# The Library

Library Management System

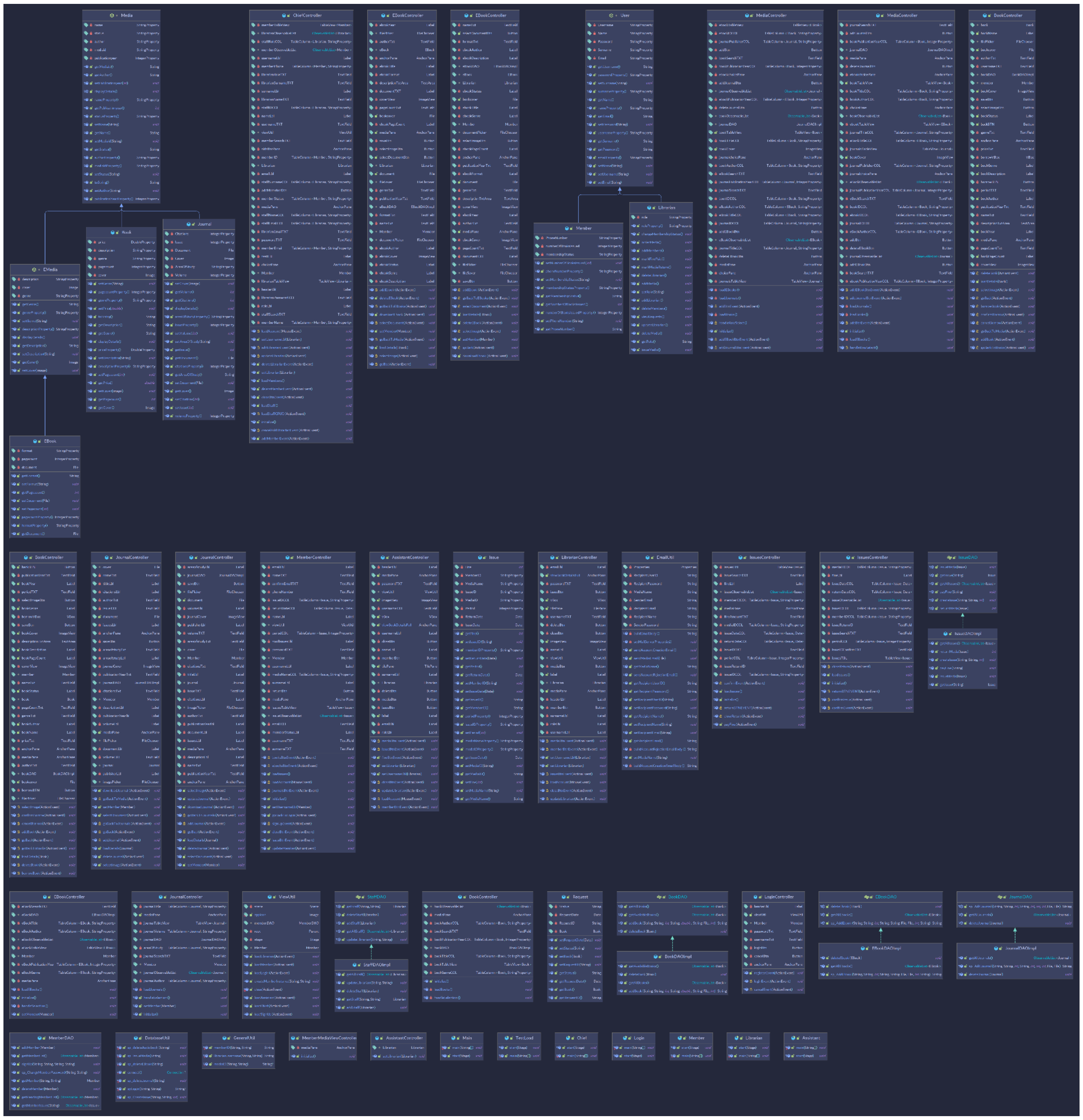
Version 1.0.0

Prepared by
Adriaan Grobler
Mia Hinda
Petrina Maharero
Sakaria Nghivafe

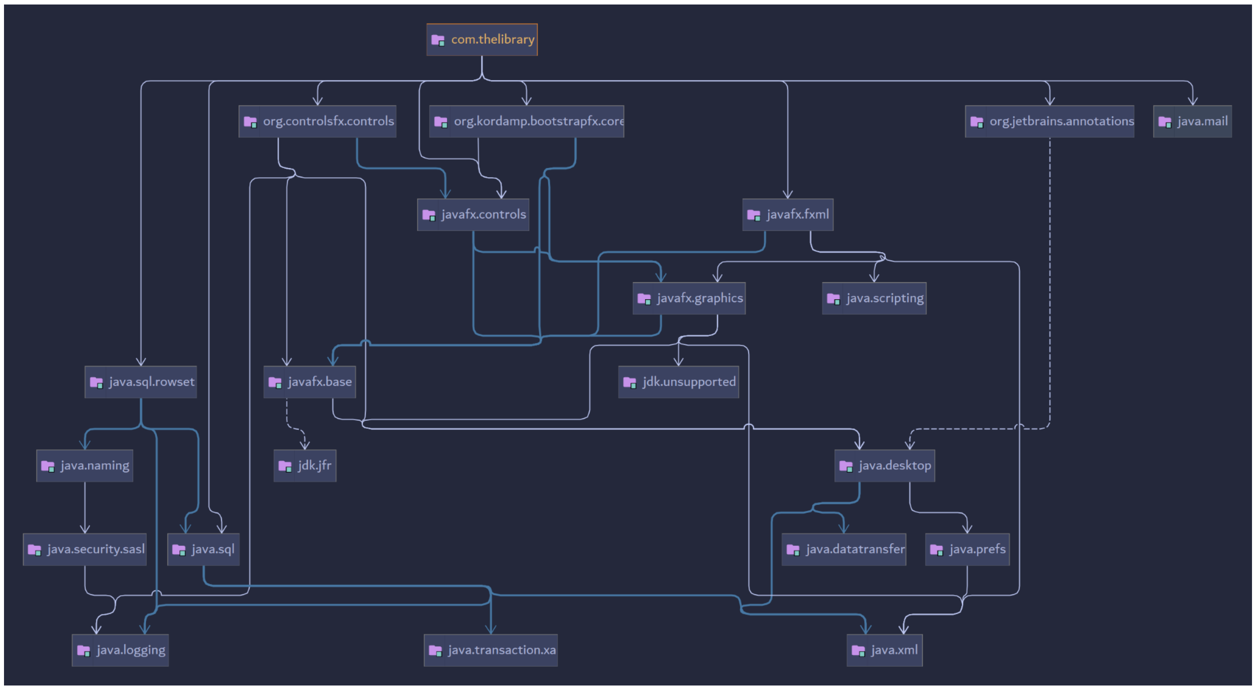
# Table of Contents

# Diagrams

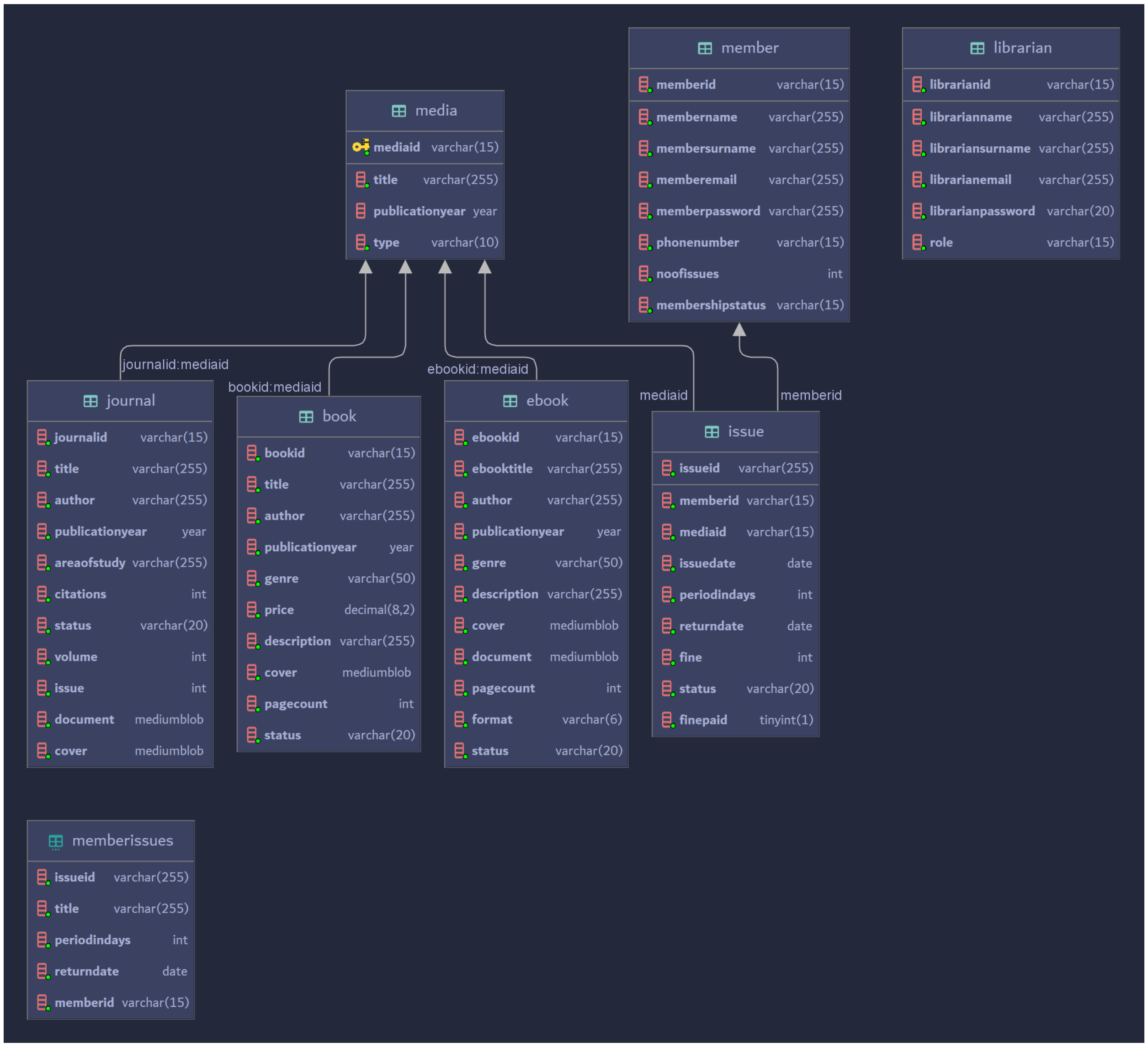
## Class Diagram



## Modules Diagram



## Entity Relation Diagram



# Introduction

## Purpose

This document describes The Library Version 1.0.0. It contains the functional and non-functional requirements of the project and provides a guideline to be the systems developers and technicians who will overlook the development and maintenance of the project.

