

COLLEGE LIBRARY MANAGEMENT SYSTEM

A Project Report

Submitted in partial fulfilment of the

Requirements for the award of the degree of

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

By

**Mohammed Kaif Shahid Ali 1066583 & Gupta Vikas Sugriv
1066558**

Under the esteemed guidance of

Mr. Sohil Luhar



DEPARTMENT OF INFORMATION TECHNOLOGY

SIDDHARTH COLLEGE OF COMMERCE & ECONOMICS, (FORT)

CSMIT (Affiliated to University of Mumbai) MUMBAI 400001

MAHARASHTRA

2023-2024

PROFORMA FOR THE APPROVAL PROJECT PROPOSAL

(Note: All entries of the proforma of approval should be filled up with appropriate and complete information. Incomplete proforma of approval in any respect will be summarily rejected.)

PNR No.:

Roll no: _____

PNR No.:

Roll no: _____

1. Name of the Student

2. Title of the Project

3. Name of the Guide

4. Teaching experience of the Guide _____

5. Is this your first submission?

Yes

☐

No

☐

Signature of the Student

Signature of the Guide

Date:

Date:

ABSTRACT

Online Library Management System is a system which maintains the information about the books present in the library, their authors, the members of library to whom books are issued, library staff and all.

This is very difficult to organize manually. Maintenance of all this information manually is a very complex task. Owing to the advancement of technology, organization of an Online Library becomes much simple. The Online Library Management has been designed to computerize and automate the operations performed over the information about the members, book issues and returns and all other operations.

This computerization of library helps in many instances of its maintenances. It reduces the workload of management as most of the manual work done is reduced.

ACKNOWLEDGEMENT

We take this occasion to thank God, almighty for blessing us with his grace and taking our endeavour to a successful culmination. We extend our sincere and heartfelt thanks to our esteemed guide, **SOHIL LUHAR**, for providing us with the right guidance and advice at the crucial junctures and for showing me the right way. We would like to thank our friends and family for the support and encouragement they have given us during the course of our work.

Table of Contents

Chapter 1: Introduction	6
1.1 Background.....	6
1.2 Objective.....	7
1.3 Purpose, Scope and Applicability.....	8
1.3.1 Purpose.....	8
1.3.2 Scope.....	8
1.3.3 Applicability.....	8
1.4 Achievement.....	9
1.5 Organization & Report.....	10
Chapter 2: System Analysis.....	11
2.1 Existing System.....	11
2.2 Proposed System.....	11
2.3 Requirement Analysis.....	12
2.4 Hardware Requirement.....	15
2.5 Software Requirement.....	16
2.6 Justification of selection of Technology.....	19
Chapter 3 System Design.....	20
3.1 Module Division	20
3.2 Data Dictionary	21
3.3 ER Diagram	22

3.4 Data Flow Diagram (level 0)	24
3.5 Flow Chart Diagram.....	25
3.6 Activity Diagram.....	28
3.7 Use Case Diagrams.....	31
3.8 Schema Diagram.....	32
Chapter 4 Implementation and Testing	33
4.1 Code	33
4.2 Testing Approach	59
4.2.1 Unit Testing	60
4.2.2 Integration System	65
4.3 Security Issues.....	66
4.4 Implementation and Approach.....	68
Chapter 5 Results and Discussion	69
5.1 Test reports	69
5.2 User documentation.....	71
Chapter 6 Conclusion and Future work.....	76
6.1 Conclusion	76
6.2 Signification of the System.....	77
6.3 Limitations of the System	78
6.4 Future Scope.....	79

CHAPTER 1: INTRODUCTION

1. INTRODUCTION:

A library management system is used to maintain library records. It tracks the records of the number of books in the library, how many books are issued, or how many books have been returned or renewed or late fine charges, etc. A digital library, also known as an e-library, is a collection of documents in an organized digital form, available on the internet or on disks.

The purpose of an e-library is to store, access, and manage magazine articles, books, audio files, images, and video files.

1.1. BACKGROUND:

E-Library Management System is an application which refers to library systems which are generally small or medium in size is used by librarian to manage the library using a computerized system where he/she can add new books and Page sources.

Books and student maintenance modules are also included in this system, which would keep track of the students using the library and also a detail description about the books a library contains. With this computerized system there will be no loss of book record or member record which generally happens when a non-computerized system is used.

All these modules are able to help librarians to manage the library with more convenience and in a more efficient way as compared to library systems which are not computerized.

The System allows librarians to maintain organized database of books Journals and track, return date of book and Various facilitate.

The development of technology and computerization revolutionized library management, leading to the emergence of library management systems. These systems were designed to automate various tasks and streamline library operations, making it easier for librarians to manage resources and serve library users effectively. Here are some key factors that contributed to the background and evolution of college library management systems.

1.2 OBJECTIVES:

- **The aims objectives are as follows:**

- i) Online book adds.
 - ii) A search column to search availability books.
 - iii) An admin login pages where admin can add books, update.
- **Efficient book management to automate the processes of cataloguing, Circulation, track, of book add book, send message, search with the making help of book ID It easierto manage library is collection.**
 - 1) User friendly interface to provide that allows Students, faculty, and staff to easily provide Search, borrow and return books.
 - 2) Inventory controls its maintain accurate inventory records, ensuring that books are available when need and reducing instances of misplaced or lost books.
 - 3) Security and privacy to to implement Security measure to protect user data and library resources ensuring privacy and data Confidentiality.
 - 4) Support for different Devices to ensures Compatibility with various devices, such as Smartphones and tablet so that user can access E-library services on the go.
 - 5) Continuous improvement to regularly update andimprove the System based on user feedback andchanging need.
 - 6) Automation to various tasks like books checkouts, renewal, and reminders, reducing manual workload forlibrary staff.

1.3 PURPOSE SCOPE AND APPLICABILITY:

1.3.1 PURPOSE:

Purpose of a college library management system is to efficiently manage or organize the resources, service, and operations within a college library. Its scope includes cataloging book, managing borrowing, and returning processes, monitoring library inventory, providing search and retrieval capabilities generating reports.

1.3.2 SCOPE:

Scope of a college library management system typically includes tasks like managing book, inventory, add book, records of books, check in/check-out, reminder, cataloging and report of book. It may also involve integrating with the college student information to system providing user friendly interface for both librarian and students.

1.3.3 APPLICABILITY:

The applicability of the system is relevant for college and other educational Institutions. with libraries as it enhances the overall library experience for the students, faculty and staff Making it easier access and utilize the available resource effectively.

1.4 ACHIEVEMENTS:

1. Improved user Experience.
2. Enhancing the system's user interface and features to provide a more intuitive and efficient experience for students and faculty.
3. Digitalization of resources and organizing library's resources, making it easier for user to access books online.
4. Advanced Search and recommendation.
5. Efficient Borrowing and return process reducing waiting times and automatic Overdue reminders.
5. Analytics and reporting developing tools for analytics generating usage reports library Staff decisions about to help Light to Make data-driven
6. Enhanced Security and to measure protect over Privacy library access resources and of from unauthorized.
7. Increased Collection Size: Expanding the library's collection by adding new books, journals, e-books, and other resources relevant to the academic needs and interests of students and faculty.
8. Technological Upgrades: Regularly updating library systems and infrastructure to provide a seamless and user-friendly experience for patrons.

1.5 ORGANIZATION AND REPORT:

This project consists of (6) chapters:

Chapter 1 is the introduction to the project. The background, objectives, purpose, scope, applicability, and achievements of the project are explained in detail in this chapter.

Chapter 2 is Survey of Technologies which includes system analysis and existing system and Without This System How Present Work Is Going and Disadvantages of Present Time Without This System. All the relevant journal, thesis and books taken from that research will be discussed in detail.

Chapter 3 is requirements and analysis; this chapter reveals the problem definition, requirements specification and planning and scheduling, software and hardware requirements and conceptual models. And besides this, it will also discuss about the process flow in detail of this research.

Chapter 4 is system design it consists of the Gantt chart and the processing and working of the development, it has data structures, algorithm design and security issues and test case design.

Chapter 5 is implementation and testing. This chapter documents all the process that involved in developing this system and the testing made the system.

Chapter 6 is result and discussion it means the result or the output we get from the project and the discussion that is can be executed in the real time.

CHAPTER 2: SYSTEM ANALYSIS

2.1 EXISTING SYSTEM:

- i) If one is not very careful then there is a possibility of issuing more than one book to a user.
- ii) There is a possibility of issuing a book to a user, whose membership is not there.
- iii) When a user requests for a book, one has to physically check for the presence of a book in the library.
- iv) Answering management query is a time-consuming process.
- v) Daily keeping a manual record of changes taking place in the library such as book being issued, book being returned etc can become cumbersome if the library size is bigger.

2.2 PURPOSED SYSTEM:

The **LIBRARY MANAGEMENT SYSTEM** is a software application which avoids more manual hours in taking the book, that need to spend in record keeping and generating reports. Maintaining of user details is complex in manual system in terms of agreements, royalty, and activities. This all must be maintained in ledgers or books. Co-coordinators needs to verify each record for small information also.

- 1.Easy search of book in the online library.
- 2.Avoid the manual work.
- 3.User need not go to the library for Issue any kind of book, he can renewal the book online.
- 4. Admin can easily add the book through online and they can update delete and change the book and they can check the records of book online borrow and return dare of user and maintain library clean.
- 5. The main goal of e-library the user gets the notification of the return book on time.

2.3 REQUIREMENT ANALYSIS:

1. NON-FUNCTIONAL REQUIREMENTS:

i) EFFICIENCY REQUIREMENT

When a library management system will be implemented librarian and user will easily access the library as searching and book transaction will be very faster.

ii) RELIABILITY REQUIREMENT

The system should accurately perform member registration, member validation, report generation, book transaction and search.

iii) USABILITY REQUIREMENT

The system is designed for a user-friendly environment so that students and staff of library can perform the various tasks easily and in an effective way.

iv) IMPLEMENTATION REQUIREMENT

In implementing the whole system, it uses node.js in front end with react.js as server-side scripting language which will be used for database connectivity and the backend the database part is developed using MongoDB no- MySQL.

v) DELIVERY REQUIREMENT

The whole system is expected to be delivered in a month of time evaluation by the project guide.

2.FUNCTIONAL REQUIREMENTS:

1.NORMAL USER: -

1.1 USER LOGIN

Description of feature: -This feature used by the user to login into system. They are required to enter user id and password before they are allowed to enter the system. The user id and password will be verified and if invalid id is their user is allowed to not enter the system.

• Functional requirements: -

- User id is provided when they register
- The system must only allow user with valid id and password to enter the system
- The system performs authorization process which decides what user level can access to.
- The user must be able to logout after they finished using system.

1.2 REGISTER NEW USER: -

This feature can be performed by all users to register new user to create account.

• Functional requirements:

- System must be able to verify information
- System must be able to delete information if information is wrong

1.3 REGISTER NEW BOOK: -

This feature allows to add new books to the library

• Functional requirements:

- System must be able to verify information
- System must be able to enter number of copies into table.
- System must be able to not allow two books having same book id.

1.5 SEARCH BOOK: -

This feature is found in book maintenance part. we can search book based on book id, book name, publication or by author name.

• Functional requirements:

- System must be able to search the database based on select search type
- System must be able to filter book based on keyword entered
- System must be able to show the filtered book in table view
- System should be able to add detailed information about events.
- System should be able to display information on notice board available in the homepage of site

1.6 Borrowing and Returning: -

- The system should allow users to check the availability of a book and reserve it if it's currently unavailable.
- Users should be able to borrow books and specify the expected return dates.
- Librarians should have the ability to process book checkouts and returns, including fine calculation for overdue books.
- Notifications should be sent to users regarding upcoming due dates and overdue books.

2.4 HARDWARE REQUIREMENTS:

- 2 GHz dual-core processor or better.
- 1-4 GB system memory.
- 25 GB of free hard drive space.
- Internet access is helpful.
- Either a DVD drive or a USB port for the installer media.
- Intel core i5 2nd generation is used as a processor because it is fast than other processors and provide reliable and stable and we can run our pc for long time.
- By using this processor, we can keep on developing our project without any worries.
- Ram 1 GB is used as it will provide fast reading and writing capabilities and will in turn support in processing.

2.5 SOFTWARE REQUIREMENTS:

➤ Operating system:

Microsoft Windows (also defied to as Windows ó Win)is a graphical operating system developed and publishedby Microsoft. It provides a way to store files, in software, play games, watch videos, and connect to the Internet. Microsoft Windows was first introduced with version 1.0 on November 10, 1983.

Windows operating system is one of the most popular computer operating systems in the world. The primary reason behind its global popularity is its very user-friendly interface. Windows operating system also provides multitasking capabilities, virtual memory management, etc.

➤ Database:

Mongo dB NoSQL databases have flexible data models,scale horizontally, have incredibly fast queries, and areeasy for developers to work with. NoSQL databases typically have very flexible schemas. A flexible schema allows you to easily make changes to your database as requirements change.

➤ Programming language:

JavaScript is used to write the whole code and develop webpages with CSS, Html for styling work. Fronted we usea react.js and backend we use node.js they are easy to use.

Node.js, react.js, JavaScript, Tailwind CSS, Express. Js.

➤ API:

Telegram: - Telegram The official Telegram Bot API documentation provides detailed information on how to use various API methods, set up bots, and handle interactions.

