DEPARTMENT LIBRARY MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

Submitted By

DHANAJAYAN K

Register No. 22PCA242505

Under the Guidance of

Dr. C. SENTHAMARAI MCA, Ph.D.

Assistant Professor of Computer Applications

DEPARTMENT OF COMPUTER APPLICATIONS

GOVERNMENT ARTS COLLEGE (AUTONOMOUS)

(Re-Accredited by NAAC with B Status) SALEM

636007

OCTOBER 2023

CERTIFICATE

This is to certify that the mini project work entitled "**DEPARTMENT LIBRARY MANAGEMENT SYSTEM"** submitted during 3rd semester to Government Arts College (Autonomous), Salem-7. In partial fulfillment requirements for the award of degree of **MASTER OF COMPUTER APPLICATIONS** is a record of project work done by [**DHANAJAYAN K], (Register No: 22PCA242505)** from June to Oct 2023 under my supervision and guidance.

Date: **Signature of the Guide**

Place: Salem-7

Head of the Department

Submitted for the Autonomous Practical Examination held on……..

Internal Examiner **External Examiner**

DECLARATION

I hereby declare that the mini project work entitled "**DEPARTMENT LIBRARY MANGEMENT SYSTEM**" submitted to Government Arts College (Autonomous) Salem-7 in partial fulfillment of the requirements for the award of degree of **MASTER OF COMPUTER APPLICATIONS** is a record of original work done by me under the supervision and guidance of **Dr. C. SENTHAMARAI MCA., Ph.D.,** Assistant Professor in Department of Computer Applications, Government Arts College (Autonomous), Salem-7.

Signature of the Candidate

(DHANAJAYAN K)

REGISTER NUMBER:

22PCA242505

Date:

Place: Salem-7.

ACKNOWLEDGEMENT

At the outset, I would like to thank and honor God who gave me the wisdom and knowledge to complete this project.

I would like to profusely thank to our college principal **Dr. N. SHENBAGALAKSHMI, M.A(Tamil)., M.A(JMC)., M.Sc. (Yoga), B.Ed., M.Phil., Ph.D., (NET)** Government Arts College (Autonomous), Salem-7 for her guidance and encouragement to successfully complete my project.

I express my gratitude **Dr. A. KANGAIAMMAL MCA, MPhil, ME, Ph.D.** Head of the Department of Computer Applications, Government Arts College (Autonomous), Salem-7 for being an unfailing source of inspiration and encouragement.

I would like to express my gratitude and sincere thanks to my guide of **Dr. C. SENTHAMARAI MCA, Ph.D.** Assistant Professor**,** Department of Computer Applications, Government Arts College (Autonomous), Salem-7 for her source of guidance and suggestions throughout the project.

I express my sincere thanks to all other staff members of computer science department for this supporting help.

This acknowledgement will be incomplete without expressing my thanks to my parents and friends for their constant support and encouragement.

CONTENTS

Abstract

The Department Library Management System is a project designed to efficiently manage and electronically store information about books, catering to the needs of students and faculty. This system offers a comprehensive solution for library managers and users to continuously monitor the availability of library resources. Users, including administrators, students, and faculty, can easily search for their desired books, simplifying the process of book retrieval.

Manual tracking of book issuance, returns, and fine calculations can be a time-consuming and error-prone task. However, this system automates these processes, ensuring accurate tracking of information such as issue dates, return deadlines, and fine details.

As a result, manual record-keeping is eliminated, reducing the chances of errors and streamlining library operations.

The Department Library Management System, built on the robust foundation of CodeIgniter 4, offers much more than just automation. It goes beyond minimizing manual effort; it fosters a paperless ecosystem within the academic institution.

This shift towards a digital environment not only streamlines library operations but also upholds data integrity.

FRONT END: PHP

BACK END: MYSQL(MariaDB)

FRAMEWORK: CodeIgniter 4

INTRODUCTION

CHAPTER-1

INTRODUCTION

Library Management System is an application which refers to library systems which are generally small or medium in size. It is used by librarian to manage the library using a computerized system where he/she can record various transactions like issue of books, return of books, addition of new books, addition of new students, calculating fine amount, etc.

Books, student and faculty maintenance modules are also included in this system which would keep track ofthe students or faculty using the library and also a detailed 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.

In addition, Budget management is a crucial component of the system, offering real-time tracking of allocated funds for book acquisition. This feature allows department heads and budget managers to stay updated on available funds, expenses, and remaining budget, sending alerts when limits are nearing. The system also simplifies the process of managing book store bills. After obtaining approval, books can be ordered from the bookstore, and the system records and tracks the expenditures, generating detailed reports for book purchases.

MODULES:

- Login Module
- Book borrow and return Module
- Books Management Module
- Users Management Module
- Budget Management Module
- Data Upload Module (Books and Users)

OBJECTIVES:

User Access Control:

To restrict access to department-owned materials only to authorized department members, including faculty, researchers, students, and staff.

Online Catalog and Search:

To provide an accessible online catalog or search interface, allowing department members to easily discover and locate materials within the department's collection.

Reporting and Analytics:

To generate reports and gather insights into departmental library usage, circulation patterns, and collection management. This data informs decisions about resource acquisition and allocation.

User-Friendly Interface:

To offer a user-friendly interface that requires minimal training, ensuring that department members can navigate and utilize the system effortlessly.

Security and Privacy:

To ensure the security and privacy of departmental library records, safeguarding sensitive information about users and their borrowing history.

SYSTEM ANALYSIS

CHAPTER-2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM:

The current system is quite outdated, lacking essential features like user logins, online book requests, and email notifications for borrowed books. It also struggles to generate necessary reports, causing inefficiencies for librarians. In contrast, the proposed system offers user logins, online reservations, and email notifications. It also streamlines reporting, making it easier for librarians to manage the library. Additionally, it incorporates a feature for book requests and suggestions from users, enhancing their overall experience. Overall, the existing system falls short in various aspects, while the proposed system promises a more efficient and user-friendly library management solution.

2.1.1 DRAWBACKS OF EXISTING SYSTEM:

- The existing system lacks any facility for faculty or student's login, while the proposed system will offer both student and teacher login features.
- The existing system doesn't have a feature for online requests of borrowed books, whereas the proposed system has an online book reservation facility.
- The existing system doesn't provide email notifications for borrowed books, whereas the proposed system includes a feature for indicating borrowed books.
- The existing system doesn't offer the facility to generate student and faculty and book issue reports, while the proposed system provides the librarian with tools to generate reports.
- The existing system doesn't have any feature for book requests and suggestions, but in the proposed system, after logging into their accounts, students and faculty can request books and provide suggestions to improve the library.
- The existing system consumes more time and energy and requires additional manpower.
- Maintenance of a large volume of data through manual methods is impractical, and searching is challenging.

2.2 PROPOSED SYSTEM:

- Proposed system is an automated Library Management System. Through our software user can add books, search books, renewal, update information, edit information, and return books in quick time. Our proposed system has the following advantages. Management software for monitoring and controlling the transactions in a library.
- The project “Library Management System” is developed in php, which mainly focuses on basic operations in a library like adding new books, and updating new information, searching books and members and return books.

2.2.1 ADVANTAGE OF PROPOSED SYSTEM:

- The new system automates tasks like adding, updating, and searching for books, reducing manual work, and making library operations smoother.
- The system provides logins for students and faculty, simplifying access. It allows online book requests and sends email notifications for borrowed books, improving user experience and communication.
- Unlike the old system, the new one empowers librarians with reporting tools. This helps in making better decisions by providing insights into student, faculty records, and book issues.
- The new system handles large data volumes effectively, overcoming manual limitations. Its user-friendly search function makes accessing the library's collection more convenient.

2.3 SYSTEM REQUIREMENTS

2.3.1 HARDWARE REQUIREMENTS:

- Processor : Pentium V 2.5 GHz
- RAM : 4 GB or Above
- Hard Disk : 250 GB or Above
- Barcode reader : Barcode laser scanners

2.3.2 SOFTWARE REQUIREMENTS:

- Operating System : Linux, Unix, Windows-7/8/10
- Web Server : Apache HTTP Server 2.4
- Technology : PHP 8.2
- Database : MariaDB 10.11
- IDE : Sublime Text editor

2.3.3 SOFTWARE DESCRIPTION:

Operating System:

The system is compatible with a variety of operating systems, providing versatility and accessibility to users on Linux, Unix, and different Windows versions

Web Server:

Apache HTTP Server serves as the robust foundation for hosting and managing web applications, ensuring secure and reliable web hosting.

PHP:

PHP is employed to create a user-friendly Department Library Management System that caters to students, faculty, and librarians, enhancing the overall project's usability.

MariaDB:

MariaDB is utilized for database connectivity, facilitating the storage and management of book records, faculty and student information, and transactional data within the project.

IDE:

Sublime Text Editor provides a user-friendly integrated development environment for coding, making the development process more efficient and organized.

2.3.4 FEASIBILITY STUDY

- The overall scope of the feasibility study was to provide sufficient information to allow a decision to be made as to whether the Department Library Management

System project should proceed and if so, its relative priority in the context of other existing Library Management Technology. The document only covers the requirements specifications for the Department Library Management System.

- This document does not provide any references to the other component of the Library Management System. All the external interfaces and the dependencies are also identified in this document. The feasibility study phase of this project had undergone through various. Steps which as describe as under:
    - Identity the origin the information at different level.
    - Identity the expectation of user from computerized system.
    - Analyze the drawback of existing system (manual system)

2.3.5 ABOUT THE SOFTWARE

The whole Project is divided in two parts the front end and the back end.

FRONT END

- The front end is designed using of HTML, PHP, CSS, and JavaScript for creating web pages and other information that can be displayed in a web browser.
- The graphical interface through which users interact with the system. It includes the design, layout, and features of the system, such as search forms, user profiles, and book reservation forms.
- JavaScript is used to add interactivity to the front-end. It allows for dynamic features like form validation, real-time search, and user interactions.

BACK END

- The database is used to store and manage information about books, users, transactions, and more. MariaDB, a relational database management system, is often used to store and retrieve data efficiently.
- Implementing security measures to protect user data and system functionality, such as user authentication, authorization, and data encryption.
- This component defines the system's rules, algorithms, and workflows. It manages tasks like book reservations, user registration, borrowing, returning, and generating reports.

SYSTEM DESIGN AND DEVELOPMENT

CHAPTER-3

SYSTEM DESIGN AND DEVELOPMENT

3.1 DATA FLOW DIAGRAM

The Department Library Management System is described in the flowchart displayed in Figure 3.1.

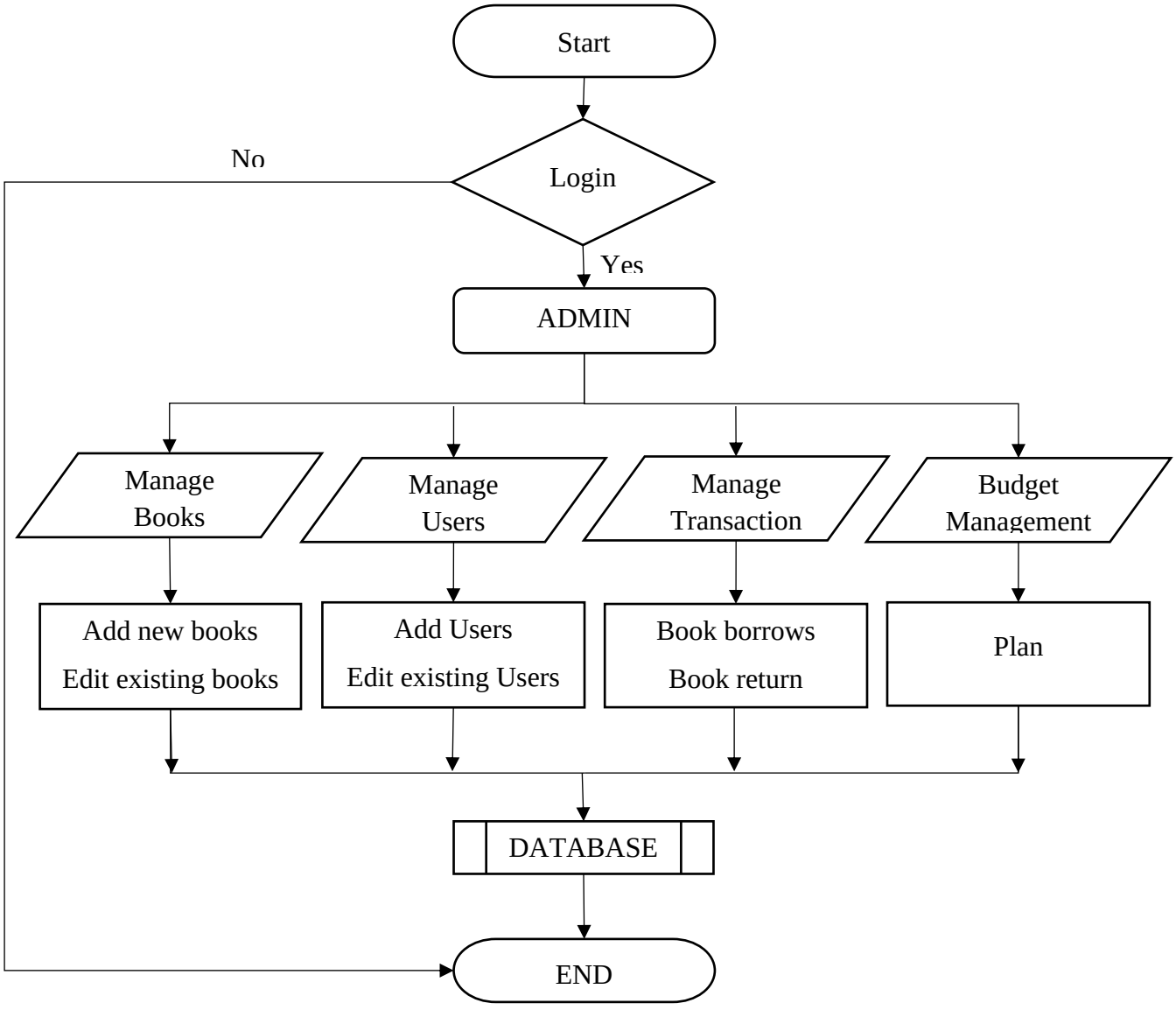

fig 3.1 Admin Flow diagram

The flowchart, shown in Figure 3.2, illustrates the Department Library Management System and its book borrowing process.