## Scope

The Library(TL) is a library management system(LMS). TL provides a digital implementation of current manual systems in place.
It is designed with librarians and library members in mind. TL provides a complete graphical user interface to facilitate library management processes and library usage from library members.
Existing or new libraries can use the system to manage its media . This includes, but is not limited to, adding and borrowing books, adding new members and librarians.
The Library is a powerful system that works well for large and small libraries. It provides a free easy-to-use system for rising libraries.

## Intended Audience

Systems developers, testers, library owners and managers are the intended audience of this document.

## Definitions and Acronyms

- ➔ TL: The Library
- ➔ GUI: Graphical User Interface
- ➔ DAO: Database Access Object
- ➔ DAOI: Database Access Object Interface
- ➔ DAOCC: Database Access Object Concrete Class
- ➔ MO: Model Object

## References

- ➔ https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document
- ➔ IEEE 830-1998 standard for writing SRS documents.
- ➔ https://www.perforce.com/resources/9-tips-writing-useful-requirements
- ➔ https://dipeshagrawal.files.wordpress.com/2018/07/srs-library-management-system.pdf

# Overall Description

## User Needs

The users of the system are members and librarians of the university who act as administrator to maintain the system. The members are assumed to have basic knowledge of computers and internet browsing. The administrators of the system should have more knowledge of the internals of the system and are able to rectify the small problems that may arise due to disk crashes, power failures and other catastrophes to maintain the system. The proper user interface, user manual, online help and the guide to install and maintain the system must be sufficient to educate the users on how to use the system without any problems. The admin provides certain facilities to the users in the form of:-

Backup and Recovery

Forgot Password

Data migration, i.e. whenever the user registers for the first time, then the data is stored in the server

Data replication i.e. if the data is lost in one branch, it is still stored with the server

Auto Recovery i.e. frequently auto saving the information

Maintaining files i.e. File Organisation  The server must be maintained regularly and it has to be updated from time to time

# Assumptions and Dependencies

## Assumptions

### Budget

The first version of The Library is local thus the currency used is the Namibian dollar and will be used throughout the project.

The cost of resources will remain the same throughout the project unless the user makes a request beyond the scope of the current version.

There will be an initial cost for instalment of the project which will be followed by periodic billing for maintenance and technical support.

### Resources

Client will have access to the necessary resources of The Library to utilise the application efficiently to complete tasks and other work in a timely fashion.

### Scope

The scope of the project will not change.

### Environment

The Library will utilise the provided digital environment that meets the requirements for the application to run. This application can be run on existing infrastructure and architecture such as already running servers and computers.

## Dependencies

The Library utilises the Finish to start (FS) dependency type where task B can only start once task A has completed. The user can only access one window at a time, complete the task then move to another window on the application.

If a user logs in they must first input their details for authentication and then they will have access to library resources and the application features.

If a user wishes to book out media from the library, then they first need to check availability of the media then they can request for it and then they can retrieve it.

Some elements of the application require approval from the staff so after sending a request for media or to join as a member the user must wait for the authorised library staff to process those requests.

The above mentioned examples confirm and demonstrate the dependency type of The Library application.

## System Features and Requirements

### Functional Requirements

#### Common Functions:

| | |
|---|---|
| Requirement ID: | LMS001 |
| Title: | Login |
| Description: | All users should login before they are able to view or modify and information on the system. |
| Priority: | 1 |

| | |
|---|---|
| Requirement ID: | LMS002 |
| Title: | Invalid credentials |
| Description: | All users will be alerted if an invalid password and/or username is entered. |
| Priority: | 2 |

| | |
|---|---|
| Requirement ID: | LMS003 |
| Title: | Search media |
| Description: | Users will be able to search for media by title, date, author or ID |
| Priority: | 1 |

## Members:

| | |
|---|---|
| Requirement ID: | LMS004 |
| Title: | Register |
| Description: | All new members are required to register before accessing the rest of the system. |
| Priority: | 3 |

| | |
|---|---|
| Requirement ID: | LMS005 |
| Title: | View member issues |
| Description: | Members will be able to view all details of media that has been issued to them. |
| Priority: | 1 |

| | |
|---|---|
| Requirement ID: | LMS006 |
| Title: | View books |
| Description: | Members must click the books button to view books. |
| Priority: | 2 |

| | |
|---|---|
| Requirement ID: | LMS007 |
| Title: | View ebooks |
| Description: | Members must click e-books button to view e-books. |
| Priority: | 2 |

| | |
|---|---|
| Requirement ID: | LMS008 |
| Title: | View journals |
| Description: | Members must click journals button to view journals. |
| Priority: | 2 |

## Chief Librarian:

| | |
|---|---|
| Requirement ID: | LMS010 |
| Title: | Search staff |
| Description: | The chief librarian can search staff accounts by their identifiers |
| Priority: | 1 |

| | |
|---|---|
| Requirement ID: | LMS011 |
| Title: | Add staff |
| Description: | The chief librarian can add new staff members and their information |
| Priority: | 1 |

| | |
|---|---|
| Requirement ID: | LMS012 |
| Title: | Filter activity |
| Description: | The chief librarian can filter all activities on system such as issuing of media by date and userID. |

| | |
|---|---|
| Priority: | 3 |

| | |
|---|---|
| Requirement ID: | LMS013 |
| Title: | Delete staff |
| Description: | The chief librarian can delete users from the system. |
| Priority: | 1 |

| | |
|---|---|
| Requirement ID: | LMS014 |
| Title: | Update staff |
| Description: | The chief librarian can update the personal information of staff members. |
| Priority: | 1 |

| | |
|---|---|
| Requirement ID: | LMS015 |
| Title: | Add members |
| Description: | The chief librarian will confirm new member registration before they can access the system. |
| Priority: | 1 |

| | |
|---|---|
| Requirement ID: | LMS016 |
| Title: | Delete members |
| Description: | The chief librarian can delete profiles of any member |
| Priority: | 1 |

| | |
|---|---|
| Requirement ID: | LMS017 |
| Title: | Search members |
| Description: | The chief librarian can search for any member on the system by their identifiers. |
| Priority: | 1 |

## Librarian:

| | |
|---|---|
| Requirement ID: | LMS018 |
| Title: | Add media |
| Description: | The librarian will add new media and their details to the system. |

| | |
|---|---|
| Priority: | 1 |

| | |
|---|---|
| Requirement ID: | LMS019 |
| Title: | Delete media |
| Description: | The librarian will delete media from system when needed. |
| Priority: | 1 |

| | |
|---|---|
| Requirement ID: | LMS020 |
| Title: | Search issues |
| Description: | The librarian can search through all issued media based on their identifiers. |
| Priority: | 2 |

| | |
|---|---|
| Requirement ID: | LMS021 |
| Title: | Confirm issue |
| Description: | The librarian will confirm the issuing of media on the system. |
| Priority: | 1 |

| | |
|---|---|
| Requirement ID: | LMS022 |
| Title: | Mark returned |
| Description: | The librarian can mark any media as returned and remove it from the issue media list. |
| Priority: | 1 |

## Assistant Librarian

| | |
|---|---|
| Requirement ID: | LMS023 |
| Title: | Mark returned |
| Description: | The assistant librarian can mark any media as returned and remove it from the issue media list. |
| Priority: | 1 |

| | |
|---|---|
| Requirement ID: | LMS024 |
| Title: | Confirm issue |

|  |  |
|---|---|
| Description: | The assistant librarian will confirm the issuing of media on the system. |
| Priority: | 1 |

## External Interface Requirements

### GUI

The software provides a good graphical interface for the user and the administrator can operate on the system, performing the required task such as create, update, and view the details of the book.

- It allows users to view quick reports like Book Issued/Returned in between particular times.
- It provides stock verification and search facility based on different criteria.
- The user interface must be customizable by the administrator
- All the modules provided with the software must fit into this graphical user interface and accomplish to the standard defined
- The design should be simple and all the different interfaces should follow a standard template
- The user interface should be able to interact with the user management module and a part of the interface must be dedicated to the login/logout module

#### Login Interface

- In case the user is not yet registered, he can enter the details and register to create his account.
- Once his account is created he can ‘Login’ which asks the user to type his username and password.
- If the user entered either his username or password incorrectly, then an error message appears.

## Search

The member or librarian can enter the type of book he is looking for and the title he is interested in,then he can search for the required book by entering the book name.

## Categories View

Categories view shows the categories of books available and provides ability to the librarian to add/edit or delete categories from the list.