The Telegram API and TDLib allow you to build your own customized Telegram clients. You are welcome to use both APIs free of charge. You can also add Telegram Widgets to your website. Designers are welcome to create Animated Stickers or Custom Themes for Telegram.

➤ **VERSION CONTROL:**

GitHub: - GitHub, Inc. is a platform and cloud-based service for software development and version control using Git, allowing developers to store and manage their code.

GitHub provides distributed version controls geared toward tracking and managing changes to software code. In line with this, several developers can work on a Git repository and track changes along the way. Further, every team member can access the database simultaneously to view previous versions.

A version control system, or VCS, tracks the history of changes as people and teams collaborate on projects together. As developers make changes to the project, any earlier version of the project can be recovered at any time.

➤ **SECURITY:**

Secure Sockets Layer (SSL): - Is a protocol that provides secure communication over the Internet. It uses both symmetric and asymmetric cryptography. The SSL protocol provides server authentication and client authentication: Server authentication is performed when a client connects to the server.

SSL provides a secure channel between two machines or devices

operating over the internet or an internal network. One common example is when SSL is used to secure communication between a web browser and a web server. This turns a website's address from HTTP to HTTPS, the 'S' standing for 'secure'.

➤ **NOTIFICATION AND REMINDER:**

Notifications for due dates, holds, and other relevant information. Using Telegram messaging Bot.

Email: - for email going to use JavaScript module name Nodemailer.js it allows as to send the send email to user. Node mailer is a module for Node.js applications to allow easy as cake email sending. The project got started back in 2010 when there was no sane option to send email messages, today it is the solution most Node.js users turn to by default.

An email notification is an email a business sends to inform subscribers about updates or changes to a service or website. For example, it could be an update about a feature update, information about a user account, billing or charge confirmation, or any regular updates.

2.6 JUSTIFICATION OF SELECTION OF TECHNOLOGY:

- i) We have seen many Programming Languages like **Java, Flutter, XML, Node.js, Apache Cordova, Ionic 5, Firebase, Django, React.js, Python, and PHP, etc.**
- ii) In our COLLEGE LIBRARY MANAGEMENT SYSTEM we are using JAVA-SCRIPT and React.JS for Front-end.
- iii) For Back-end we have selected Node.js.
- iv) For Database we are using MongoDB

CHAPTER 3: SYSTEM DESIGN

3.1 MODULE DIVISION:

This module division help us to divide the overall drawback into units and develop module individually. In this topic we have detail of the project module and functionally of the module using Figures.

➤ Login Page:

- A login page is an entry page in an e-library website that requires user identification and authentication.
- In a login page by entering a email address and password combination is used.

➤ Home Page:

- A Home Page is generally the primary page which visitor navigating it.
- A Home Page is the default front page of a site.

➤ BOOKS:

- A admin can add update and delete the book at any time.
- Only admin can do this
- Admin add book any time as user requirements.

➤ E-BOOKS:

- The user can read the book online without reach a college.
- The user saves the time and energy.
- The all types of books available online like magazine, e-books, newspapers, etc...

➤ Notifications:

- The user get notification when they have returned the book to college on time.
- With the help of E-mail.

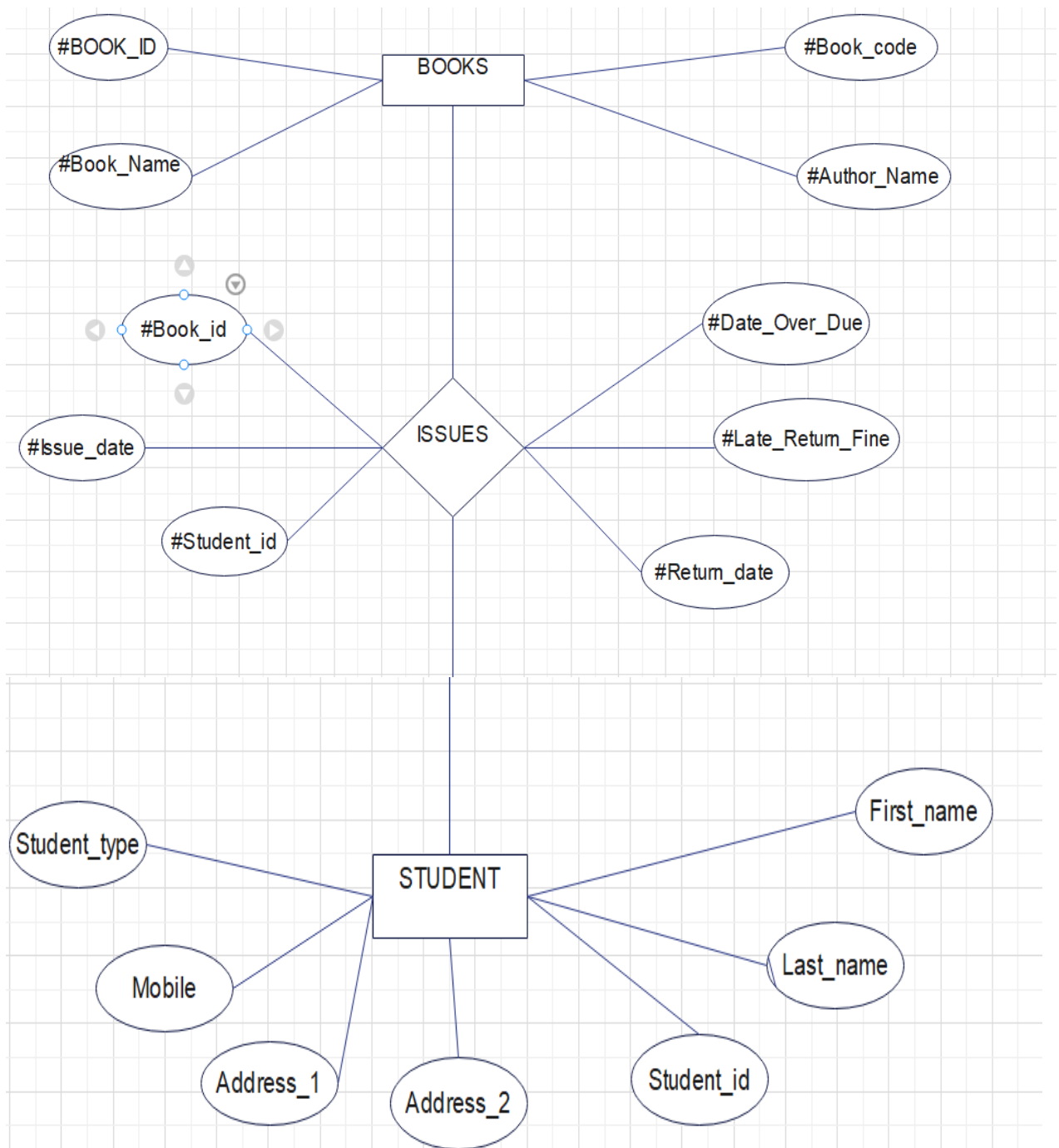
3.2 DATA DICTIONARY:

Credentials:

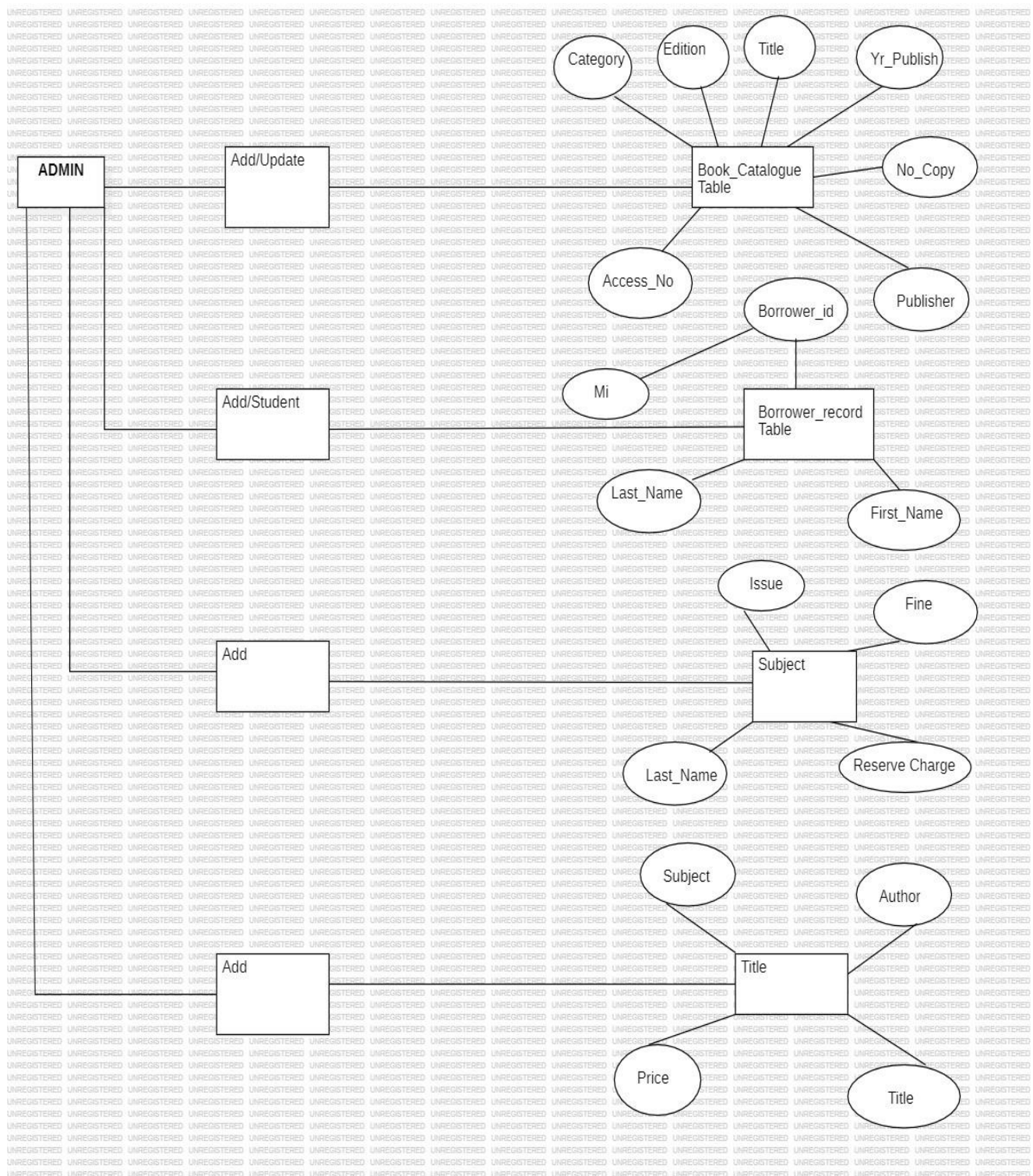
- **User Collection:** It consists of email and password.
- **Email:** E-mail should be valid.
- **Password:** For authorizing the user
- **Admin:** The username and password are automatically defined by system to admin for login.
- **Book id:** The admin gives a book id to find the book easily by users.
- **E-Book:** Unique Key to identify each group of books.

3.3 ENTITY RELATIONSHIP – (ER) DIAGRAM:

- ER model stands for and Entity- Relationship Model.
- This model is used to define the data elements and relationships for a specified system.

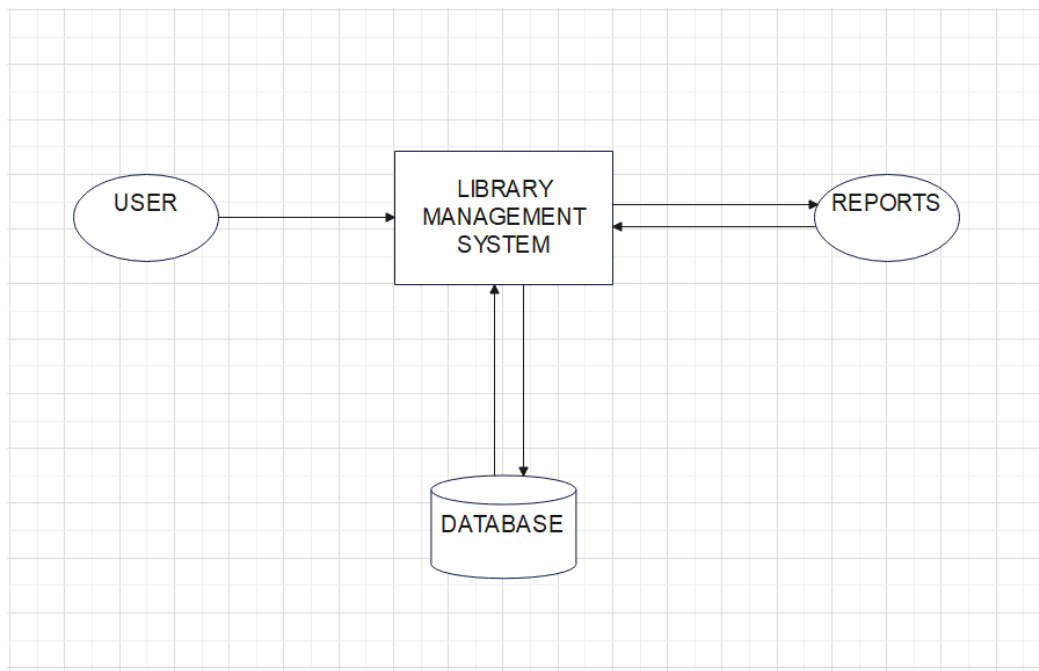


ER DIAGRAM FOR ADMIN FOR BOOK:



3.4 DATA FLOW DIAGRAM (DFD):

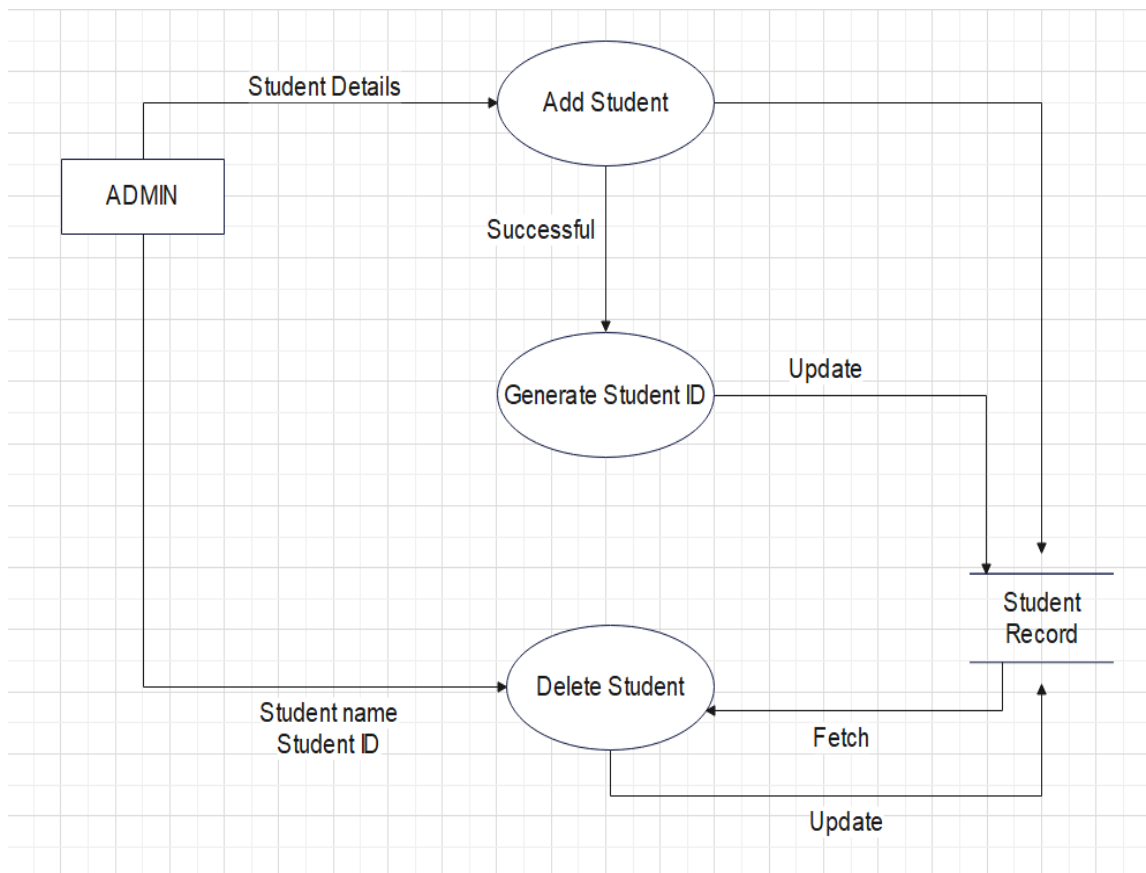
- It shows the flow of data between various functions of system and specifies how the current system is implemented.
- Data Flow Diagram (DFD) is easy to understand and quite effective when the required design is not clear, and the user wants a notational language for communication.



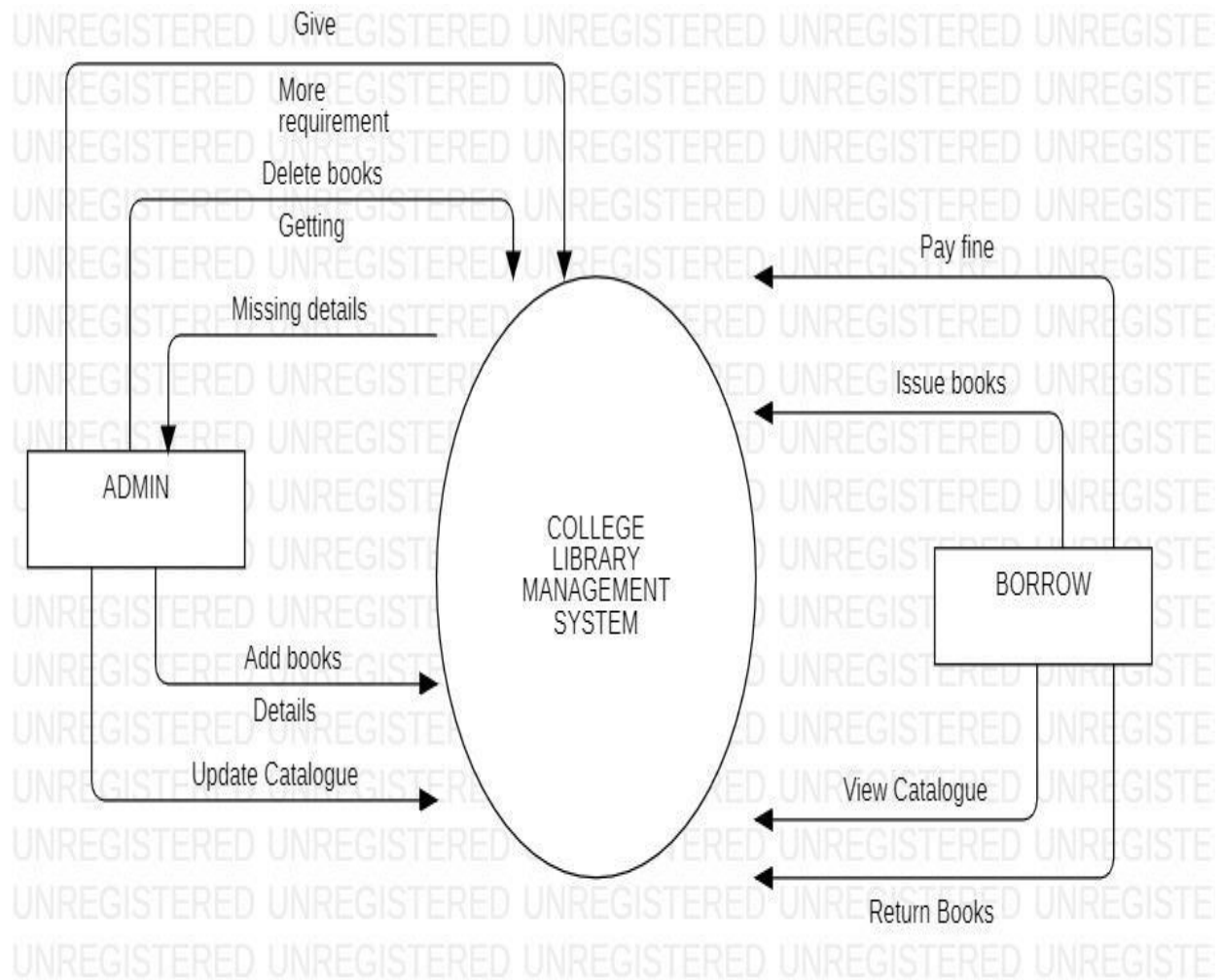
3.5 FLOW CHART:

- A computer flowchart is a type of diagram that represents a computer algorithm or process.
- A computer flowchart can help to visualize the logic, structure, and steps of a program or system.

