**DEVELOPMENT PROJECT**

**De Montfort University**

**Library Automation Desk Project Documentation**

**Name: Maruan Makky**

**P-Number: P2701018**

**Module Code: CTEC3451**

**Module Leader: HOSSEIN MALEKMOHAMADI**

**SUPERVISER: LIPIKA DEKA**

**Words: 8397**

**1. SYNOPSIS**

* 1. **Title of the Project :**

**“Library Automation System”**

* 1. **Objectives :**
* The web application is used for the automation of library system.
* This web application is capable of managing Book Issues, Returns, and Magazine Subscriptions. Generating various Reports for Record-Keeping according to end user requirements.
* Providing library membership facility to students and staffs .
* Easy circulation of Books to members.
* Ensure Data Security.
* User Friendly and Reliable.
  1. **Description of the Project :**
* This web application is for automation of library.
* Can provide membership to members.
* Can read and write information about any Member.
* Can update, create, and delete the record of Membership as per requirement.
* Magazine Subscription management facility.
* Keeps the track of issues and submission of Books.
* Powerful Search Facility for Book and Member Search based on various Search Criteria.
* Quick Report View like Book Issues/Returned etc.
* College Configuration Maintenance.
* Stock verification facility
* Members Fine Calculating facility.
  1. **Software Used :**

Front End : HTML, CSS, PHP

Back End : MySQL

* 1. **Present Working System :**
* Details of Books and Members were maintained manually.
* Circulations of Books were also maintained manually.
  1. **Proposed Working System :**

The new System proposed with the following feature:

* Front end should be PHP and MySQL should be the back end.
* The website can be used only by the authorized person.
* The system should be flexible, i.e any changes should be easily updated in to the database without many changes to the coding system.

**1.7 Hardware Requirements:**

* Memory : 1GB of available RAM
* Hard Disk : 5GB of available disk space
* Monitor : Color monitor capable of 800 × 600

Resolution

* Key Board : Any Standard Keyboard
* Mouse : Any Standard Mouse
* Pen Drive : 4 GB
* Processor Type : Intel Pentium processor; 450 MHz or

Equivalent processor like AMD processor.

* CD-ROM drive : Any Standard CD-ROM drive.
* Operating System : Microsoft Windows 2000 OR Windows XP
  1. **Software Requirements :**
* Web Browser : Internet Explorer, Mozilla Firefox, Google chrome, etc
* Front End : PHP
* Back End : MySQL

# SOFTWARE REQUIREMENT SPECIFICATION

# Introduction

This document serves as the Software Requirements Specification (SRS) for the Library Management Software, specifically detailing the requirements for the "Library Automation Desk".

## Purpose

The purpose of this document is to presents a comprehensive description of the Library Automation Desk, a proposed enhancement to the existing library management system. It outlines the system's objectives, features, interfaces, operational constraints, and responses to external stimuli. Initially designed for stakeholders and developers, the document also aims to bridge the communication gap between students and librarians in the current offline system, where physical interactions predominate in book issuance and returns. Due to the significant challenges encountered with the offline system, an online application has been developed to streamline operations and improve accessibility for college students and staff. The primary objective of this project is to automate all the functional aspects of a college library, managing the catalog and maintaining records of all transactions concerning the available books.

The newly developed system incorporates advanced features that facilitate various stakeholders, including students, staff, and librarians, by managing book issues, returns, and magazine subscriptions efficiently. Additionally, it generates diverse reports for record-keeping tailored to the end users' requirements. The current web application also extends library membership facilities to students and staff, enhancing user engagement and operational efficiency.

## Intended Audience and Reading Suggestions

This document is intended for staff, librarians, and students and seeks to elucidate the fundamental concept underpinning the Library Automation Desk in a comprehensible manner. It facilitates an easy understanding for both developers and clients regarding the system's functionalities and framework. Additionally, this document outlines the objectives for future development, providing a roadmap for subsequent enhancements and iterations of the system.

## Project Scope and Product Features

The main scope of this project are:

* The system is convenient and flexible to be used. It saves their time, efforts, money and resources.
* User Friendliness is provided in the application with various controls provided by system Rich User Interface.
* The user information files can be stored in centralized database which can be maintained by the system.
* This web application is for automation of library. It also involves maintaining the database of entering new books and the record of books that have been retrieved or issued, with their respective dates.
* Can provide membership to members.
* Can read and write information about any Member.
* Can update, create, and delete the record of Membership as per requirement.
* Magazine Subscription management facility.
* Powerful Search Facility for Book and Member Search based on various Search Criteria.
* Quick Report View like Book Issues/Returned etc.
* College Configuration Maintenance.
* Book Stock verification facility
* Members Fine Calculating facility.

## Definitions, acronyms, abbreviations:

* SRS – Software requirements specifications.
* PHP - A server side scripting language.
* MySQL - The database that will be used for this project.
* http – Hyper text transfer protocol
* www – World wide web

## Overview:

The subsequent sections of this document are organized into three chapters, followed by appendices. Chapter Two offers a comprehensive overview of the system's functionality and its interactions with other systems. It further details the constraints of the system and assumptions made about the product. Chapter Three delineates the requirements specifications in meticulous detail and describes the various system interfaces. Various specification techniques have been employed to articulate the requirements with greater precision for diverse audiences.

# Overall Description:

The subsequent subsections offer a comprehensive overview of the Software Requirements Specification (SRS) for the product "Library Automation Desk." Initially, the document delineates the overall product concept. Following this, the external interface requirements are outlined, followed by a concise description of the product's components and features. The final section details the non-functional requirements of the product, ensuring a thorough understanding of its specifications and expectations.

## Product Perspective:

Given the growing popularity and accessibility of the internet, it is imperative for the educational sector to leverage this expansive resource. The administrative user interface is designed to handle consistent information integral to organizational activities, requiring robust authentication mechanisms for data collection. This interface facilitates various transactional processes such as data insertion, deletion, and updating, and provides capabilities for executive data searches.

Meanwhile, the operational and generic user interface enables users to engage with the system effectively, allowing transactions involving existing data and required services. This interface also supports users in managing their personal information in a tailored manner, accommodating individual preferences and needs through its flexible design.

## Product Functions:

The product encompasses the following functionalities:

The library system delivers online, real-time information regarding book availability within the library, as well as user data. The primary aim of this project is to diminish the extent of manual labor required in library operations. The software is adept at managing book issuances and returns, calculating fines, and generating a variety of reports tailored to the specific needs of end users. In this system, the librarian serves as the administrator, responsible for overseeing member activities and book management.

Additionally, the status of book issues and returns by members is consistently maintained within the library's database.

## User Classes and Characteristics:

The system offers varied services tailored to the specific roles of its users, whether they are members or librarians. The librarian functions as the primary controller, possessing all the privileges associated with an administrator. Within this capacity, the librarian is authorized to issue books to members and has the capability to view the diverse array of books available in the library.

## Operating Environment:

These are the major components to run this software. The server needs following tools to run the website:

* Apache Server
* MySQL Server

## Design and Implementation Constraints

CO-1: The system’s design, code, and maintenance documentation shall conform to the Process Impact Intranet Development Standard, Version

CO-2: The system shall use the current corporate standard MySQL database engine.

CO-3: All HTML code shall conform to the HTML 4.0 standard.

CO-4: All scripts shall be written in PHP.

## User Documentation

UD-1: The system shall provide an online hierarchical and cross-linked help system in HTML that describes and illustrates all system functions.

UD-2: The online tutorial will be provided.

## Assumptions and Dependencies

* Internet connection required.
* 24X7 uptime server connection required.
* One assumption about the product is that it will always be used on mobile phones that have enough performance.
* The user should have basic computer knowledge. They should be trained to handle the features provided by the system.
* The browser version should be used which have html5 support

# Specific Requirements

This section contains all of the functional and quality requirements of the system. It gives a detailed description of the system and all its features.

## External Interface Requirements:

This section offers a comprehensive description of all inputs and outputs associated with the system. It also includes detailed information on the hardware, software, and communication interfaces utilized by the system. Additionally, this section presents basic prototypes of the user interface to provide a clearer understanding of how users will interact with the system.

### User interface:

The primary user classes anticipated to utilize this product include:

* **Administrator**: The administrator functions as a superuser and is considered the owner of the site. They possess comprehensive privileges and are responsible for tasks such as adding and managing staff or librarian details.
* **Librarian**: Librarians are tasked with adding and managing book stock details. They facilitate the issuance and return of books, roles that are typically handled by designated agents.
* **Staff**: Staff members include lecturers or employees of the college. They have the capability to request books online, enhancing accessibility and convenience.
* **Student**: Students can register online by providing their name, email ID, registration number, and password. College staff are responsible for verifying student accounts. Once verified, students are able to request or issue books.

### Hardware Interface:

* **Operating system:** Windows XP / Windows 7
* **Hard disk:** 40 GB hard disk
* **RAM :** 1 GB RAM
* **Processor:** Intel Pentium or above

### Software interface:

* XAMPP
* Apache server
* PHP 5.4
* MYSQL server 5.5
* IDE: Notepad++

### Communication interface:

The project will utilise the HTTP protocol for communication over the internet, and for intranet communication, it will employ the TCP/IP protocol suite.

## Functional Requirements

The project has following modules:

* **Student Account module:**

This module records student registration details, including name, roll number, password, contact number, email ID, and other pertinent information. For security purposes, student accounts are subject to manual verification by an administrator or staff member. Upon successful registration, students may log into the website using their credentials. If the entered credentials are valid, the system redirects the student to their account page, where they can view details of issued and returned books.

This system has following sub modules:

* Registration module :- This module stores student registration details by entering name, roll number, password, contact number, email id, etc.
* Login module :- . After the registration student can login to the website by entering login credentials.
* Student profile module :- if the student enters valid login credentials then the page redirects to the student account page.
* Change password module :- The module allows you to configure actions the user must perform before changing their password.
* **Books entry module:**

This feature enables librarians to add new books to the library's collection. Librarians are required to input details such as the book's category, title, front cover image, author, price, and the number of pages it contains. Additionally, librarians have the ability to enter and manage the inventory of copies for each individual book. Furthermore, the system is designed to carefully assign a unique code to each book, ensuring accurate tracking and cataloguing.

* This module has following sub modules:
* Add, Edit, Delete books
* Book stock entry
* **Book transaction module:** This module encompasses the issuance and return of books.
* **Issue book:** This function enables librarians to issue books based on requests. Should the requested books be out of stock, the system places the request on a waiting list.
* **Return book:** This function allows librarians to log details of returned books. Upon entry of return details, the system updates both the book stock information and the student's records.
* **Librarian Account module:** This encompasses all library staff members who are tasked with entering records into the system and monitoring various activities such as the issuance and return of books, as well as tracking instances of book unavailability.