## Librarian’s Control Panel

This control panel will allow librarians to add/remove users; add, edit, or remove a resource. And manage lending options.

## System Features

- The users of the system should be provided the surety that their account is secure. This is possible by providing :

- User authentication and validation of members using their unique member ID

- Proper monitoring by the administrator, which includes updating account status, showing a pop-up if the member attempts to issue several books that exceed the limit provided by the library policy, assigning fine to members who skip the date of return.

- Proper accountability which includes not allowing a member to see another member's account. Only the administrator will see and manage all member accounts.

## Nonfunctional Requirements

### Security

- The system uses a secured database hosted locally on the servers of the library
- There are 4 classes of users, Assistant Librarian, Librarian, Chief Librarian and Member each with different database access constraints
- Normal users, Members, will not be able to edit any data in the database apart from their own
- Users need to put in their credentials correctly before they can access into the system

### Capacity

- The system will require at least 1 GB of storage for the installation
- The system will require 4 GB of storage for storing data in the database
- Further, the storage will need to increase as the database grows and more information is added to the system

### Compatibility

#### Minimum Hardware System Requirements

Processor: Intel Core i5 10th Generation or Greater
Memory: 4 GB RAM
Storage: 8 GB internal storage drive

### Minimum Software System Requirements

Operating System: Windows 10
Database Engine: MySQL V8.0.25
Java Runtime Environment

## Reliability and Availability

- The database will have to be online at all times
- The user only needs to have access to the system when using it

## Maintainability and Manageability

- Fixing components vary between different components
- Average time to fix a component would be a day or two
- The administrator can easily manage this system

## Error Handling

- TL handles expected and unexpected errors
- TL handles errors in a way that prevents the loss of information and long downtime periods

## Performance

- TL will accommodate a large volume of media and users with minimal fault
- GUI response will take no longer than 10 seconds
- Retrieval and updating of information will take no longer than 10 seconds

## Interfaces

### Login Interface

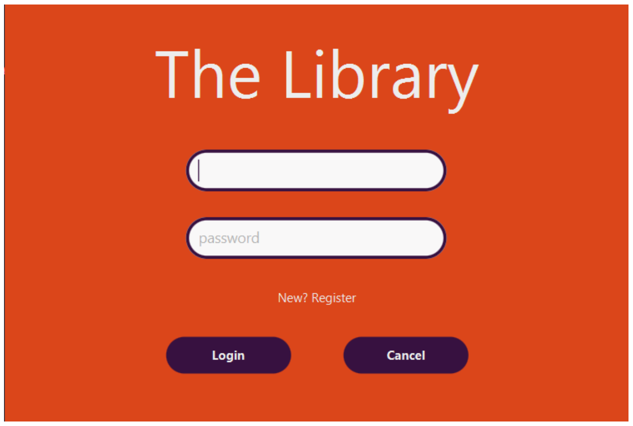
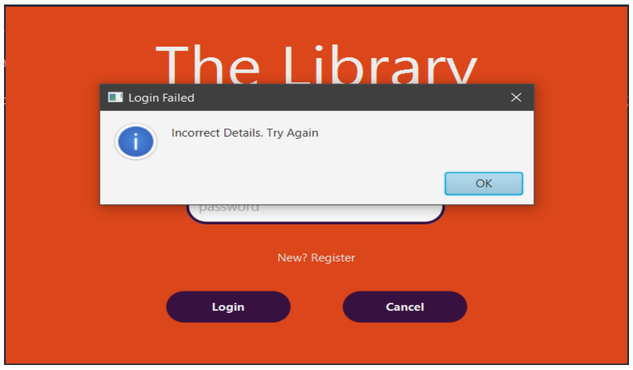


Figure 1: Login



Incorrect Details. Try Again

Figure 2: Incorrect Credentials

## SignUp Interface

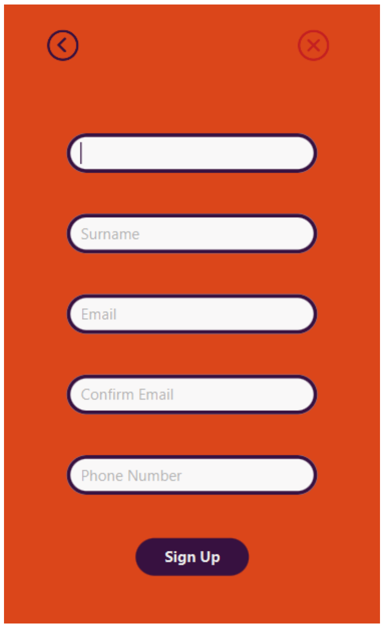


Figure: 3 Sign Up

## Member Interface

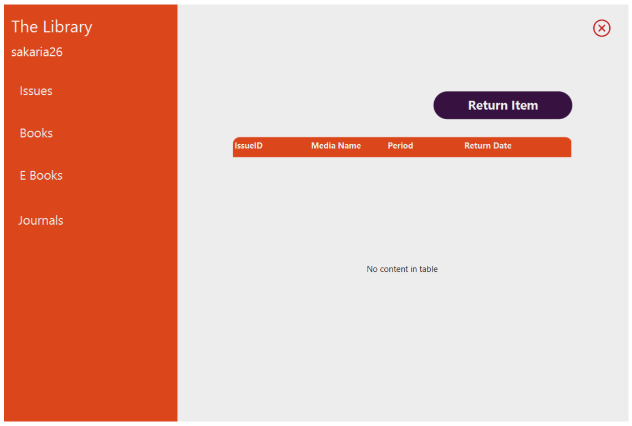
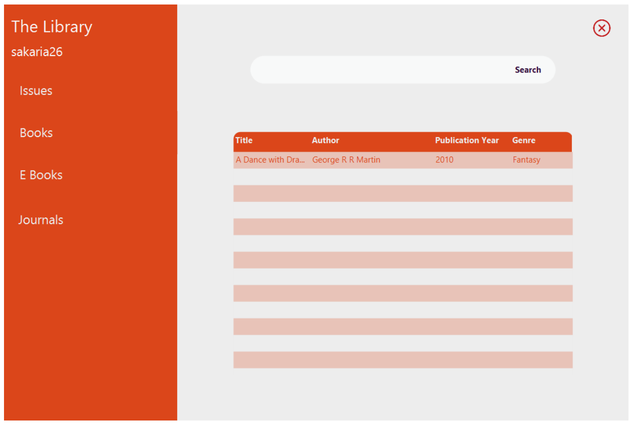


Figure 4: Member Home Page

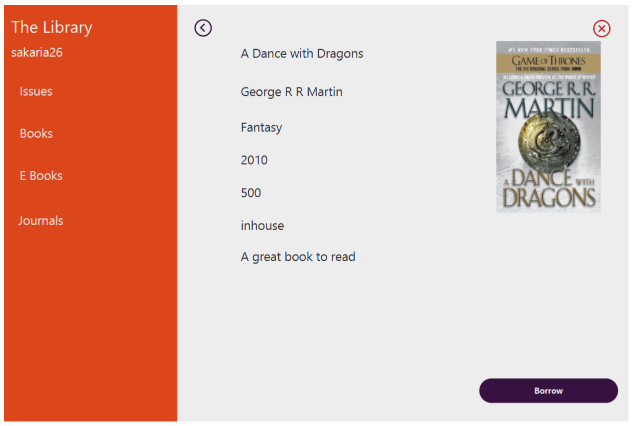
Here the member sees all their issues and can return their item through this interface. However a librarian needs to mark that they have returned the issue



| Title | Author | Publication Year | Genre |
|---|---|---|---|
| A Dance with Dra... | George R R Martin | 2010 | Fantasy |

Figure 5: Book View for Members

Here the members view all books that are currently available for them to borrow



A Dance with Dragons

George R R Martin

Fantasy

2010

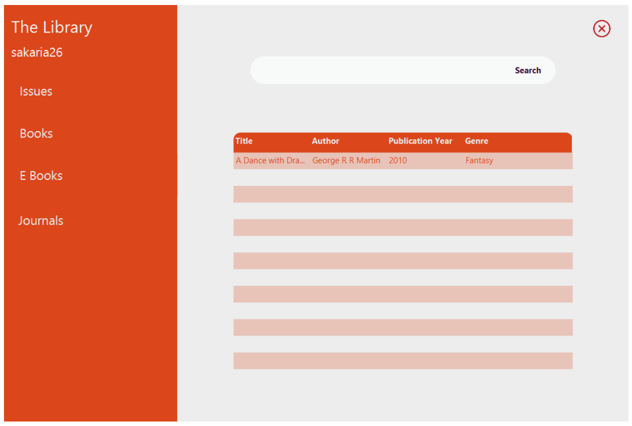
500

inhouse

A great book to read

Figure 6: Borrowing Book View

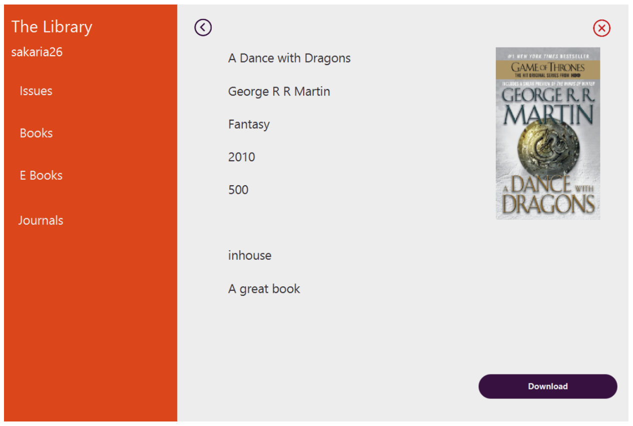
Here the members view the details of the book they wish to borrow



| Title | Author | Publication Year | Genre |
|---|---|---|---|
| A Dance with Dra... | George R R Martin | 2010 | Fantasy |

