='news-competition-event'>
                     <p>Competion-2</p>
                     <p>Lorem Ipsum is simply dummy text of the printing and
typesetting
                        industry.</p>
                  </div>
                  <div className='news-competition-event'>
                     <p>Competion-3</p>
```

```
                <p>Lorem Ipsum is simply dummy text of the printing and
typesetting
                    industry.</p>
                </div>
                <div className='news-competition-event'>
                <p>Competion-4</p>
                <p>Lorem Ipsum is simply dummy text of the printing and
typesetting
                    industry.</p>
                </div>
                <div className='news-competition-event'>
                <p>Competion-5</p>
                <p>Lorem Ipsum is simply dummy text of the printing and
typesetting
                    industry.</p>
                </div>
            </div>
        </div>
        <div className='news-empty'></div>
        <div>
            <h1 className='news-subtitle'>Online Quiz</h1>
            <div>
                <div className='news-quiz-event'>
                <p>Quiz-1</p>
                <p>Lorem Ipsum is simply dummy text of the printing and
typesetting
                    industry.</p>
                </div>
                <div className='news-quiz-event'>
                <p>Quiz-2</p>
                <p>Lorem Ipsum is simply dummy text of the printing and
typesetting
```

```
                    industry.</p>
                </div>
                <div className='news-quiz-event'>
                    <p>Quiz-3</p>
                    <p>Lorem Ipsum is simply dummy text of the printing and
typesetting
                    industry.</p>
                </div>
                <div className='news-quiz-event'>
                    <p>Quiz-4</p>
                    <p>Lorem Ipsum is simply dummy text of the printing and
typesetting
                    industry.</p>
                </div>
                <div className='news-quiz-event'>
                    <p>Quiz-5</p>
                    <p>Lorem Ipsum is simply dummy text of the printing and
typesetting
                    industry.</p>
                </div>
            </div>
        </div>
        <div className='news-empty'></div>
    </div>
  </div>
  )
}

export default News
```

Allbooks.js

```
import React from "react";
import "./Allbooks.css";

function Allbooks() {
  return (
    <div className="books-page">
      <div className="books">
        <div className="book-card">
          <img
            src="https://encrypted-tbn0.gstatic.com/images?
q=tbn:ANd9GcQp16xiXu1ZtTzbLy-eSwEK4Ng6cUpUZnuGbQ&usqp=CAU"
            alt=""
          ></img>
          <p className="bookcard-title">Wings Of Fire</p>
          <p className="bookcard-author">By Pranavdhar</p>
          <div className="bookcard-category">
            <p>Auto Biography</p>
          </div>
          <div className="bookcard-emptybox"></div>
        </div>
        <div className="book-card">
          <img
            src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQ-
Rb2t6jA5ml7n57qdTZbAOWX1qSfsLCbaOA&usqp=CAU"
            alt=""
          ></img>
          <p className="bookcard-title">The Power Of Your Subconscious
Mind</p>
          <p className="bookcard-author">By Joseph</p>
```

```
        <div className="bookcard-category">
          <p>Psychology</p>
        </div>
        <div className="bookcard-emptybox"></div>
      </div>
      <div className="book-card">
        <img
          src="https://encrypted-tbn0.gstatic.com/images?
q=tbn:ANd9GcRRFiDRQ7a-Oo-CnMmnbIMApP1Cq9B5bYx-UA&usqp=CAU"
          alt=""
        ></img>
        <p className="bookcard-title">Elon Musk</p>
        <p className="bookcard-author">By Elon</p>
        <div className="bookcard-category">
          <p>Auto Biography</p>
        </div>
        <div className="bookcard-emptybox"></div>
      </div>
      <div className="book-card">
        <img
          src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQ-
Rb2t6jA5ml7n57qdTZbAOWX1qSfsLCbaOA&usqp=CAU"
          alt=""
        ></img>
        <p className="bookcard-title">The Subtle Art Of Not Giving A Fuck</p>
        <p className="bookcard-author">By Mark Manson</p>
        <div className="bookcard-category">
          <p>COMIC</p>
        </div>
        <div className="bookcard-emptybox"></div>
      </div>
```

```
      </div>
    </div>
  );
}

export default Allbooks;
```

Signin.js

```
import React, { useContext, useState } from 'react'
import './Signin.css'
import axios from 'axios'
import { AuthContext } from '../Context/AuthContext.js'
import Switch from '@material-ui/core/Switch';

function Signin() {
    const [isStudent, setIsStudent] = useState(true)
    const [admissionId, setAdmissionId] = useState()
    const [employeeId,setEmployeeId] = useState()
    const [password, setPassword] = useState()
    const [error, setError] = useState("")
    const { dispatch } = useContext(AuthContext)

    const API_URL = process.env.REACT_APP_API_URL

    const loginCall = async (userCredential, dispatch) => {
        dispatch({ type: "LOGIN_START" });
        try {
            const res = await axios.post(API_URL+"api/auth/signin", userCredential);
            dispatch({ type: "LOGIN_SUCCESS", payload: res.data });
```