This module has following sub modules:

* Librarian Login module :- ThisIncludes all the library staff who are required to enter the records.
  + Change password :- The module allows you to configure actions the user must perform before changing their password.
  + Librarian Profile module :- if the librarian enters valid login credentials then the page redirects to their account page.
* **Penalty module:** This function will maintain a record of penalty reports. If a student delays the return of a book, a daily penalty will be calculated. The user is required to pay this penalty to the librarian. Upon receipt of the penalty payment, the system will generate a receipt.
  + Penalty calculation :- If the student takes time to return book then the penalty will calculate for each and every day.
  + Penalty receipt :- . The user needs to pay penalty to the librarian.after making penalty payment the system will generate the receipt.
* **Report module:** As indicated by its name, this module is dedicated to generating a variety of useful reports.

The system generates following report:

* Book issue report :-

This functionality allows librarians to issue books based on requests. Should the books be out of stock, the system places these requests on a waiting list.

* Return book report:-

This functionality enables librarians to record details of book returns. Once the return is logged, the system updates both the book stock information and the associated student records.

* Penalty report :-This report records penalties incurred. If a student delays the return of a book, a penalty is accrued daily until the book is returned.
* Book stock report :-

Module which contain the complete details about stock.

* **Library configuration module:** In this module librarian can set and configure number of days to keep the book and fine for delay return.

This module has following sub modules:

* Days to keep the book:- The student can take four book and one for the reference purpose and other three books can issue for 15 days.
* Fine settings :- The fine will be charges if the students will not return book within 15 days. Student must return the reference book at the same day without any fail.

# Other Nonfunctional Requirements

## Performance Requirements

This section the performance requirements expected from this Project:

* The performance of the system should be fast and accurate.
* Library management system shall handle expected and non-expected errors in ways that prevent loss in information and long downtime period.

## Safety Requirements

No safety requirements have been identified.

## Security Requirements

* The database may get crashed at any certain time due to virus or operating system failure. Therefore this required to take the database backup so that the database is not lost. Proper UPS /Inverter facility should be there in case of power supply failure.

## Software Quality Attributes

* If the connection between the user and the system is broken prior to an order being either confirmed or canceled, the System shall enable the user to recover an incomplete order.
* **RELIABILITY**

It is tested for all the constraints at development stage

* **AVAILABILITY**

This system will only available till the system on which it is installed is running.

* **SECURITY**

This system is provided with authentication without which no user can pass. So only the legitimate users are allowed to use the application. If the legitimate users share the authentication information then the system is open to outsiders.

* **MAINTAINABILITY**

There will be no maintenance required for the website. The database is provided by the end-user and therefore is maintained by this user.

* **PORTABILITY**

The system works anywhere with the internet connection.

**SYSTEM DESIGN**

# Introduction :

* **System Analysis :**

The system analysis approach necessitates a thorough examination of all components of the system. Analysts are required to consider each element of the system, including their inputs, outputs, controls, feedback, and the environment in which the system operates, during the construction phase.

* **System Design :** The objective of the system design phase is to create a model or representation of the system that can be utilised to construct the actual system. The focus during this phase is on translating the system requirements into detailed design specifications.

# Applicable Documents :

The document used in system design is Software Requirement Specification Document.

# Functional Decomposition :

The project has following modules:

* **Student Account module:** This module captures student registration details such as name, roll number, password, contact number, and email ID. For security purposes, student accounts are subject to manual verification.
* **Books entry module:** This feature allows librarians to add new books to the library's collection. Librarians must enter details such as the book category, title, front page, author, price, and number of pages. Additionally, librarians manage the inventory of each book, ensuring each receives a unique book code..
* **Book transaction module:** This module contains issues book,return book.
* **Librarian Account module:** This module is essential for all library staff, enabling them to enter records into the system and monitor activities such as book issues, returns, and tracking the non-availability of books.
* **Penalty module:** This module records penalties. If a student delays returning a book, a daily penalty is calculated. The student must pay this penalty to the librarian, after which the system generates a receipt.
* **Report module:** As the name implies, this module is dedicated to generating various useful reports.
* **Library configuration module:** In this module, librarians can configure the number of days a book can be kept and the fines for delayed returns

# Program Description:

* 1. **Context Flow Diagram**

A context flow diagram represents a top-level data flow diagram. It comprises a single process node that encapsulates the function of the entire system in relation to external entities. In this diagram, the entire system is considered as a single process, and all its inputs, outputs, sinks, and sources are identified and depicted.

**Context Flow Diagram (Level 0)**

****

* 1. **Data Flow Diagram** :

A Data Flow Diagram (DFD) is a graphical representation that illustrates the flow of data within an information system, aiding in the visualization of data processing. It is standard practice for designers to create a context-level DFD that depicts the interactions between the system and external entities. This initial DFD provides an overview and is subsequently expanded to detail the system being modeled. A DFD outlines the movement of data through various transformations or processes within the system, offering a functional view that transforms inputs into desired outputs. These diagrams are particularly useful during problem analysis, as they help conceptualize the system’s operational flow.

Moreover, Data Flow Diagrams are valuable tools for communicating to end-users how their input impacts the system's structure—from order processing and dispatch to restocking. They reveal how systems are developed and provide insights into the data management practices, highlighting how data is registered, stored, and maintained by relevant authorities.

**Notations in the DFD**

|  |  |
| --- | --- |
| **Symbol** | **Description** |
|  | The circle or bubble represents a process. A process is named and each process is represented by a named circle. |
|  | The source or sink is represented as a rectangular box. The source or sink is the net originator or the consumer of the data that flows in the system. |
|  | The arrow represents the flow of data through the system. The labeled arrows enter or leave the bubbles. |
|  | The database is represented with the open box symbol. |

**Top Level DFD :**

****

**DFD Level 2.1:**

****

**Description of component:**

* **Input** :
* Branchdetails
* Librarian details
* Login details
* **Process Definition:**
  + Adding branch details
  + Adding librarian details
  + Login process
* **Output definition:**
* Branched,branchname
* Loginid,password

**DFD Level 2.2:**

****

**Description of component:**

* **Input** :
* bookcategory details
* book details
* book stock detail
* **Process Definition:**
  + Adding bookcategory
  + Adding bookdetails
  + Adding bookstoc
* **Output definition:**
* bookcategoryid,bookcategory
* bookname ,bookdescription,bookimg,bookimg
* bookid,bookname
* bookid,qty,cost

**DFD Level 2.3**

****

**Description of component:**

* **Input** :
* Course details
* Registration details
* Login credentials
* Student detail
* **Process Definition:**
  + Adding course
  + Registration process
  + Validate registration detail
  + Login process
  + Update profile
* **Output definition:**
* Coursed,course
* Rollno,password
* Emailed,contactno, studentimg

**DFD Level 2.4:**

****

**Description of component:**

* **Input** :
* Selectedbook
* bookdetails
* **Process Definition:**
  + Searchbook
  + Adding to bag
  + Issue book
* **Output definition:**
* studentid,bookid,bookingdate
* transtype,borrowdate,returndate
* Bookissue report

**DFD Level 2.5:**

****

**Description of component:**

* **Input** :
* Finesettings
* Bookreturn details
* bookreturn details
* penalty details
* **Process Definition:**
  + Finesettings process
  + Returning book
  + Calculate penalty or fine
* **Output definition:**
* daytokeep,penaltycost,no.books
* transactionid,bookid,cost
* bookid,returndate,status

**DATA BASE DESIGN**

## Introduction

* **Database :** A database is a structured collection of related data, varying in size and complexity. Utilizing the concept of a database facilitates efficient storage and retrieval of data. The primary function of a database is to furnish information tailored to the specific requirements of the system utilizing it.
* **Database Design :** Database design precedes the construction of a database and aims to fulfill the needs of end-users within the designated information system. It entails defining the requisite data and data structures comprising the database. The physical implementation of the database is achieved using MySQL.

The database Library for is organized into 10 tables:

## book

## book category

## book\_stock

## branch

## course

## finesettings

## librarian

## penalty

## student

## transaction

Each entity can be described as follows along with its attributes:

## Table name: book:

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Type** | **Index** | **Description** |
| ***Bookid*** | int(10) | Primary key | **BookID** |
| bookcategoryid | int(10) | Not null | bookcategoryid |
| bookname | Varchar(50) | Not null | bookname |
| bookdescription | Text | Null | bookdescription |
| bookimg | Varchar(100) | Null | bookimg |
| bookcost | Float(10,2) | Not null | bookcost |
| Author | Varchar(25) | Null | author |
| Status | Varchar(10) | Not null | bookstatus |

## Table name: bookcategory

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Type** | **Index** | **Description** |
| ***bookcategoryid*** | int(10) | Primary key | **Bookcategory ID** |
| bookcategory | varchar(50) | Not null | bookcategory |
| bookdescription | text | Not null | bookdescription |
| Status | varchar(10) | Not null | Bookcategory status |

## Table name:book\_stock

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Type** | **Index** | **Description** |
| ***Book\_stock\_id*** | int(10) | Primary key | **Book-stock-ID** |
| Bookid | int(10) | Not null | bookid |
| branchid | int(10) | Not null | branchid |
| purchasedate | date | Not null | purchasedate |
| Qty | int(10) | Not null | qty |
| Cost | Float(10,2) | Not null | cost |
| Status | varchar(10) | Not null | Book-stock status |

## Table name:branch

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Type** | **Index** | **Description** |
| ***branchid*** | int(10) | Primary key | **Branch ID** |
| branchname | varchar(50) | Not null | Branch name |
| branchdescription | text | Not null | Branch description |
| Status | Varchar(10) | Not null | Branch status |

## Table name:course

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Type** | **Index** | **Description** |
| ***courseid*** | int(10) | Primary key | **CourseID** |
| branchid | int(10) | Not null | branchid |
| Course | varchar(50) | Not null | course |
| coursenote | Text | Not null | coursenote |
| Status | Varchar(10) | Not null | Course status |

## Table name:finesetting

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Type** | **Index** | **Description** |
| ***daytokeep*** | int(10) | Not null | **Daytokeep** |
| penaltycost | float (10,2) | Not null | penaltycost |
| nobooks | int(10) | Not null | nobooks |

## Table name:librarian

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Type** | **Index** | **Description** |
| ***Librarian\_id*** | int(10) | Primary key | **Librarian\_ID** |
| Name | varchar(25) | Not null | name |
| Type | varchar(10) | Not null | type |
| Loginid | varchar(25) | Not null | loginid |
| password | varchar(100) | Not null | password |
| Status | varchar(10) | Not null | Librarian status |

## Table name: penalty

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Type** | **Index** | **Description** |
| ***penaltyid*** | int(10) | Primary key | **PenaltyID** |
| Penalty\_type | varchar(25) | Not null | Penalty\_type |
| transactionid | int(10) | Not null | transactionid |
| Bookid | int(10) | Not null | bookid |
| Cost | Float(10,2) | Not null | cost |
| Penaltydate | Date | Not null | Penaltydate |
| Status | Varchar(10) | Not null | Penalty status |

## Table name:student

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Type** | **Index** | **Description** |
| ***studentid*** | int(10) | Primary key | **StudentID** |
| courseid | int(10) | Not null | courseid |
| studentname | varchar(25) | Not null | studentname |
| studentimg | varchar(100) | Not null | studentimg |
| Rollno | varchar(25) | Not null | Rollno |
| password | varchar(100) | Not null | password |
| EmailId | varchar(100) | Not null | emailid |
| contactno | varchar(15) | Not null | contactno |
| Status | varchar(15) | Not null | Student status |

## Table name: transaction

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Type** | **Index** | **Description** |
| ***transactionid*** | int(10) | Primary key | **TransactionID** |
| studentid | int(10) | Not null | studentid |
| Bookid | int(10) | Not null | bookid |
| transtype | Varchar(50) | Not null | transtype |
| bookingdate | datetime | Not null | bookingdate |
| borrowdate | datetime | Not null | borrowdate |
| returndate | datetime | Not null | returndate |
| Status | Varchar(10) | Not null | Transaction status |

## Entity-Relationship Diagram:

An Entity-Relationship (ER) diagram is a specialized graphic representation that elucidates the relationships between entities within a database. Typically, ER diagrams employ symbols to denote three distinct types of information. Entities are commonly depicted using boxes, relationships using diamonds, and attributes using ovals.