## Figure 7: Ebook View for Member

Here the members view all ebooks that are in the system



A Dance with Dragons

George R R Martin

Fantasy

2010

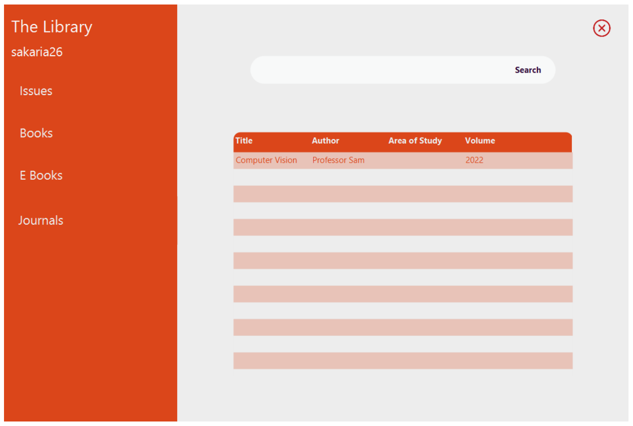
500

inhouse

A great book

## Figure 8: Download Ebook View

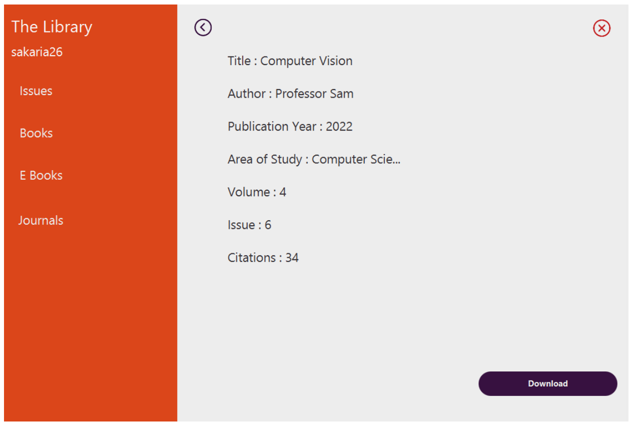
Here the members view the details of the ebook and can download it



| Title | Author | Area of Study | Volume |
|---|---|---|---|
| Computer Vision | Professor Sam | | 2022 |

## Figure 9: Journals View For Members

Here the members view all journals that are in the system



Title : Computer Vision

Author : Professor Sam

Publication Year : 2022

Area of Study : Computer Scie...

Volume : 4

Issue : 6

Citations : 34

Figure 10: Download Journal View for Members

Here the members view the details of the journal and can download it

## Librarian Interface

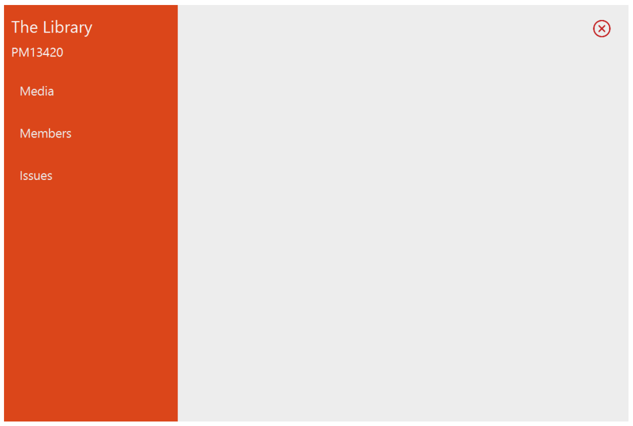
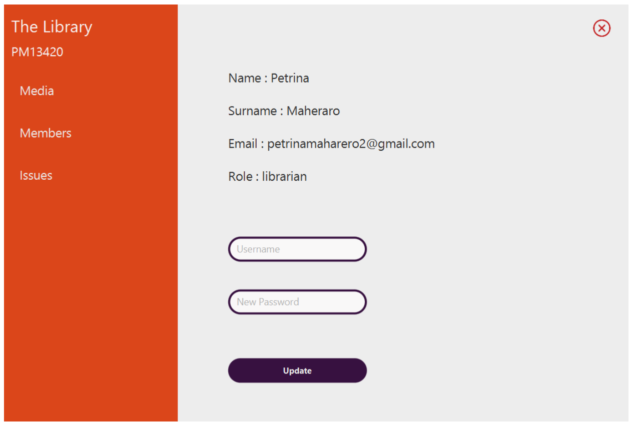


Figure 11: Librarian Home Page

This is the home page for when a librarian logs in



Name : Petrina

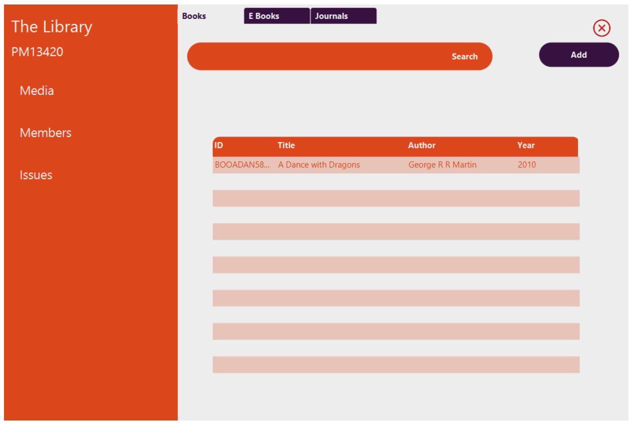
Surname : Maheraro

Email : petrinamaharero2@gmail.com

Role : librarian

Figure 12: Librarian Account information

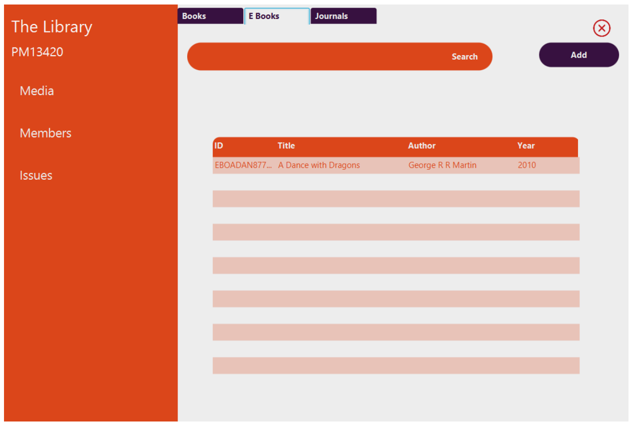
Here the librarian views their account details



| ID | Title | Author | Year |
|---|---|---|---|
| BOOADAN58... | A Dance with Dragons | George R R Martin | 2010 |

Figure 13: Librarian Books View

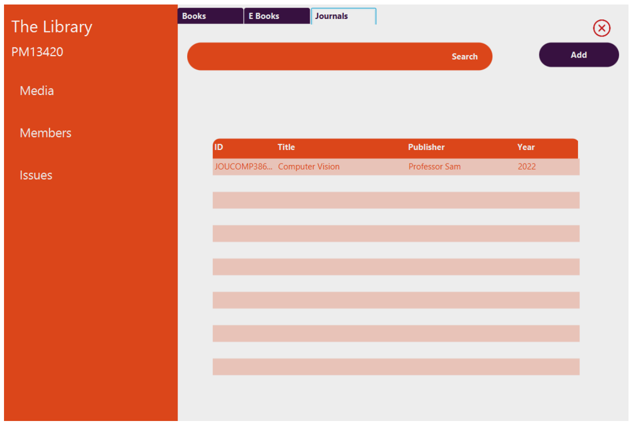
Here the librarians view all books in the system



| ID | Title | Author | Year |
|---|---|---|---|
| EBOADAN877... | A Dance with Dragons | George R R Martin | 2010 |

Figure 14: Librarian Ebooks View

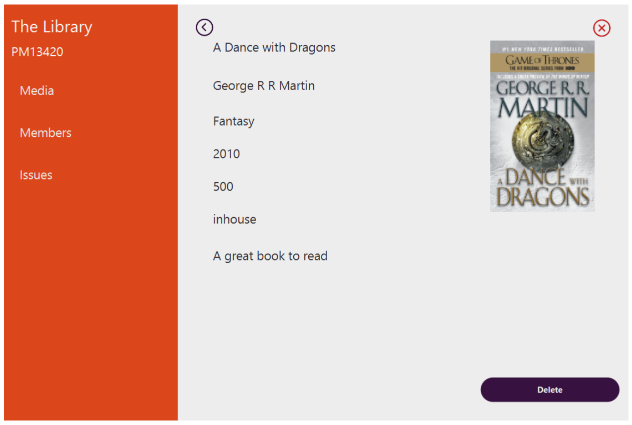
Here the librarians view all ebooks in the system



| ID | Title | Publisher | Year |
|---|---|---|---|
| JOUCOMP386... | Computer Vision | Professor Sam | 2022 |