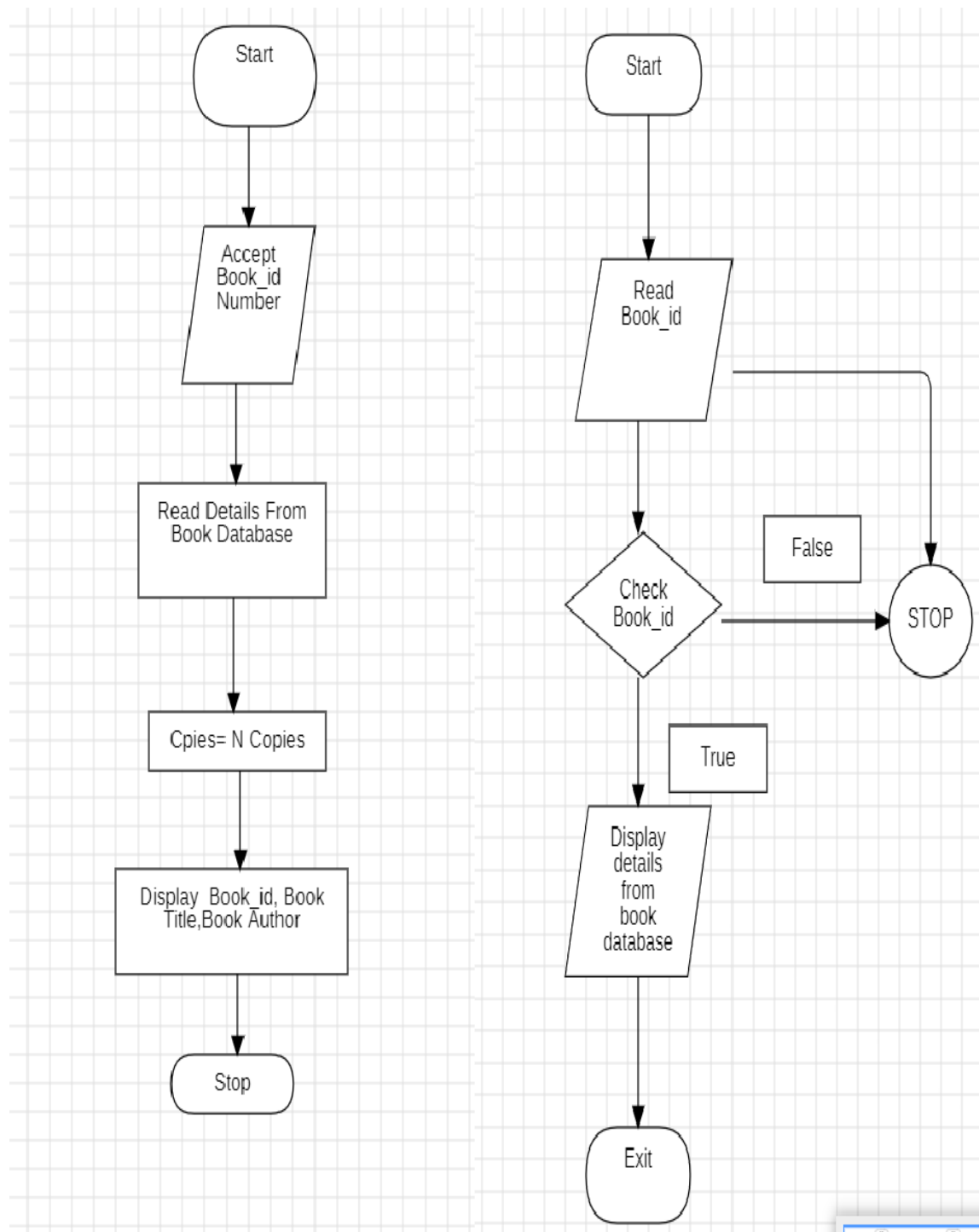
➤ **FLOW CHART FOR STUDENT CREATION FROM LIBRARY MANAGEMENT SYSTEM.**



➤ **FLOW CHART 0-LEVEL (CONTENT LEVEL):**

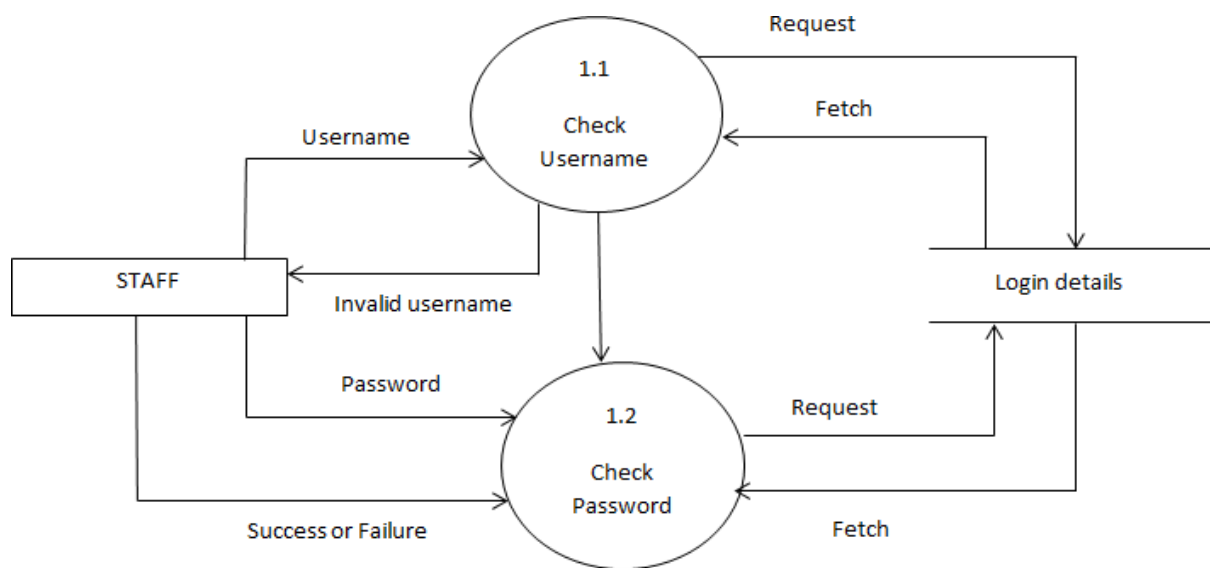


➤ **FLOW CHART FOR BOOK ENTRY PROGRAM:**

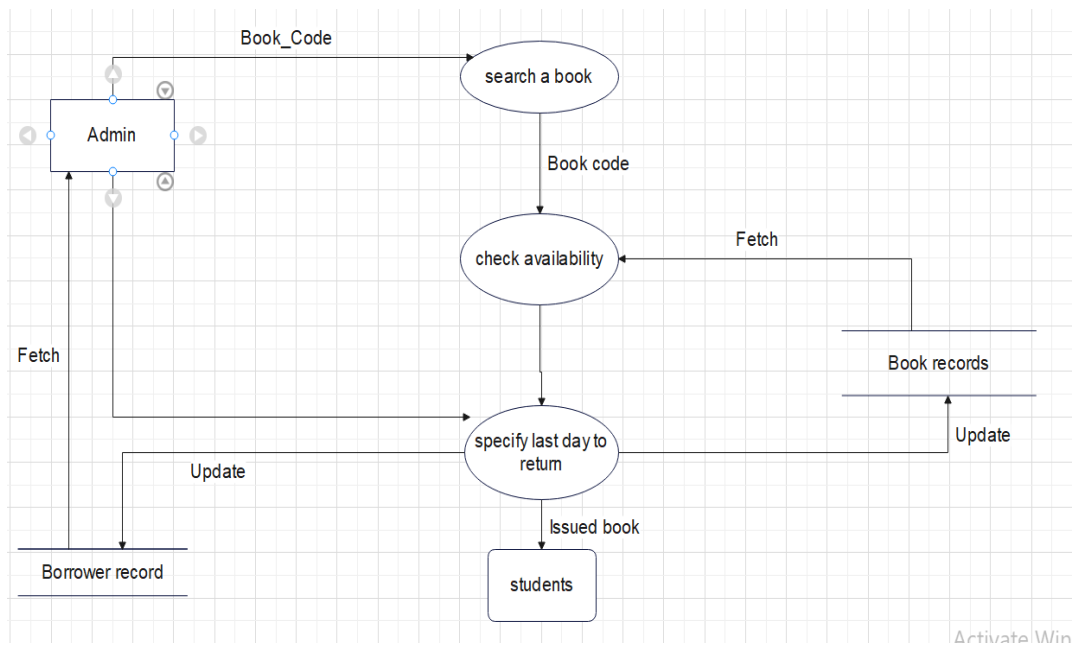


3.6 ACTIVITY DIAGRAM:

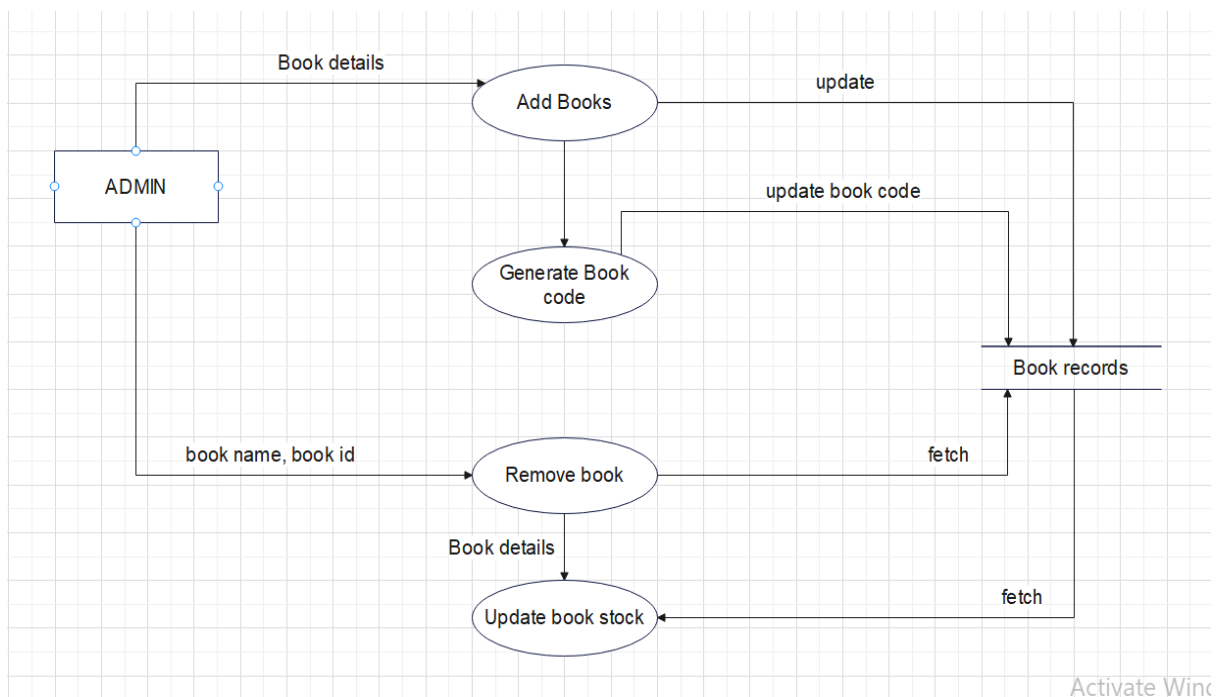
- Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.
- Activity diagram is basically a flowchart to represent the flow from one activity to another activity.
- **AUTHETICATION FOR VALID-USERNAME AND PASSWORD:**



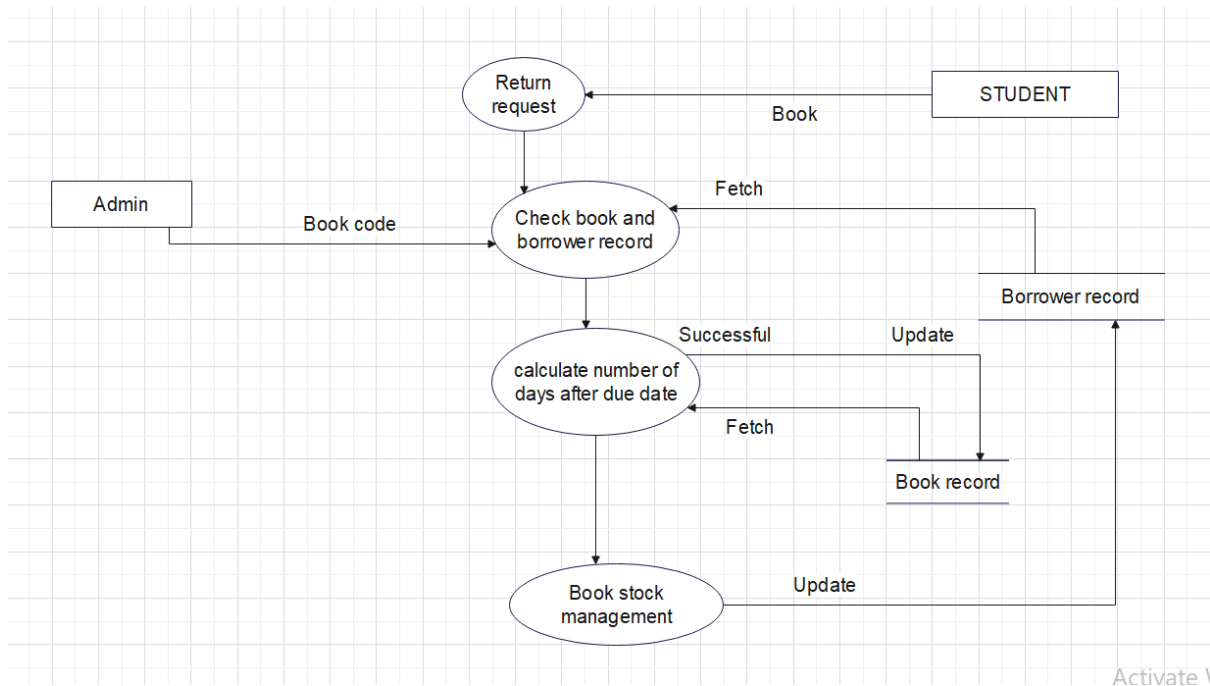
➤ ISSUES BOOK:



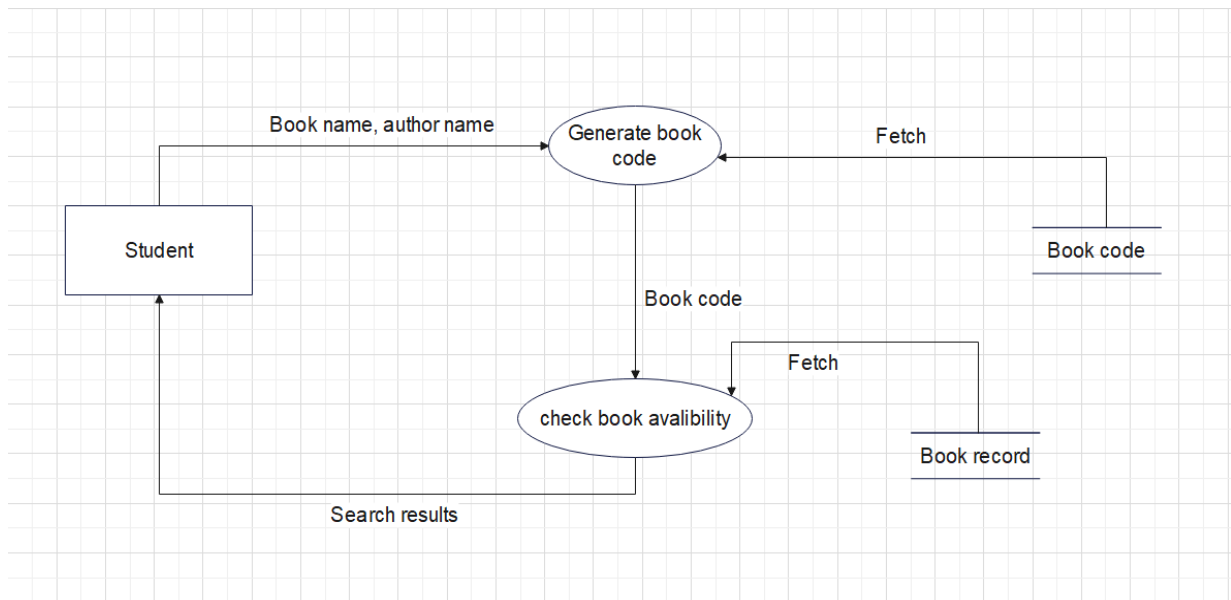
➤ BOOK STOCK MANAGEMENT:



➤ **RETRUN BOOK:**

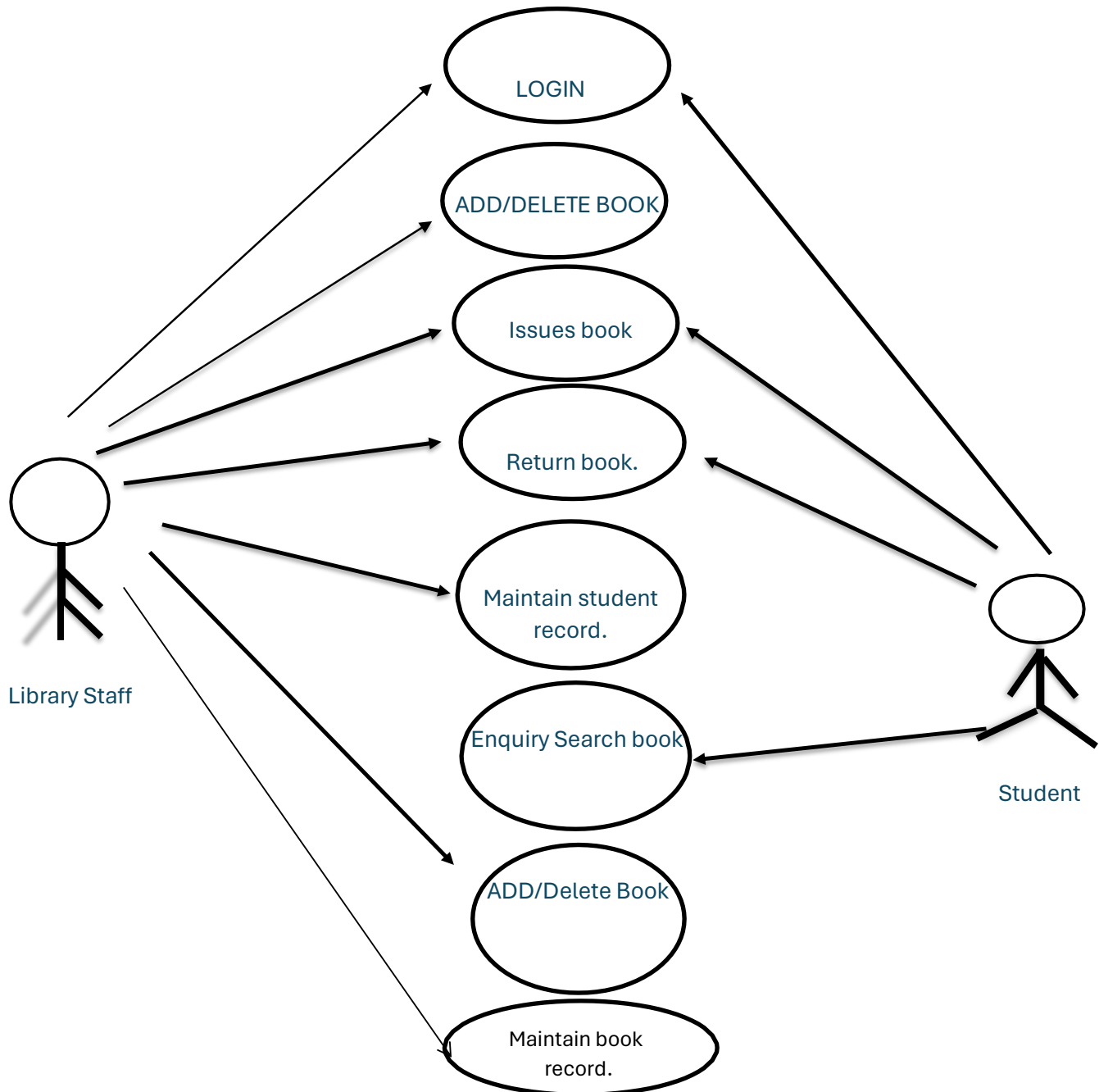


➤ **SEARCH:**



3.7 USE CASE DIAGRAM:

- Use Case Diagram are drawn to capture the functional requirements of a system.
- Use Case Diagram consists of Actors, use cases and their relationships.