The Symbols are shown in below table:

|  |  |  |
| --- | --- | --- |
| **Name** | **Notation** | **Description** |
| Entity |  | Entity is represented by a box within the ERD. Entities are abstract concepts, each representing one or more instances of the concept in question. An entity might be considered a container that holds all of the instances of a particular thing in a system. Entities are equivalent to database tables in a relational database, with each row of the table representing an instance of that entity. |
| Relationship |  | Relationships are represented by Diamonds. A relationship is a named collection or association between entities or used to relate to two or more entities with some common attributes or meaningful interaction between the objects. |
| Attributes |  | Attributes are represented by Oval. An attribute is a single data item related to a database object. The database schema associates one or more attributes with each database entity. |

**ER Diagram:**

****

**DETAIL DESIGN**

**1. Introduction:**

  The detailed design phase commences subsequent to the system phase, once the system has been thoroughly reviewed and certified. The primary objective of this phase is to formulate the internal logic for each of the modules identified during system design.

While system design primarily entails identifying modules, detailed design focuses on delineating the logic for these modules. In essence, system design addresses the question of which components are necessary, whereas detailed design concerns itself with how these components can be implemented within the software.

The software design process comprises two key events. Initially, the emphasis is on determining the requisite modules for the system, specifying these modules, and establishing their interconnections. This phase is referred to as system design or top-level design. Here, decisions regarding whether the module specifications can be fulfilled are made. Subsequently, at the detailed design or logic design level, an extension of the system design, the focus is on further elaborating the system design, which governs the primary structural characteristics of the system.

System design significantly influences the testability, modifiability, and efficiency of the system. Consequently, a substantial portion of the design efforts in software engineering is dedicated to creating an effective system design.

**2. Applicable documents:**

      The detailed design phase draws upon the system documents, with the primary reference being the system design. Additionally, it incorporates considerations from the data structure, making the second application document the database design.

**3. Structure of software package:**

        The software package consists of following functional components.

* Functional component 1: Registration.
* Functional component 2: Book issue.
* Functional component 3: Book return.
* Functional component 4: Search/view book details.
* Functional component 5: Update book details.
* Functional component 6: Payment (Fine).

**3.1 Structure charts:**

A structured chart is a graphic representation of the control logic of processing functions or module representations within a system. It is widely employed as a method for system design. In a structured chart, each program module is depicted by a rectangular box. Modules at the top level of the structured chart call the modules at lower levels. The connections between modules are illustrated by lines connecting the rectangular boxes. These connections denote the data flows between the called and calling modules.

**4. Module Decomposition:**

* **Student Account module:** This module serves to store student registration details, including name, roll number, password, contact number, and email address. For security purposes, student accounts undergo manual verification by an administrator or staff member. Upon successful registration, students can log in to the website using their credentials. Valid login credentials redirect the user to their student account page, where they can view details of issued and returned books.
* **Books entry module:** This feature enables librarians to add new books to the library's inventory. Librarians are required to input details such as the book's category, title, front page, author, price, and number of pages. Additionally, librarians can manage the quantity of each book in stock. The system ensures the allocation of unique book codes to individual books.
* **Book transaction module:** This module contains issues book,return book.
* **Librarian Account module:** This module encompasses all library staff members responsible for entering records into the system and monitoring activities such as book issuance, returns, and inventory management.
* **Penalty module:** This will store penalty report. If the student takes time to return book then the penalty will calculate for each and every day. The user needs to pay penalty to the librarian. After making penalty payment the system will generate receipt.
* **Report module:** As the name suggest, this module helps only for generating various useful reports.
* **Library configuration module:** In this module librarian can set and configure number of days to keep the book and fine for delay return.

# Modular Decomposition Of Components :

**5.1. Student Account module:**

**5.1.1. Identification Of Modules :**

The modules identified in this component are:

* Registration module
* Login module
* Student profile module
* Changed password module

**5.1.2. Structure chart for student module:**

****

**4.2. Books entry module:**

**4.2.1. Identification Of Modules:**

The modules identified in this component are:

* Add,Edit,Delete books
* Book stock entry

**4.2.2. Structure chart for Books entry module:**

****

**4.3. Book transaction module::**

**4.3.1. Identification Of Modules:**

The modules identified in this component are:

* + - Issue book
    - Return book

****

**4.4. Librarian Account module:**

**4.4.1 Identification Of Modules:**

The modules identified in this component are:

* Librarian login module
* Changed password
* Librarian profile module

**4.4.2 Structure chart for Llibrarian Account module:**

****

**4.5. Penalty module:**

**4.5.1. Identification Of Modules:**

The modules identified in this component are:

* Penalty calculation
* Penalty receipt

**4.5.2. Structure chart for Penalty:**



**4.6. Report module:**

**4.6.1.: Identification Of Modules**

The modules identified in this component are:

* Book issue report
* Return book report
* Penalty report
* Book stock report

**4.6.2. Structure chart for report:**



**4.7. Library configuration module:**

**4.7.1. Identification Of Modules:**

The modules identified in this component are:

* Days to keep the book
* Fine settings

**4.7.2. Structure chart for library configuration :**



**CODING**

# INTRODUCTION

* **PHP (PHP HYPERTEXT PREPROCESSOR)**

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

* PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
* PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
* It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
* PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
* PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
* PHP is forgiving: PHP language tries to be as forgiving as possible.
* PHP Syntax is C-Like.

**Common uses of PHP**

* PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
* PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
* You add, delete, modify elements within your database through PHP.

## MySQL

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons −

* MySQL is released under an open-source license. So you have nothing to pay to use it.
* MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
* MySQL uses a standard form of the well-known SQL data language.
* MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
* MySQL works very quickly and works well even with large data sets.
* MySQL is very friendly to PHP, the most appreciated language for web development.
* MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
* MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

**HTML**

HTML stands for **H**yper**t**ext **M**arkup **L**anguage, and it is the most widely used language to write Web Pages.

* **Hypertext** refers to the way in which Web pages (HTML documents) are linked together. Thus, the link available on a webpage is called Hypertext.
* As its name suggests, HTML is a **Markup Language** which means you use HTML to simply "mark-up" a text document with tags that tell a Web browser how to structure it to display.

Originally, HTML was developed with the intent of defining the structure of documents like headings, paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers.

Now, HTML is being widely used to format web pages with the help of different tags available in HTML language.

  HTML was created by Berners-Lee in late 1991 but "HTML 2.0" was the first standard HTML specification which was published in 1995. HTML 4.01 was a major version of HTML and it was published in late 1999. Though HTML 4.01 version is widely used but currently we are having HTML-5 version which is an extension to HTML 4.01, and this version was published in 2012.

## HTML Documents

All HTML documents must start with a document type declaration: <!DOCTYPE html>.

The HTML document itself begins with <html> and ends with </html>.

The visible part of the HTML document is between <body> and </body>.

**CSS (Cascading Style Sheets)**

**C**ascading **S**tyle **S**heets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs,variations in display for different devices and screen sizes as well as a variety of other effects.

CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

## Advantages of CSS

* **CSS saves time** − You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
* **Pages load faster** − If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.
* **Easy maintenance** − To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
* **Superior styles to HTML** − CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.

**JAVASCRIPT**

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

JavaScript was first known as **LiveScript,** but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. JavaScript made its first appearance in Netscape 2.0 in 1995 with the name **LiveScript**. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

The [ECMA-262 Specification](http://www.ecma-international.org/publications/index.html) defined a standard version of the core JavaScript language.

* JavaScript is a lightweight, interpreted programming language.
* Designed for creating network-centric applications.
* Complementary to and integrated with Java.
* Complementary to and integrated with HTML.
* Open and cross-platform

## Advantages of JavaScript

The merits of using JavaScript are −

* **Less server interaction** − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
* **Immediate feedback to the visitors** − They don't have to wait for a page reload to see if they have forgotten to enter something.
* **Increased interactivity** − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
* **Richer interfaces** − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

**6.2 Source Coding**

**CODING FOR DB CONNECTION:**

<?php

$con = mysqli\_connect("localhost","root","","library");

if(mysqli\_connect\_errno($con))

{

echo "Database not connected" .mysqli\_connect\_error();

}

?>

**CODING FOR LOGIN PAGE:**

<?php

include("header.php");

if(isset($\_SESSION["librarian\_id"]))

{

echo "<script>window.location='dashboard.php';</script>";

}

if(isset($\_SESSION["studentid"]))

{

echo "<script>window.location='studentpanel.php';</script>";

}

if(isset($\_POST[submit]))

{

if($\_POST[usertype] == "Student")

{

$sql="SELECT \* FROM student WHERE rollno='$\_POST[rollno]' AND password='$\_POST[password]' AND status='Active'";

$qsql = mysqli\_query($con,$sql);

if(mysqli\_num\_rows($qsql) == 1)

{

$rs = mysqli\_fetch\_array($qsql);

$\_SESSION["studentid"] = $rs[studentid];

echo "<script>window.location='studentpanel.php';</script>";

}

else

{

echo "<script>alert('You have entered invalid login credentials...');</script>";

echo "<script>window.location='login.php';</script>";

}

}

if($\_POST[usertype] == "Admin" || $\_POST[usertype] == "Librarian")

{

$sql="SELECT \* FROM librarian WHERE type='$\_POST[usertype]' AND loginid='$\_POST[rollno]' AND password='$\_POST[password]' AND status='Active'";

$qsql = mysqli\_query($con,$sql);

if(mysqli\_num\_rows($qsql) == 1)

{

$rs = mysqli\_fetch\_array($qsql);

$\_SESSION["librarian\_id"] = $rs[librarian\_id];

$\_SESSION["type"] = $rs[type];

echo "<script>window.location='dashboard.php';</script>";

}

else

{

echo "<script>alert('You have entered invalid login credentials...');</script>";

echo "<script>window.location='login.php';</script>";

