CS 4347.002 Database System

Library Database Management System

Instructor: ***Jalal Omer***

Organized by:

Jeremiah Joseph
Anthony Chen
Priyanka Ganesan
Mit Patel
Sreyleak Le

Date: May 2nd, 2022

Table of Content

1. Introduction

In this system, we are designing a library database. There will be two primary users of the database: readers and staff. These people should have distinctive views of the database and different permissions of what they can access.

Readers are patrons of the library system who should have access to the books and resources available in the library. Generally, readers should check out books, and filter through the catalog. Moreover, information regarding their checked out books and future return dates for the books must be viewable to the reader. Moreover, viewability for the cost of overdue books should also be available to the reader.

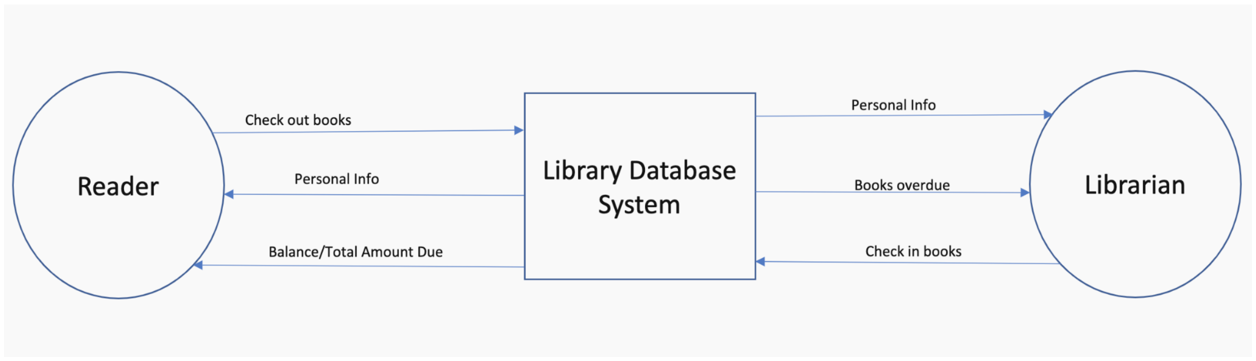
The staff has Librarians which have their own requirements. The Librarians should be able to see what upcoming overdue books for each reader, see what books are overdue for each reader, see how much each reader owes to the library, view their salary information, and view their hours worked.

There are many pieces of information to be contained in the system. The system should have the login id, name, password, and email of every system user. This includes the librarians. For librarians, the system should collect the information of the hours worked and the salary. The login id should be unique to every user. For readers, there should be information on books overdue, and how much money is due. The system should also contain information about the books. Each book should have a publish date, rating, page count, title, author, publisher, genre, copies, and ISBN. The ISBN should be unique to each book.

The following report will first talk about the basic system description and function and nonfunction requirements. We will then explain the conceptual and logical database design. We will then show the normalization and database creation process. We will finally give an overview of the UI and other future work that can be done on the system.

2. System Requirements

Context Diagram:



Interface Requirements:

<u>Librarians:</u>

- The librarians should have a menu where they can log in before getting authorization to view other interface components.
- Librarians when looking at the screen should only be able to see their own information (no other librarians).
- The librarians should have a window where they can see what a specific reader has checked out and if there are any books overdue.
- Librarians should have a menu where they can check in books.

<u>Readers:</u>

- The readers should have a menu where they can log in before getting authorization to view other interface components.
- Readers should have a window where they can see only their own information (no other readers).
- They should have a window where they can view all the books.
- They should have away from their user interface to see if a book is able to be checked out.
- They should have a way from their user interface to check out a book if the book is able to be checked out.
- The reader should have a menu where they can see how many books they have overdue and fees they have due.