```
    }
    catch (err) {
      dispatch({ type: "LOGIN_FAILURE", payload: err })
      setError("Wrong Password Or Username")
    }
  }

  const handleForm = (e) => {
    e.preventDefault()
    isStudent
    ? loginCall({ admissionId, password }, dispatch)
    : loginCall({ employeeId,password }, dispatch)
  }

  return (
    <div className='signin-container'>
      <div className="signin-card">
        <form onSubmit={handleForm}>
          <h2 className="signin-title"> Log in</h2>
          <p className="line"></p>
          <div className="persontype-question">
            <p>Are you a Staff member ?</p>
            <Switch
              onChange={() => setIsStudent(!isStudent)}
              color="primary"
            />
          </div>
          <div className="error-message"><p>{error}</p></div>
          <div className="signin-fields">
            <label htmlFor={isStudent?"admissionId":"employeeId"}>
<b>{isStudent?"Admission ID":"Employee ID"}</b></label>
```

```
                    <input className='signin-textbox' type="text"
placeholder={isStudent?"Enter Admission ID":"Enter Employee ID"}
name={isStudent?"admissionId":"employeeId"} required onChange={(e) =>
{ isStudent?setAdmissionId(e.target.value):setEmployeeId(e.target.value) }}/>
                    <label htmlFor="password"><b>Password</b></label>
                    <input className='signin-textbox' type="password" minLength='6'
placeholder="Enter Password" name="psw" required onChange={(e) =>
{ setPassword(e.target.value) }} />
                    </div>
                <button className="signin-button">Log In</button>
                <a className="forget-pass" href="#home">Forgot password?</a>
            </form>
            <div className='signup-option'>
                <p className="signup-question">Don't have an account? Contact
Librarian</p>
            </div>
          </div>
      </div>
   )
}

export default Signin
```

BACKEND

Models.js

```
import mongoose from "mongoose";

const BookSchema = new mongoose.Schema({
```

```
bookName:{
    type:String,
    require:true
},
alternateTitle:{
    type:String,
    default:""
},
author:{
    type:String,
    require:true
},
language:{
    type:String,
    default:""
},
publisher:{
    type:String,
    default:""
},
bookCountAvailable:{
    type:Number,
    require:true
},
bookStatus:{
    type:String,
    default:"Available"
},
categories:[{
    type: mongoose.Types.ObjectId,
```

```
        ref: "BookCategory"
    }],
    transactions:[{
        type:mongoose.Types.ObjectId,
        ref:"BookTransaction"
    }]
},
{
    timestamps:true
})

export default mongoose.model("Book",BookSchema)
```

User.js

```
import mongoose from "mongoose";

const UserSchema = new mongoose.Schema({
    userType: {
        type: String,
        require: true
    },
    userFullName: {
        type: String,
        require: true,
        unique: true
    },
    admissionId: {
        type: String,
        min: 3,
```

```
    max: 15,
},
employeeId: {
    type: String,
    min: 3,
    max: 15,
},
age: {
    type: Number
},
gender: {
    type: String
},
dob: {
    type: String
},
address: {
    type: String,
    default: ""
},
mobileNumber: {
    type: Number,
    require: true
},
photo: {
    type: String,
    default: ""
},
email: {
    type: String,
```

```
        require: true,
        max: 50,
        unique: true
    },
    password: {
        type: String,
        require: true,
        min: 6
    },
    points: {
        type: Number,
        default: 0
    },
    activeTransactions: [{
        type: mongoose.Types.ObjectId,
        ref: "BookTransaction"
    }],
    prevTransactions: [{
        type: mongoose.Types.ObjectId,
        ref: "BookTransaction"
    }],
    isAdmin: {
        type: Boolean,
        default: false
    }
},
    {
        timestamps: true
    });
```

```
export default mongoose.model("User", UserSchema);
```

UserTransactions.js

```
import mongoose from "mongoose"

const BookTransactionSchema = new mongoose.Schema({
   bookId: {
      type: String,
      require: true
   },
   borrowerId: { //EmployeeId or AdmissionId
      type: String,
      require: true
   },
   bookName: {
      type: String,
      require: true
   },
   borrowerName: {
      type: String,
      require: true
   },
   transactionType: { //Issue or Reservation
      type: String,
      require: true,
   },
   fromDate: {
      type: String,
```