}

}

}

?>

<!-- ABOUT -->

<section id="about">

<div class="container">

<div class="row">

<div class="col-md-offset-3 col-md-12 col-sm-12">

<div class="entry-form">

<form action="" method="post">

<h2>Login Panel</h2>

<input type="text" name="rollno" class="form-control" placeholder="LOGIN-ID OR Roll Number" required="">

<input type="password" name="password" class="form-control" placeholder="Your password" required="">

<select name="usertype" class="form-control">

<option value="">Select User type</option>

<option value="Student">Student</option>

<option value="Librarian">Librarian</option>

<option value="Admin">Admin</option>

</select>

<button class="submit-btn form-control" id="form-submit" name="submit" type="submit">Login</button>

</form>

</div>

</div>

</div>

</div>

</section>

<?php

include("footer.php");

?>

**CODING FOR REGISTER PAGE:**

<?php

include("header.php");

if(isset($\_POST[submit]))

{

$imgname= rand(). $\_FILES["studentimg"]["name"];

move\_uploaded\_file($\_FILES["studentimg"]["tmp\_name"],"imgstudent/".$imgname);

if(isset($\_GET[editid]))

{

//4. update statement starts here

$sql = "UPDATE student SET courseid ='$\_POST[courseid]',studentname='$\_POST[studentname]'";

if($\_FILES["studentimg"]["name"] != "")

{

$sql = $sql . ",studentimg='$imgname'";

}

$sql = $sql . " ,rollno='$\_POST[rollno]',password='$\_POST[password]',emailid='$\_POST[emailid]',contactno='$\_POST[contactno]',status='$\_POST[status]' WHERE studentid='$\_GET[editid]'";

$qsql = mysqli\_query($con,$sql);

echo mysqli\_error($con);

if(mysqli\_affected\_rows($con) ==1)

{

echo "<script>alert('student record Updated successfully..');</script>";

echo "<script>window.location='viewstudent.php';</script>";

}

//4. update statement ends here

}

else

{

$sql = "INSERT INTO student(courseid,studentname,studentimg,rollno,password,emailid,contactno,status) VALUES('$\_POST[courseid]','$\_POST[studentname]','$imgname','$\_POST[rollno]','$\_POST[password]','$\_POST[emailid]','$\_POST[contactno]','Active')";

$qsql = mysqli\_query($con,$sql);

echo mysqli\_error($con);

if(mysqli\_affected\_rows($con) ==1)

{

echo "<script>alert('Student Registration done successfully..');</script>";

echo "<script>window.location='login.php';</script>";

}

}

}

?>

<!-- ABOUT -->

<section id="about">

<div class="container">

<div class="row">

<div class="col-md-offset-3 col-md-12 col-sm-12">

<div class="entry-form-register">

<form role="form" action="" method="post" enctype="multipart/form-data" onsubmit="return validateform()" name="frmregister">

<h2>Registration Panel</h2>

<select name="courseid" class="form-control">

<option value="">Select Course</option>

<?php

$sqlcourse ="SELECT \* FROM course WHERE status='Active'";

$qsqlcourse = mysqli\_query($con,$sqlcourse);

while($rscourse = mysqli\_fetch\_array($qsqlcourse))

{

if($rscourse[courseid] == $rsedit[courseid])

{

echo "<option value='$rscourse[courseid]' selected>$rscourse[course]</option>";

}

else

{

echo "<option value='$rscourse[courseid]'>$rscourse[course]</option>";

}

}

?>

</select>

<input type="text" name="studentname" class="form-control" placeholder="Name">

<input type="file" name="studentimg" class="form-control" placeholder="Profile Image">

<input type="text" name="rollno" class="form-control" placeholder="Roll number">

<input type="password" name="password" class="form-control" placeholder="Enter Password">

<input type="password" name="conformpassword" class="form-control" placeholder="Confirm password">

<input type="email" name="emailid" class="form-control" placeholder="Email ID">

<input type="text" name="contactno" class="form-control" placeholder="Contact No">

<button class="submit-btn form-control" name="submit" id="form-submit" type="submit">Register</button>

</form>

</div>

</div>

</div>

</div>

</section>

<?php

include("footer.php");

?>

**CODING FOR LOGOUT PAGE:**

<?php

session\_start();

session\_destroy();

header ("Expires: Mon, 26 Jul 1997 05:00:00 GMT"); // Date in the past

header ("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");

header ("Cache-Control: no-cache, must-revalidate"); // HTTP/1.1

header ("Pragma: no-cache");

echo "<script>window.location='dashboard.php';</script>";

?>

**SYSTEM TESTING**

**7.1 Introduction**

Testing is a fundamental process aimed at identifying errors within software. It plays a crucial role in quality assurance and ensures the reliability of software systems. The outcomes of testing also serve as valuable insights during the maintenance phase.

**7.1.1 Psychology of Testing**

The primary goal of testing is often to demonstrate that a program operates without errors. However, the essence of testing lies in detecting any potential errors present within the program. Therefore, testing should not commence with the assumption that the program functions flawlessly. Testing involves the execution of a program with the specific objective of identifying errors.

**7.2 Testing Objectives**

The primary objective of testing is to uncover errors systematically and efficiently within a software system. A successful test is one that reveals previously unidentified errors. A well-designed test case is characterized by its ability to detect errors if they exist. It is essential to acknowledge that tests may not always detect all potential errors. Furthermore, successful testing ensures that the software adheres to quality and reliability standards.

**7.3 System Testing**

Software testing is a critical component of software quality assurance and serves as the ultimate validation of specifications, design, and coding. System testing involves subjecting the software system to various test scenarios using diverse test data. The preparation of comprehensive test data is pivotal in the system testing process. Following test data preparation, the system undergoes rigorous testing to identify and rectify any errors discovered. These errors and their corresponding corrections are meticulously documented for future reference. Consequently, a series of testing procedures are executed on the system before it is deemed ready for implementation.

**7.4 Levels of Testing**

1. Unit testing
2. Integrated testing
3. Validation testing
4. Output testing
5. User Acceptance testing

* **Unit Testing:**

Unit testing concentrates verification efforts on the smallest unit of software, i.e., the module. It utilises the detailed design and the process specifications to test and uncover errors within the module's boundaries. Each module must successfully pass unit testing before integration testing commences. In this project, each service is considered a module and has been tested with various sets of inputs during and after development to ensure error-free operation. Inputs are validated when received from the user. This method facilitates the early detection of errors in the system's individual units.

* **Integrated Testing:**

Following unit testing, integration testing is conducted to ensure that modules integrate smoothly. The primary focus is on testing the interfaces between modules, effectively assessing the system's design through module interactions. Throughout this project, the integration of all modules was scrutinized by testing various input combinations to evaluate the impact on the functionality of the services.

* **Validation Testing:**

After integration testing, when the software has been fully assembled and interfacing errors corrected, validation testing begins. This phase tests the system in conditions that can reasonably be expected by the customer, ensuring it meets the system requirement specifications.

* **Output Testing:**

Once validation testing is complete, the next step is output testing, which is critical as no system is useful if it does not produce the correct output in the desired format. This involves generating and testing the output/report, considering both screen displays and printed formats to ensure accuracy and appropriateness of the output.

* **User Acceptance Testing:**

Acceptance testing uses realistic client data to demonstrate that the software functions satisfactorily. This phase focuses on the external behaviour of the system, rather than the internal logic of the program. Test cases are selected to exercise the maximum attributes of an equivalence class simultaneously, ensuring comprehensive coverage and validation of the system’s functionality from the user’s perspective.

**7.5 Test Cases:**

**Test Unit: Admin Component**

* **Adding Login page**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Serial No.** | **Condition**  **To be Tested** | **Test Data** | **Expected Output** | **Remarks** |
| 1. | If we are not enter the valid roll no or LOGIN ID. | LOGIN ID or Roll no | Please fill out this field. | SUCCESSFUL |
| 2. | If we only put the valid roll no or LOGIN ID and after pressing the login button. | Your password | Please fill out this field. | SUCCESSFUL |
| 3. | If we are putting the correct user name and wrong password and after selecting the valid login credential. | Select the user type | Kindly select the status | SUCCESSFUL |

* + **Adding Registration form:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Serial No.** | **Condition**  **To be Tested** | **Test Data** | **Expected Output** | **Remarks** |
| 1 | If we are not selected any field. | Select Course | Kindly select the fields. | SUCCESSFUL |
| 2 | If we are not entered the name. | Name | Please enter the name. | SUCCESSFUL |
| 3 | If we are not entered the valid roll no. | Roll no | Please enter the valid roll no. | SUCCESSFUL |
| 4 | If we are not entered the valid password. | Enter Password | Please enter the valid password. | SUCCESSFUL |
| 5 | If the password is not same as previous password. | Confirm password | Please enter the valid password once again. | SUCCESSFUL |
| 6 | If we are not entered the valid emailid. | Email id | Kindly insert the emailid. | SUCCESSFUL |
| 7 | If we are not suppose to enter contact no. | Contact no. | Please enter the contact no. | SUCCESSFUL |
| 8 | If we are not suppose to add any image files on the choose File. | Choose file | Kindly upload Image. | SUCCESSFUL |

* + **Adding admin/branch**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Serial No.** | **Condition**  **To be Tested** | **Test Data** | **Expected Output** | **Remarks** |
| 1. | If we are not enter the branch name. | Branch name | Please fill out this field. | SUCCESSFUL |
| 2. | If the branch description not to entered. | Branch description | Please fill out this field. | SUCCESSFUL |
| 3. | If we not selected any status | Branch status | Kindly select the status | SUCCESSFUL |

* **Admin/book:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Serial No.** | **Condition**  **To be Tested** | **Test Data** | **Expected Output** | **Remarks** |
| 1. | If you are not selected category. | Book category | Please fill out this field. | SUCCESSFUL |
| 2. | If we are not enter the book name. | Book name | Please enter book name. | SUCCESSFUL |
| 3 | If we are not enter the book cost fields. | Book cost | Please enter book cost. | SUCCESSFUL |
| 4 | If we are not enter book author fields. | Book author | Please enter book author. | SUCCESSFUL |
| 5 | If we not selected any status. | Book status | Kindly select status. | SUCCESSFUL |
| 6 | If we are not suppose to add any image files on the choose File. | Book Image | Kindly upload Image. | SUCCESSFUL |
| 7 | If we are not enter book description. | Book description | Book description should not be empty. | SUCCESSFUL |

* **Admin/Book category:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Serial No.** | **Condition**  **To be Tested** | **Test Data** | **Expected Output** | **Remarks** |
| 1. | If we are not enter the valid book category. | Book category | Please fill out this fields. | SUCCESSFUL |
| 2. | If we are not suppose the book description. | Book description | Book description should not be empty.. | SUCCESSFUL |
| 3. | If we are not selected status. | Book status | Kindly select status | SUCCESSFUL |