Functional Requirements:

Readers:

- The reader should be able to log into their account with their given username and password.
- The reader should be able to check out books from the collection.
- The reader should be able to search the catalog of books
- The reader should be able to filter the catalog based on various criteria(publish date,rating, copies of the book, page count, title, author, publisher, genre, ISBN(primary_key))
- The reader should be able to view a list of their book return dates
- The reader should be able to view a list of costs from overdue books

Librarians:

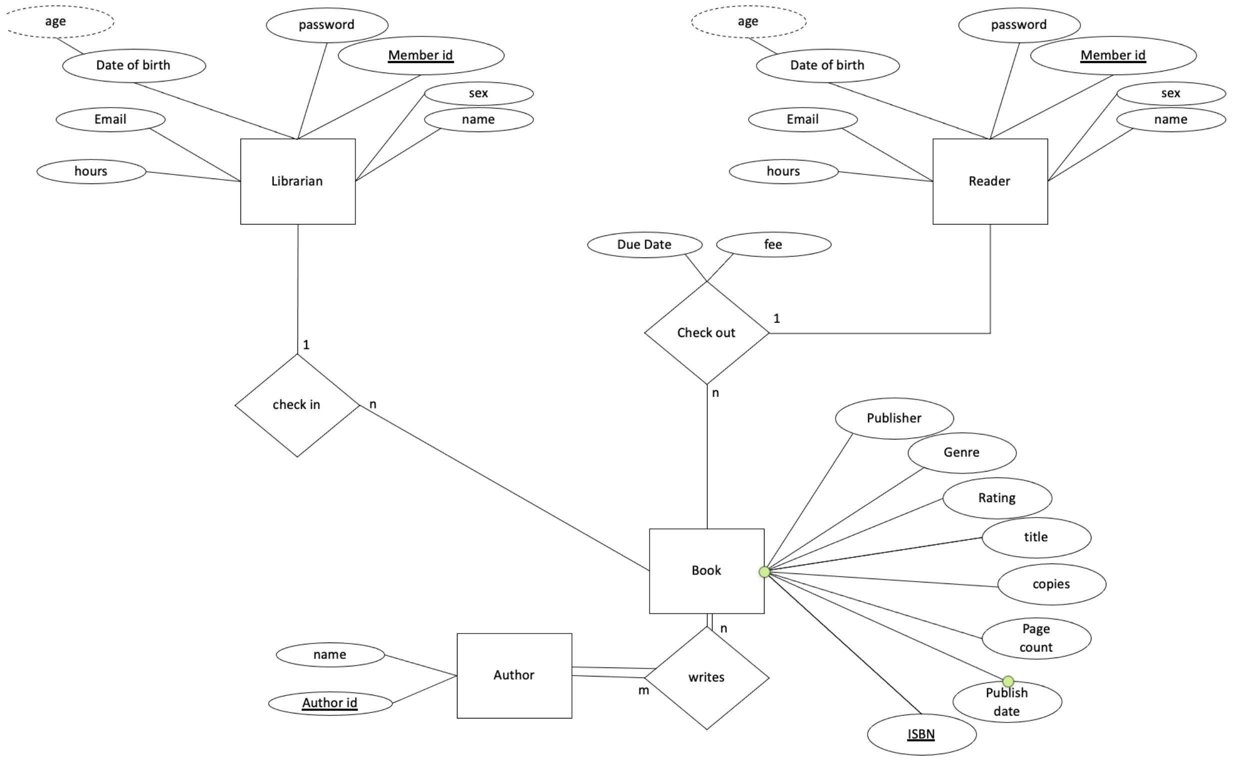
- The librarian should be able to see readers who have overdue books
- The librarian should be able to view hours worked
- The librarian should be able to see readers who have books that are due soon
- The librarian should be able to see the amount of library debt the readers have
- The librarian should be able to view their salary information

Nonfunctional Requirements:

- All users are required to log in to their account
- Searches will take less than 5 seconds.
- The system should be able to hold up to 4,000 books
- The design should be able to service up to 100 readers at one time
- All information is stored redundantly and updated weekly to prevent data loss
- The system should be accessible to multiple platforms
- The database should be able to be integrated with a user interface

3. Conceptual Design of the Database

ER Diagram:



Data Dictionary:

TABLE LIBRARIAN

PK	#	Name	Data Type	Nullable	Description
=>	1	Staff id	VARCHAR(30)		A unique Id for each Librarian
	2	name	VARCHAR(30)	Y	Librarian’s Name
	3	salary	FLOAT(5,2)		How much librarian makes per hour

	4	hours	INT(3)	Y	Hours per week a librarian works
	5	date of birth	DATE	Y	Librarian’s Date of Birth
	6	email	VARCHAR (26)	Y	Librarian’s Email
	7	password	VARCHAR (26)		Librarian’s password
	8	sex	CHAR (1)	Y	Librarian’s sex

TABLE READER

PK	#	Name	Data Type	Nullable	Description
=>	1	Reader_Id	VARCHAR(30)		A unique Id for each Reader
	2	fname	VARCHAR(30)	Y	Reader’s First Name
	3	lname	VARCHAR(30)	Y	Reader’s Last Name
	4	DateOfBirth	DATE	Y	Reader’s Date of Birth
	5	Email	VARCHAR(26)	Y	Reader’s Email
	6	Passwords	VARCHAR(26)		Reader’s Password
	7	Sex	CHAR(1)	Y	Reader’s Gender

TABLE BOOK

PK	#	Name	Data Type	Nullable	Description
=>	1	ISBN	VARCHAR(30)		Unique ISBN ID

	2	Title	VARCHAR(30)	Y	Title of the book
	3	Copies	INT(3)	Y	Number of copies of the book in the database
	4	Genre	VARCHAR(30)	Y	Genre of the book
	5	publisher	VARCHAR(30)	Y	Name of the publishing agency
	6	Page count	INT(3)	Y	Number of pages in the book
	7	rating	CHAR(1)	Y	Rating of the book
	8	Publish date	DATE	Y	Date the book was published

TABLE CHECKOUT

PK	#	Name	Data Type	Nullable	Description
=>	1	Member_id	VARCHAR(30)		A unique Id for each Reader
	2	ISBN	VARCHAR(15)		Unique ISBN ID
	3	Due date	DATE	Y	The day that book should be return
	4	Daily fee	INT	Y	Late fee/day

TABLE AUTHOR

PK	#	Name	Data Type	Nullable	Description
=>	1	Author_ID	VARCHAR(30)		Unique ID for each Author in the System
	2	fname	VARCHAR(20)	Y	Author’s First Name
	3	lname	VARCHAR(20)	Y	Author’s Last Name

TABLE WRITES

PK	#	Name	Data Type	Nullable	Description
=>	1	Author_ID	VARCHAR(30)		Unique ID for each Author in the System
	2	ISBN	VARCHAR(15)	N	Unique ISBN ID (Book)

Business Rules and Integrity Constraints:

Business Rules:
- A librarian’s salary can never be lowered. When updated it can only be increased.
- The library cannot have more than 30 books from the same author
- A reader cannot check out a book that has no more remaining copies.

Integrity Constraints:
- A librarian or reader has to have a unique member id/ username.
- A book should have a unique ISBN
- An author should have a unique author id
- A book that is checked out must have a corresponding ISBN in the book table as well as a corresponding member id from the reader table.
- In the writes table, the author id should have a corresponding author in the author table. Similarly, the ISBN should correspond to an ISBN in the book column
- When a reader is deleted from the system the books checked out by the reader should have the member id go to null on the checked-out table.

- If an author id is deleted the books he/she wrote in the write column will have the author id go to null.
- Any updates to the member ids, the ISBN numbers, or the author ids will update any of the tables that reference them.