Figure 15: Librarian Journals View

Here the librarians view all journals in the system



A Dance with Dragons

George R R Martin

Fantasy

2010

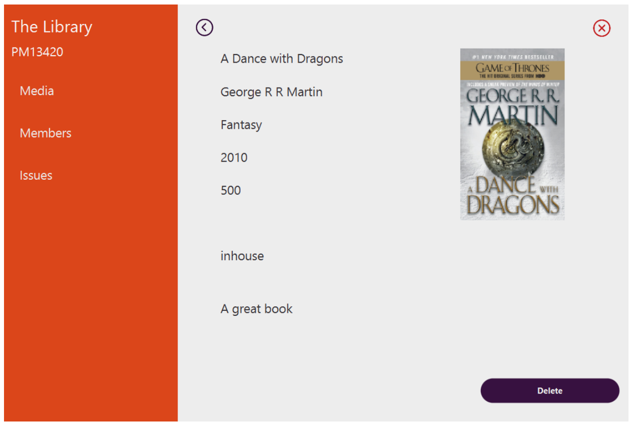
500

inhouse

A great book to read

Figure 16: Librarian Book Details View

Here the librarian view all the details of the book they selected



A Dance with Dragons

George R R Martin

Fantasy

2010

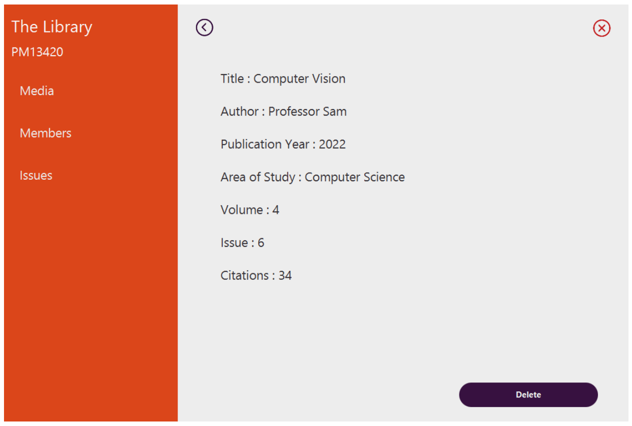
500

inhouse

A great book

Figure 17: Librarian Ebooks Details View

Here the librarian view all the details of the ebook they selected



Title : Computer Vision

Author : Professor Sam

Publication Year : 2022

Area of Study : Computer Science

Volume : 4

Issue : 6

Citations : 34

Figure 18: Librarian Journals Details View

Here the librarian views all the details of the journal they selected

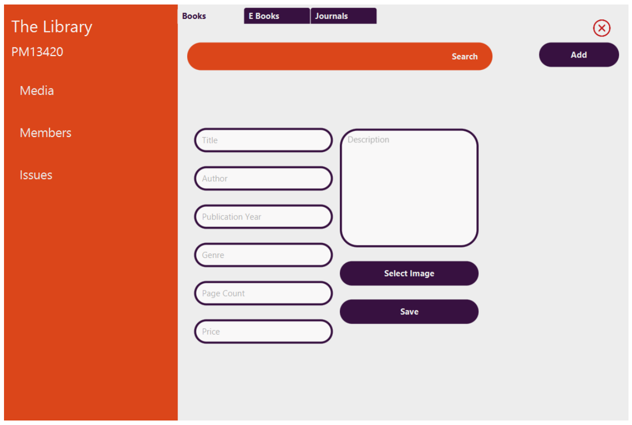


Figure 19: Adding Book

Here, the librarians add a book to the system

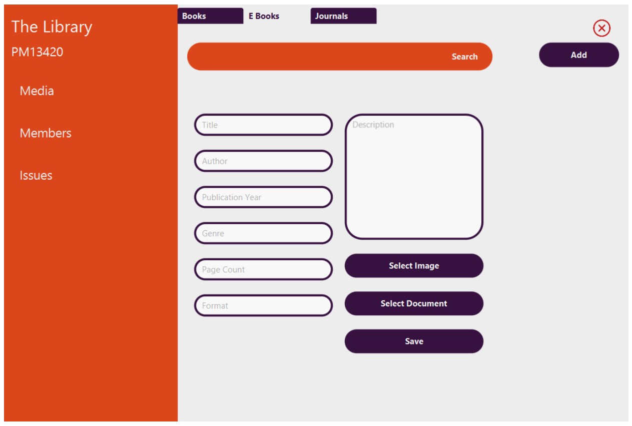


Figure 20: Adding Book

Here, the librarians add a ebook to the system

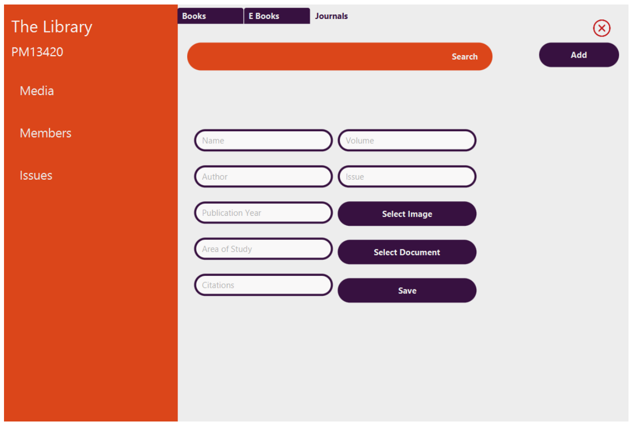
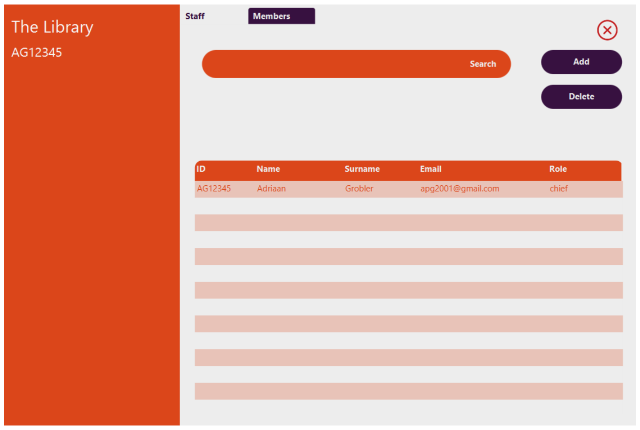


Figure 21: Adding Journal

Here, the librarians add a journal to the system

## Chief Librarian Interface



| ID | Name | Surname | Email | Role |
|---|---|---|---|---|
| AG12345 | Adriaan | Grobler | apg2001@gmail.com | chief |

Figure 22: staff view for chief librarian

Here the chief librarian views all the librarians in the system and can add and delete librarians

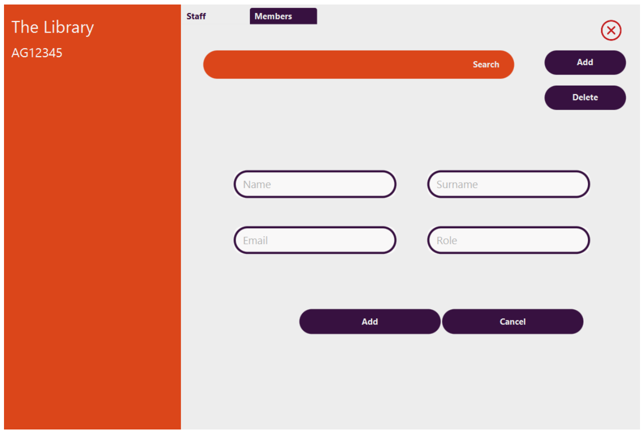
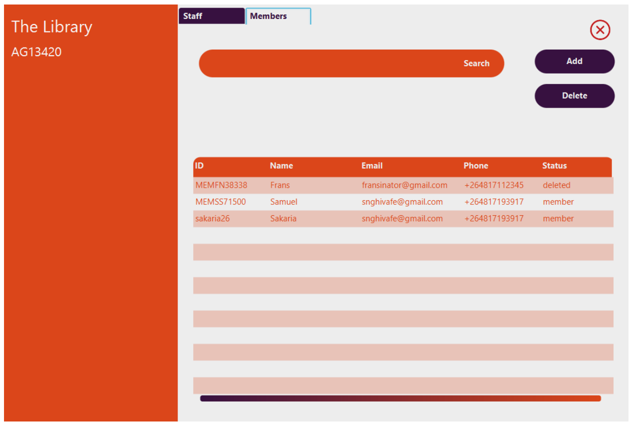


Figure 23: The add staff view for chief librarian

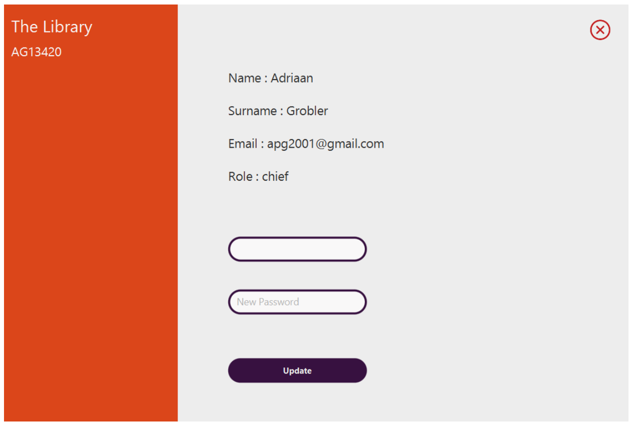
Here the chief librarian adds a librarian to the system



| ID | Name | Email | Phone | Status |
|---|---|---|---|---|
| MEMFN38338 | Frans | fransinator@gmail.com | +264817112345 | deleted |
| MEMSS71500 | Samuel | snghivafe@gmail.com | +264817193917 | member |
| sakaria26 | Sakaria | snghivafe@gmail.com | +264817193917 | member |

Figure 24: member view for chief librarian

Here the chief librarian views all the members and can add or delete members



Name : Adriaan

Surname : Grobler

Email : apg2001@gmail.com

Role : chief

Figure 25: member view for chief librarian

Here the chief librarian can view their account details and update their password here