3.8 SCHEMA DESIGN:



CHAPTER 4: IMPLEMENTATION AND TESTING

4.1 CODE:

Login Page:

```
import axios from "axios";
import { useState } from "react";
import { useDispatch } from "react-redux";
import { useNavigate } from "react-router-dom";
import { setAuthUser } from "../store/auth/slice";

export const SignIn = () => {
  const [userDetails, setUserDetails] = useState({
    email: "",
    password: "",
  });
6
  const handleUserDetails = (e) => {
    const name = e.target.name;
    const value = e.target.value;
    setUserDetails({ ...userDetails, [name]: value });
  };

  const navigate = useNavigate();
  const dispatch = useDispatch();

  const handleLogin = () => {
    axios
      .post("http://localhost:3000/users/login", userDetails)
      .then((res) => {
        console.log(res.data);
        dispatch(setAuthUser(res.data));
        navigate("/home");
      })
      .catch((error) => alert(JSON.stringify(error)));
  };
};
```

```

return (
  <section className="bg-gray-50 dark:bg-gray-900">
    <div className="py-8 px-4 mx-auto max-w-screen-xl lg:py-16 grid lg:grid-cols-2
gap-8 lg:gap-16">
      <div className="flex flex-col justify-center">
        <h1 className="mb-4 text-4xl font-extrabold tracking-tight leading-none text-gray-
900 md:text-5xl lg:text-6xl dark:text-white">
          We invest in the world's potential
        </h1>
        <p className="mb-6 text-lg font-normal text-gray-500 lg:text-xl dark:text-gray-
400">
          Welcome to the Siddharth College E-Library. Library management is important
because, Libraries are the storehouses of knowledge and information. Libraries perform
various tasks like collecting books, arranging them systematically, conservation and
preservation of those books, dissemination of information sources, etc.
        </p>
        <a
          href="#"
          className="text-blue-600 dark:text-blue-500 hover:underline font-medium text-lg
inline-flex items-center"
        >
          <svg
            className="w-3.5 h-3.5 ms-2 rtl:rotate-180"
            aria-hidden="true"
            xmlns="http://www.w3.org/2000/svg"
            fill="none"
            viewBox="0 0 14 10"
          >
            <path
              stroke="currentColor"
              strokeLinecap="round"
              strokeLinejoin="round"
              strokeWidth={2}
              d="M1 5h12m0 0L9 1m4 4L9 9"
            />
          </svg>
        </a>
      </div>
    </div>
  </div>

```

```

<div className="w-full lg:max-w-xl p-6 space-y-8 sm:p-8 bg-white rounded-lg
shadow-xl dark:bg-gray-800">
  <h2 className="text-2xl font-bold text-gray-900 dark:text-white">
    Sign in to College Library
  </h2>
  <div>
    <label
      htmlFor="email"
      className="block mb-2 text-sm font-medium text-gray-900 dark:text-white"
    >
      Your email
    </label>
    <input
      type="email"
      name="email"
      value={userDetails.email}
      onChange={handleUserDetails}
      id="email"
      className="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-
lg focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5 dark:bg-gray-700
dark:border-gray-600 dark:placeholder-gray-400 dark:text-white dark:focus:ring-blue-500
dark:focus:border-blue-500"
      placeholder="name@company.com"
      required=""
    />
  </div>
  <div>
    <label
      htmlFor="password"
      className="block mb-2 text-sm font-medium text-gray-900 dark:text-white"
    >
      Your password
    </label>
    <input
      type="password"
      name="password"
      id="password"
      value={userDetails.password}

```

```

        onChange={handleUserDetails}
        placeholder=""
        className="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-
lg focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5 dark:bg-gray-700
dark:border-gray-600 dark:placeholder-gray-400 dark:text-white dark:focus:ring-blue-500
dark:focus:border-blue-500"
        required=""
    />
</div>
<button
    type="submit"
    className="w-full px-5 py-3 text-base font-medium text-center text-white bg-
blue-700 rounded-lg hover:bg-blue-800 focus:ring-4 focus:ring-blue-300 sm:w-auto
dark:bg-blue-600 dark:hover:bg-blue-700 dark:focus:ring-blue-800"
    onClick={() => handleLogin()}
>
    Login to your account
</button>
<div className="text-sm font-medium text-gray-900 dark:text-white">
    Not registered yet?{ " " }
    <a
        className="text-blue-600 hover:underline dark:text-blue-500"
        onClick={() => navigate("/register")}
    >
        Create account
    </a>
</div>
</div>
</div>
</div>
</section>
);
};

```

REGISTER PAGE:

```
import { Link, useNavigate } from "react-router-dom";
import { useState } from "react";
import axios from "axios";
export const SignUp = () => {
  const [user, setUser] = useState({
    name: "",
    email: "",
    password: "",
    phone_no: "",
    stream: "",
  });
  const navigate = useNavigate();
  const createUser = () => {
    axios
      .post("http://localhost:3000/users", user)
      .then(() => navigate("/login"))
      .catch((error) => alert(error.response.data));
  };

  const handleInputs = (e) => {
    const name = e.target.name;
    const value = e.target.value;
    setUser({ ...user, [name]: value });
  };
  return (
    <
      <div className="h-screen flex items-center justify-center bg-neutral-400">
        { /* Right side with input div */ }
        <section className="bg-gray-50 dark:bg-gray-900"></section>

        <div className="w-full bg-white rounded-lg shadow dark:border md:mt-0 sm:max-w-md xl:p-0 dark:bg-gray-800 dark:border-gray-700">
          <div className="p-6 space-y-4 md:space-y-6 sm:p-8">
            <h1 className="text-xl font-bold leading-tight tracking-tight text-gray-900 md:text-2xl dark:text-white">
              Create and account
            </h1>
          </div>
        </div>
      </div>
    )
  );
};
```

```

</h1>
<div>
  <label
    htmlFor="name"
    className="block mb-2 text-sm font-medium text-gray-900 dark:text-white"
  >
    Enter your Name
  </label>
  <input
    type="text"
    name="name"
    id="name"
    className="bg-gray-50 border border-gray-300 text-gray-900 sm:text-sm
rounded-lg focus:ring-primary-600 focus:border-primary-600 block w-full p-2.5 dark:bg-
gray-700 dark:border-gray-600 dark:placeholder-gray-400 dark:text-white dark:focus:ring-
blue-500 dark:focus:border-blue-500"
    placeholder="Enter your Name"
    required="name"
    value={user.name}
    onChange={handleInputs}
  />
</div>
<div>
  <label
    htmlFor="email"
    className="block mb-2 text-sm font-medium text-gray-900 dark:text-white"
  >
    Your email
  </label>
  <input
    type="email"
    name="email"
    id="email"
    className="bg-gray-50 border border-gray-300 text-gray-900 sm:text-sm
rounded-lg focus:ring-primary-600 focus:border-primary-600 block w-full p-2.5 dark:bg-
gray-700 dark:border-gray-600 dark:placeholder-gray-400 dark:text-white dark:focus:ring-
blue-500 dark:focus:border-blue-500"
    placeholder="name@company.com"
    required="email"

```

```

        value={user.email}
        onChange={handleInputs}
      />
    </div>
    <div>
      <label
        htmlFor="password"
        className="block mb-2 text-sm font-medium text-gray-900 dark:text-white"
      >
        Password
      </label>
      <input
        type="password"
        name="password"
        id="password"
        placeholder="••••••••"
        className="bg-gray-50 border border-gray-300 text-gray-900 sm:text-sm
rounded-lg focus:ring-primary-600 focus:border-primary-600 block w-full p-2.5 dark:bg-
gray-700 dark:border-gray-600 dark:placeholder-gray-400 dark:text-white dark:focus:ring-
blue-500 dark:focus:border-blue-500"
        required="password"
        value={user.password}
        onChange={handleInputs}
      />
    </div>
    <div>
      <label className="block mb-2 text-sm font-medium text-gray-900 dark:text-
white">
        Phone Number
      </label>
      <input
        type="number"
        name="phone_no"
        placeholder="Enter your Phone No"
        className="bg-gray-50 border border-gray-300 text-gray-900 sm:text-sm
rounded-lg focus:ring-primary-600 focus:border-primary-600 block w-full p-2.5 dark:bg-
gray-700 dark:border-gray-600 dark:placeholder-gray-400 dark:text-white dark:focus:ring-
blue-500 dark:focus:border-blue-500"
        required="number"

```



```

        value={user.phone_no}
        onChange={handleInputs}
      />
    </div>
    <div>
      <label
        htmlFor="Stream"
        className="block mb-2 text-sm font-medium text-gray-900 dark:text-white"
      >
        Stream
      </label>
      <input
        type="text"
        name="stream"
        id="stream"
        className="bg-gray-50 border border-gray-300 text-gray-900 sm:text-sm
rounded-lg focus:ring-primary-600 focus:border-primary-600 block w-full p-2.5 dark:bg-
gray-700 dark:border-gray-600 dark:placeholder-gray-400 dark:text-white dark:focus:ring-
blue-500 dark:focus:border-blue-500"
        placeholder="Enter your Stream"
        required="stream"
        value={user.stream}
        onChange={handleInputs}
      />
    </div>
    <div className="h-28 w-full mb-4">
      <p className="text-white font-bold text-xl mb-4 px-14">
        Upode Your fees receipt
      </p>
      <div className="h-10 w-full bg-slate-400 mb-4 px-10 py-2">
        <input
          type="file"
          className="dark:ring-offset-gray-800 dark:border-gray-600 text-black font-
bold px-4 rounded"
        />
      </div>
    </div>
    <div className="flex items-start">

```

```

    <div className="flex items-center h-5">
      <input
        id="terms"
        aria-describedby="terms"
        type="checkbox"
        className="w-4 h-4 border border-gray-300 rounded bg-gray-50 focus:ring-3
focus:ring-primary-300 dark:bg-gray-700 dark:border-gray-600 dark:focus:ring-primary-
600 dark:ring-offset-gray-800"
        required=""
      />
    </div>
  </div>
  <button
    className="w-full text-white bg-primary-600 hover:bg-primary-700 focus:ring-4
focus:outline-none focus:ring-primary-300 font-medium rounded-lg text-sm px-5 py-2.5
text-center dark:bg-primary-600 dark:hover:bg-primary-700 dark:focus:ring-primary-800"
    onClick={() => createUser()}
  >
    Create an account
  </button>
  <p className="text-sm font-light text-gray-500 dark:text-gray-400">
    Already have an account?
    <Link
      className="text-xl font-light text-gray-500 dark:text-gray-400 px-5 rounded"
      to="/login"
    >
      Sign In
    </Link>
  </p>
</div>
</div>
</div>
</>
);
};

```

HOME PAGE:

```
import Header from "../common/Header";
import { SearchBar } from "../common/SearchBar";
import { Ebooks } from "../pages/Ebooks";
export const Home = () => {
  return (
    <>
      <Header />
      <div className="flex justify-center p-10">
        <SearchBar />
      </div>
      <Ebooks />
    </>
  );
};
```

DASHBOARD:

```
import { SideBar } from "../Admindashboard/SideBar";
import { UpperNavbar } from "../Admindashboard/UpperNavbar";
/* eslint-disable react/no-unknown-property */
export const Dashboard = () => {
  return (
    <>
      <UpperNavbar />
      <SideBar />
    </>
  );
};
```

SideBar:

```
import { useState } from "react";
import { useNavigate } from "react-router-dom";

export function SideBar() {
  const BookContant = [
    {
      id: 1,
      action: "Add Book",
      path: "/dashboard/addBooks",
    },
    {
      id: 2,
      action: "Update Book",
      path: "/dashboard/updateBooks",
    },
    {
      id: 3,
      action: "Delete Book",
      path: "/dashboard/deleteBooks",
    },
    {
      id: 4,
      action: "Apply Books",
      path: "/dashboard/applyBooks",
    },
    {
      id: 5,
      action: "Return Books",
      path: "/dashboard/returnBooks",
    },
  ];
  const navigate = useNavigate();
  const [activeState, setActiveState] = useState(BookContant[0].id);
  return (
```