4. Logical Database Schema

Database Schema:

Librarian (Staff id, name, salary, hours, date of birth, email, password, sex)

Reader (Reader id, name, date of birth, email, password, sex)

Book (ISBN, publisher, publish date, title, genre, age rating, book type, page count, word count, copies)

Checked Out(Reader id, ISBN, Due Date, Daily fee)

- FK: Reader id ref Reader, ISBN ref Book

Author(Author id, name)

- FK: Author id

Writes(Author id, ISBN**)**

- FK: Author id ref Author, ISBN ref Book

SQL Table Creation Code:

```
-- Code to build the normalized tables of the database

   DROP DATABASE IF EXISTS `Library Management System`;
   CREATE DATABASE `Library Management System`;
   USE `Library Management System`;

   CREATE TABLE Librarian (
                 Staff_ID VARCHAR(30) NOT NULL,
      name VARCHAR(30),
      Salary FLOAT(5,2) NOT NULL,
      Hours INT(3) DEFAULT 0,
      DateOFBirth DATE,
      Email VARCHAR (26),
      Passwords VARCHAR (26) NOT NULL,
      Sex CHAR (1),

      PRIMARY KEY (Staff_ID)
);

CREATE TABLE Reader (
                Reader_ID VARCHAR(30) NOT NULL,
   name VARCHAR(30),
   DateOFBirth DATE,
   Email VARCHAR (26),
   Passwords VARCHAR (26) NOT NULL,
   Sex CHAR (1),

   PRIMARY KEY (Reader_ID)
);

 CREATE TABLE Book (
                ISBN VARCHAR(13) NOT NULL,
   Publisher VARCHAR(100),
   PublisherDate DATE,
   Title VARCHAR (40),
   Genre VARCHAR (40),
   AgeRating INT(3),
   PageCount INT(4),
   Copies INT (2),

   PRIMARY KEY (ISBN)
);

CREATE TABLE CheckedOut (
                Reader_ID VARCHAR(30),
   ISBN VARCHAR(13),
   DueDate DATE,
   DailyFee INT(2),

   PRIMARY KEY (READER_ID, ISBN),

   FOREIGN KEY (Reader_ID) REFERENCES Reader(Reader_ID)
                        ON UPDATE Cascade,
   FOREIGN KEY (ISBN) REFERENCES Book(ISBN)
                        ON UPDATE Cascade

 );
```

```
CREATE TABLE Author (
  Author_ID VARCHAR(11) NOT NULL,
  name VARCHAR(20),

  PRIMARY KEY (Author_ID)
  );

    CREATE TABLE Writes (
            Author_ID VARCHAR(11) ,
            ISBN varchar(13) ,

  PRIMARY KEY (Author_ID, ISBN),
  FOREIGN KEY (Author_ID) REFERENCES Author(Author_ID)
                  ON DELETE Cascade
     ON UPDATE Cascade,
            FOREIGN KEY (ISBN) REFERENCES Book(ISBN)
                  ON DELETE Cascade
     ON UPDATE Cascade
  );
```

Expected Operation and Data Volumes:

We're expecting several SELECTS, INSERTS, and DELETES for data retrieval as well as book check in and check out respectively.

As for the volume we expect SELECTS to be the bulk of the operations on the database as they are used for retrieving: user information, user login, staff login, user checked out books, user overdue books, user overdue books cost, staff information, etc. While INSERTS and DELETES, will be roughly equal as they are used for checking out books and checking in respectively and much less comparatively to the SELECTS. Overall we expect around 500 SELECTS and 150 INSERTS and DELETES, considering the current user count of our system.

5. Functional Dependencies/ Normalization

Normalization:

<u>Original:</u>

Librarian (<u>Staff id</u>, name, salary, hours, date of birth, email, password, sex)

- FD: Staff_Id -> {name, salary, hours, date of birth, email, password, sex}

Reader (<u>Reader id</u>, name, date of birth, email, password, sex)

- FD: Reader_Id -> {name, salary, hours, date of birth, email, password, sex}

Book (<u>ISBN</u>, publisher, publish date, title, genre, age rating, book type, page count, word count, copies)

- FD: ISBN -> {publisher, publish date, title, genre, age, rating, page count, word count, copies}

Checked Out(<u>Reader id, ISBN</u>, Due Date, Daily fee)

- FD: {Reader_id, ISBN} -> {Due Date, Daily fee}

Author(<u>Author id</u>, name)

- FD: Author_id -> name

Writes(<u>Author id, ISBN</u>**)**

<u>1NF:</u>

Librarian (<u>Staff id</u>, name, salary, hours, date of birth, email, password, sex)