## Assistant Librarian Interface

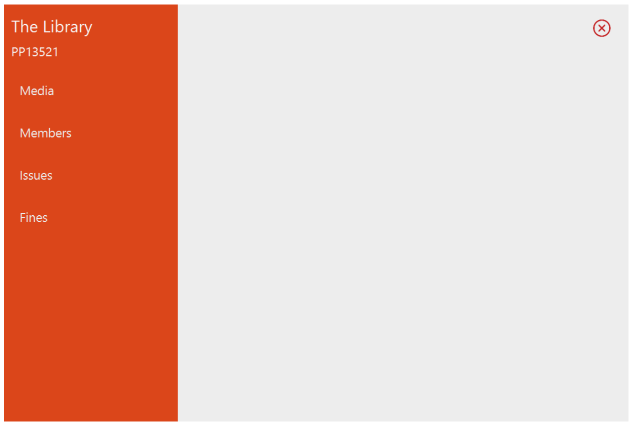
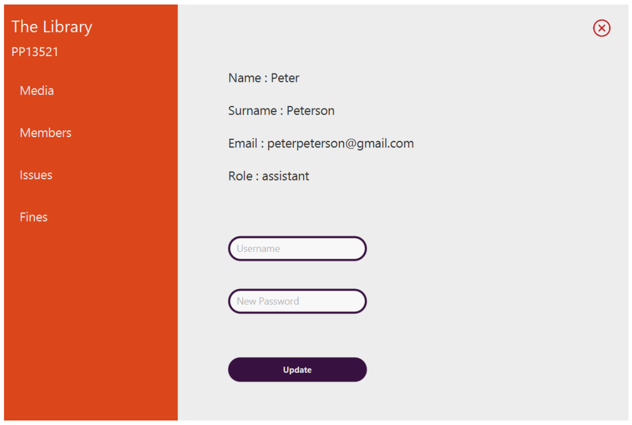


Figure 26: Assistant Librarian Home page

This is the landing page for assistant librarians



Name : Peter

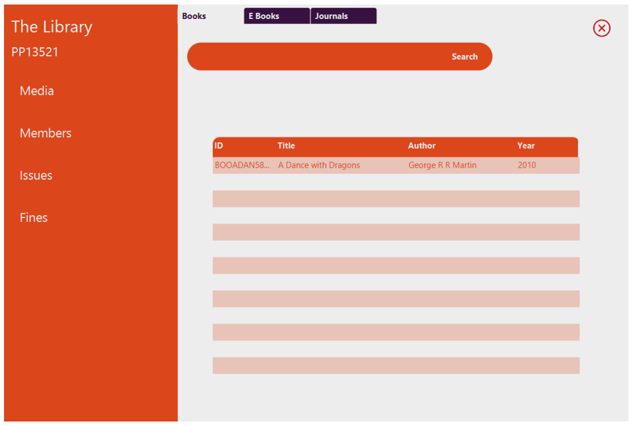
Surname : Peterson

Email : peterpeterson@gmail.com

Role : assistant

Figure 27: Assistant Librarian Account Page

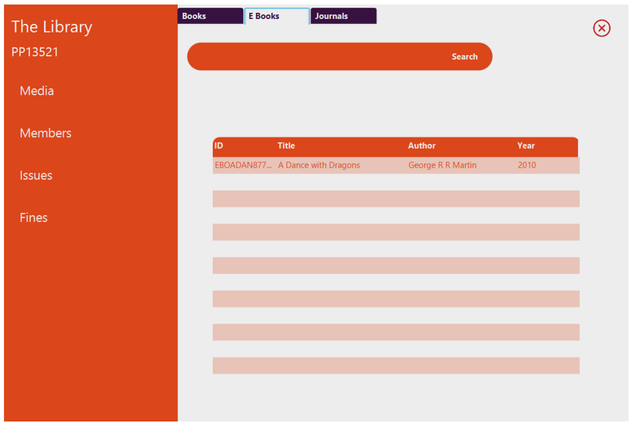
This is the page showing all details of the assistant librarian and can be updated here



| ID | Title | Author | Year |
|---|---|---|---|
| BOOADAN58... | A Dance with Dragons | George R R Martin | 2010 |

Figure 28: Assistant Librarian Book View

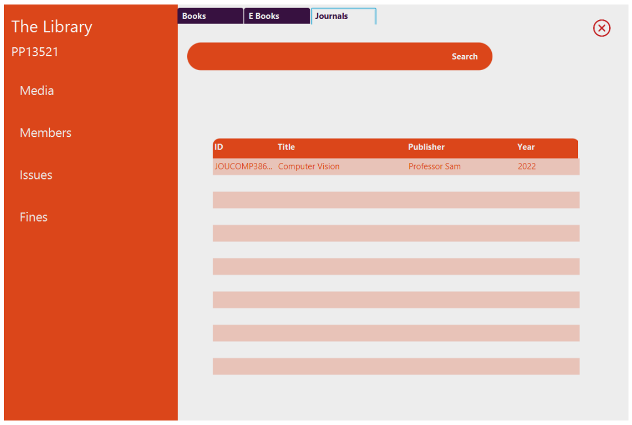
Here assistant librarians view all the books in the system



| ID | Title | Author | Year |
|---|---|---|---|
| EBOADAN877... | A Dance with Dragons | George R R Martin | 2010 |

Figure 29: Assistant Librarian Ebook View

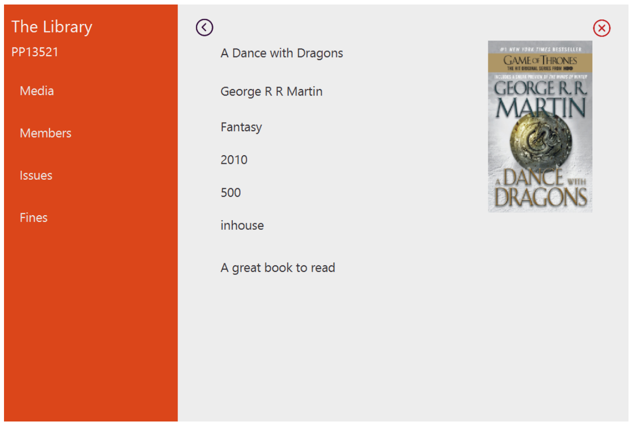
Here assistant librarians view all the ebooks in the system



| ID | Title | Publisher | Year |
|---|---|---|---|
| JOUCOMP386... | Computer Vision | Professor Sam | 2022 |

Figure 29: Assistant Librarian Journal View

Here assistant librarians view all the journals in the system



A Dance with Dragons

George R R Martin

Fantasy

2010

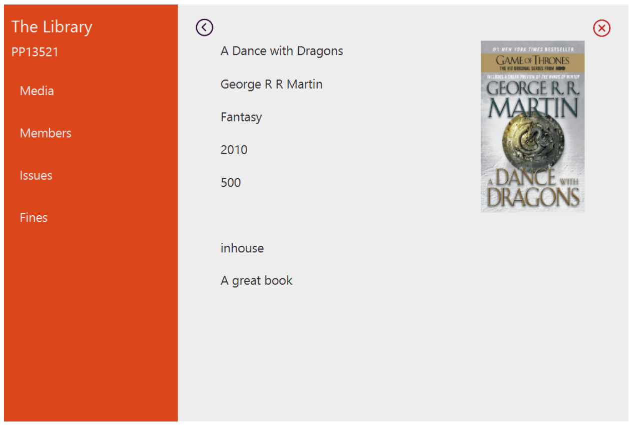
500

inhouse

A great book to read

Figure 30: Assistant Librarian Book Details View

Here assistant librarians view the details of the selected book



A Dance with Dragons

George R R Martin

Fantasy

2010

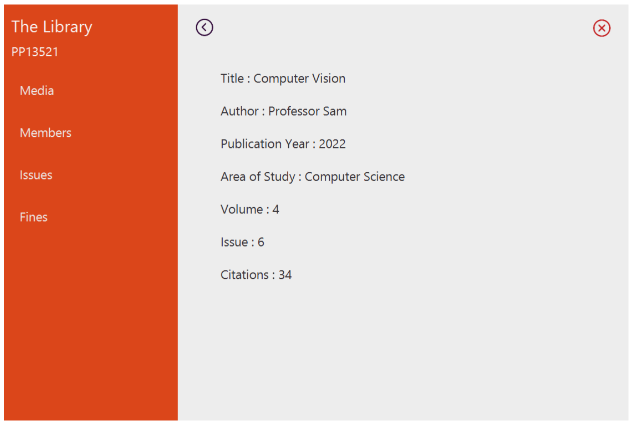
500

inhouse

A great book

Figure 31: Assistant Librarian Ebook Details View

Here assistant librarians view the details of the selected ebook



Title : Computer Vision

Author : Professor Sam

Publication Year : 2022

Area of Study : Computer Science

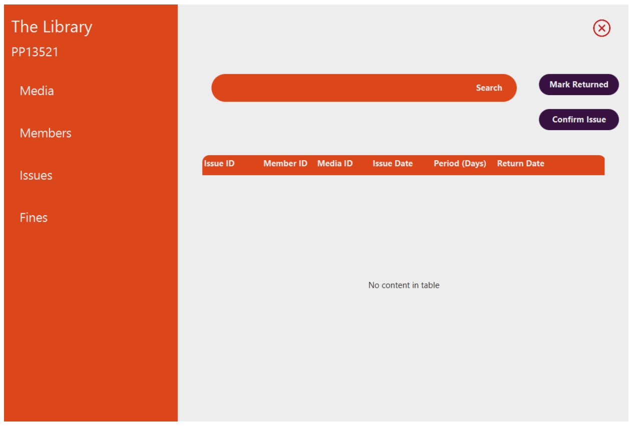
Volume : 4

Issue : 6

Citations : 34

Figure 32: Assistant Librarian Journal Details View

Here assistant librarians view the details of the selected journal



| Issue ID | Member ID | Media ID | Issue Date | Period (Days) | Return Date |
|---|---|---|---|---|---|