* **Admin/Course:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Serial No.** | **Condition**  **To be Tested** | **Test Data** | **Expected Output** | **Remarks** |
| 1. | If you are not select the branch | Select branch. | Please enter the valid fields. | SUCCESSFUL |
| 2. | If the course name not entered. | Course | Please enter the course. | SUCCESSFUL |
| 3. | If the course note did not entered. | Course note | Please You should fill out course note. | SUCCESSFUL |
| 4. | If we are not selected status. | Course status | Kindly select the status. | SUCCESSFUL |

* **Admin/Librarian:**

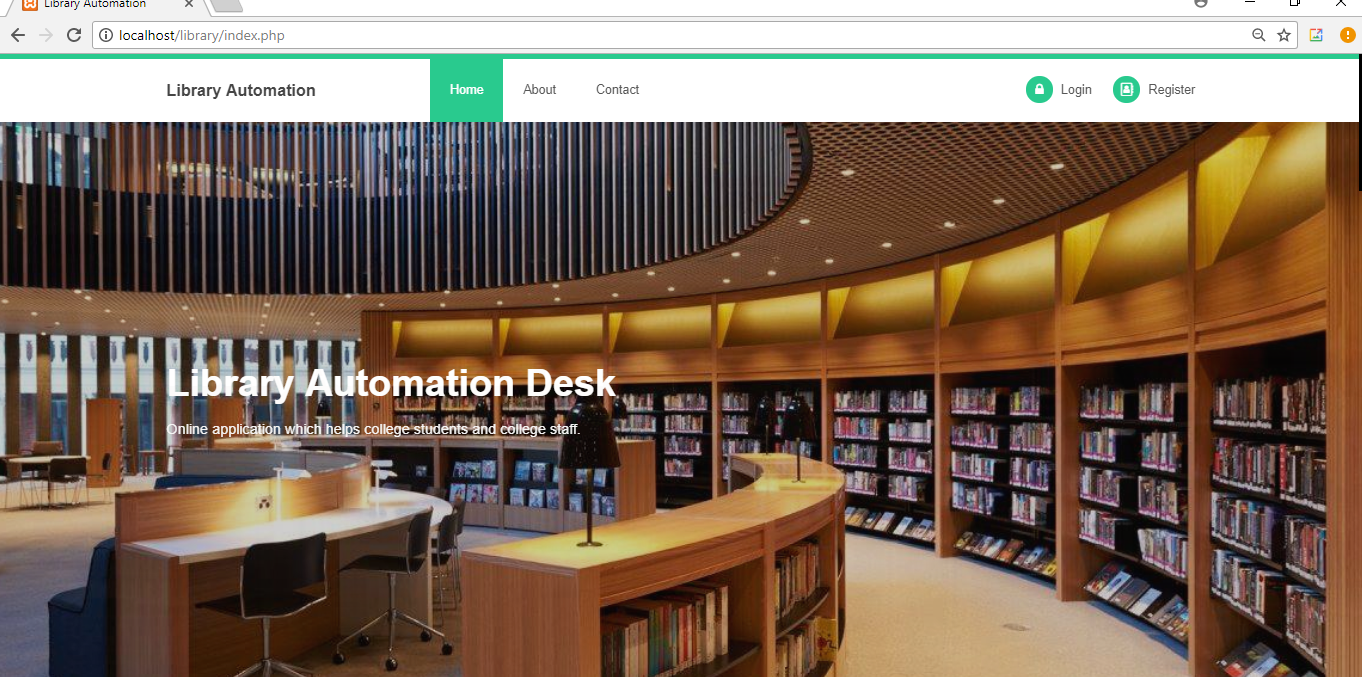
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Serial No.** | **Condition**  **To be Tested** | **Test Data** | **Expected Output** | **Remarks** |
| 1 | If we are not entered the name. | Name | Please enter the name. | SUCCESSFUL |
| 2 | If we are not selected the type. | Type | Please select the type. | SUCCESSFUL |
| 3 | If we are not entered the valid login id | Login id | Enter the valid Login id. | SUCCESSFUL |
| 4 | If we are not entered the password field. | Password | Enter the valid password. | SUCCESSFUL |
| 5 | If we are not entered the same password. | Confirm password | Enter the valid password once again. | SUCCESSFUL |
| 6 | If we are not selected any status. | Status | Kindly select the status | SUCCESSFUL |

* **Admin/Student:**

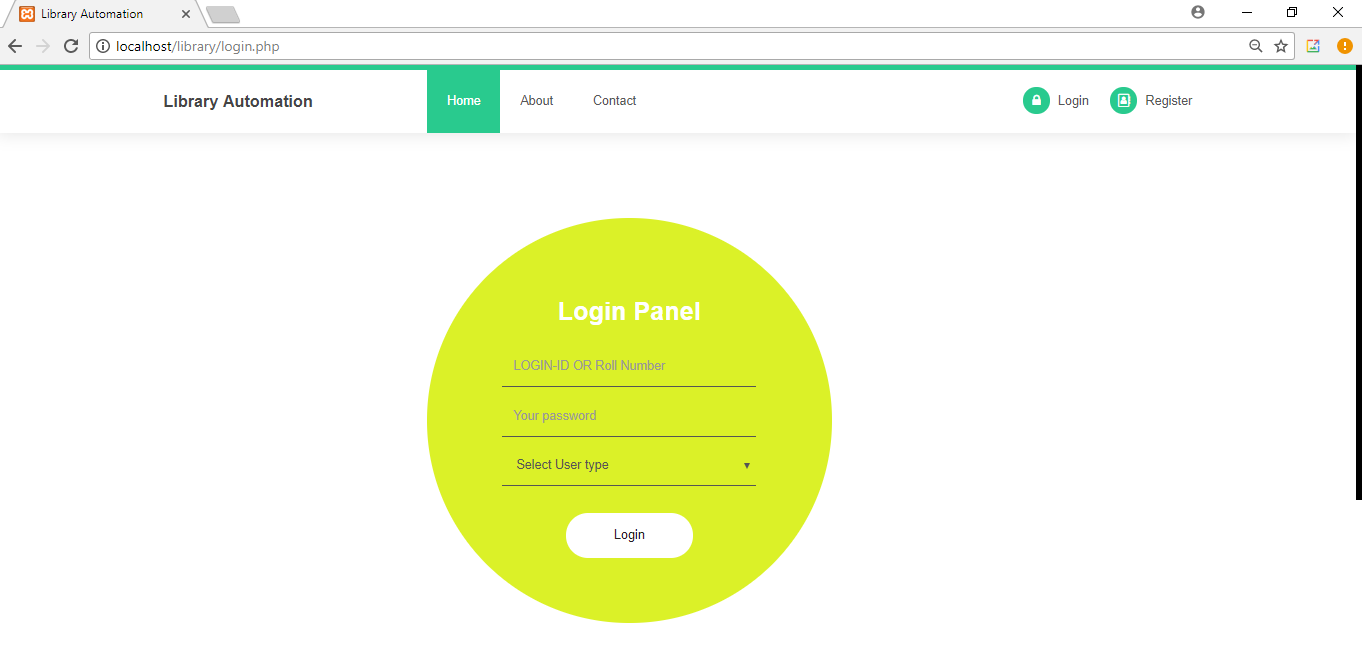
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Serial No.** | **Condition**  **To be Tested** | **Test Data** | **Expected Output** | **Remarks** |
| 1 | If we are not selected any course. | Course | Please fill out the fields. | SUCCESSFUL |
| 2 | If we are not entered the student name. | Student name | Please enter the student name. | SUCCESSFUL |
| 3 | If we are not entered the valid roll no. | Roll no | Please enter the valid roll no. | SUCCESSFUL |
| 4 | If we are not entered the valid password. | Password | Please enter the valid password. | SUCCESSFUL |
| 5 | If the password is not same as previous password. | Confirm password | Please enter the valid password once again. | SUCCESSFUL |
| 6 | If we are not entered the valid emailid. | Email id | Kindly insert the emailid. | SUCCESSFUL |
| 7 | If we are not suppose to enter contact no. | Contact no. | Please enter the contact no. | SUCCESSFUL |
| 8 | If we are not selected status | Status | kindly select status. | SUCCESSFUL |
| 9 | If we are not suppose to choose the image file. | Student image | Kindly upload the image. | SUCCESSFUL |