<

<aside className="flex h-screen w-52 flex-col overflow-y-auto border-r bg-white px-

8 ">

```
<div className="mt-6 flex flex-1 flex-col justify-between">
  <nav className="-mx-3 space-y-6 ">
    <div className="space-y-3 ">
      <a
        className="flex transform items-center rounded-lg px-3 py-2 text-gray-600
transition-colors duration-300 hover:bg-gray-950 hover:text-slate-100"
        href="#"
      >
        <span className="text-xl font-bold ">DASHBOARD</span>
      </a>
    </div>
    <div className="space-y-3 ">
      <label className="text-xl font-bold mt-3">Books</label>
      {BookContant.map((book) => (
        <a
          key={book.id}
          className={
            activeState !== book.id
              ? "flex transform items-center rounded-lg px-3 py-2 text-gray-600 transition-
colors duration-300 hover:bg-gray-950 hover:text-slate-100"
              : "flex transform items-center rounded-lg px-3 py-2 text-white transition-
colors duration-300 bg-gray-950 hover:text-slate-100"
          }
          onClick={() => {
            setActiveState(book.id);
            navigate(`${book.path}`);
          }}
        >
          <span className="mx-2 text-base text-center font-bold">
            {book.action}
          </span>
        </a>
      ))}
    </div>
    <div className="space-y-3 ">
      <label className="text-xl font-bold mt-3">Department</label>
      <a
```

```

        className="flex transform items-center rounded-lg px-3 py-2 text-gray-600
transition-colors duration-300 hover:bg-gray-950 hover:text-slate-100"
        href="#"
    >
    <span className="mx-2 text-base text-center font-bold">
        B.Com
    </span>
</a>
<a
    className="flex transform items-center rounded-lg px-3 py-2 text-gray-600
transition-colors duration-300 hover:bg-gray-950 hover:text-slate-100"
    href="#"
    >
    <span className="mx-2 text-base text-center font-bold">
        Bsc.IT
    </span>
</a>
<a
    className="flex transform items-center rounded-lg px-3 py-2 text-gray-600
transition-colors duration-300 hover:bg-gray-950 hover:text-slate-100"
    href="#"
    >
    <span className="mx-2 text-base text-center font-bold">
        B.M.S
    </span>
</a>
<a
    className="flex transform items-center rounded-lg px-3 py-2 text-gray-600
transition-colors duration-300 hover:bg-gray-950 hover:text-slate-100"
    href="#"
    >
    <span className="mx-2 text-base text-center font-bold">
        B.M.M
    </span>
</a>
<a
    className="flex transform items-center rounded-lg px-3 py-2 text-gray-600
transition-colors duration-300 hover:bg-gray-950 hover:text-slate-100"

```

```

        href="#"
    >
        <span className="mx-2 text-base text-center font-bold">
            Jr.college
        </span>
    </a>
</div>

<div className="space-y-3 ">
    <label className="text-xl font-bold mt-3">Notice</label>
    <a
        className="flex transform items-center rounded-lg px-3 py-2 text-gray-600
transition-colors duration-300 hover:bg-gray-950 hover:text-slate-100"
        href="#"
    >
        <span className="mx-2 text-base text-center font-bold">
            Previous notice
        </span>
    </a>
    <a
        className="flex transform items-center rounded-lg px-3 py-2 text-gray-600
transition-colors duration-300 hover:bg-gray-950 hover:text-slate-100"
        href="#"
    >
        <span className="mx-2 text-base text-center font-bold">
            New notice
        </span>
    </a>
</div>
</nav>
</div>
</aside>
</>
);
}

```

Upper Navbar:

```
import { Menu } from "@headlessui/react";
import { BookOpenIcon } from "@heroicons/react/24/outline";
export const UpperNavbar = () => {
  return (
    <>
      <div className="bg-gray-800">
        <div className="mx-auto max-w-7xl px-2 sm:px-6 lg:px-8">
          <div className="relative flex h-16 items-center justify-between">
            <div className="absolute inset-y-0 left-0 flex items-center sm:hidden">
              { /* Mobile menu button */ }
            </div>
            <div className="flex flex-1 items-center justify-center sm:items-stretch sm:justify-start">
              <div className="flex flex-shrink-0 items-center">
                <BookOpenIcon className="h-8 w-auto" color="white" />
                <span className="text-white ml-2 mb-1">
                  Siddharth College E-library
                </span>
              </div>
            </div>
          </div>
          { /* Profile dropdown */ }
          <Menu as="div" className="relative ml-3">
            <div>
              <Menu.Button className="relative flex rounded-full bg-gray-800 text-sm focus:outline-none focus:ring-2 focus:ring-white focus:ring-offset-2 focus:ring-offset-gray-800">
                <span className="absolute -inset-1.5" />
                <span className="sr-only">Open user menu</span>
                
              </Menu.Button>
            </div>
          </Menu>
        </div>
      </div>
    </>
  );
}
```



```

        </Menu.Button>
      </div>
    </Menu>
  </div>
</div>
</div>
</div>
</div>
</div>
</div>
);
};

```

Add Books:

```

import axios from "axios";
import { Dashboard } from "../Dashboard";
import { useState } from "react";
import { ToastContainer, toast } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";

```

```

export const AddBooks = () => {
  const [addBooks, setBooks] = useState({
    title: "",
    author: "",
    publisher: "",
    department: "",
    semester: "",
    url: "",
    num_of_copies: 1,
  });

```

```

  console.log(addBooks);

```

```

  const handleFileChange = (event) => {
    const selectedFile = event.target.files[0];
    const reader = new FileReader();

```

```

    reader.onloadend = () => {
      const base64String = reader.result;

```

```

    setBooks({ ...addBooks, url: base64String });
  };

  if (selectedFile) {
    reader.readAsDataURL(selectedFile);
  }
};

const handleAddBook = () => {
  axios
    .post("http://localhost:3000/books", addBooks)
    .then(() => toast.success("Book added successfully"))
    .catch((error) => toast.error(error.response.data));
};

const handleInputs = (e) => {
  const { name, value } = e.target;
  setBooks({ ...addBooks, [name]: value });
};

return (
  <>
    <Dashboard />
    <div className="w-full bg-white translate-x-96 mx-96 -translate-y-full ">
      <div className="rounded-lg shadow dark:border md:mt-0 sm:max-w-md xl:p-0 dark:bg-gray-800 dark:border-gray-700">
        <div className="p-6 space-y-4 md:space-y-6 sm:p-8">
          <h1 className="text-xl font-bold leading-tight tracking-tight text-gray-900 md:text-2xl dark:text-white">
            Add New Book
          </h1>
          <div>
            <label className="block mb-2 text-sm font-medium text-gray-900 dark:text-white">
              Title
            </label>
            <input
              type="text"

```

```

        name="title"
        id="title"
        className="bg-gray-50 border border-gray-300 text-gray-900 sm:text-sm
rounded-lg focus:ring-primary-600 focus:border-primary-600 block w-full p-2.5 dark:bg-
gray-700 dark:border-gray-600 dark:placeholder-gray-400 dark:text-white dark:focus:ring-
blue-500 dark:focus:border-blue-500"
        placeholder="Book Title"
        value={addBooks.title}
        onChange={handleInputs}
    />
</div>
<div>
    <label className="block mb-2 text-sm font-medium text-gray-900 dark:text-
white">
        Author Name
    </label>
    <input
        type="text"
        name="author"
        id="author"
        placeholder="Author Name"
        className="bg-gray-50 border border-gray-300 text-gray-900 sm:text-sm
rounded-lg focus:ring-primary-600 focus:border-primary-600 block w-full p-2.5 dark:bg-
gray-700 dark:border-gray-600 dark:placeholder-gray-400 dark:text-white dark:focus:ring-
blue-500 dark:focus:border-blue-500"
        value={addBooks.author}
        onChange={handleInputs}
    />
</div>
<div>
    <label className="block mb-2 text-sm font-medium text-gray-900 dark:text-
white">
        Publisher
    </label>
    <input
        type="text"
        name="publisher"
        id="publisher"
        placeholder="Publisher Name"

```

```
        className="bg-gray-50 border border-gray-300 text-gray-900 sm:text-sm rounded-lg focus:ring-primary-600 focus:border-primary-600 block w-full p-2.5 dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400 dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500"
```

```
        value={addBooks.name}
```

```
        onChange={handleInputs}
```

```
    />
```

```
</div>
```

```
<div>
```

```
  <label className="block mb-2 text-sm font-medium text-gray-900 dark:text-white">
```

```
    Department
```

```
  </label>
```

```
  <input
```

```
    type="text"
```

```
    name="department"
```

```
    id="department"
```

```
    placeholder="Department Name"
```

```
    className="bg-gray-50 border border-gray-300 text-gray-900 sm:text-sm rounded-lg focus:ring-primary-600 focus:border-primary-600 block w-full p-2.5 dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400 dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500"
```

```
    value={addBooks.name}
```

```
    onChange={handleInputs}
```

```
  />
```

```
</div>
```

```
<div>
```

```
  <label className="block mb-2 text-sm font-medium text-gray-900 dark:text-white">
```

```
    Semster
```

```
  </label>
```

```
  <input
```

```
    type="number"
```

```
    name="semester"
```

```
    id="Semster"
```

```
    placeholder="Semster"
```

```
    className="bg-gray-50 border border-gray-300 text-gray-900 sm:text-sm rounded-lg focus:ring-primary-600 focus:border-primary-600 block w-full p-2.5 dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400 dark:text-white dark:focus:ring-
```

```

blue-500 dark:focus:border-blue-500"
    value={addBooks.semester}
    onChange={handleInputs}
  />
</div>
<div>
  <label className="block mb-2 text-sm font-medium text-gray-900 dark:text-
white">
    Number of Copies
  </label>
  <input
    type="number"
    name="num_of_copies"
    id="Semster"
    placeholder="Semster"
    className="bg-gray-50 border border-gray-300 text-gray-900 sm:text-sm
rounded-lg focus:ring-primary-600 focus:border-primary-600 block w-full p-2.5 dark:bg-
gray-700 dark:border-gray-600 dark:placeholder-gray-400 dark:text-white dark:focus:ring-
blue-500 dark:focus:border-blue-500"
    value={addBooks.num_of_copies}
    onChange={handleInputs}
  />
</div>

<p className="text-white font-bold text-xl mb-4 px-14">
  Upload Book image
</p>
<div className="h-10 w-full bg-slate-400 mb-4 px-10 py-2">
  <input
    type="file"
    id="image"
    className="dark:ring-offset-gray-800 dark:border-gray-600 text-black font-
bold px-4 rounded"
    onChange={handleFileChange}
  />
</div>

<button

```

```

        type="submit"
        className="w-full hover:bg-slate-950 text-white bg-primary-600 hover:bg-
primary-700 focus:ring-4 focus:outline-none focus:ring-primary-300 font-medium
rounded-lg text-sm px-5 py-2.5 text-center dark:bg-primary-600 dark:hover:bg-primary-
700 dark:focus:ring-primary-800"
        onClick={() => handleAddBook()}
      >
      Add Book
    </button>
  </div>
</div>
</div>
<ToastContainer />
</>
);
};

```

BACK-END:

Index.js:

```

const mongoose = require("mongoose");
const bodyParser = require("body-parser");
const cors = require("cors");
const express = require("express");
const multer = require("multer");
const BookRouter = require("./book/book.routes");
const UserRouter = require("./user/user.routes");
const app = express();
app.use(bodyParser.json({ limit: "10mb" }));
app.use(
  cors({
    origin: "*",
  })
);

const connection = mongoose
  .connect("mongodb://127.0.0.1:27017/library")

```

```

.then(() => console.log(" Backend is Connected!"));

app.get("/", function (req, res) {
  res.status(200).json({ message: "get user details" });
});

app.use("/users", UserRouter);
app.use("/books", BookRouter);

app.listen(3000, () => {
  console.log("Server listening on port 3000");
});

module.exports = connection;

```

Book Controller:

```

const Book = require("../models/book.model");

const BookController = () => {};

BookController.getAllBooks = async (req, res) => {
  try {
    const allBooks = await Book.find();
    if (!allBooks) {
      return res.status(404).send("Not book found");
    }
    return res.status(200).json(allBooks);
  } catch (error) {
    return res.status(500).send(error);
  }
};

BookController.getBookById = async (req, res) => {
  try {
    const bookId = req.params.id;
    const book = await Book.findById(bookId);

```

```

    if (!book) {
      return res.status(404).json("Book not found");
    }
    return res.status(200).json(book);
  } catch (error) {
    return res.status(500).send(error);
  }
};

```

```

BookController.updateBook = async (req, res) => {
  try {
    const bookData = req.body;
    const bookId = req.params.id;

    const book = await Book.updateOne({ _id: bookId }, bookData);
    console.log({ book });
    if (!book) {
      return res.status(500).json("Error occurred while adding book");
    }
    return res
      .status(200)
      .json({ message: "Book updated successfully", data: book });
  } catch (error) {
    return res.status(500).json(error);
  }
};

```

```

BookController.addBook = async (req, res) => {
  try {
    const bookData = req.body;
    const book = await Book.create(bookData);
    if (!book) {
      return res.status(500).json("Error occurred while adding book");
    }
    return res
      .status(201)
      .json({ message: "Book added successfully", data: book });
  }
};

```



```

    } catch (error) {
      return res.status(500).json(error);
    }
  };