No content in table

Figure 33: Assistant Librarian Issues View

Here assistant librarians view the issues that are currently not returned

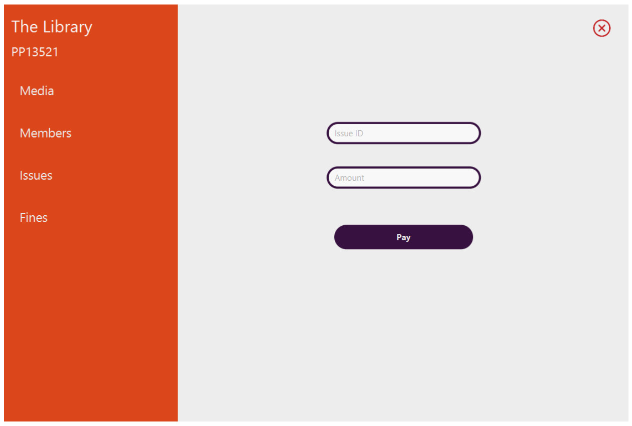
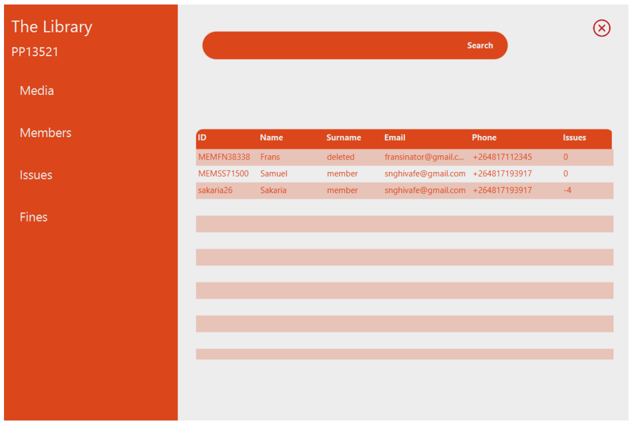


Figure 34: Assistant Librarian Fines View

Here assistant librarians input any fines that need to be paid



| ID | Name | Surname | Email | Phone | Issues |
|---|---|---|---|---|---|
| MEMFN38338 | Frans | deleted | fransinator@gmail.c... | +264817112345 | 0 |
| MEMSS71500 | Samuel | member | snghivafe@gmail.com | +264817193917 | 0 |
| sakaria26 | Sakaria | member | snghivafe@gmail.com | +264817193917 | -4 |

Figure 33: Assistant Librarian Members View

Here assistant librarians view the members in the system

# Technologies

## Development

- Language: Java 17
- IDE: IntelliJ Idea Ultimate
- Framework: JavaFX

## Design Pattern

- The system uses the DAO Design Pattern
- The DAO is a design pattern used to separate low level data accessing API from high level business services
- Components of DAO
  - DAOI
    - Interfaces defining standard operations to be performed on model
  - DAOCC
    - Implements DAOI
    - Gets data from database
  - MO
    - Class containing attributes and getters and setters
    - Used to store retrieved data using DAO Class

## Database

- MySQL 8.0.25

# Database Creation

- The code provided below is how the database is created

## Tables

```
CREATE TABLE `media` (
                    `mediaid` varchar(15) NOT NULL,
                    `title` varchar(255) NOT NULL,
                    `publicationyear` year NOT NULL,
                    `type` varchar(10) NOT NULL,
                    PRIMARY KEY (`mediaid`),
                    KEY `idx_Media` (`mediaid`)
);

drop table book;
CREATE TABLE `book` (
                   `bookid` varchar(15) NOT NULL,
                   `title` varchar(255) NOT NULL,
                   `author` varchar(255) NOT NULL,
                   `publicationyear` year NOT NULL,
                   `genre` varchar(50) NOT NULL,
                   `price` decimal(8,2) NOT NULL,
                   `description` varchar(255) NOT NULL,
                   `cover` mediumblob NOT NULL,
                   `pagecount` int NOT NULL,
                   `status` varchar(20) NOT NULL DEFAULT 'inhouse',
                   UNIQUE KEY `bookid` (`bookid`),
                   KEY `idx_Book` (`bookid`),
                   CONSTRAINT `bookid` FOREIGN KEY (`bookid`) REFERENCES `media`
(`mediaid`)
);

drop table ebook;
CREATE TABLE `ebook` (
                    `ebookid` varchar(15) NOT NULL,
                    `ebooktitle` varchar(255) NOT NULL,
                    `author` varchar(255) NOT NULL,
                    `publicationyear` year NOT NULL,
                    `genre` varchar(50) NOT NULL,
                    `description` varchar(255) NOT NULL,
                    `cover` mediumblob NOT NULL,
                    `document` mediumblob NOT NULL,
                    `pagecount` int NOT NULL,
                    `format` varchar(6) NOT NULL,
                    `status` varchar(20) NOT NULL DEFAULT 'inhouse',
                    UNIQUE KEY `ebookid` (`ebookid`),
                    KEY `idx_EBook` (`ebookid`),
                    CONSTRAINT `ebookid` FOREIGN KEY (`ebookid`) REFERENCES `media`
(`mediaid`)
);

drop table journal;
CREATE TABLE `journal` (
                     `journalid` varchar(15) NOT NULL,
                     `title` varchar(255) NOT NULL,
                     `author` varchar(255) NOT NULL,
                     `publicationyear` year NOT NULL,
                     `areaofstudy` varchar(255) NOT NULL,
                     `citations` int NOT NULL,
                     `status` varchar(20) NOT NULL DEFAULT 'incomplete',
                     `volume` int NOT NULL,
                     `issue` int NOT NULL,
                     `document` mediumblob NOT NULL,
                     `cover` mediumblob NOT NULL,
                     UNIQUE KEY `journalid` (`journalid`),
                     KEY `idx_Journal` (`journalid`),
                     CONSTRAINT `journalid` FOREIGN KEY (`journalid`) REFERENCES `media`
(`mediaid`)
);

drop table member;
CREATE TABLE `member` (
                    `memberid` varchar(15) NOT NULL,
                    `membername` varchar(255) NOT NULL,
                    `membersurname` varchar(255) NOT NULL,
                    `memberemail` varchar(255) NOT NULL,
                    `memberpassword` varchar(255) NOT NULL,
                    `phonenumber` varchar(15) NOT NULL,
                    `noofissues` int NOT NULL DEFAULT '0',
                    `membershipstatus` varchar(15) NOT NULL DEFAULT 'pending',
                    PRIMARY KEY (`memberid`),
                    KEY `idx_Member` (`memberid`)
);

drop table librarian;
CREATE TABLE `librarian` (
                       `librarianid` varchar(15) NOT NULL,
                        `librarianname` varchar(255) NOT NULL,
                        `librariansurname` varchar(255) NOT NULL,
                        `librarianemail` varchar(255) NOT NULL,
                        `librarianpassword` varchar(20) NOT NULL,
                        `role` varchar(15) NOT NULL,
                        PRIMARY KEY (`librarianid`),
                        KEY `idx_Librarian` (`librarianid`)
);

drop table issue;
CREATE TABLE `issue` (
                    `issueid` varchar(255) NOT NULL,
                    `memberid` varchar(15) NOT NULL,
                    `mediaid` varchar(15) NOT NULL,
                    `issuedate` date NOT NULL,
                    `periodindays` int NOT NULL,
                    `returndate` date NOT NULL,
                    `fine` int NOT NULL DEFAULT '0',
                    `status` varchar(20) DEFAULT 'pending',
                    `finepaid` tinyint(1) NOT NULL DEFAULT '0',
                    PRIMARY KEY (`issueid`),
                    KEY `memberid` (`memberid`),
                    KEY `mediaid` (`mediaid`),
                    KEY `idx_Issue` (`issueid`),
                    CONSTRAINT `mediaid` FOREIGN KEY (`mediaid`) REFERENCES `media`
(`mediaid`),
                    CONSTRAINT `memberid` FOREIGN KEY (`memberid`) REFERENCES
`member` (`memberid`)
);
```