```
        require: true,
    },
    toDate: {
        type: String,
        require: true,
    },
    returnDate: {
        type: String
    },
    transactionStatus: {
        type: String,
        default: "Active"
    }
},
    {
        timestamps: true
    }
);

export default mongoose.model("BookTransaction", BookTransactionSchema)
```

-

Users.js

```
import express from "express";
import User from "../models/User.js";
```

```
const router = express.Router()

/* Getting user by id */
router.get("/getuser/:id", async (req, res) => {
    try {
        const user = await
User.findById(req.params.id).populate("activeTransactions").populate("prevTrans
actions")
        const { password, updatedAt, ...other } = user._doc;
        res.status(200).json(other);
    }
    catch (err) {
        return res.status(500).json(err);
    }
})

/* Getting all members in the library */
router.get("/allmembers", async (req,res)=>{
    try{
        const users = await
User.find({}).populate("activeTransactions").populate("prevTransactions").sort({_
id:-1})
        res.status(200).json(users)
    }
    catch(err){
        return res.status(500).json(err);
    }
})

/* Update user by id */
router.put("/updateuser/:id", async (req, res) => {
```

```
    if (req.body.userId === req.params.id || req.body.isAdmin) {
        if (req.body.password) {
            try {
                const salt = await bcrypt.genSalt(10);
                req.body.password = await bcrypt.hash(req.body.password, salt);
            } catch (err) {
                return res.status(500).json(err);
            }
        }
        try {
            const user = await User.findByIdAndUpdate(req.params.id, {
                $set: req.body,
            });
            res.status(200).json("Account has been updated");
        } catch (err) {
            return res.status(500).json(err);
        }
    }
    else {
        return res.status(403).json("You can update only your account!");
    }
})

/* Adding transaction to active transactions list */
router.put("/:id/move-to-activetransactions" , async (req,res)=>{
    if(req.body.isAdmin){
        try{
            const user = await User.findById(req.body.userId);
            await user.updateOne({$push:{activeTransactions:req.params.id}})
            res.status(200).json("Added to Active Transaction")
```

```
        }
        catch(err){
            res.status(500).json(err)
        }
    }
    else{
        res.status(403).json("Only Admin can add a transaction")
    }
})

/* Adding transaction to previous transactions list and removing from active
transactions list */
router.put("/:id/move-to-prevtransactions", async (req,res)=>{
    if(req.body.isAdmin){
        try{
            const user = await User.findById(req.body.userId);
            await user.updateOne({$pull:{activeTransactions:req.params.id}})
            await user.updateOne({$push:{prevTransactions:req.params.id}})
            res.status(200).json("Added to Prev transaction Transaction")
        }
        catch(err){
            res.status(500).json(err)
        }
    }
    else{
        res.status(403).json("Only Admin can do this")
    }
})

/* Delete user by id */
router.delete("/deleteuser/:id", async (req, res) => {
```

```
    if (req.body.userId === req.params.id || req.body.isAdmin) {
        try {
            await User.findByIdAndDelete(req.params.id);
            res.status(200).json("Account has been deleted");
        } catch (err) {
            return res.status(500).json(err);
        }
    } else {
        return res.status(403).json("You can delete only your account!");
    }
})

export default router
```

transactions.js

```
import express from "express"
import Book from "../models/Book.js"
import BookTransaction from "../models/BookTransaction.js"

const router = express.Router()

router.post("/add-transaction", async (req, res) => {
    try {
        if (req.body.isAdmin === true) {
            const newtransaction = await new BookTransaction({
                bookId: req.body.bookId,
                borrowerId: req.body.borrowerId,
                bookName: req.body.bookName,
                borrowerName: req.body.borrowerName,
```

```
            transactionType: req.body.transactionType,
            fromDate: req.body.fromDate,
            toDate: req.body.toDate
        })
        const transaction = await newtransaction.save()
        const book = Book.findById(req.body.bookId)
        await book.updateOne({ $push: { transactions: transaction._id } })
        res.status(200).json(transaction)
    }
    else if (req.body.isAdmin === false) {
        res.status(500).json("You are not allowed to add a Transaction")
    }
  }
  catch (err) {
    res.status(504).json(err)
  }
})

router.get("/all-transactions", async (req, res) => {
  try {
    const transactions = await BookTransaction.find({}).sort({ _id: -1 })
    res.status(200).json(transactions)
  }
  catch (err) {
    return res.status(504).json(err)
  }
})