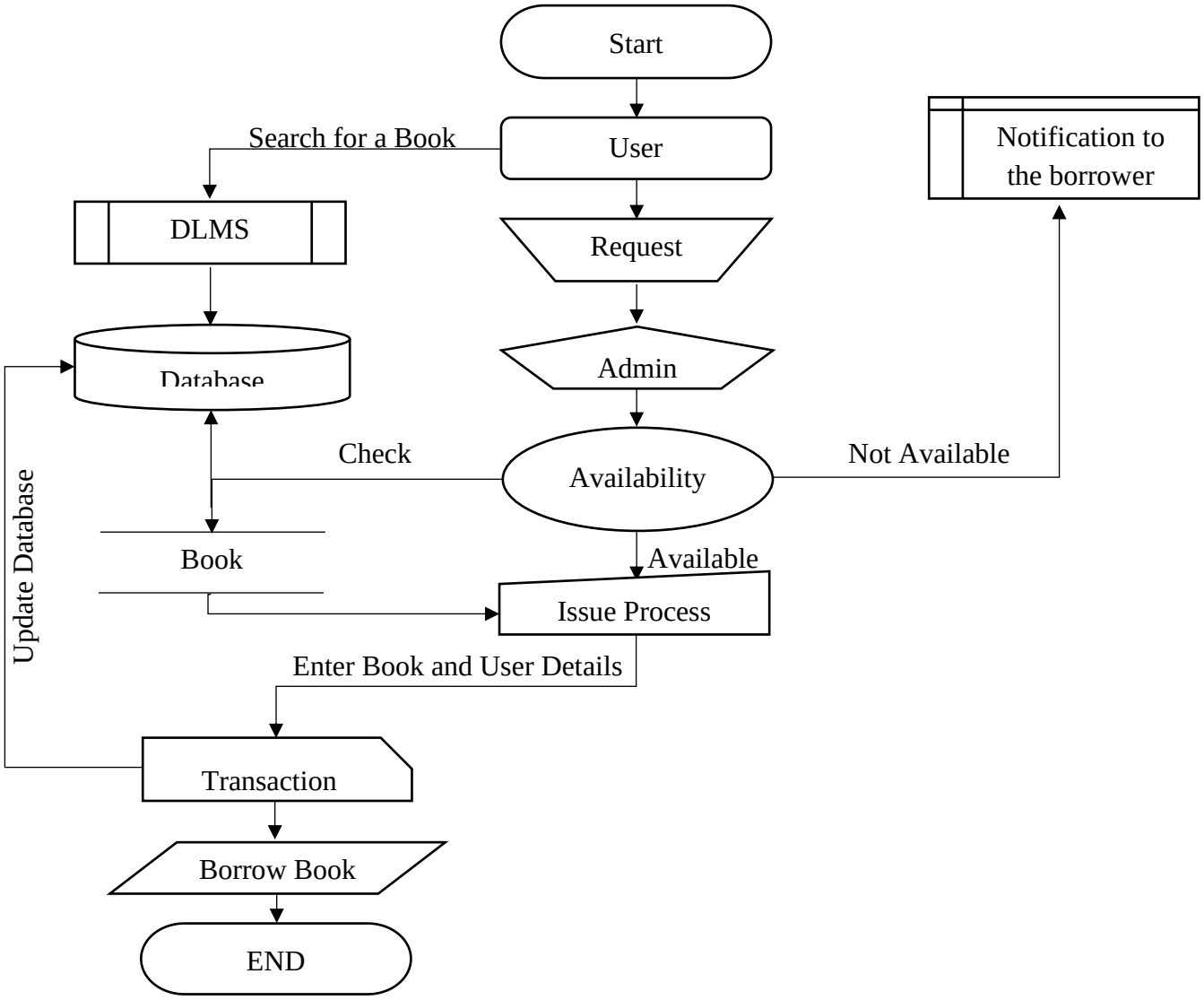

fig 3.2 Borrow book flow diagram

The flowchart presented in Figure 3.3, the Department Library Management System is described, highlighting the book returning process

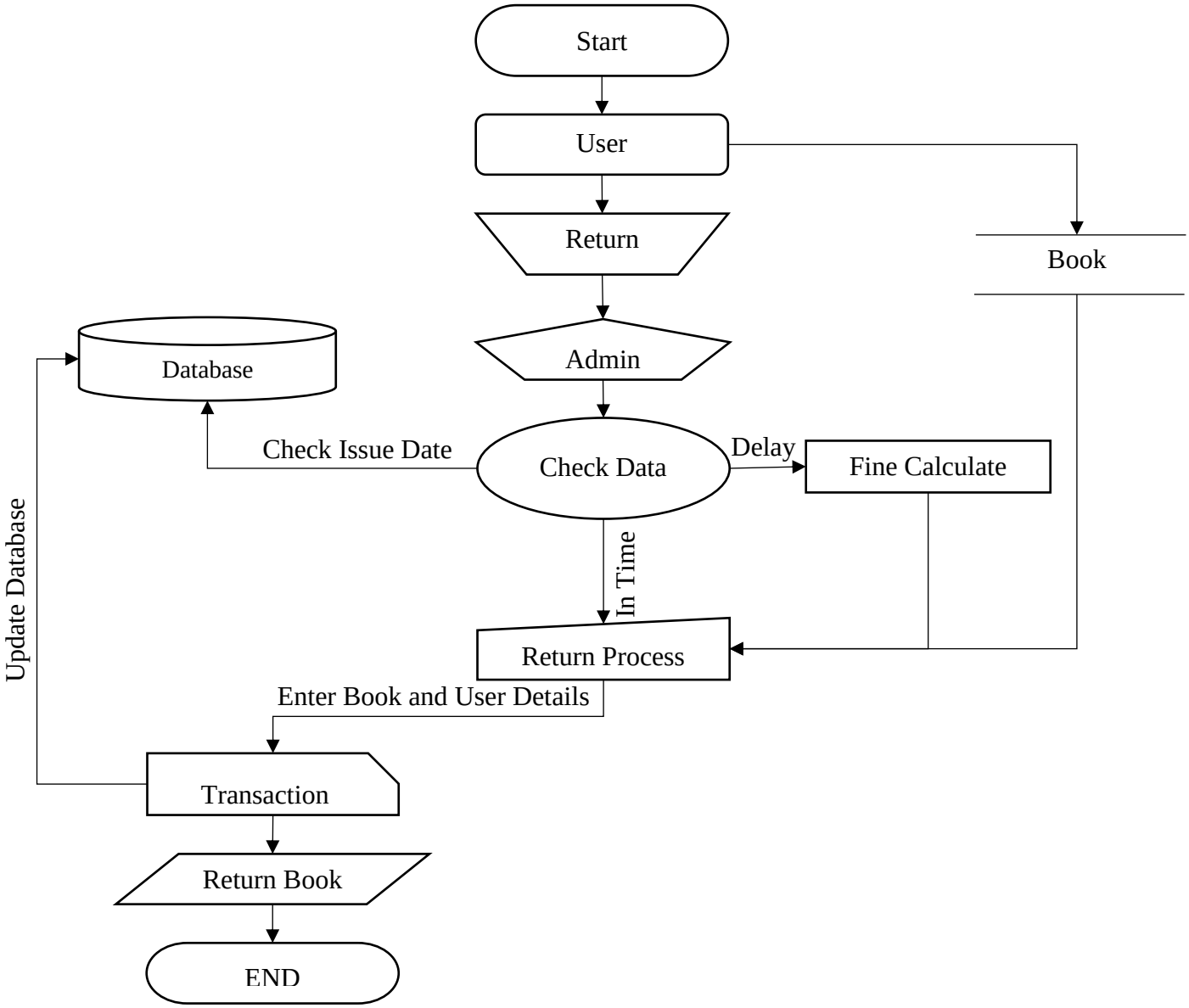

fig 3.3 Book Return flow diagram

3.2 MODULES DESCRIPTION

- Login Module
- Book borrow and return Module
- Books Management Module
- Users Management Module
- Budget Management Module
- Data Upload Module (Books and Users)

Login Module:

In this module Student and Students can login the account though the valid login credentials which is provided by the admin. Also, student can update his/her profile like mobile number, password etc.

- Admin Login
- Student Login
- Faculty Login

Book Borrow and Return Module:

The "Book Borrow and Return" module is an essential component of the library system, enabling students and faculty to borrow and return books efficiently. It ensures the smooth circulation of materials and streamlines daily library operations. This module enhances user experience by providing a straightforward process for accessing books and maintaining accountability for borrowed items, benefiting both library staff and patrons.

Books Management Module

The "Books Management Module" is the central hub of the library system, empowering librarians to catalog, organize, and upkeep the vast book collection efficiently. This module simplifies book-related tasks like categorization, labeling, and real-time availability tracking. It ensures that books are well-organized, readily accessible, and in prime condition for users. Additionally, the module's reporting and analytics feature aids librarians in making informed decisions about the library's book collection, ultimately enhancing the library experience for students, faculty, and other patrons.

Users Management Module

The "Users Management Module" serves as a critical component in the Library Management System. This module is indispensable for organized and secure user administration, empowering librarians to manage access, communicate effectively, and create a user-friendly environment, ultimately benefiting the entire library community.

Budget Management Module

The "Budget Management Module (Plan)" is essential for maintaining financial transparency, controlling expenditures, and making informed decisions about resource allocation in the library. It ensures that the library operates within its financial means while striving to enhance its offerings and services.

Data Upload Module (Books and Users)

This module provides a user-friendly interface for librarians and administrators to upload bulk data, such as information about books and user records, into the library system.

3.3 TABLE DESIGN

In department library management system 6 tables are created for the proposed project. They are:

- Admin Table
- Borrow Books Table
- Books Table
- Lib Planning Table
- Books Table
- Lib Planning Table
- Staff Table
- Student Table

In the proposed Department Library Management System, multiple tables have been created to efficiently handle data. These tables are essential for tasks such as tracking book borrowings, cataloging books, managing administration, and storing user information. This structured approach ensures data integrity and enables streamlined library management.

Admin Table:

This table may store information about administrators or librarians who manage the library system. It could include data like names, login credentials, and permissions.

Borrow Books Table:

This table likely records information about borrowed books, including details of the books, the borrowers, due dates, and any fines or penalties associated with late returns.

Books Table:

The Books Table would store comprehensive data about the library's collection, including book titles, authors, categories, availability, and more.

Lib Planning Table:

This table might be used for budget planning and tracking. It could store financial data related to the library's budget, expenses, and allocations.

Staff Table:

The Staff Table would likely contain information about library staff, including librarians, support personnel, and their contact details.

Student Table:

This table is probably used to maintain records of students who are library users. It may include student IDs, names, contact information, and borrowing histories.

Settings Table:

The Settings Table could store various settings and configurations for the library management system, allowing administrators to customize the system to their specific needs.

TABLE 3.3.1 ADMIN TABLE

Admin Table	
Column	**Type**
Aname(Primary key)	Varchar
Apass	Varchar
Amail	Varchar
Role	Char

Refer Appendix fig. 3.3.1

Aname: Stands for "Admin Name" and stores the name of an administrator.

Apass: Represents "Admin Password" and stores the password for the administrator.

Amail: Refers to "Admin Email" and stores the email address of the administrator.

Role: Denotes the "Role" and specifies the role or designation of the administrator, typically as a character code.