```

```

BookController.deleteBook = async (req, res) => {
  try {
    const bookId = req.params.id;
    console.log(bookId);
    const book = await Book.findByIdAndDelete(bookId);
    if (!book) {
      return res.status(500).json("Error occurred while deleting book");
    }
    return res
      .status(200)
      .json({ message: "Book deleted successfully", data: book });
  } catch (error) {
    return res.status(500).json(error);
  }
};

```

```

module.exports = BookController;

```

Book Models:

```

const mongoose = require("mongoose");
const Schema = mongoose.Schema;

```

```

const BookModel = new Schema(
  {
    title: String,
    author: String,
    publisher: String,
    department: String,
    semester: Number,
    url: String,
    num_of_copies: Number,
  },

```

```
    { timestamps: true }  
  );  
  
const Book = mongoose.model("book", BookModel);  
  
module.exports = Book;
```

USER MODELS:

```
const mongoose = require("mongoose");  
const Schema = mongoose.Schema;  
  
const UserModel = new Schema(  
  {  
    name: String,  
    email: { type: String, unique: true },  
    password: String,  
    phone_no: Number,  
    stream: String,  
    isAdmin: {  
      type: Boolean,  
      default: false,  
    },  
  },  
  { timestamps: true }  
);  
  
const User = mongoose.model("user", UserModel);  
  
module.exports = User;
```

USER BOOK MODELS:

```
const mongoose = require("mongoose");  
const Schema = mongoose.Schema;  
  
const UserBookModel = new Schema(  

```

```
{
  bookId: {
    type: mongoose.Types.ObjectId,
    ref: "book",
  },
  userId: {
    type: mongoose.Types.ObjectId,
    ref: "user",
  },
  num_of_copies: Number,
},
{ timestamps: true }
);

const UserBook = mongoose.model("user_book", UserBookModel);

module.exports = UserBook;
```

4.2 TESTING APPROACH:

- A test approach is the test strategy implementation of a project, defines how testing would be carried out.
- There are two types of Testing approach:
 - **Proactive:**
 - An approach in which the test design process is initiated as early as possible in order to find and fix the defects before the build is created.
 - **Reactive:**
 - An approach in which the testing is not started until after design and coding are completed.

4.2.1 UNIT TESTING:

- Unit Testing is a type of software testing where individual units or components of a software are tested.
- Unit Testing is done the development (coding phase) of an application by the Developers.

➤ **Test Case Design for Login Page:**

Test CaseId	Test Case Name	Test Case Description	Test Input	Expected Output	Actual Output	Test Case Status- (P/F)
1	Email id	Only Registered Email are Allowed	nick@gmail. com	Login Successful	Login Successful	Pass
			nick@ab c.com	Invalid Email	Invalid Email	Fail
2	Password	To verify Entered password is corrected	1234567	Login Successful	Login Successful	Pass
			012345	Incorrected password	Incorrected password	Fail
3	Password	Password length should be more than 7	1234567	Login Successful	Login Successful	Pass
			1234	Invalid password	Invalid password	Fail
4	Login	To verify email and password should be blank	nick@gmail.com 1234567	Login Successful	Login Successful	Pass
			Fields are empty	Please Enter your Email and Password	Please Enter your Email and Password	Fail

➤ **Test Case Design for Register Page:**

Test Case Id	Test Case Name	Test Case Description	Test Input	Expected Output	Actual Output	Test Case Status-(P/F)
1	Full Name	Only valid Full Name are allowed	Nick Jaiswal	Register Successfully	Register Successfully	Pass
			Nick Jaiswal@145\$	Invalid Full Name	Invalid Full Name	Fail
2	Email	Only Email are Allowed	nick@gmail.com	Register Successfully	Register Successfully	Pass
			nick@ab c.com	Invalid Email	Invalid Email	Fail
3	Phone Number	Phone Number are allowed	96325874123	Register Successfully	Register Successfully	Pass
			96325874123658	Invalid password	Invalid password	Fail
4	Password	Password length should be more than 7	Nick12345	Register Successfully	Register Successfully	Pass
			23427	Invalid password	Invalid password	Fail
5	Register	To verify Username, Password, Confirm Password Should be not be left blank	Nick Jaiswal Nick1234 CP: Nick1234	Register Successfully	Register Successfully	Pass
			Fields are empty	Please Enter Username, Password, Confirm Password	Please Enter Username, Password, Confirm Password	Fail

➤ **Test Case Design for Add Books:**

C	Test Case Name	Test Case Description	Test Input	Expected Output	Actual Output	Test Case Status -(P/F)
1	Book id	Enter your room id (Admin create a book id: Only string)	54896	Book id added Successfully	Book id Successfully	Pass
			dfdsfd	Invalid Book id	Invalid Book id	Fail

➤ **Test Case Design for No of Copy available:**

Test Case Id	Test Case Name	Test Case Description	Test Input	Expected Output	Actual Output	Test Case Status-(P/F)
1	No_of cpoy	Only Number	Valid	Check no of copy available	Available	Pass
			Invalid	Not available	Apply	Fail

➤ **Test Case Design for Borrow Table:**

Test CaseId	Test Case Name	Test Case Description	Test Input	Expected Output	Actual Output	Test Case Status- (P/F)
1	Borrow_id	To verify borrow_id is valid or not	Valid	Borrow id check successful	Borrow id check successful	Pass
		If invalid = fail	Invalid	Borrow id check not successful	Borrow id check not successful	Fail

➤ **Test Case Design for Requirement Table:**

Field name	Type	Constrains
ID	Text	Primary key
Title	Number	Not null
Author	Number	Not null
Publisher	Number	Not null
No of copy	Number	Not null

➤ **Test Case Design for Book Catalog Table:**

Field Name	Type	Constrains
Title	Text	Not null
Edition	Text	Not null
Author	Text	Not null
Publisher	Text	Not null
Yr_publish	Text	Not null
No copy	Number	Not null
Available copy	Number	Not null
Borrow copy	Number	Not null

4.2.2 INTEGRATION TESTING:

- Integration Testing is defined as a type of testing where software modules are integrated logically and tested as a group.
- A typical software project consists of multiple software modules, coded by different programmers.

Sr No.	Test Case Name	Type	Description	Pass /Fail (P/F)
1	User Login	Valid or Invalid	Valid if correct	Pass
			Invalid if incorrect	Fail
2	Book details	Book id check correct or not	Book id valid	Pass
			Book id invalid	Fail
3	Internet	Detected	Internet Detected	Pass
			Internet not detected	Fail

- In this type of testing, we test various integration of the project module by providing the input. The primary objective is to test the module interfaces to ensure that no errors are occurring when one module invokes the other module.

4.3 SECURITY ISSUES:

SSL stands for Secure Sockets Layer, and it is a cryptographic protocol that provides secure communication over a computer network. SSL is used to protect data in transit between two applications, such as a web browser and a web server.

SSL can be used to protect the security of college library management systems in a few ways. For example, SSL can be used to encrypt the following data:

- **User login credentials**
- **Library resource metadata (e.g., book titles, authors, ISBNs)**
- **Lending transaction data**
- **Reservation data**

By encrypting this data, SSL can help to protect it from unauthorized access and interception.

Here are some specific examples of how SSL can be used to mitigate security issues for college library management systems:

- **Unauthorized access to library resources:** SSL can help to protect against unauthorized access to library resources by encrypting user login credentials.
 - This means that even if an attacker is able to obtain a user's login credentials, they will not be able to use them to access library resources unless they also have the user's SSL certificate.
- **Theft or damage to library resources:** SSL can help to protect against theft or damage to library resources by encrypting library resource metadata and lending transaction data. This means that even if an attacker is able to access this data, they will not be able to understand it without the decryption key.
- **Data breaches:** SSL can help to protect against data breaches by encrypting all sensitive data in transit. This means that even if an attacker is able to intercept data as it is being transmitted between two applications,

they will not be able to read it.

- **Privacy violations:** SSL can help to protect against privacy violations by encrypting all user data in transit. This means that even if an attacker is able to intercept user data, they will not be able to read it

4.4 IMPLEMENTATION AND APPROACH:

- In a web application, there is a complexity to implementing a College Library Management System.
- However, depending on the specific requirements and our tools, it opens up several other approaches to.
- Depending on the requirements of the specific tools, it can open up several other approaches to us.
- MongoDB database which can be used to user get book details online to save the time and functionality.

i) Network Communication:

To establish the connection is proper, you'll need to use Network Communication with the other user.

ii) User Interface:

- You'll need to design and implement the user interface for the book, including views for the local and remote add book, buttons for add book and delete and update books, etc...
- User can apply for book any time for any book one user can apply one book at a time.
- E-Books are also available for users.

CHAPTER 5: RESULTS AND DISCUSSIONS

5.1 TEST REPORT:

- In Our Project, **COLLEGE LIBRARY MANAGEMENT SYSTEM** we tested the Web ability to Create add and update and delete book by admin, e-books, online apply books, return book on time and various things are included.
- During the Testing Phase, we identified and fixed several issues related to user experience, including the Web responsiveness, check book details are available and which book need the user and the requirement of users.
- We also performed rigorous Security Testing to ensure the Website if protected user data and privacy.
- The Security Testing included testing for Vulnerabilities, Data Protection, and Access Control Mechanisms.
- Based on the Test Result, we are confident that the College Library Management System meet the requirements of the user and provides seamless and secure Book reading ans learning experience.
- We have Documented all our Test Cases and their Result in the Test Report, including the Environment, Tools, Test Cases and Results.

➤ In Conclusion, we are pleased to report that our College Library Management System with Firebase Authentication, Firebase Storage, and Book details sharing its Alb or not capabilities have passed all our Test and is ready for Deployment.

➤ We believe that website will provide and Excellent User Experience and meet the needs of the users, and we look forward to seeing the web being used by out user.

5.2 USER DOCUMENTATION:

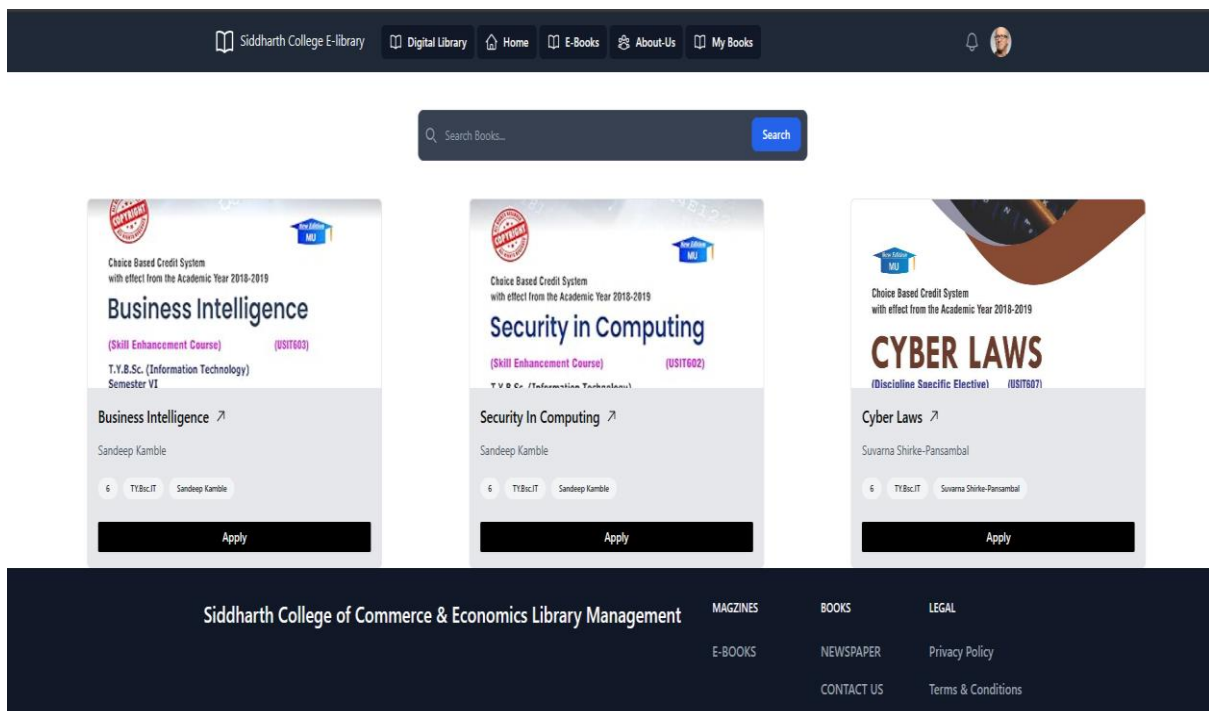
- **SIGN IN / LOGIN:**
- Open the College Library Management System

The screenshot shows the login interface of the College Library Management System. On the left, the title "College Library Management System" is displayed in large white font. Below it, a welcome message states: "Welcome to the Siddharth College E-Library. Library management is important because, Libraries are the storehouses of knowledge and information. Libraries perform various tasks like collecting books, arranging them systematically, conservation and preservation of those books, dissemination of information sources, etc". A small blue arrow points to the right. On the right side, there is a dark gray box titled "Sign in to College Library". Inside this box, there are two input fields: "Your email" (containing "name@company.com") and "Your password". Below these fields is a blue button labeled "Login to your account". At the bottom of the box, it says "Not registered yet? [Create account](#)". At the very bottom of the page, there is a footer with the text "Siddharth College of Commerce & Economics Library Management" on the left, and a grid of links on the right: MAGZINES, E-BOOKS, BOOKS, NEWSPAPER, CONTACT US, LEGAL, Privacy Policy, and Terms & Conditions.