**SCREEN SHOTS**

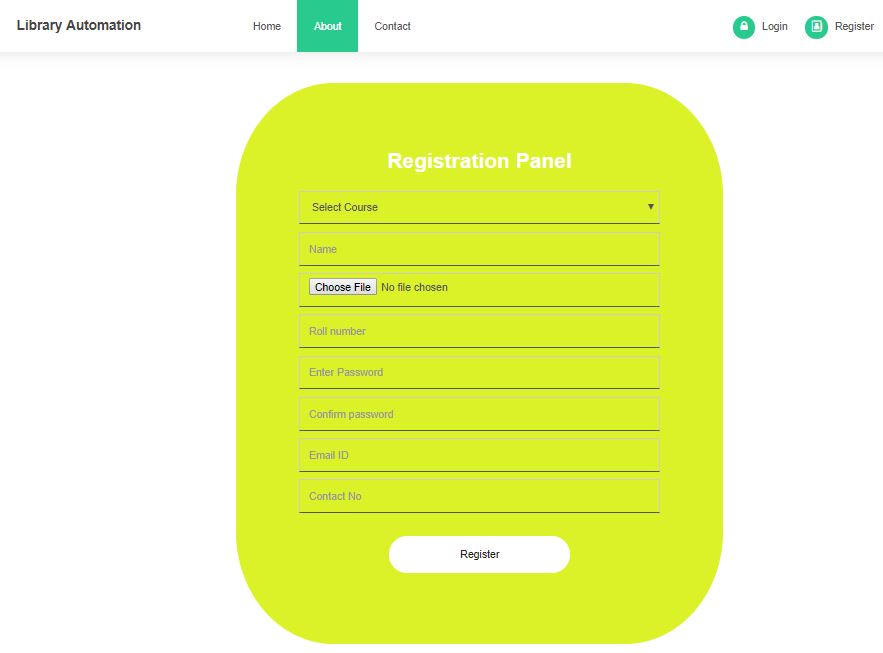
**Main page:**

****

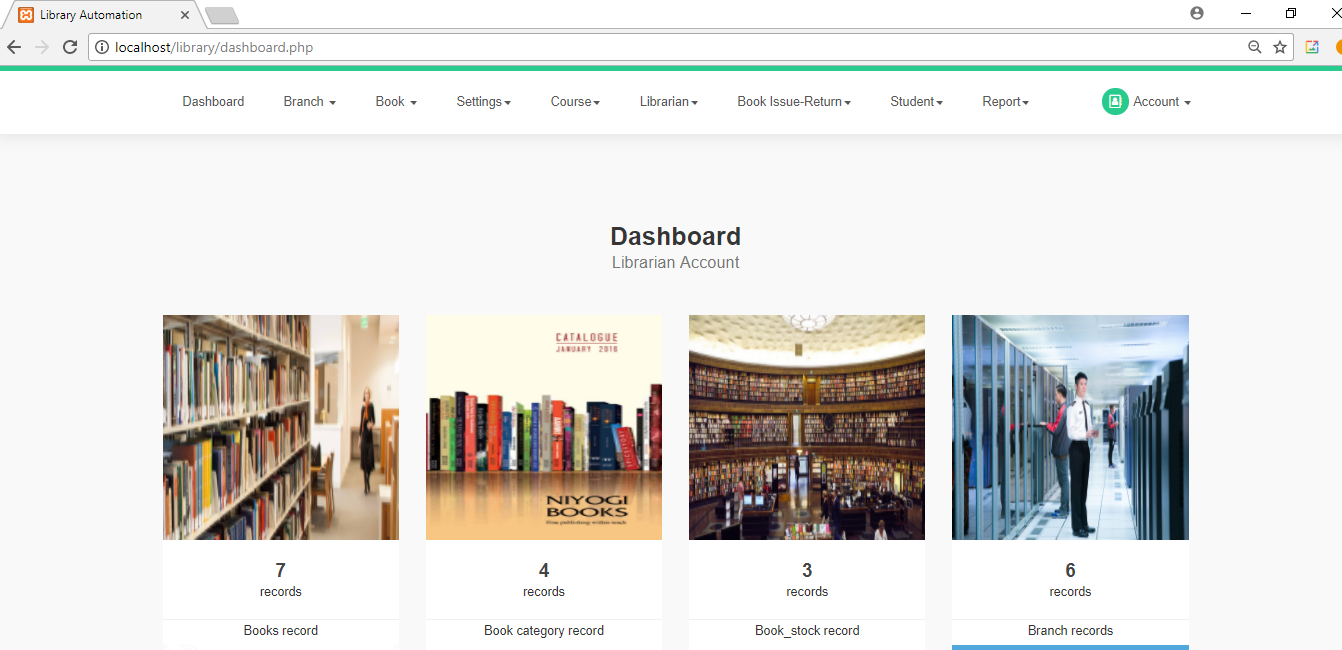
**Loginpage:**



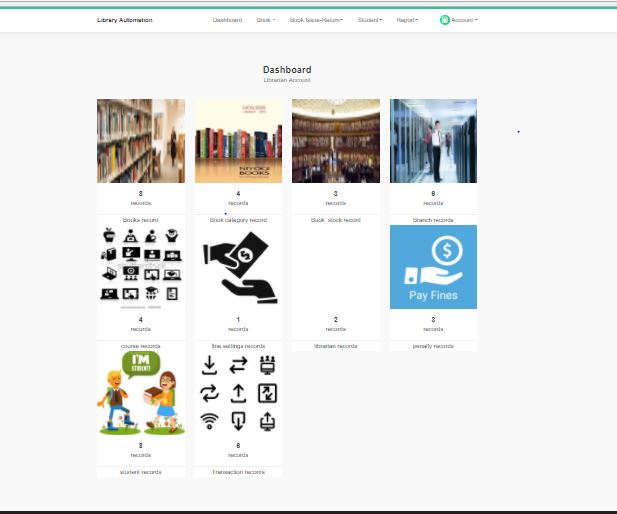
**Registration page:**

****

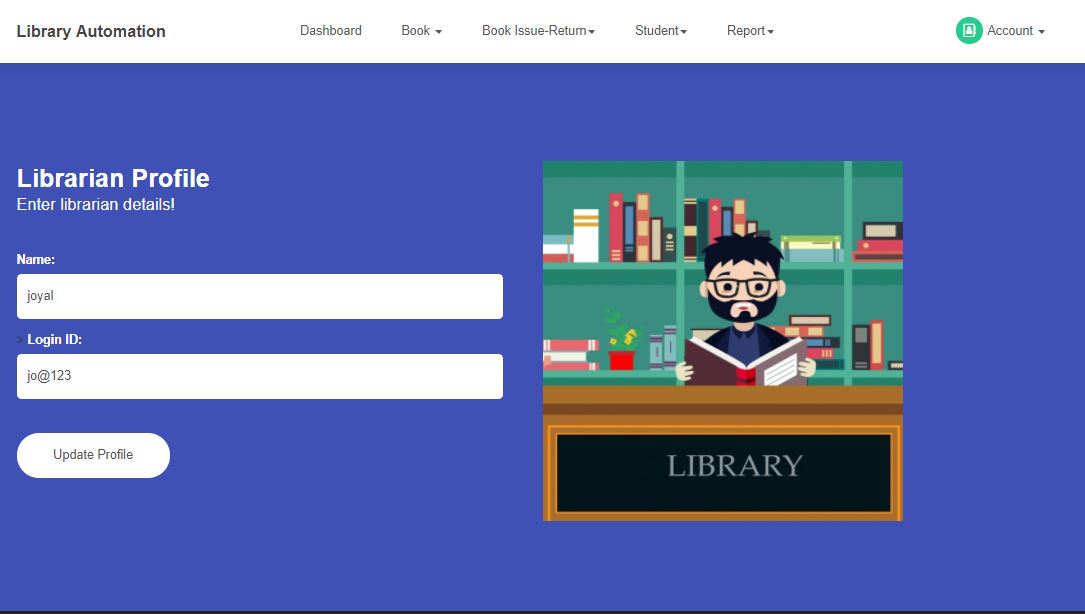
**Adminpage:**

****

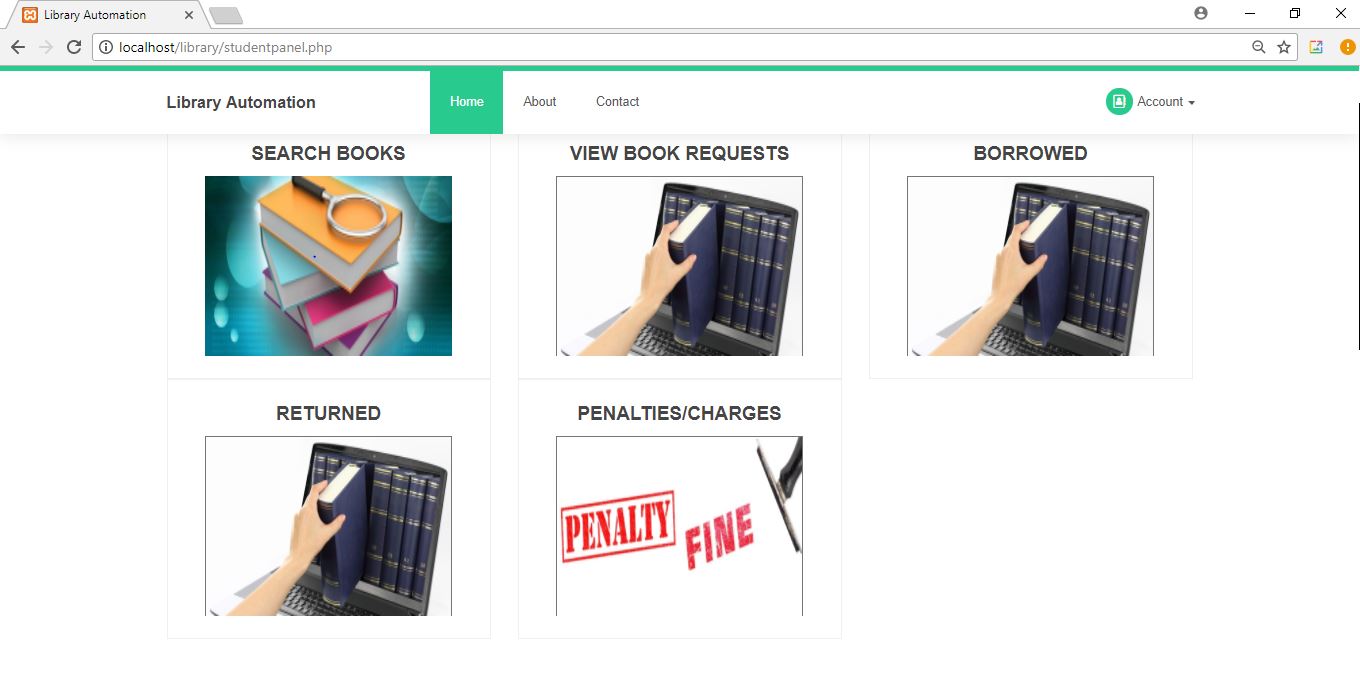
**Librarian page:**

****

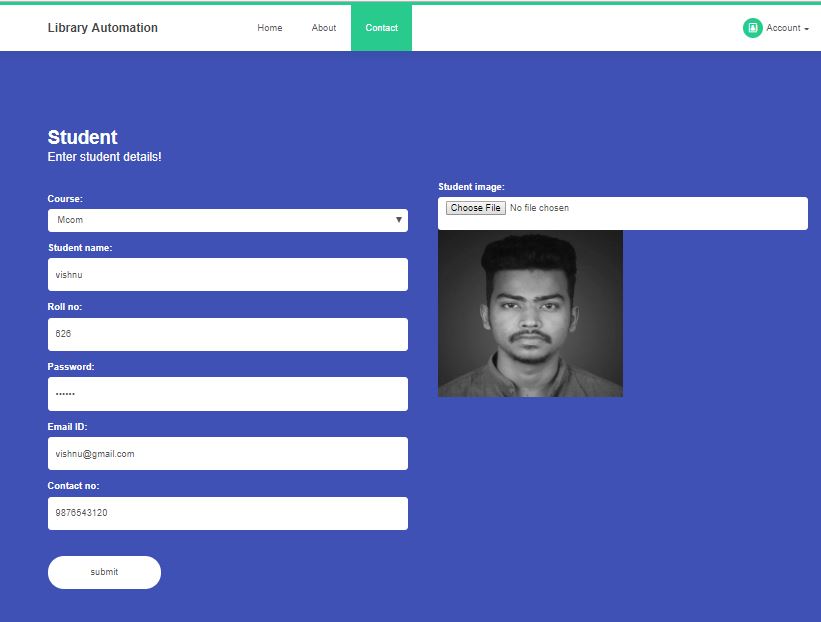
**librarianProfile:**

****

**Studentpage:**

****

**Studentprofilepage:**

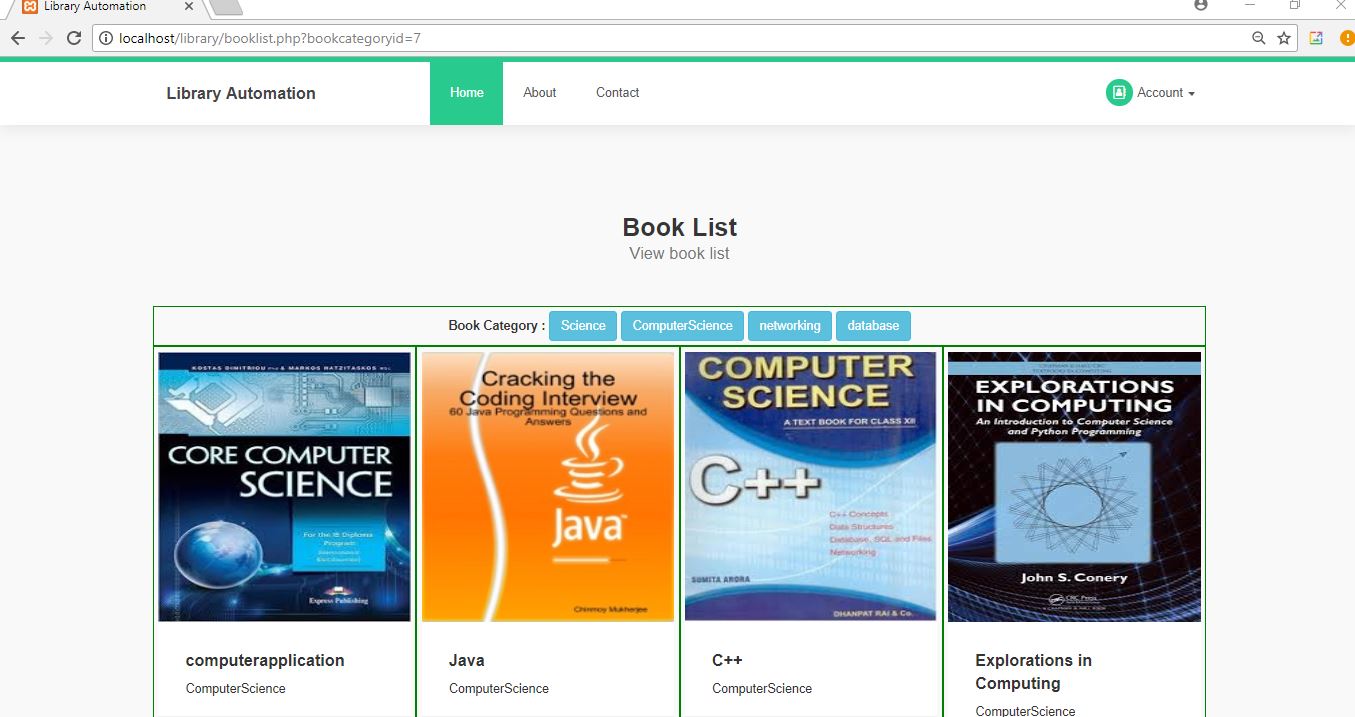
****

**Viewbook page:**

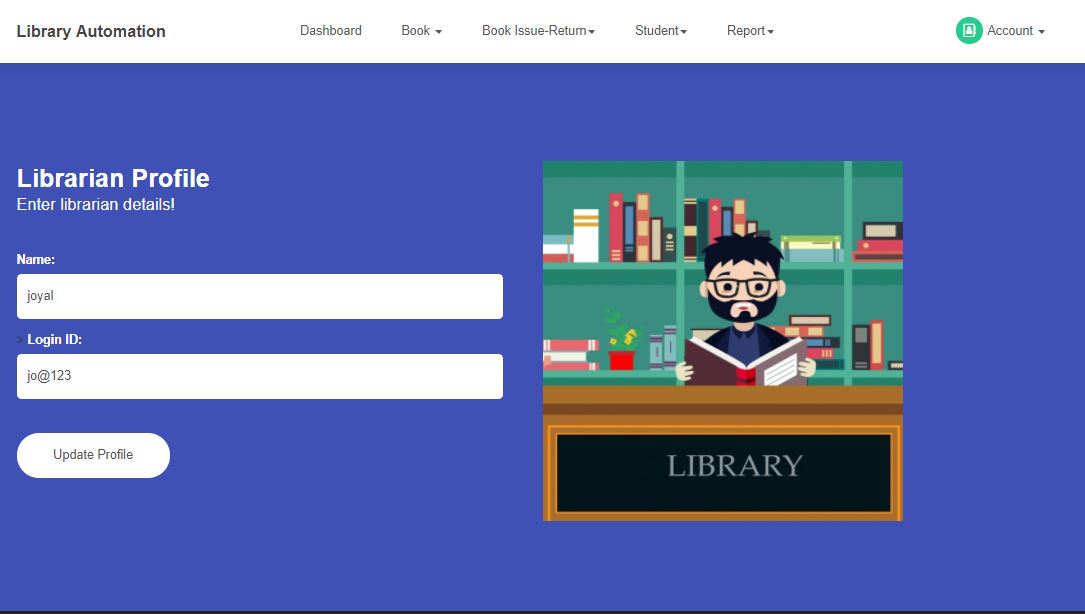
**A screenshot of a computer

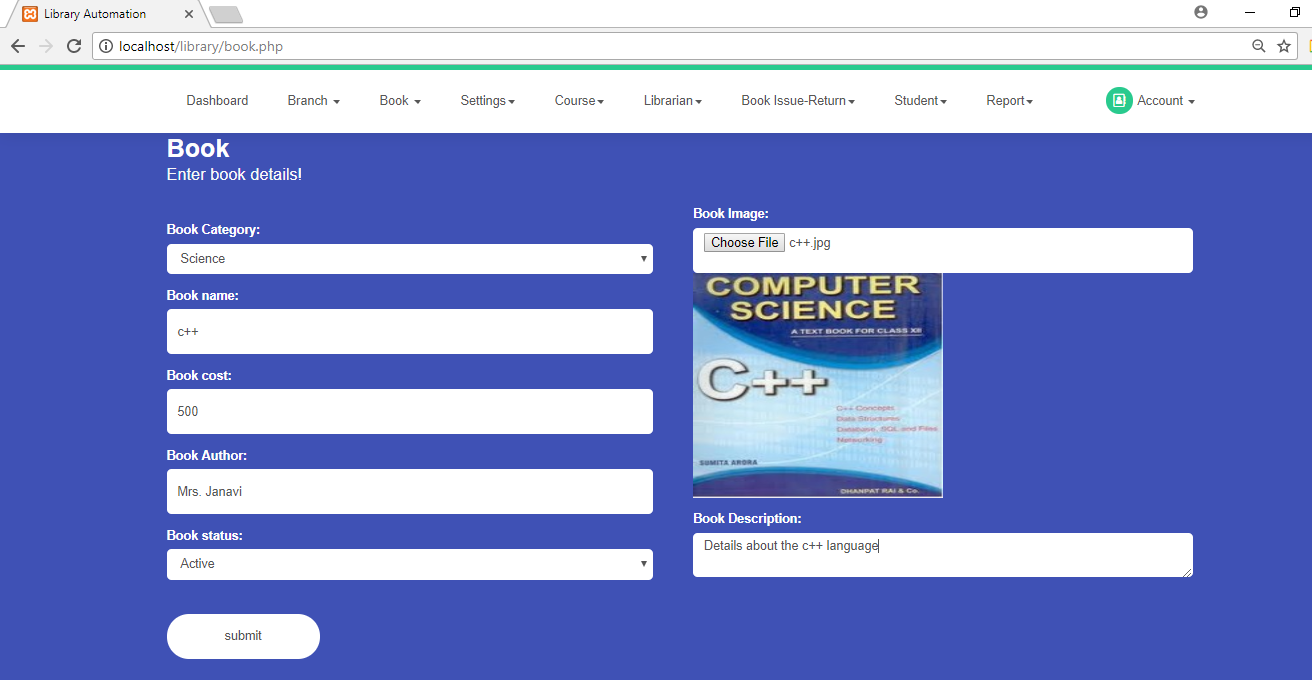
Description automatically generated**

**Booklist:**

****

**Librarianprofile:**

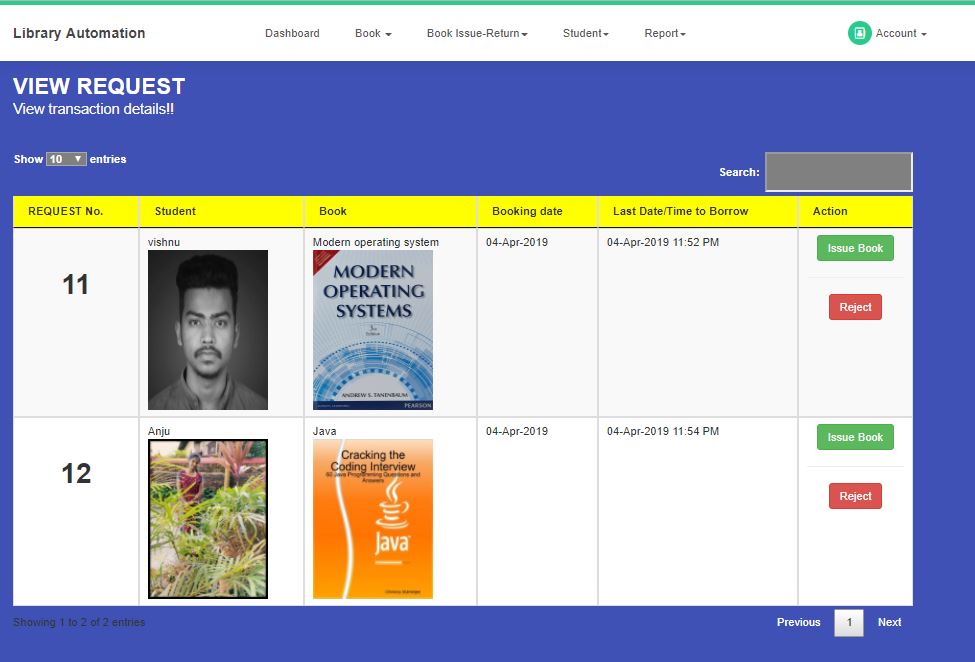
**Addbook:**

****

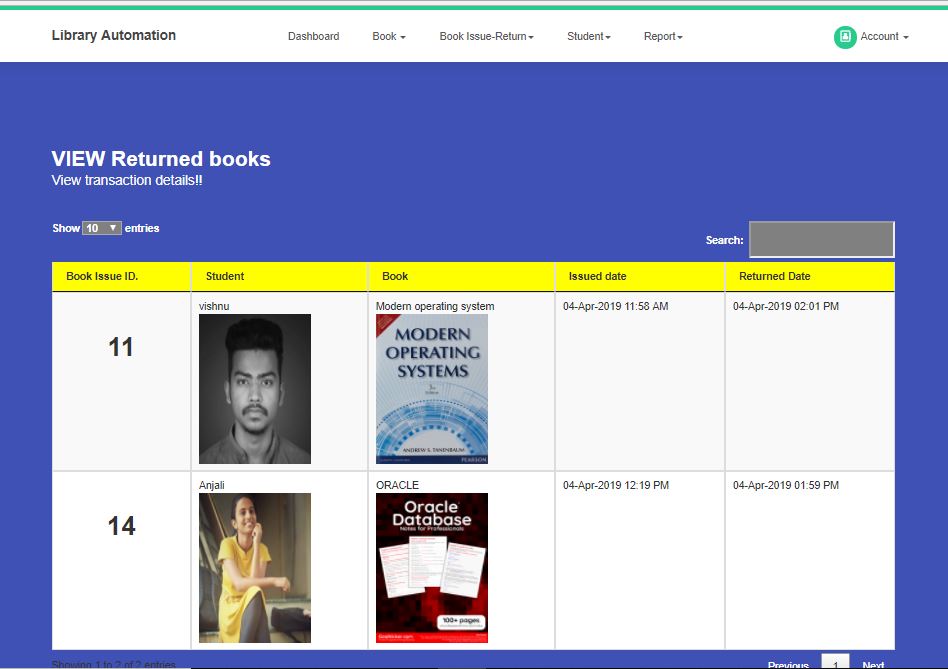
**View requestpage:**

****

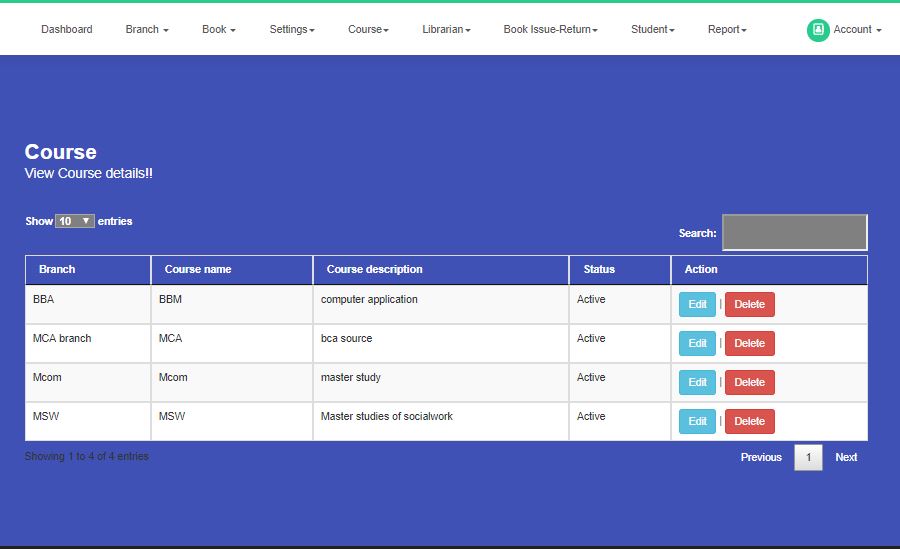
**View issue request:**

****

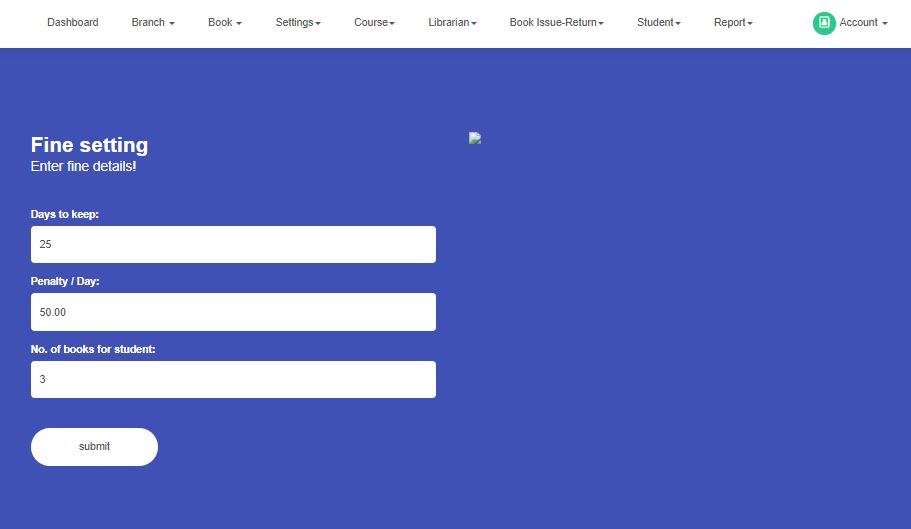
**Bookreturn page:**

****

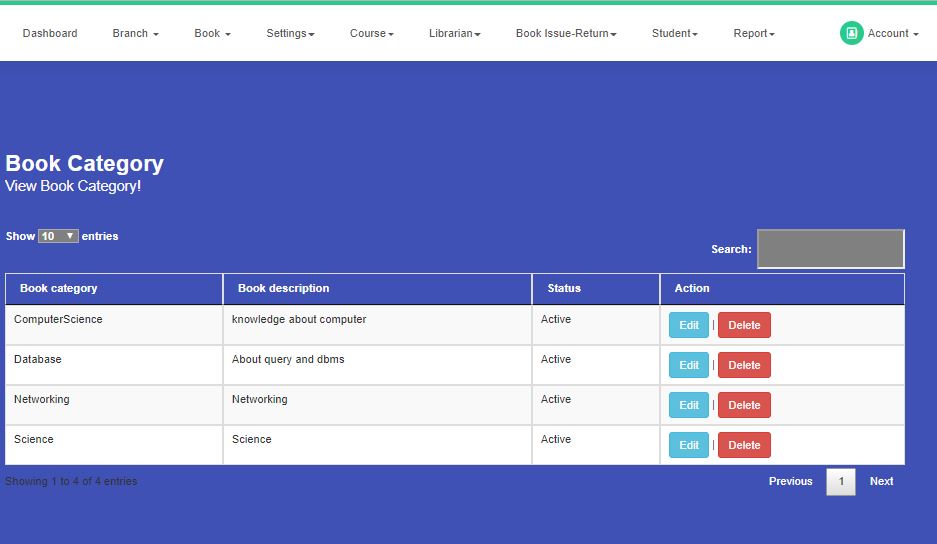
**Course page:**

****

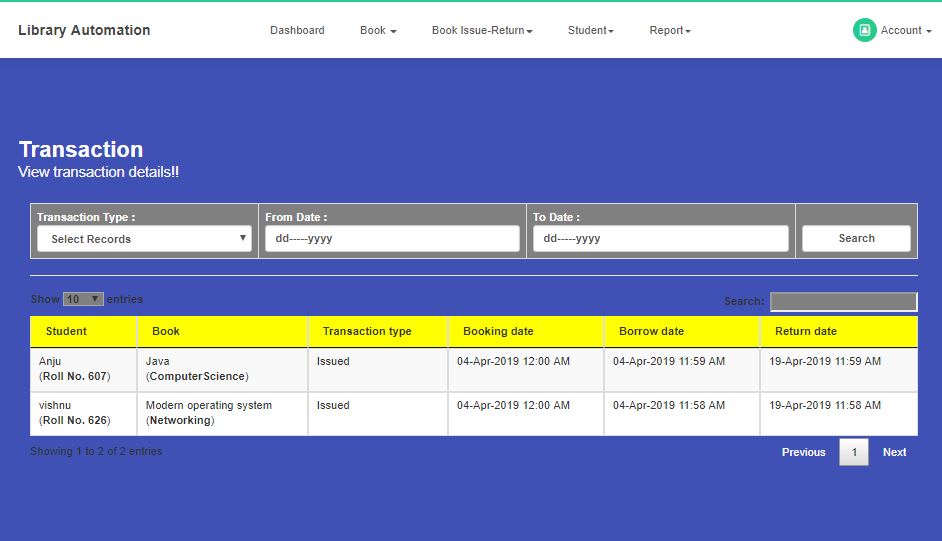
**Finesetting:**

****

**Bookcategory page:**

****

**Transaction page:**

****

**CONCLUSION**

This website offers a computerized library management system that benefits both students and library staff. It facilitates the entire process online, allowing students to search for books and staff to generate reports and manage book transactions. Additionally, the system features a student login where students can view the status of issued books, request new books, or provide suggestions.