TABLE 3.3.2 BORROW BOOKS TABLE

Borrow Books Table	
Column	**Type**
Sbid (Primary key)	Int
Sid (Foreign key)	Int
Bid (Foreign key)	Int
Request_date	Timestamp
Return_date	Date
Returned_date	Datetime
Is_returned	Int
Role	Varchar
Remark	Varchar
Status	Int

Refer Appendix fig. 3.3.2

Sbid: Stands for "Student Borrow ID" and stores a unique identifier for each student's book borrowing record.

Sid: Represents "Student ID" and stores the unique identifier for each student.

Bid: Refers to "Book ID" and stores the unique identifier for each book in the library.

Request_date: Records the date and time when a book borrowing request is made.

Return_date: Stores the expected return date for the borrowed book.

Returned_date: Records the date and time when the borrowed book is returned.

Is_returned: Indicates whether the book has been returned (1 for returned, 0 for not returned).

Role: Denotes the role or designation related to the borrowing transaction.

Remark: Provides a field for adding comments or remarks related to the borrowing.

Status: Represents the status of the borrowing transaction

TABLE 3.3.3 BOOKS TABLE

Books Table	
Column	**Type**
Bid (Primary key)	Int
Bno	varchar
plan_id	Int
Bcode	varchar
Title	varchar
Aname	varchar
Publication	varchar
Price	varchar
year_of_publication	varchar
Language	varchar
Edition	varchar
shelf_id (Foreign Key)	Int
Remark	varchar
Status	Int

Refer Appendix fig. 3.3.2

Bid: Stands for "Book ID" and stores a unique identifier for each book in the library.

Bno: Represents the "Book Number" and stores a unique identification code for each book.

Plan_id: Refers to "Plan ID" and stores the unique identifier for a particular book acquisition plan.

Bcode: Denotes the "Book Code" and stores a code or identifier associated with the book.

Title: Stores the title of the book.

Aname: Represents the "Author Name" and stores the name of the book's author.

Publication: Records the name of the publisher of the book.

Price: Stores the price or cost of the book.

Year_of_publication: Records the year in which the book was published.

Language: Denotes the language in which the book is written.

Edition: Stores information about the edition of the book.

Shelf_id: Stands for "Shelf ID" and stores the unique identifier for the shelf where the book is located.

Remark: Provides a field for adding comments or remarks related to the book.

TABLE 3.3.4 LIB_PLANNING TABLE

Lib_Planning Table	
Column	**Type**
Id (Primary key)	Int
Category	Varchar
Year	Varchar
plan_status	Varchar
Billno	Varchar
Noofbooks	Int
Amount	Int
Remark	Varchar
Status	Int

Refer Appendix fig. 3.3.2

Id: Stores a unique identifier associated with the record.

Category: Represents the category or type related to the record.

Year: Records the year or time frame relevant to the record.

Plan_statue: Indicates the status of the plan, such as "approved," "pending," or "completed."

Billno: Stores the bill number associated with the record.

Noofbooks: Represents the number of books related to the record.

Amount: Records the monetary amount or cost associated with the record.

Remare: Provides a field for adding comments or remarks related to the record.

Status: Represents the status of the record.

TABLE 3.3.5 STAFF TABLE

Staff Table	
Column	**Type**
Sid (Primary key)	Int
Regno	Varchar
Spass	Varchar
Sname	Varchar
Semail	Varchar
contact	Varchar

gender	enum('male','female')
Image	varchar
Validity	date
Remark	Varbinary

Refer Appendix fig. 3.3.2

Sid: Represents "Student ID" and stores a unique identifier for each student.

Regno: Stands for "Registration Number" and stores the registration number of the student.

Spass: Stores the password associated with the student's account.

Sname: Records the name of the student.

Semail: Stores the email address of the student.

contact: Stores the contact or phone number of the student.

gender: Represents the gender of the student and is typically limited to 'male' or 'female.'

Image: Stores the path or reference to the student's image or photo.

Validity: Records the date until which the student's account is valid.

Remark: Provides a field for adding binary data or additional remarks related to the student.

TABLE 3.3.6 SETTINGS TABLE

Settings Table	
Column	**Type**
Id (Primary key)	Int
app_decp	varchar
app_logo	varchar
Fine	Int
fine_stf_days	Int
fine_std_days	Int
smtp_host	varchar
smtp_port	varchar
smtp_user	varchar
smtp_pass	varchar
smtp_sec_type	enum('ssl','tls')

Refer Appendix fig. 3.3.2

Id: Stores a unique identifier associated with the record.

app_decp: Represents the description or name of the application.

app_logo: Stores the path or reference to the application's logo or icon.

Fine: Records the amount of fine imposed for overdue items.

fine_stf_days: Indicates the number of days allowed for staff members to return items before incurring fines.

fine_std_days: Indicates the number of days allowed for students to return items before incurring fines.

smtp_host: Stores the hostname or address of the SMTP (email) server.

smtp_port: Records the port number used for SMTP server communication.

smtp_user: Stores the username for authenticating with the SMTP server.

smtp_pass: Stores the password for authenticating with the SMTP server.

smtp_sec_type: Represents the security type used for SMTP communication, which can be either 'ssl' or 'tls'.

SYSTEM DESIGN:

3.4 Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system.

The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- Login Form
- Borrow form
- Return form
- Add book form
- Add student form
- Add staff form

3.4.1 Login form

A login form is a simple interface where users provide their credentials, typically a username and password, to access a system or application. Refer Appendix 3.4.1 Login form.

3.4.2 Borrow form

This form is used when a user, such as a student or faculty member, wants to borrow a book from the library. It includes fields for entering book details, borrower information, and due date. Refer Appendix 3.4.2 Borrow form.

3.4.3 Return form

The return form is utilized when a user returns a book to the library. It collects information about the returned book, the condition of the book, and any fines if applicable. Refer Appendix 3.4.3 Return form.

3.4.4 Add book form

This form is used by library staff to add new books to the library's collection. It includes fields for entering book details like title, author, category, and more. Refer Appendix 3.4.4 Add book form.

3.4.5 Add student form

Library administrators use this form to add new students to the system. It typically includes fields for student information such as name, registration number, and more. Refer Appendix 3.4.5 Add student form.

3.4.6 Add staff form

Library administrators or staff can use this form to add new staff members to the system. It includes fields for staff information, including their name, role, contact details, and more. Refer Appendix 3.4.6 Add staff form.

3.5 OUTPUT DESIGN:

- A quality output is one, which meets the requirements of the end user and presents the information clearly.
- In any system results of processing are communicated to the users and to other system through outputs.
- In output design it is determined how the information is to be displaced for immediate need and also the hard copy output.
- It is the most important and direct source information to the user.
- Efficient and intelligent output design improves the system’s relationship to help user decision-making

TESTING AND IMPLEMENTATION

CHAPTER-4

TESTING AND IMPLEMENTATION

4.1 SYSTEM TESTING

System testing is the state of implementation, which is aimed at ensuring that the System Works accurate and efficient as expect before, live operation, commences. It certifies that the whole set of programs hang together system testing requires a test plan, that consist of several key activities and step for run program, string, system and user acceptance testing. the implementation of newly design package is important in adapting a successful new system

Testing is important stage in software development system test is implementation should be as confirmation that all is correct and opportunity to show the user that the system works as they expected it accounts the largest percentage of technical effort in software development process

Testing phase development phase that validates the code against the functional specification. testing is a vital to the achievement of the system goals the objective of testing is to discover errors, to fulfill this objective a series of test step such as the unit test, integration, validation and system test were planned and executed.

UNIT TESTING

Unit testing, also known as module testing, is a critical phase in the software development process. In this project, the following modules have undergone unit testing:

- **Login module**
    - This module was tested with valid usernames and passwords, ensuring that user authentication functions correctly.
- **Borrow module**
    - The unit testing for this module focused on validating the tracking of book availability during the borrowing process.
- **Upload module**
    - This module was tested to verify and ensure the successful import of data.

INTEGRATION TESTING

Integration testing evaluates the efficiency of the interfaces between various modules in the system. In this project, integration testing has been conducted for the following modules.

- **Admin Module**
    - The integration testing for this module assessed the interactions between different functions, such as login, data insertion, updating records, tracking, and removal of items.

VAILIDATION TESTING

Software validation is achieved through a series of test that demonstrates the conformity and requirement thus the proposed system under consideration has to be tested by validation and found to be working satisfactorily.

- **User Authentication**
    - Verify that user logins work correctly, and users can access the system using valid credentials while being denied access with invalid or unauthorized ones.

- **Book Borrowing and Returning**
    - Test the book borrowing and returning processes to ensure that they function seamlessly, and users can borrow and return books without issues.

- **Data Import**
    - Validate the data upload module's ability to import book and user data from external sources accurately, without data loss or corruption.

4.2 SYSTEM IMPLEMENTATION

Implementation is the final and important phase, the most critical stage in achieving a successful new system and giving the user confidence. That the new system will work be effective. The system can be implemented only after through testing is done and if it found to working according to the specification.

CONCLUSION

CHAPTER-5

CONCLUSION

Conclusion

- The Department Library Management System streamlines library operations and enhances efficiency.
- Users, including students and faculty, benefit from a user-friendly interface and simplified book borrowing processes.
- User data is securely managed, ensuring privacy and data protection.
- Efficient budget management optimizes resource allocation, contributing to financial transparency.
- Reporting and analytics tools provide valuable insights for informed decision-making.
- Automation and data upload modules reduce manual workloads, saving time for staff and users.
- Effective communication through the notification system enhances user engagement.
- The system accommodates library growth through scalability.
- Resource accessibility is improved, ensuring that library materials are readily available.
- User accountability is encouraged, reducing the risk of loss or damage to library items.
- Data analysis capabilities provide valuable data-driven insights.
- The project marks a significant step forward in library modernization, contributing to a more effective library management system.

FUTURE ENHANCEMENTS

Mobile Application

Develop a mobile app for convenient access to the library system on smartphones and tablets, offering on-the-go services.

RFID Technology

Integrate RFID technology for automated book check-in and check-out, further improving the efficiency of library operations.

Digital Library Integration

Incorporate e-books and digital resources into the system, expanding the library's digital collection.

Interactive User Portal

Develop an interactive user portal for user-driven content, such as book reviews, ratings, and discussion forums.

Data Backup and Recovery

Implement robust data backup and recovery solutions to safeguard against data loss.

Multi-Library Integration

If applicable, allow for the integration of multiple Department Libraries within the same institution, making it easier for users to access materials from different departments.

Enhanced Reporting

Develop more advanced reporting and analytics tools for deeper insights into library usage and trends.

REFERENCE

1. Doyle, Matt. Beginning PHP 5.3 (Wrox Programmer to Programmer), 2009.
2. Lerdorf, Rasmus. PHP Pocket Reference, O'Reilly, 2000.
3. Professional CodeIgniter Paperback – Import, 25 July 2008.
4. Nixon, Robin. Learning PHP, MySQL, JavaScript, and CSS: A Step-by-Step Guide to Creating Dynamic Websites, O'Reilly, 2012.
5. Efficient MySQL Performance: Best Practices and Techniques 1st Edition. O'Reilly Media; 1st edition (January 4, 2022)
6. Welling, Luke and Thomson, Laura. PHP and MySQL Web Development, Third Edition, Sams, 2008.