## Procedures

```
create
  procedure sp_AddBook(IN thebookid varchar(15), IN thebooktitle varchar(250),
                                          IN thebookauthor varchar(100), IN thepublicationyear year,
                                          IN thegenre varchar(50), IN theprice decimal(10, 2),
                                          IN thedescription varchar(250), IN thecover mediumblob,
                                          IN thepagecount int, OUT wassuccessful tinyint(1))
begin
   declare numberofrowsbefore int;
   declare numberofrowsafter int;
   select count(*) into numberofrowsbefore from book;

   insert into media
   values
      (thebookid, thebooktitle, thepublicationyear, 'book');
   insert into book
   (bookid, title, author, publicationyear, genre, price, description, cover, pagecount)
   values
      (thebookid,
       thebooktitle,
       thebookauthor,
       thepublicationyear,
       thegenre,
       theprice,
       thedescription,
       thecover,
       thepagecount);

   select count(*) into numberofrowsafter from book;

   if(numberofrowsafter>numberofrowsbefore)
   then
         set wassuccessful = true;
   else
         set wassuccessful = false;
   end if;
end;


create
   definer = root@localhost procedure sp_AddEbook(IN theebookid varchar(15), IN theebooktitle
varchar(250),
                                                   IN theebookauthor varchar(100), IN thepublicationyear year,
                                                   IN thegenre varchar(50), IN thedescription varchar(250),
                                                   IN thecover mediumblob, IN thedocument mediumblob,
                                                   IN thepagecount int, IN theformat varchar(10),
                                                   OUT wassuccessful tinyint(1))
begin
   declare numberofrowsbefore int;
   declare numberofrowsafter int;

   select count(*) into numberofrowsbefore from ebook;
```

```
  insert into media
  values
  (theebookid, theebooktitle, thepublicationyear, 'ebook');

  insert into ebook
  (ebookid, ebooktitle, author, publicationyear, genre, description, cover, document, pagecount,
format)
  values
     (theebookid,
      theebooktitle,
      theebookauthor,
      thepublicationyear,
      thegenre,
      thedescription,
      thecover,
      thedocument,
      thepagecount,
      theformat);


  select count(*) into numberofrowsafter from ebook;

  if(numberofrowsbefore<numberofrowsafter)
  then
     begin
        set wassuccessful = true;
     end;
  else
     begin
        set wassuccessful = false;
     end;
  end if;
end;

create
  definer = root@localhost procedure sp_AddJournal(IN thejournalid varchar(15), IN
thejournaltitle varchar(250),
                                       IN thejournalauthor varchar(100), IN thepublicationyear year,
                                       IN theareaofstudy varchar(100), IN thecitations int,
                                       IN thevolume int, IN theissue int, IN thedocument mediumblob,
                                       IN thecover mediumblob, OUT wassuccessful tinyint(1))
begin
  declare numberofrowsbefore int;
  declare numberofrowsafter int;
  select count(*) into numberofrowsbefore from journal;

  insert into media
  values
  (thejournalid, thejournaltitle, thepublicationyear, 'journal');
  insert into journal
  (journalid, title, author, publicationyear, areaofstudy, citations, volume, issue, document, cover)
  values
     (thejournalid,
      thejournaltitle,
      thejournalauthor,
      thepublicationyear,
      theareaofstudy,
      thecitations,
      thevolume,
      theissue, thedocument, thecover);

  select count(*) into numberofrowsafter from journal;

  if(numberofrowsbefore<numberofrowsafter)
  then
     begin
        set wassuccessful = true;
     end;
  else
     begin
        set wassuccessful = false;
     end;
  end if;
end;

create
  definer = root@localhost procedure sp_AddLibrarian(IN thelibrarianid varchar(15), IN
thelibrarianname varchar(50),
                                                   IN thelibrariansurname varchar(50),
                                                   IN thelibrarianemail varchar(250),
                                                   IN thelibrarianpassword varchar(50), IN therole varchar(20))
begin
  INSERT INTO librarian
  VALUES (thelibrarianid, thelibrarianname, thelibrariansurname, thelibrarianemail,
thelibrarianpassword, therole);
end;
```

```
create
  definer = root@localhost procedure sp_AddMember(IN thememberid varchar(15))
begin
  update member
  set membershipstatus = 'member'
  where memberid = thememberid;
end;

create
  definer = root@localhost procedure sp_ChangeMemberPassword(IN thememberid
varchar(15), IN thepassword varchar(50))
begin
  update member
     set memberpassword = thepassword
  where memberid = thememberid;
end;

create
  definer = root@localhost procedure sp_CreateIssue(IN themediaid varchar(15), IN
thememberid varchar(15),
                                              IN theperiod int)
begin
  declare theissuedate date;
  declare thereturndate date;
  declare theissueid varchar(15);
  declare numberofrows int;

  select count(*) into numberofrows from issue;
  set theissuedate = curdate();
  set thereturndate = date_add(theissuedate, interval theperiod day);
  set theissueid = concat('iss-', month(theissuedate),'-',numberofrows+1);

  insert into issue
     (issueid, memberid, mediaid, issuedate, periodindays, returndate)
     value
     (theissueid, thememberid, themediaid, theissuedate, theperiod, thereturndate);

  update book
     set status = 'pendingborrow'
  where bookid = themediaid;
end;

create
  definer = root@localhost procedure sp_DeleteBook(IN thebookid varchar(15))
```

```
begin
  delete from book where bookid = thebookid;
end;

create
  definer = root@localhost procedure sp_DeleteEBook(IN theebookid varchar(15))
begin
  delete from ebook where ebookid = theebookid;
end;

create
  definer = root@localhost procedure sp_DeleteJournal(IN thejournalid varchar(15))
begin
  delete from journal where journalid = thejournalid;
end;

create
  definer = root@localhost procedure sp_DeleteLibrarian(IN thelibrarianid varchar(15))
begin
  DELETE FROM librarian WHERE (librarianid = thelibrarianid);
end;

create
  definer = root@localhost procedure sp_DeleteMember(IN thememberid varchar(15))
begin
  update member
  set membershipstatus = 'deleted'
  where memberid = thememberid;
end;

create
  definer = root@localhost procedure sp_IssueMedia(IN theissueid varchar(15))
begin
  declare theissuedate date;
  declare theperiod int;
  declare thereturndate date;
  declare themediaid varchar(15);

  set theissuedate = curdate();
  select periodindays into theperiod from issue where issueid = theissueid;
  set thereturndate = date_add(theissuedate, interval theperiod day);
  select mediaid into themediaid from issue;

  update issue
  set issuedate = theissuedate, returndate = thereturndate, status = 'issued'
  where issueid = theissueid;

  update book
  set status = 'issued'
  where bookid = themediaid;
end;

create
  definer = root@localhost procedure sp_Login(IN userid varchar(15), IN userpassword
varchar(50),
                                        OUT usertype varchar(20))
begin
  if exists(select * from member where memberid = userid and memberpassword =
userpassword and membershipstatus = 'member')
  then
     set usertype = 'member';
  else if exists(select * from librarian where librarianid = userid and librarianpassword =
userpassword)
  then
     select role into usertype from librarian where librarianid = userid and librarianpassword =
userpassword;
  else
  set usertype = 'nonexistent';
  end if;
  end if;
end;

create
  definer = root@localhost procedure sp_MemberIssues(IN thememberid varchar(15))
begin
  select issueid , title  , periodindays , returndate
  from media m
  right join issue i on m.mediaid = i.mediaid where m.mediaid = i.mediaid and i.memberid =
thememberid;
end;

create
  definer = root@localhost procedure sp_MemberSignUp(IN thememberID varchar(20), IN
thememberName varchar(100),
                                              IN thememberSurname varchar(100), IN thememberEmail
varchar(100),
                                              IN thememberPhoneNumber varchar(15))
begin
```

```
   declare randomNumber int;
   set randomNumber = FLOOR(RAND()*(9999-1000+1)+1000);
   insert into member
   (memberid, membername, membersurname, memberemail, memberpassword,
phonenumber)
   values
       (thememberID, thememberName, thememberSurname, thememberEmail, randomNumber,
thememberPhoneNumber);
end;

create
   definer = root@localhost procedure sp_PayFine(IN thisissueID varchar(15))
begin
   update issue
   set finepaid = true
   where issueid = thisissueID ;
end;

create
   definer = root@localhost procedure sp_RequestBook(IN therequestid varchar(15), IN
thebookid varchar(15))
begin
   insert into mediarequest
       (requestid, mediaid)
       values
       (therequestid, thebookid);
end;

create
   definer = root@localhost procedure sp_ReturnMedia(IN theissueid varchar(15), OUT thefine
int)
begin
   declare thereturndate date;
   declare currentdate date;
   declare themediaid varchar(15);
   declare thememberid varchar(15);

   select returndate, mediaid, memberid into thereturndate, themediaid, thememberid from issue
where issueid = theissueid;
   set currentdate = curdate();
   set thefine = 50*datediff(currentdate, thereturndate)+1;

   if  (thefine < 0)
   then
```

```
        set thefine = 0;
    end if;
    update issue
    set fine = thefine, status = 'returned'
    where issueid = theissueid;

    update member
    set noofissues = noofissues-1
    where memberid = thememberid;

    update book
    set status = 'inhouse'
    where bookid = themediaid;

end;

create
    definer = root@localhost procedure sp_SearchBook(IN thesearchcriteria varchar(50), OUT
thebookname varchar(100),
                                                     OUT thebookauthor varchar(250), OUT thepublicationyear
year,
                                                     OUT thegenre varchar(50), OUT thepagecount int,
                                                     OUT thecover mediumblob, OUT thedescription varchar(250))
begin
    select bookname,
        bookauthor,
        publicationyear,
        genre,
        pagecount,
        cover,
        description into thebookname,
            thebookauthor,
            thepublicationyear,
            thegenre,
            thepagecount,
            thecover,
            thedescription from books
    where bookid like concat('%', thesearchcriteria, '%')
       or bookname like concat('%', thesearchcriteria, '%')
       or bookauthor like concat('%', thesearchcriteria, '%')
       or publicationyear like concat('%', thesearchcriteria, '%')
       or genre like concat('%', thesearchcriteria, '%')
       or price like concat('%', thesearchcriteria, '%')
       or description like concat('%', thesearchcriteria, '%')
    or pagecount like concat('%', thesearchcriteria, '%')
    or status like concat('%', thesearchcriteria, '%');
end;

create
  definer = root@localhost procedure sp_UpdateLibrarian(IN thelibrarianid varchar(15), IN
thelibrarianpassword varchar(50))
begin
  UPDATE librarian
  SET librarianpassword = thelibrarianpassword
  WHERE librarianid = thelibrarianid;
end;

create
  definer = root@localhost procedure sp_ViewMedia(IN themediaid varchar(15), IN
themediatype varchar(15))
begin
  if (themediatype = 'book')
     then
     select * from books where bookid = themediaid;

     else  if (themediatype = 'ebook')
     then
     select * from ebooks where ebookid = themediaid;

     else  if (themediatype = 'audiobook')
     then
     select * from audiobooks where audiobookid = themediaid;

     else  if (themediatype = 'journal')
     then
     select * from journals where journalid = themediaid;

     else  if (themediatype = 'magazine')
     then
     select * from magazine where magazineid = themediaid;

     end if;
     end if;
     end if;
     end if;
  end if;
end;
```

```
create
   definer = root@localhost procedure sp_ViewReturn()
begin
   select *
   from pendingreturns;
end;
```