- Name can be divisible. Therefore we can divide this attribute into f name and l name
- New table is:
    - **Librarian** (<u>Staff id</u>, fname, lname, salary, hours, date of birth, email, password, sex)

Reader (<u>Reader id</u>, name, date of birth, email, password, sex)

- Name can be divisible. Therefore we can divide this attribute into f name and l name
- New table is:

    - **Reader** (Reader id, fname, lname, date of birth, email, password, sex)

Book (ISBN, publisher, publish date, title, genre, age rating, page count, word count, copies)

- All values are atomic and singular so is 1NF

Checked Out(Reader id, ISBN, Due Date, Daily fee)

- All values are atomic and singular so is 1NF

Author(Author id, name)

- Name can be divisible. Therefore we can divide this attribute into f name and l name
- New table is:
    - **Author**(Author id, fName, lName)

Writes(Author id, ISBN**)**

- All values are atomic and singular so is 1NF

2NF:

Librarian (Staff id, fname, lname, salary, hours, date of birth, email, password, sex)

- Table is 2NF as there are no partial dependencies

Reader (Reader id, fname, lname, date of birth, email, password, sex)

- Table is 2NF as there are no partial dependencies

Book (ISBN, publisher, publish date, title, genre, age rating, page count, word count, copies)

- Table is 2NF as there are no partial dependencies

Checked Out(Reader id, ISBN, Due Date, Daily fee)

- Table is 2NF as there are no partial dependencies

Author(Author id, fName, lName)

- Table is 2NF as there are no partial dependencies

Writes(Author id, ISBN**)**

- Table is 2NF as there are no partial dependencies

3NF:

Librarian (Staff id, fname, lname, salary, hours, date of birth, email, password, sex)
- Table is 3NF as there are no transitive dependencies
- FD: Staff_id -> {fname, lname, salary, hours, date of birth, email, password, sex}

Reader (Reader id, fname, lname, date of birth, email, password, sex)
- Table is 3NF as there are no transitive dependencies
- FD: Reader_id -> {fname, lname, salary, hours, date of birth, email, password, sex}

Book (ISBN, publisher, publish date, title, genre, age rating, page count, word count, copies)
- Table is 3NF as there are no transitive dependencies
- FD: ISBN -> {publisher, publish date, title, genre, age rating, page count, word count, copies}

Checked Out(Reader id, ISBN, Due Date, Daily fee)
- Table is 3NF as there are no transitive dependencies
- FD: {Reader_id, ISBN} -> {Due Date, Daily fee}

Author(Author id, fName, lName)
- Table is 3NF as there are no transitive dependencies
- FD: Author_id -> {fName, lName}

Writes(Author id, ISBN**)**
- Table is 3NF as there are no transitive dependencies