OUTLINE LINKS

- https://github.com/
- https://www.codeigniter.com/userguide3/
- https://www.tutorialspoint.com/codeigniter/index.htm
- https://www.javatpoint.com/codeigniter-tutorial

APPENDIX

ADMIN TABLE:

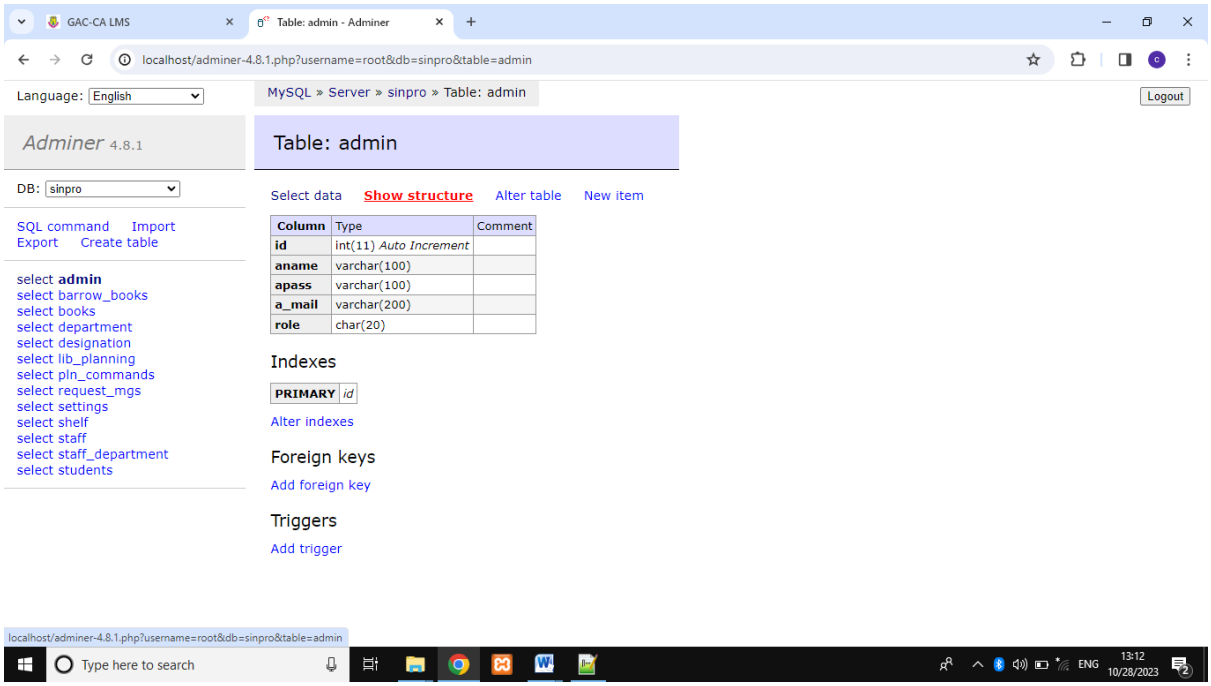

Fig 3.3.1 Admin Table

BORROW TABLE:

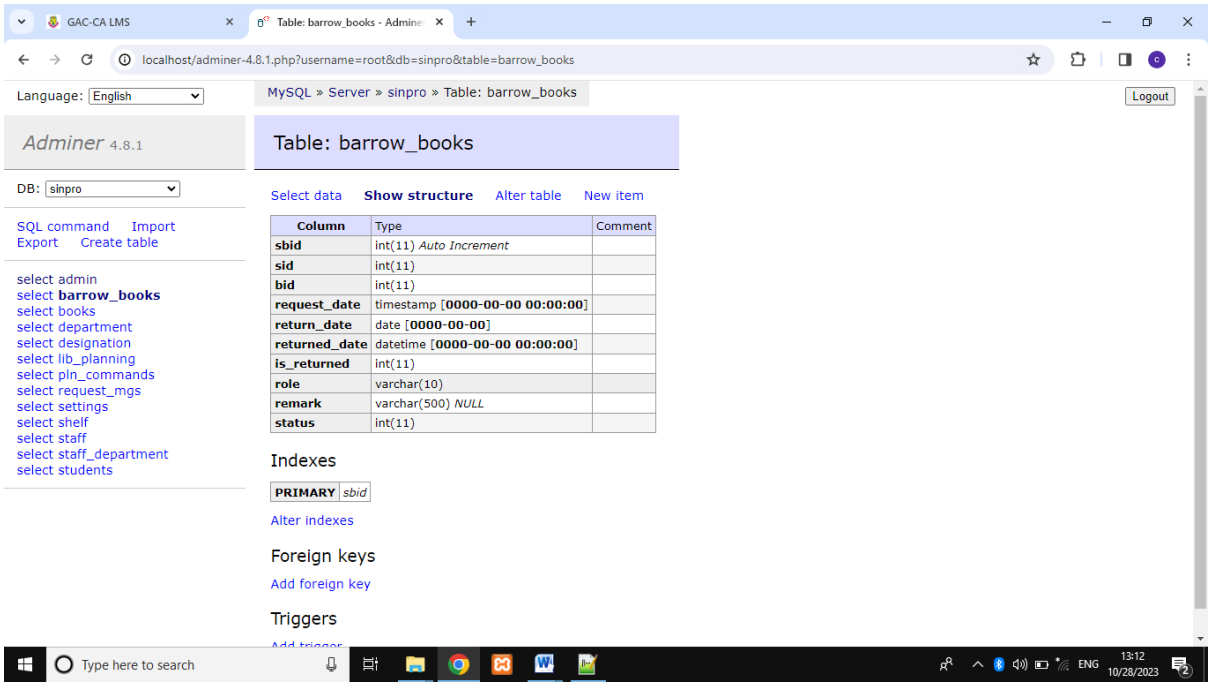

Fig 3.3.2 Borrow Table

BOOK TABLE:

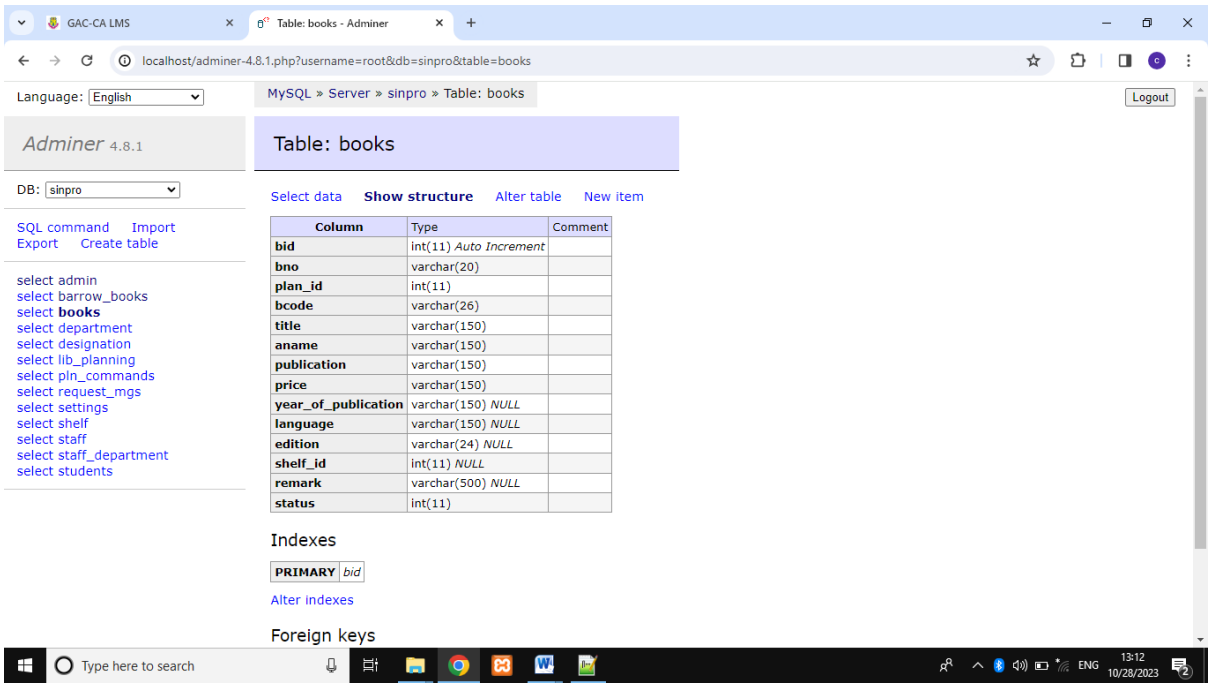

Fig 3.3.3 Admin Table

LIBRARY PLAN TABLE:

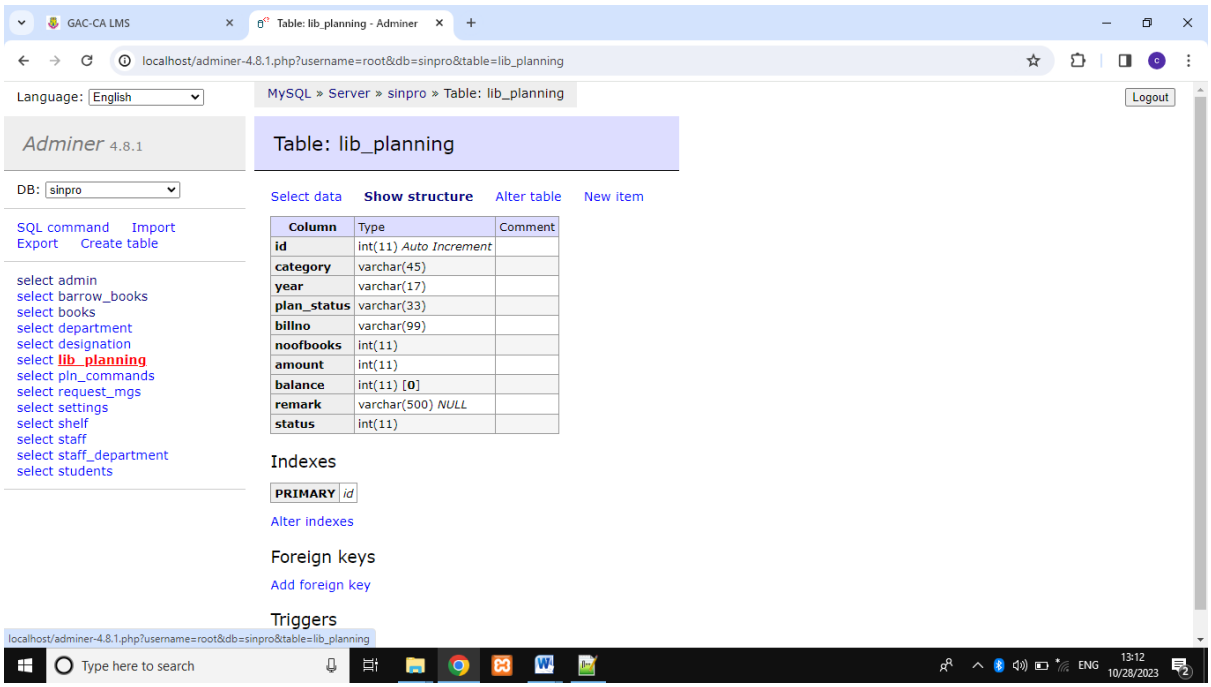

Fig 3.3.4 Library Plan Table

STAFF TABLE:

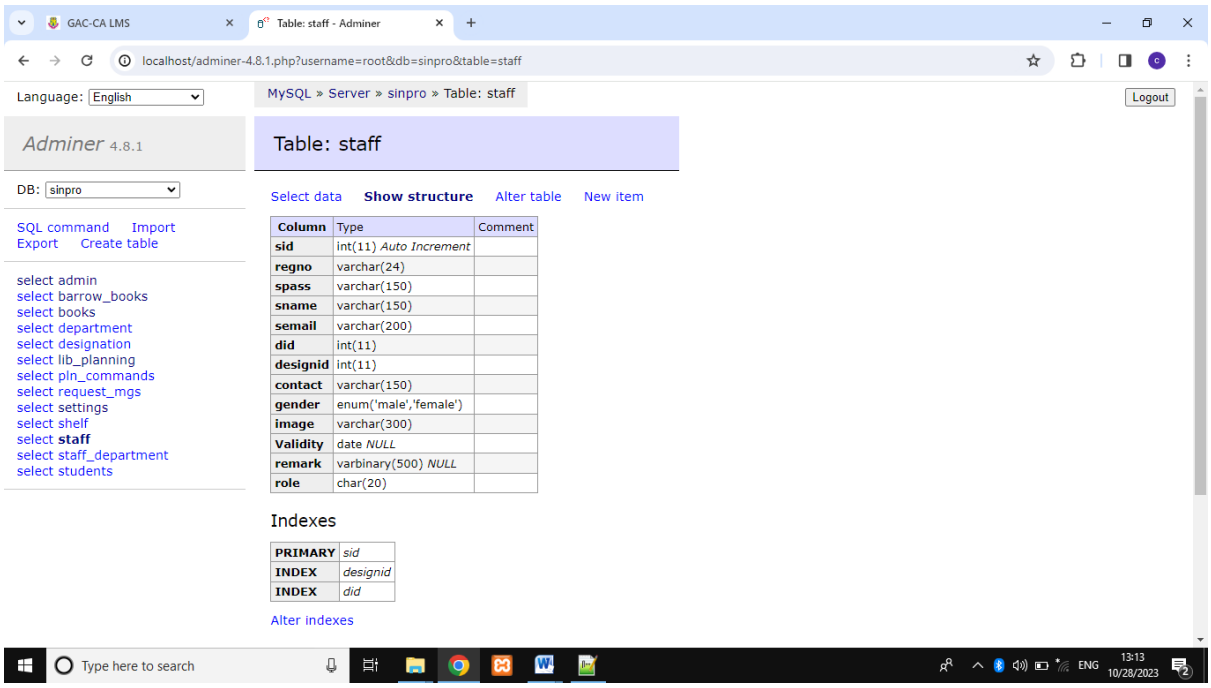

Fig 3.3.5 Admin Table

STUDENT TABLE:

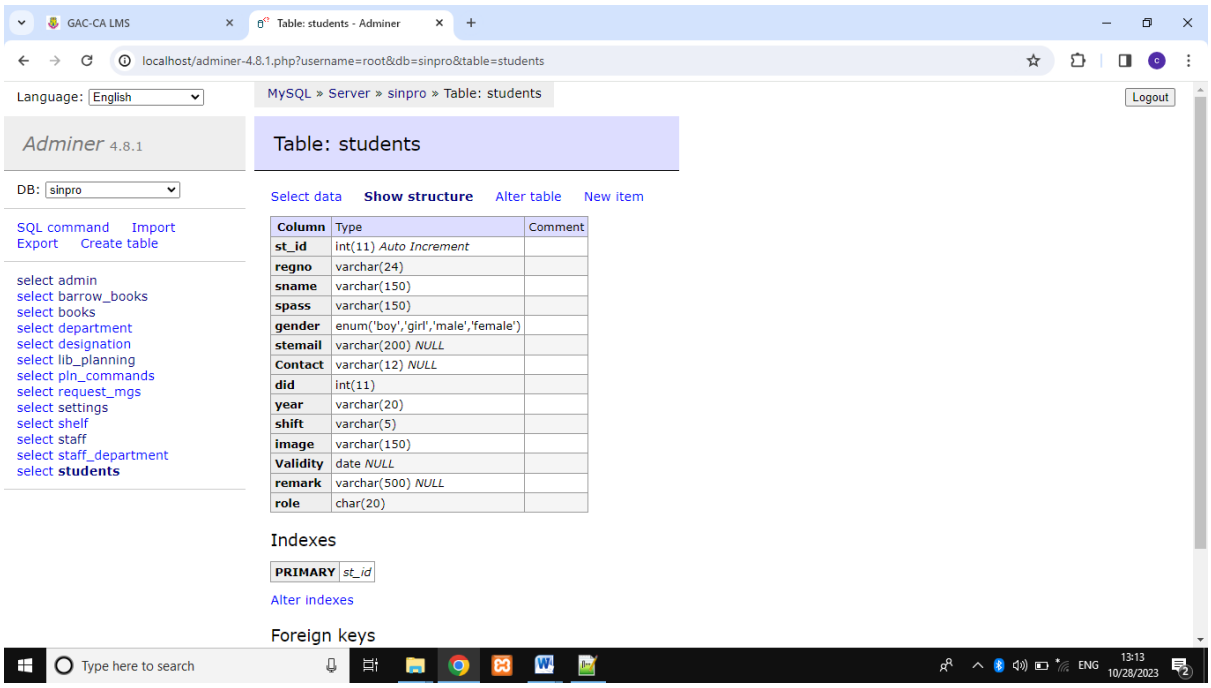

Fig 3.3.6 Admin Table

SETTINGS TABLE:

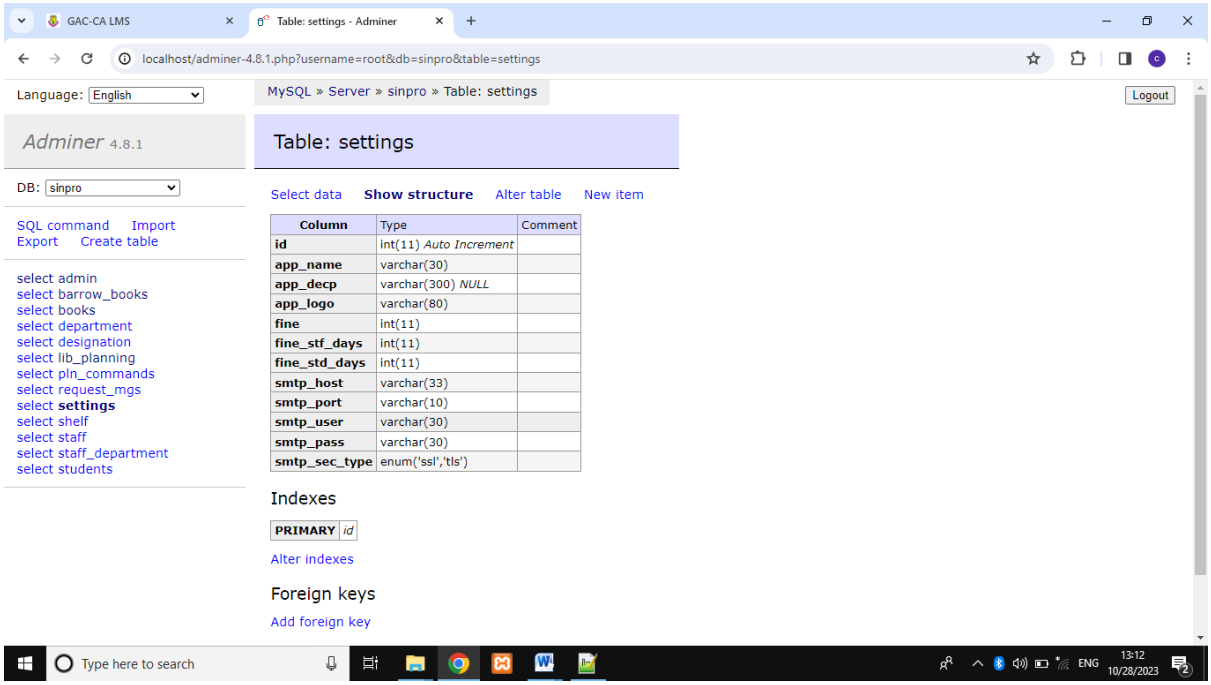

Fig 3.3.7 Admin Table

LOGIN PAGE:

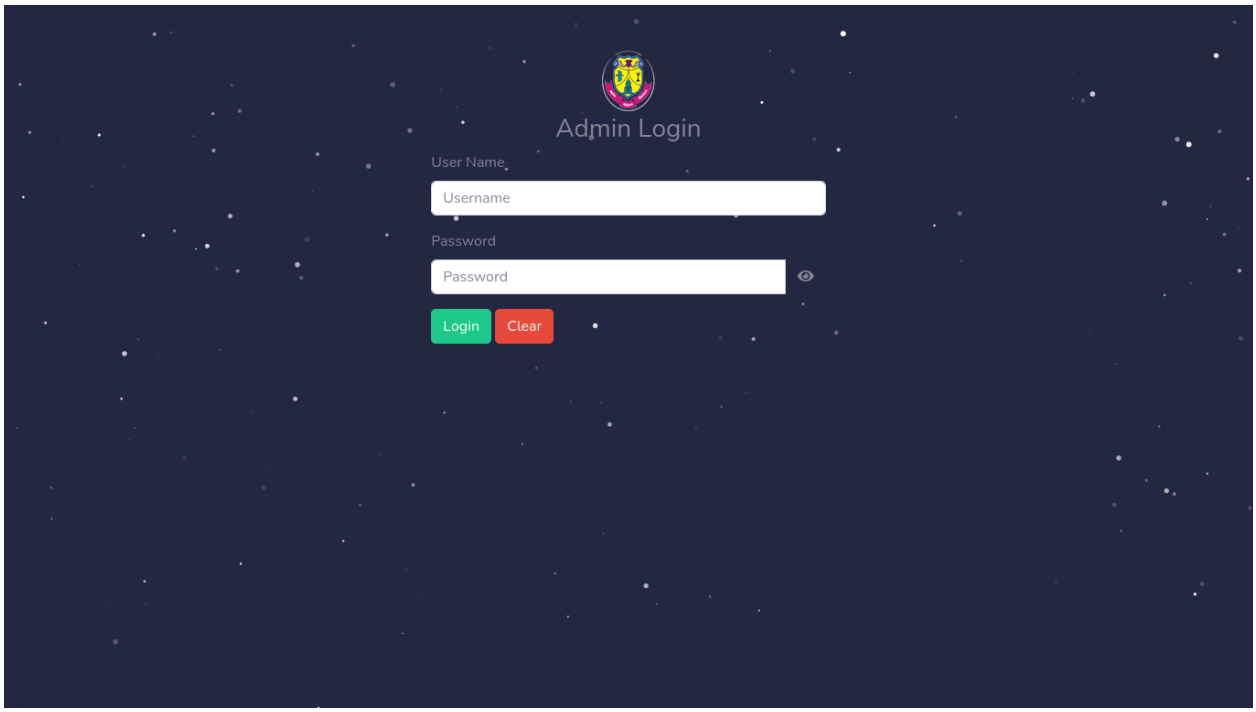

Fig 3.4.1 Login page for Admin

BORROW FORM:

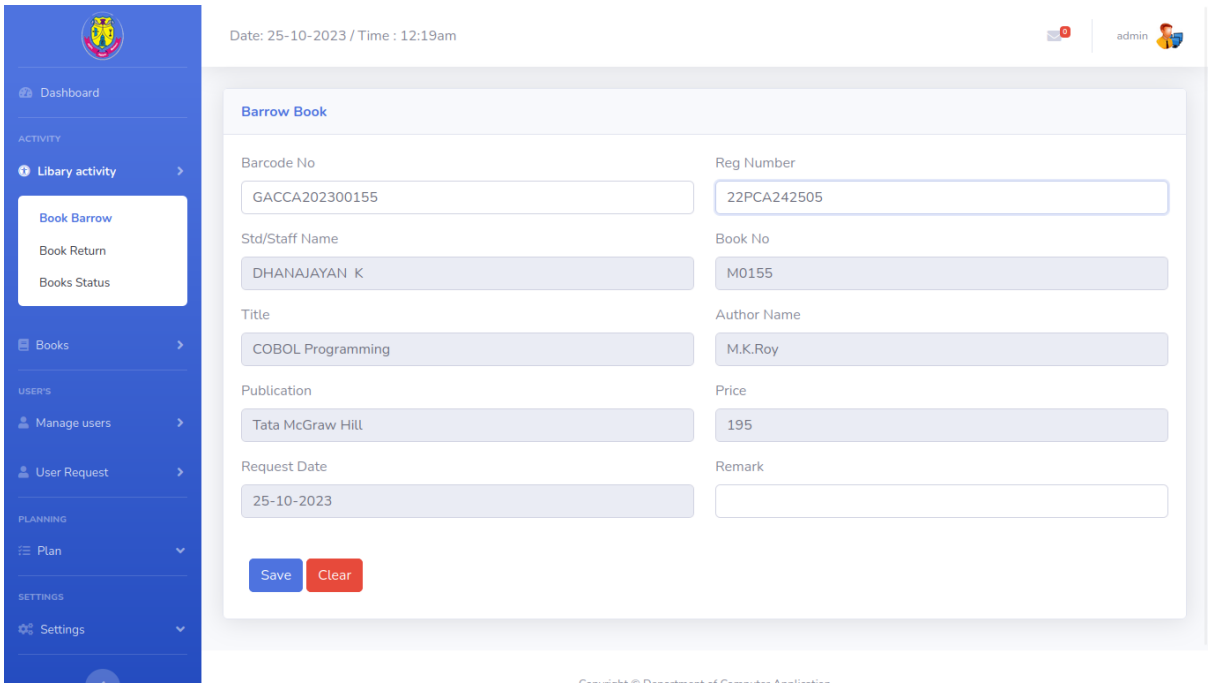

Fig 3.4.2 Borrow Form

RETURN FORM:

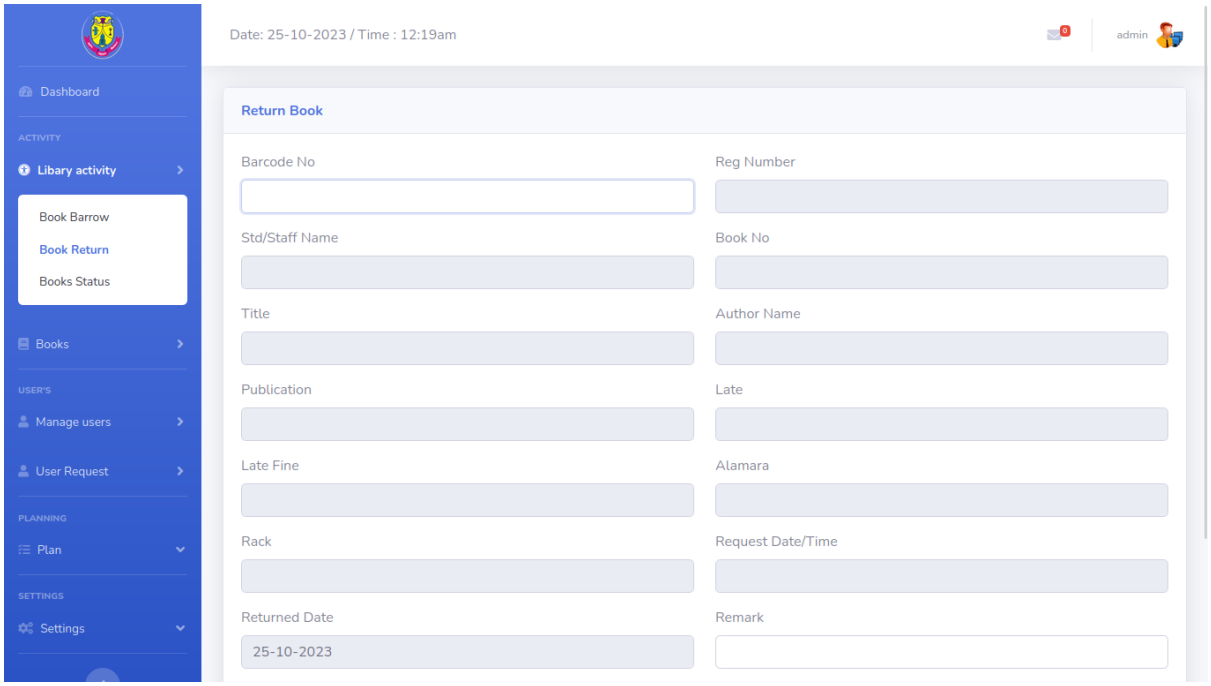

Fig 3.4.3 Borrow Form

ADD BOOK FORM:

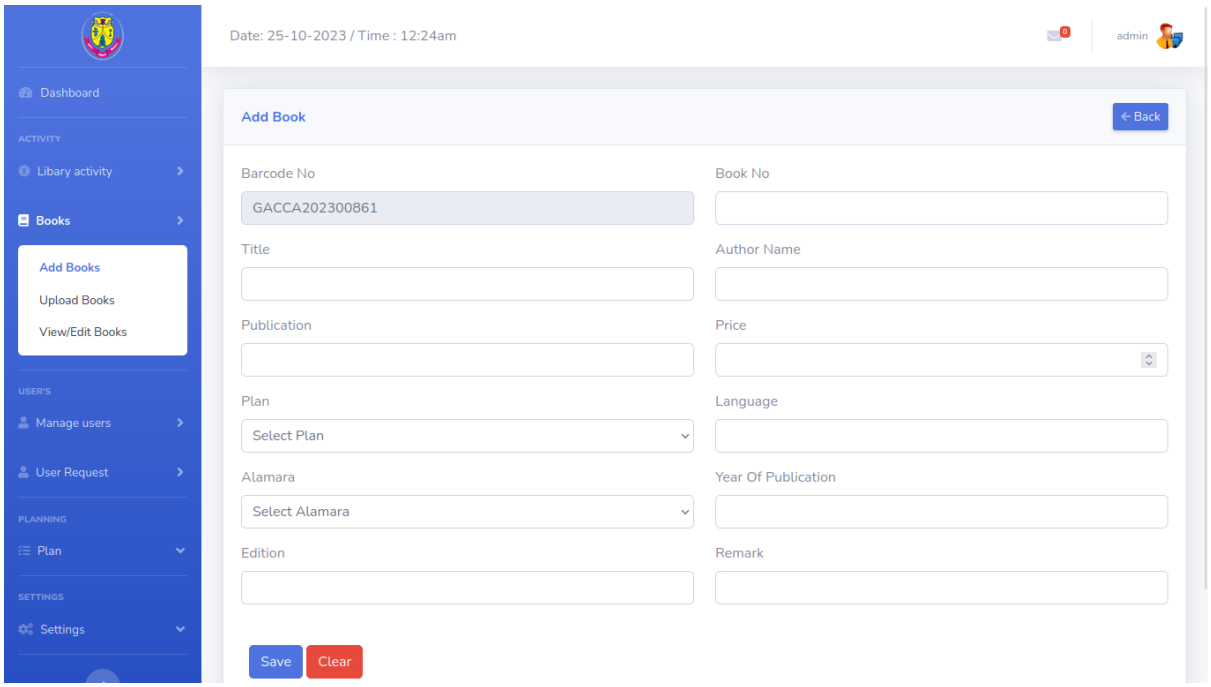

Fig 3.4.4 Add Book Form

ADD STUDENT FORM:

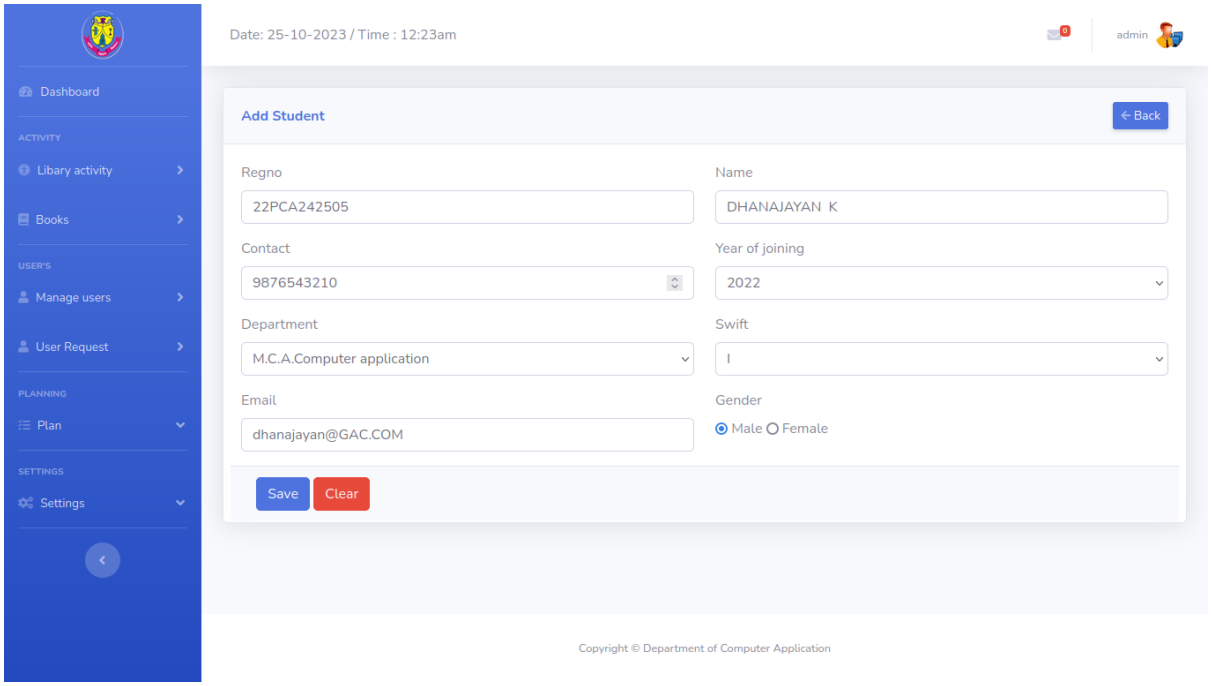

Fig 3.4.5 Add Student Form

ADD STAFF FORM:

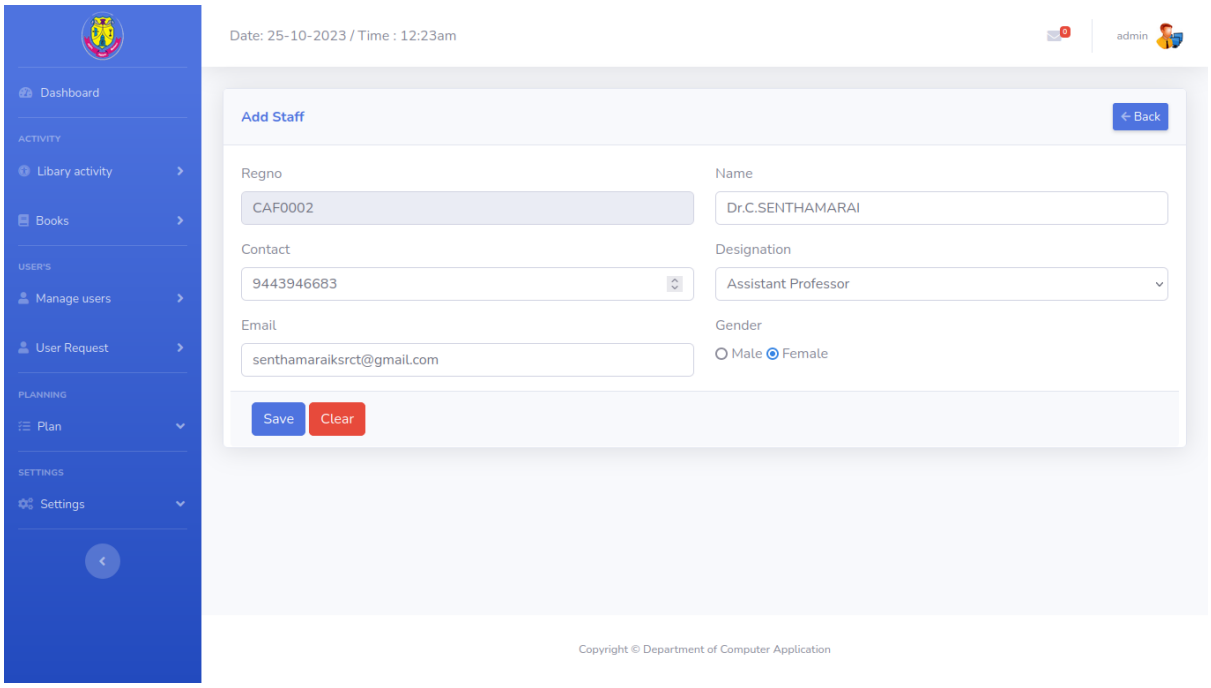

Fig 3.4.6 Add Student Form

ADMIN DASHBOARD:

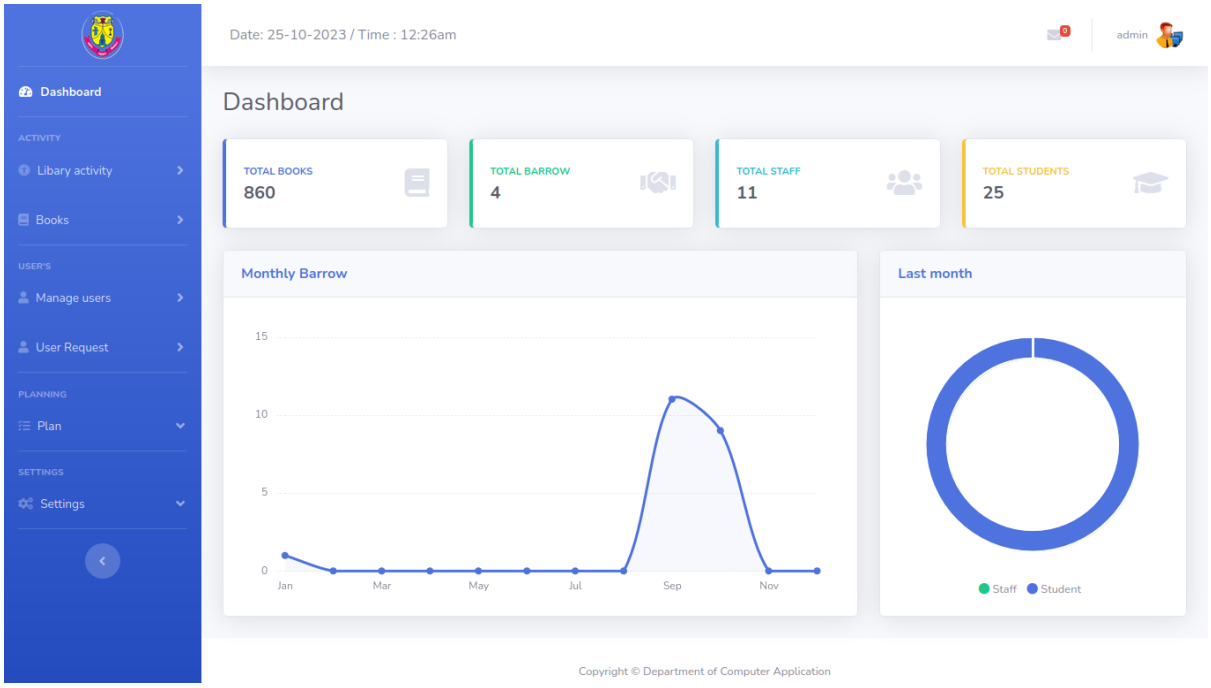

Fig 3.5.1 Admin dashboard

STUDENT LIST:

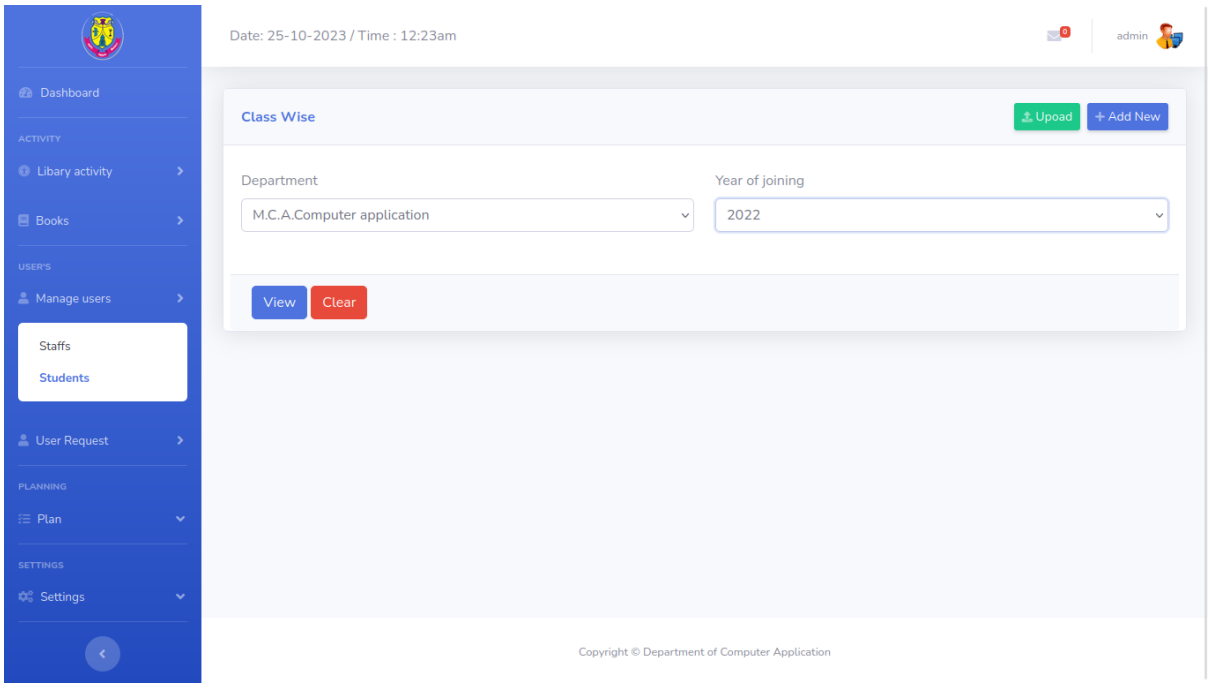

Fig 3.5.2 Student list section

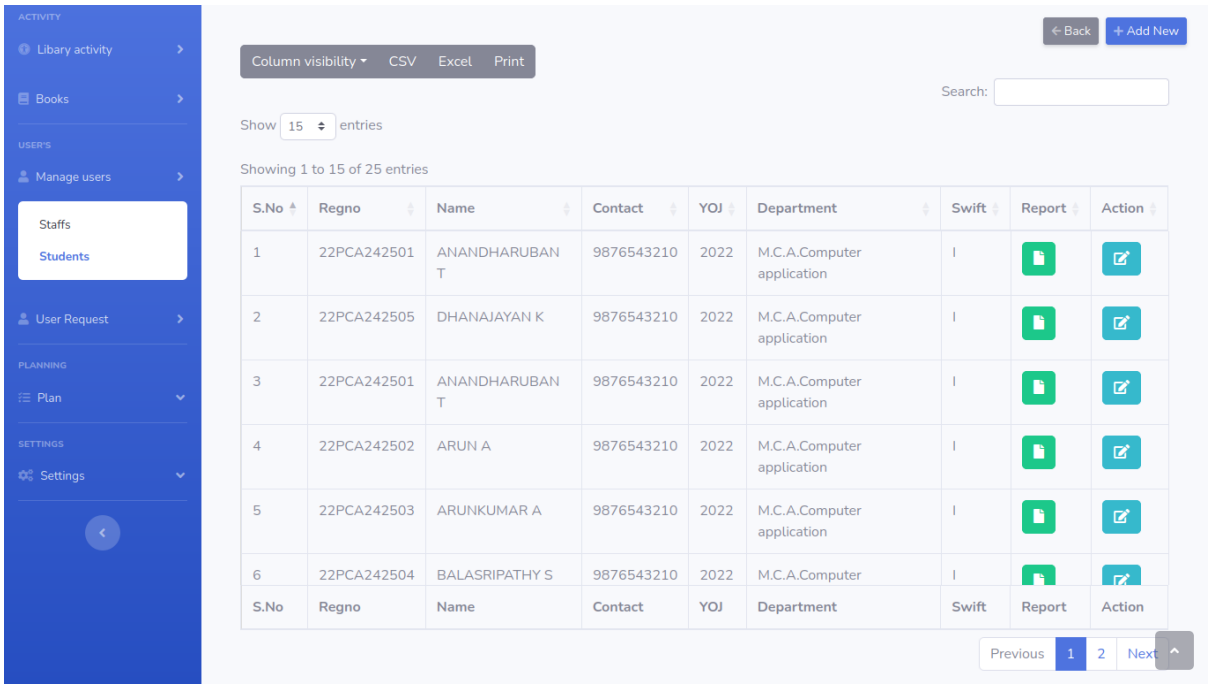

Fig 3.5.3 Student list

STAFF LIST:

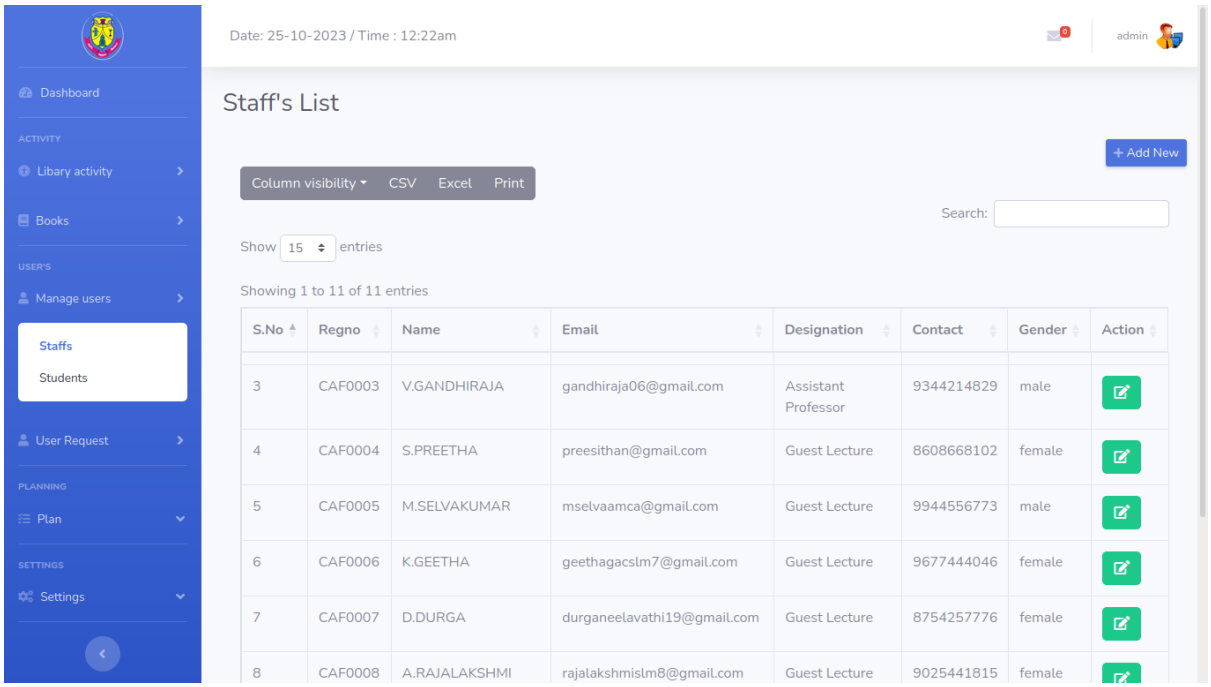

Fig 3.5.4 Staff's list

STAFF DASHBOARD:

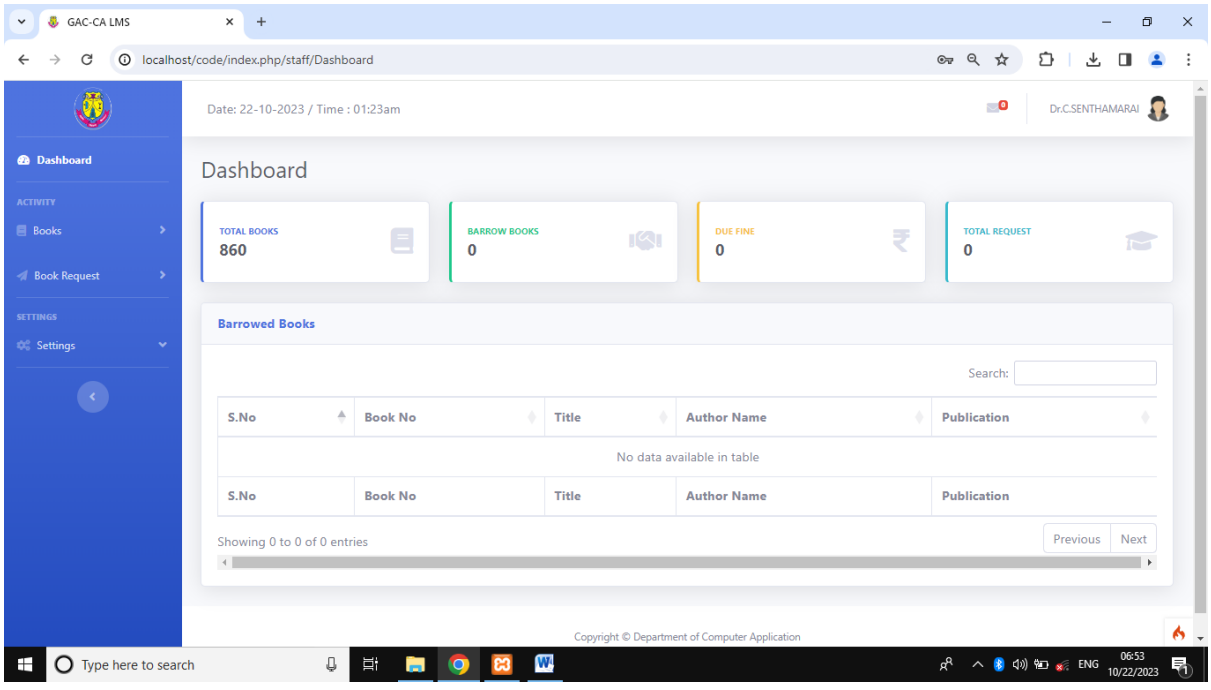

Fig 3.5.5 staff dashboard

BOOK LIST:

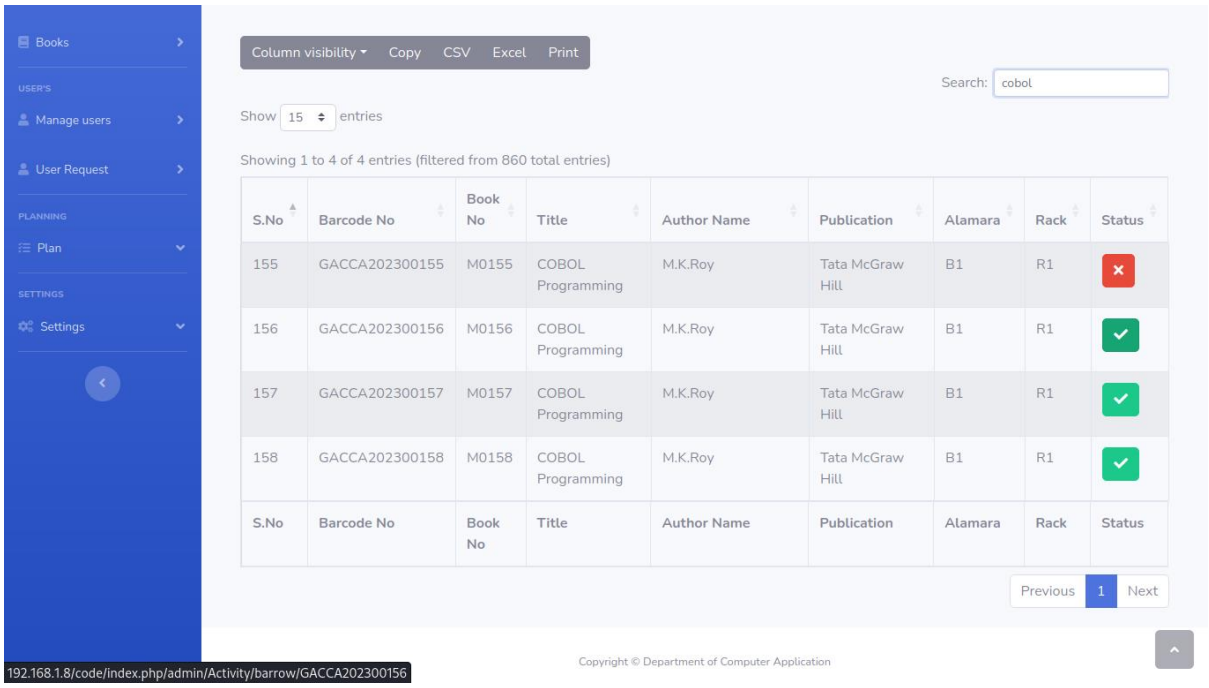

Fig 3.5.6 check the book list

PLAN LIST:

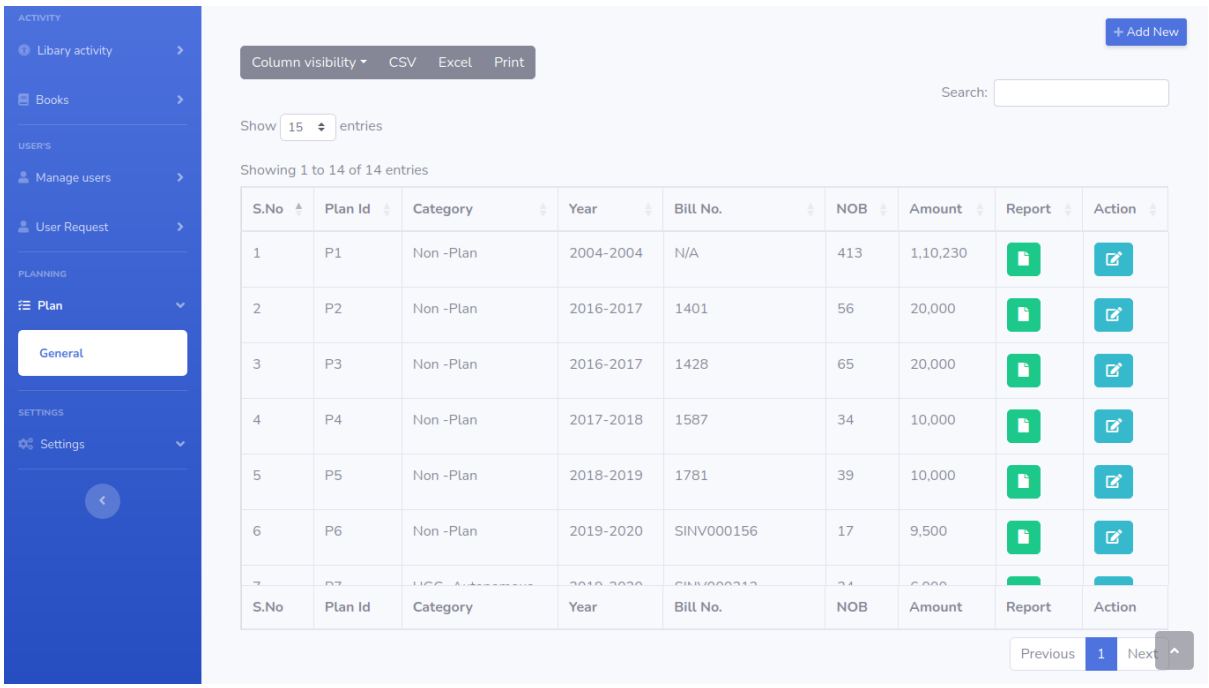

Fig 3.5.7 Budget plan list

BARROWED BOOKS LIST:

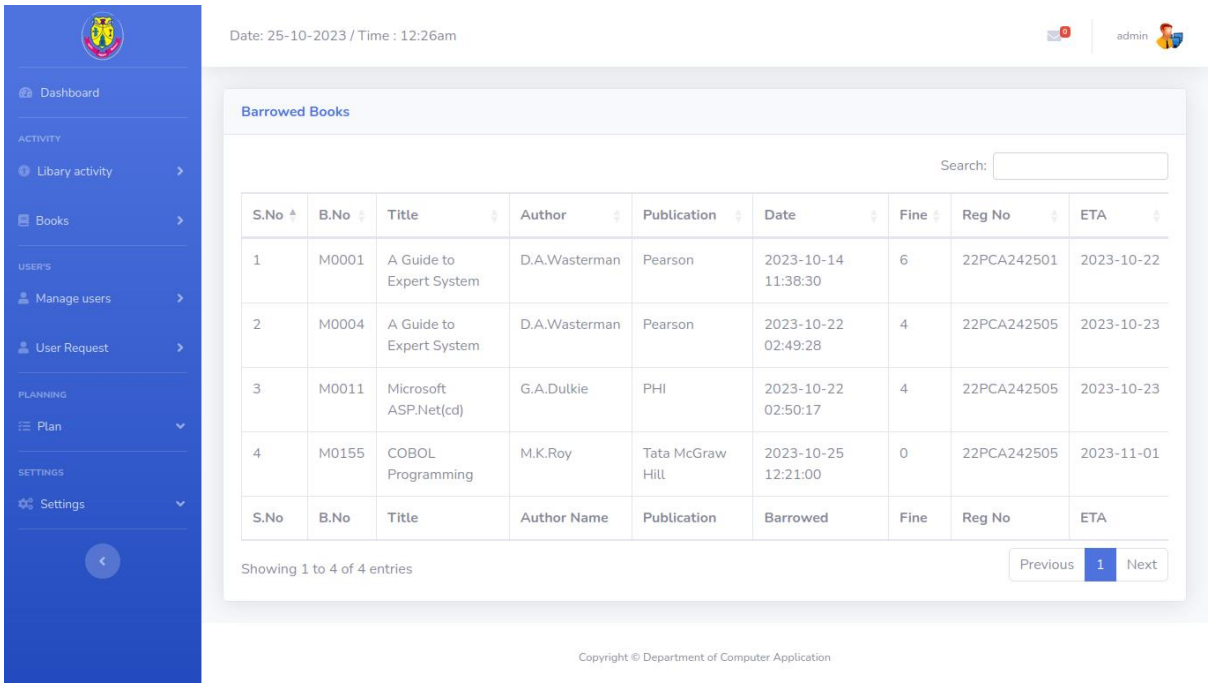

Fig 3.5.8 User and barrowed books list

BORROWED BOOK EMAIL:

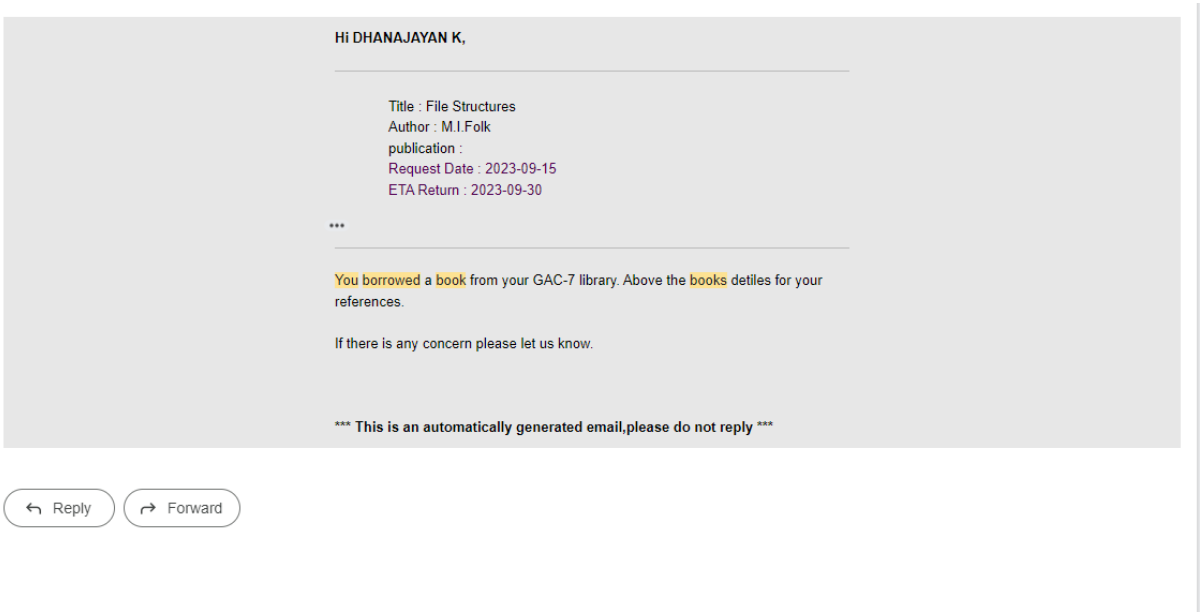

Fig 3.5.9 users borrowed notification mail

SAMPLE CODE

Activity.php:

```
<?php
namespace App\Controllers\admin;
use App\Controllers\BaseController;
use App\Models\BooksModel;
use App\Models\OtherModel;
use App\Models\BarrowBooksModel;
use App\Models\SettingsModel;
use App\Controllers\Mail;
use CodeIgniter\HTTP\RequestInterface;
use CodeIgniter\Validation\ValidationInterface;
class Activity extends BaseController
{
    public function __construct()
    {
        date_default_timezone_set("Asia/Kolkata");
        $this->booksModel = new BooksModel();
        $this->barrowbooksModel = new BarrowBooksModel();
        $this->otherModel = new OtherModel();
        $this->settingsModel = new SettingsModel();
        $this->mail = new Mail();
        // Inject the ValidationInterface into the controller
        $this->validation = \Config\Services::validation();
    }

    public function Barrow($var = null)
```

```
{
    $session = session();
    if ($session->get('role')!="admin") {
        return redirect()->to('/');
    }
    if($this->request->getMethod() === 'post'){
        $bcode = $this->request->getPost('bcode');
        $regno = $this->request->getPost('regno');
        $sname = $this->request->getPost('sname');
        $bno = $this->request->getPost('bno');
        $title = $this->request->getPost('title');
        $aname = $this->request->getPost('aname');
        $publication = $this->request->getPost('publication');
        $price = $this->request->getPost('price');
        $alamara = $this->request->getPost('alamara');
        $rack = $this->request->getPost('rack');
        $remark = $this->request->getPost('remark');
        // Set the validation rules
        $validationRules = [
            'bcode' => 'required',
            'regno' => 'required',
            'sname' => 'required',
            'bno' => 'required',
        ];

        // Set custom error messages for each field
        $validationMessages = [
```

```
            'bcode' => [
                'required' => 'The Barcode No field is required.'
            ],
            'regno' => [
                'required' => 'The Register Number field is required.'
            ],
            'sname' => [\
                'required' => 'The Std/Staff Name field is required.'
            ],
            'bno' => [
                'required' => 'The Book No field is required.'
            ]//,
        ];
        $validation = \Config\Services::validation();
        $validation->setRules($validationRules, $validationMessages);
        if($this->validation->withRequest($this->request)->run()){
            $res = $this->otherModel->getUserDet($regno);
            $book = $this->booksModel->getBookDetail($bcode);
            $Appfine = $this->settingsModel->getAppfine();
            $return_date = ($res['role']=="staff") ? date("Y-m-d", strtotime("+". $Appfine['fine_stf_days']." days")) : date("Y-m-d", strtotime("+". $Appfine['fine_std_days']." days")) ;
            $request_date = date("Y-m-d h:i:s");
            $data = [
                'sid' => $res['sid'],
                'bid' => $book['bid'],
                'role' => $res['role'],
                'returned_date' => '0000-00-00',
```

```
            'status' => 1,
            'is_returned' => 0,
            'return_date' => $return_date,
            'request_date' => $request_date,
            'remark' => $remark,
        ];
        if ($book['status']==1) {
            if($this->barrowbooksModel->setBarroeBook($data)){
                $this->booksModel->updateBook($bcode,['status' => 0]);
                try{
                    // mail
                    $subject = '(TESTING) You borrowed a book from CA GAC-7 library';

                    $body = str_replace(array('{name}', '{caname}', '{ctitle}', '{cpublic}',
'{crdate}', '{cetdate}'),array($sname, $aname, $title, $publication, date("d-M-Y",
strtotime($request_date)), date("d-M-Y", strtotime($return_date)),
),file_get_contents(base_url('assets/Template/mail.phtml')));
                    // mail trigger (calling send mail function)
                   $this->mail->sendmail($res['email'],$sname,$subject,$body);
                } catch(Exception $e){
                }
                $session->setFlashdata('msg', 'Book Barrowed.');
            } else{
                $session->setFlashdata('msg', 'Book Barrowed Failed.');
            }
        } else{
            $session->setFlashdata('msg', 'Please verify if the book is borrowed or
unavailable in the library.');
        }
```

```
        } else {
            // Validation failed
            $errors = $this->validation->getErrors();
        }
    }
        echo view('Others/header');
        echo view('Admin/AdminBookBarrow');
        echo view('Others/fooder');
}
public function Return()
{
    $session = session();
    if($this->request->getMethod() === 'post'){
        $bcode = $this->request->getPost('bcode');
        $regno = $this->request->getPost('regno');
        $sname = $this->request->getPost('sname');
        $bno = $this->request->getPost('bno');
        $remark = $this->request->getPost('remark');
        // Set the validation rules
        $validationRules = [
            'bcode' => 'required',
            'regno' => 'required',
            'bno' => 'required',
        ];
        // Set custom error messages for each field
        $validationMessages = [
            'bcode' => [
```

```
                'required' => 'The Barcode No field is required.'
            ],
            'regno' => [
                'required' => 'The Register Number field is required.'
            ],
            'sname' => [
                'required' => 'The Std/Staff Name field is required.'
            ],
            'bno' => [
                'required' => 'The Book No field is required.'
            ]
        ];
        $validation = \Config\Services::validation();
        $validation->setRules($validationRules, $validationMessages);
        if($this->validation->withRequest($this->request)->run()){
            $book = $this->booksModel->getBookDetail($bcode);
            if ($this->barrowbooksModel->setBookreturn($book['bid'],['is_returned' =>
1,'returned_date' => date("Y-m-d h:i:s"),'remark' => $remark ])) {
                $this->booksModel->updateBook($bcode,['status' => 1]);
            } else{
            }
        } else {
            $errors = $this->validation->getErrors();
        }
    }
    echo view('Admin/AdminBookReturn');
  }
}
```