router.put("/update-transaction/:id", async (req, res) => {
  try {
```

```
        if (req.body.isAdmin) {
            await BookTransaction.findByIdAndUpdate(req.params.id, {
                $set: req.body,
            });
            res.status(200).json("Transaction details updated successfully");
        }
    }
    catch (err) {
        res.status(504).json(err)
    }
})

router.delete("/remove-transaction/:id", async (req, res) => {
    if (req.body.isAdmin) {
        try {
            const data = await BookTransaction.findByIdAndDelete(req.params.id);
            const book = Book.findById(data.bookId)
            console.log(book)
            await book.updateOne({ $pull: { transactions: req.params.id } })
            res.status(200).json("Transaction deleted successfully");
        } catch (err) {
            return res.status(504).json(err);
        }
    } else {
        return res.status(403).json("You dont have permission to delete a book!");
    }
})

export default router
```

categories.js

```
import express from "express";
import BookCategory from "../models/BookCategory.js";

const router = express.Router();

router.get("/allcategories", async (req, res) => {
  try {
    const categories = await BookCategory.find({});
    res.status(200).json(categories);
  } catch (err) {
    return res.status(504).json(err);
  }
});

router.post("/addcategory", async (req, res) => {
  try {
    const newcategory = await new BookCategory({
      categoryName: req.body.categoryName,
    });
    const category = await newcategory.save();
    res.status(200).json(category);
  } catch (err) {
    return res.status(504).json(err);
  }
});

export default router;
```

books.js

```
import express from "express"
import Book from "../models/Book.js"
import BookCategory from "../models/BookCategory.js"

const router = express.Router()

/* Get all books in the db */
router.get("/allbooks", async (req, res) => {
   try {
      const books = await Book.find({}).populate("transactions").sort({ _id: -1 })
      res.status(200).json(books)
   }
   catch (err) {
      return res.status(504).json(err);
   }
})

/* Get Book by book Id */
router.get("/getbook/:id", async (req, res) => {
   try {
      const book = await Book.findById(req.params.id).populate("transactions")
      res.status(200).json(book)
   }
   catch {
      return res.status(500).json(err)
   }
```

```
})

/* Get books by category name*/
router.get("/", async (req, res) => {
    const category = req.query.category
    try {
        const books = await BookCategory.findOne({ categoryName:
category }).populate("books")
        res.status(200).json(books)
    }
    catch (err) {
        return res.status(504).json(err)
    }
})

/* Adding book */
router.post("/addbook", async (req, res) => {
    if (req.body.isAdmin) {
        try {
            const newbook = await new Book({
                bookName: req.body.bookName,
                alternateTitle: req.body.alternateTitle,
                author: req.body.author,
                bookCountAvailable: req.body.bookCountAvailable,
                language: req.body.language,
                publisher: req.body.publisher,
                bookStatus: req.body.bookSatus,
                categories: req.body.categories
            })
            const book = await newbook.save()
```

```
            await BookCategory.updateMany({ '_id': book.categories }, { $push:
{ books: book._id } });
            res.status(200).json(book)
        }
        catch (err) {
            res.status(504).json(err)
        }
    }
    else {
        return res.status(403).json("You dont have permission to delete a book!");
    }
})

/* Addding book */
router.put("/updatebook/:id", async (req, res) => {
    if (req.body.isAdmin) {
        try {
            await Book.findByIdAndUpdate(req.params.id, {
                $set: req.body,
            });
            res.status(200).json("Book details updated successfully");
        }
        catch (err) {
            res.status(504).json(err);
        }
    }
    else {
        return res.status(403).json("You dont have permission to delete a book!");
    }
})
```

```
/* Remove book  */
router.delete("/removebook/:id", async (req, res) => {
   if (req.body.isAdmin) {
      try {
         const _id = req.params.id
         const book = await Book.findOne({ _id })
         await book.remove()
         await BookCategory.updateMany({ '_id': book.categories }, { $pull:
{ books: book._id } });
         res.status(200).json("Book has been deleted");
      } catch (err) {
         return res.status(504).json(err);
      }
   } else {
      return res.status(403).json("You dont have permission to delete a book!");
   }
})

export default router
```

MILESTONE

MEETING WITH THE SUPERVISOR

Date of the meet	Mode	Comments by the supervisor	Signature of the Supervisor

Bibliography

Books

1. "Smart Parking Systems" by S. S. Iyengar et al., published by Springer, 2019.
2. "Online Parking Management Systems" by A. K. Singh et al., published by CRC Press, 2018.

3. "Parking Management Systems" by R. L. Cheu et al., published by Wiley, 2017.

Online Resources

1. "Online Parking System" by GitHub, 2020.
2. "Smart Parking System" by ResearchGate, 2019.
3. "Parking Management System" by ScienceDirect, 2018.