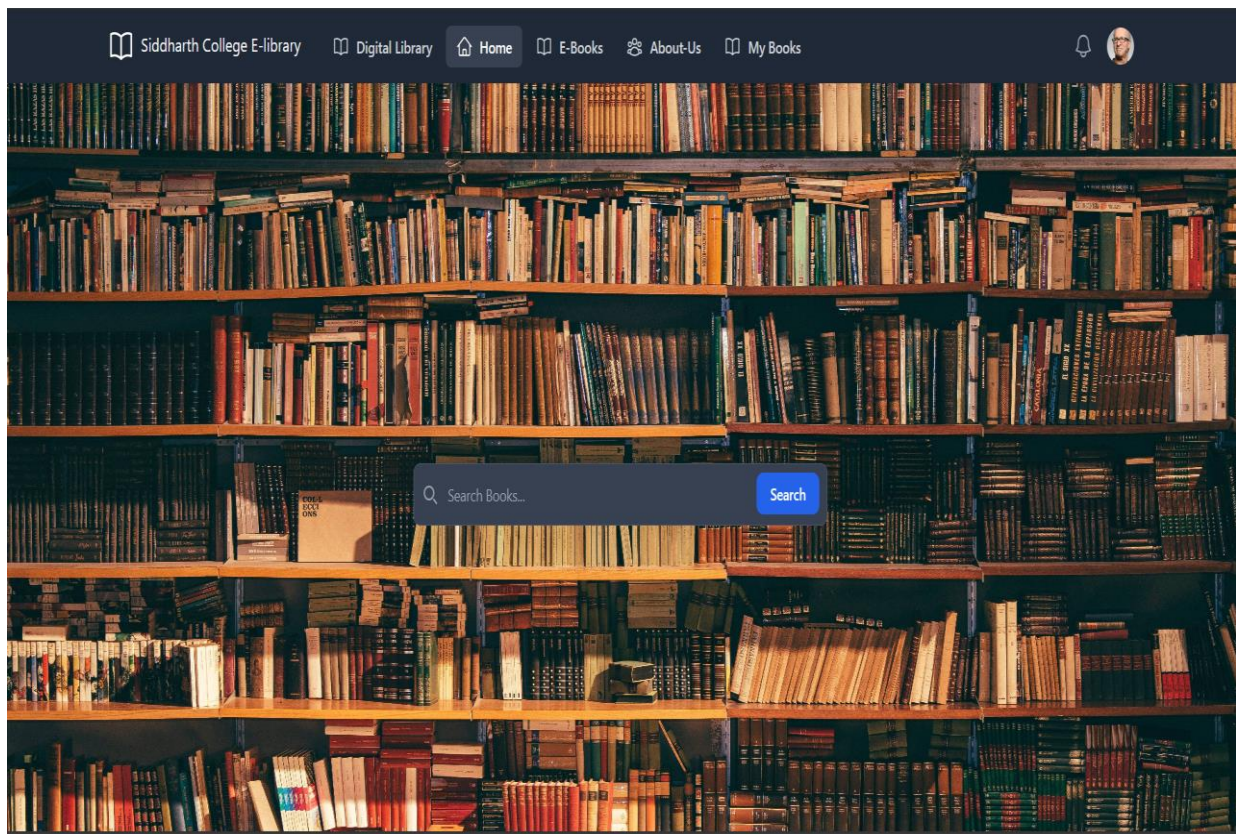
- **SING UP/ REGISTER PAGE:**

The screenshot shows the registration form titled "Create and account". It contains several input fields: "Enter your Name", "Your email" (with "name@company.com" entered), "Password" (with "*****" entered), "Phone Number" (with "Enter your Phone No" entered), and "Stream" (with "Enter your Stream" entered). Below these fields is a section titled "Uplode Your fees receipt" (note the typo) with a "Choose File" button and the text "No file chosen". At the bottom, there is a checkbox labeled "I accept the Terms and Conditions" and a "Create an account" button. A link "Already have an account? [Sign In](#)" is located at the very bottom of the form.

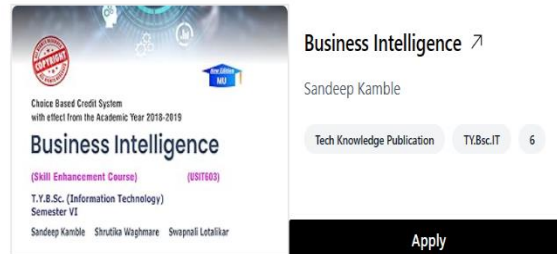
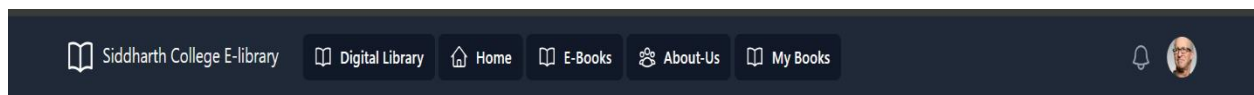
➤ HOME PAGE:



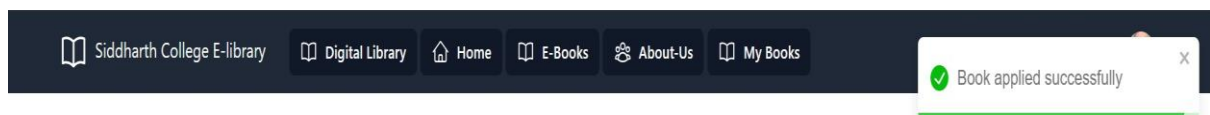
➤ SEARCH ENGINE:




➤ **APPLY BOOK PAGE:**



➤ **AFTER APPLY INTERFACE:**



➤ ABOUT US PAGE:




Siddharth College Commerce and Economics


About Us


Siddharth College of Commerce and Economics, established in 1953, recognized under Sections 2(f) and 12 (B) of the UGC Act is one of the premier affiliated Colleges of the University of Mumbai. Our College was the first Commerce College started by the People's Education Society and is the second oldest Commerce College in Mumbai. The College is situated in South Mumbai which is the financial heart of Mumbai and India. Institutions like the RBI Headquarters, the Bombay Stock Exchange and the head offices of many nationalized banks, including the SBI are within walking distance of the College.


The College building, Anand Bhavan (formerly Albert Building) is a Grade II Heritage structure. The built-in area includes the College office the offices of the Self-financed Programmes, the library, 3 computer labs, the Examination department, the NCC, NSS and Gymkhana rooms, the Principal's cabin, Professors' Common Room, 14 other class rooms and 1 small room for tutorials as well as the Ph.D. Research Centre. The library lends itself to a quiet and reflective ambience for the students as well as the faculty members. It is well-stocked with books. Besides this, the College also has 8 additional





➤ BOOK RECORD:


 Siddharth College E-library


 Digital Library


 Home



 E-Books

 About-Us

 My Books







Choice Based Credit System
with effect from the Academic Year 2018-2019

Business Intelligence

(Skill Enhancement Course) (USIT603)

T.Y.B.Sc. (Information Technology)
Semester VI

Business Intelligence ↗


6


TY.Bsc.IT


Sandeep Kamble


return


➤ **USER RETURN BOOK:**


 Siddharth College E-library



 Digital Library



 Home

 E-Books

 About-Us

 My Books

 books returned successfully 



Choice Based Credit System
with effect from the Academic Year 2018-2019

Business Intelligence

(Skill Enhancement Course) (USIT603)

T.Y.B.Sc. (Information Technology)
Semester VI

Business Intelligence ↗

6

TY.Bsc.IT

Sandeep Kamble

return

CHAPTER 6: CONCLUSION AND FUTURE WORK:

6.1 CONCLUSION:

- The main objective of College Library Management System is to keep the user connected and up to date about books with the developing world.
- College Library Management System have valuable tools because they have no limitations.
- Library Management Software is a valuable tool for educational institutions, enabling efficient management of library resources and operations. By automating tasks, streamlining processes, and enhancing accessibility, it simplifies the librarian's role and offers a seamless experience for students and staff.
- With its numerous benefits, this software 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.
- The Library Management System is much more user-friendly, faster in operation and easy to manage than the manual one. Through the use of it, the librarian can manage the whole data of the library in a single database in different tables with a much more security than the traditional way.
- The Library Management System allows the user to store the book details and the person's details.

- This software allows storing the details of all the data related to library.
- The implementation of the system will reduce data entry time and provide readily calculated reports.

6.2 SIGNIFICATION OF SYSTEM:

- The purpose of a library management system is to manage & track the daily work of the library such as issuing books, return books, due calculations, etc.
- Library management systems facilitate the administrators to keep an eye on the library department's all functions. Also, it enables librarians and users to save time on daunting tasks and enhances efficiency. By using this sort of library management system, the college management would be able to follow the work outline and fineness of different librarians' capabilities.
- Additionally, they get an opportunity to know how well-maintained the record of issued books and collection is. The librarian and the administration department can access various reports to implement new improvements.
- An important aspect of library management is planning and maintaining library facilities. Successful planning is defined as "active planning that ensures an organization will have the right people in the right place at the right time for the right job".
- A User-Friendly Interface, Clear Instructions, and Minimal Technical Requirement are some of the features that make a system easy to use.
- A College Library Management System refers to the software components used for book details.

6.3 LIMITATION OF SYSTEM:

- **Technical issues:** Online library management systems rely on computers, servers, and networks to function. Any technical issues with the system can result in downtime, slow response times, or data loss.
- **Data security:** Online library management systems store sensitive information, such as user details and borrowing records. Therefore, ensuring the security of the data is critical to prevent data breaches, hacking, or unauthorized access.
- **User skills:** Online library management systems may require users to have a certain level of technical skills to use the system effectively. Users who are not familiar with technology may find it challenging to navigate the system, leading to errors or frustration.
- **Equipment and infrastructure:** Online library management systems require suitable equipment and infrastructure to function effectively. Libraries that lack the necessary equipment, such as computers or internet access, may find it challenging to use the system.
- **Cost:** Online library management systems can be expensive to install and maintain. Smaller libraries or institutions with limited budgets may find it challenging to invest in such a system.
- **System upgrades:** Online library management systems require periodic upgrades to keep up with changing technologies and user needs. Upgrading the system can be time-consuming and costly.
- **Dependence on the internet:** Online library management systems require a stable internet connection to function. In areas with poor internet connectivity, the system may not work correctly, leading to delays or even system downtime.

6.4 **FUTURE SCOPRE OF THE PROJECT:**

- **Enhance efficiency:** Monitoring daily data on the total number of volumes issued, unreturned, reissued, and available can be a time-consuming chore for a librarian. A school library management system boosts the effectiveness of a library by enabling all tasks to be accomplished with a single click, making the work of a librarian easier.
Signing into individual accounts allows students to access the catalogue, their book status, and other information. In addition to that, they can be informed of due dates for returning books, notifications to pay late fines, and more through SMS.
- **Keep track of data:** Any educational institution requires data, and library books are a vital asset. Manual data management raises risks such as data misplacement and data input errors. Using a library management system, the whole catalogue can be maintained along with the details of library books, reissued, unreturned, and available. They can be retrieved with a few easy clicks. This functionality also makes it easy for management to keep track of all existing materials. For example, if the librarian requires the current number of a specific genre, the system can instantly provide the count.
Because it is computer-based, this programmed keeps a more accurate record of the available materials. It enables the librarian to organize the books by title, author, publication date, or whatever works best for the library.
- **Boost productivity:** Having a management system in libraries can significantly streamline overall efficiency. With records of the books available with a single click, a portal for real-time analysis, and a direct connection with students, the system can handle the majority of the tasks, saving the team a substantial amount of time.
- **Saves time:** The conventional way of managing library activities can take a while. During the examination period, the number of students using the library surges,

causing students to wait even longer than usual. At this time, using library management software can be incredibly advantageous. The library team can issue books to students promptly while also using their track record to efficiently distribute the books. Students can also check the catalogue to see if the book they require is presently available. This can save both students and the library staff a significant amount of time.

The future scope of library management systems will relieve the stress of manual labour by automating complex activities and saving time.

- **Economical:** Educational institutions have a budget for technological investments. As a result, some institutes may regard such systems as a significant financial investment. The sheer volume of materials in a library makes it difficult for librarians to physically track books and other publications. But technology automates this type of labour. This saves labor costs in addition to having low maintenance costs and efficiency.
- **Management of resources:** Managing the demand and availability of books can be difficult for librarians. With the assistance of a library management system, they can swiftly verify the records, see the history, and issue the books accordingly. They can make book issuance and return faster and smarter with the help of SMS notifications and alerts.
- **Effective collection of late fees:** Late fines from students who fail to return books on time make it difficult for the librarian to collect. This system enables students to get automated SMS notifications and messages regarding books issued, late library fees, due dates, and more. As a result, the library can verify a student's book record and issue payments the next time they come to pay their fees.
- **Safety:** System updates and maintenance are performed on a regular basis to guarantee that user databases are always confidential and safe. A library management system is regularly upgraded to ensure that it is error-free and can

handle large user increases.

- **Improves reporting and monitoring:** Self-updating records in an automated library management system offer dynamic reporting and oversight. It enables more efficient bookings, distribution of material, and user tracking.

- **Both ends find it easier to use:** The benefits of using a library management system extend beyond management. This tool will also help end users. Furthermore, the library management system allows both parties to view materials in circulation as well as their current state. This improves communication and the flow of procedures in the library.