There is also a teacher’s login feature, enabling teachers to upload lecture notes and offer suggestions to the library. Teachers can also post information about workshops or events occurring within the college or nearby institutions on the online notice board.

Looking to the future, there is potential to expand this system by incorporating additional features such as online lecture videos and tutorials uploaded by teachers, an online assignment submission facility, and a group chat feature where students can discuss various engineering topics. These enhancements aim to make the system more interactive, user-friendly, and comprehensively meet the needs of all users.

**BIBILOGRAPHY**

**Book References:**

* Meloni, J. Sams Teach Yourself PHP, MySQL and Apache All in One.
* Nixon, R. Learning PHP, MySQL, JavaScript, and CSS: A Step-by-Step Guide to Creating Dynamic Websites.

**Website References:**

* W3Schools. PHP Tutorial. Available at: [http://www.w3schools.com/php/default.asp](http://www.w3schools.com/php/default.asp%20) (Accessed: [12/01/2024).
* TutorialsPoint. PHP Tutorial. Available at: [http://www.tutorialspoint.com/php/](http://www.tutorialspoint.com/php/%20) (Accessed: 12/01/2024).
* W3Schools. MySQL Tutorial. Available at: [http://www.w3schools.com/php/php\_mysql\_intro.asp](http://www.w3schools.com/php/php_mysql_intro.asp%20) (Accessed: 05/02/2024).
* TutorialsPoint. MySQL Tutorial. Available at: [http://www.tutorialspoint.com/mysql/index.htm](http://www.tutorialspoint.com/mysql/index.htm%20) (Accessed: 06/02/2024).
* W3Schools. JavaScript Tutorial. Available at: [http://www.w3schools.com/js/default.asp](http://www.w3schools.com/js/default.asp%20) (Accessed: 06/02/2024).
* TutorialsPoint. JavaScript Tutorial. Available at: [http://www.tutorialspoint.com/javascript/index.htm](http://www.tutorialspoint.com/javascript/index.htm%20) (Accessed: 10/02/2024).
* W3Schools. CSS Tutorial. Available at: [http://www.w3schools.com/css/default.asp](http://www.w3schools.com/css/default.asp%20) (Accessed: 13/02/2024).
* TutorialsPoint. CSS Tutorial. Available at: [http://www.tutorialspoint.com/css/index.htm](http://www.tutorialspoint.com/css/index.htm%20) (Accessed: 20/02/2024).
* W3Schools. HTML Tutorial. Available at: [http://www.w3schools.com/html/default.asp](http://www.w3schools.com/html/default.asp%20) (Accessed: 25/02/2024).
* TutorialsPoint. HTML Tutorial. Available at: [http://www.tutorialspoint.com/html/index.htm](http://www.tutorialspoint.com/html/index.htm%20) (Accessed: 02/03/2024).
* TutorialsPoint. HTML5 Tutorial. Available at: [http://www.tutorialspoint.com/html5/index.htm](http://www.tutorialspoint.com/html5/index.htm%20) (Accessed: 11/03/2024).
* W3Schools. AJAX Tutorial. Available at: [http://www.w3schools.com/ajax/default.asp (](http://www.w3schools.com/ajax/default.asp%20()Accessed: 21/03/2024).
* TutorialsPoint. AJAX Tutorial. Available at: [http://www.tutorialspoint.com/ajax/index.htm](http://www.tutorialspoint.com/ajax/index.htm%20) (Accessed: 26/03/2024).