SQL Code After Normalization:

```
-- Code to build the normalized tables of the database

DROP DATABASE IF EXISTS `Library Management System`;
CREATE DATABASE `Library Management System`;
USE `Library Management System`;

CREATE TABLE Librarian (
	Staff_ID INT(11) NOT NULL,
        fname VARCHAR(30),
        lname VARCHAR(30),
        Salary FLOAT(5,2) NOT NULL,
        Hours INT(3) DEFAULT 0,
        DateOFBirth DATE,
        Email VARCHAR (26),
        Passwords VARCHAR (26),
        Sex CHAR (1),

        PRIMARY KEY (Staff_ID)
   );

CREATE TABLE Reader (
        Reader_ID INT(11) NOT NULL,
        fname VARCHAR(30),
        lname VARCHAR(30),
        DateOFBirth DATE,
        Email VARCHAR (26),
        Passwords VARCHAR (26),
        Sex CHAR (1),

        PRIMARY KEY (Reader_ID)
   );

CREATE TABLE Book (
        ISBN VARCHAR(13) NOT NULL,
        Publisher VARCHAR(100),
        PublisherDate DATE,
        Title VARCHAR (40),
        Genre VARCHAR (40),
        AgeRating INT(3),
        PageCount INT(4),
        Copies INT (2),

        PRIMARY KEY (ISBN)
   );

CREATE TABLE CheckedOut (
        Reader_ID INT(11),
        ISBN VARCHAR(13),
        DueDate DATE,
        DailyFee INT(2),

        PRIMARY KEY (READER_ID, ISBN),

        FOREIGN KEY (Reader_ID) REFERENCES Reader(Reader_ID)
                ON UPDATE Cascade,
        FOREIGN KEY (ISBN) REFERENCES Book(ISBN)
                ON UPDATE Cascade

    );

CREATE TABLE Author (
        Author_ID INT(11) NOT NULL,
        fname VARCHAR(20),
        lname VARCHAR(20),

        PRIMARY KEY (Author_ID)
      );

CREATE TABLE Writes (
        Author_ID INT(11) ,
        ISBN varchar(13) ,

        PRIMARY KEY (Author_ID, ISBN),
        FOREIGN KEY (Author_ID) REFERENCES Author(Author_ID)
        ON DELETE Cascade
        ON UPDATE Cascade,
        FOREIGN KEY (ISBN) REFERENCES Book(ISBN)
        ON DELETE Cascade
        ON UPDATE Cascade
      );

CREATE TABLE Adds (
        Staff_ID INT(11) ,
        ISBN VARCHAR(13) ,

        PRIMARY KEY (Staff_ID, ISBN),
        FOREIGN KEY (Staff_ID) REFERENCES Librarian(Staff_ID)
                ON DELETE Cascade
        ON UPDATE Cascade,
                FOREIGN KEY (ISBN) REFERENCES Book(ISBN)
                ON DELETE Cascade
        ON UPDATE Cascade
      );
```

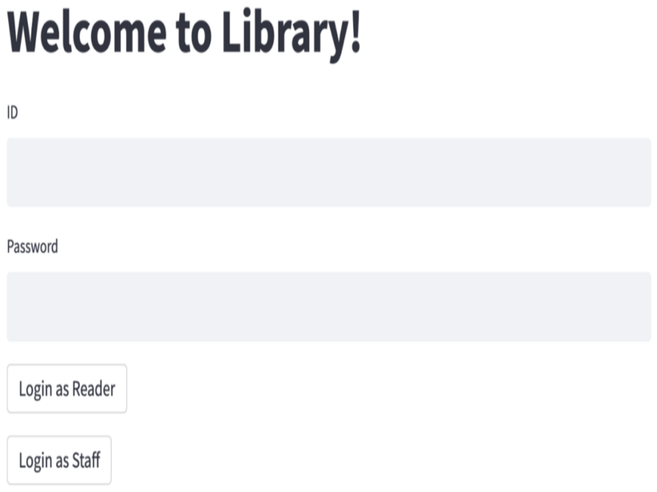
6. The Database System

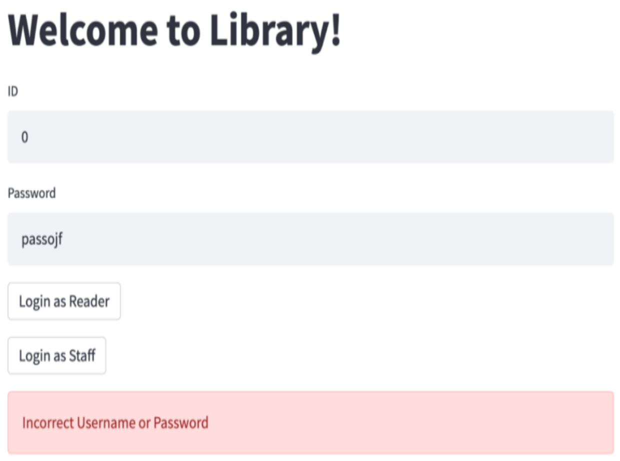
How to Install/Run System:
In order to run the system the first step is installing Python3 and MySQL. From there there are other libraries in python to install. This can be done using pip. You need to use pip to install pandas, numbpy, mysql.connector, and streamlit. The next step is to run the database. Next you need to go to the streamlit folder on terminal. Finally you need to use the command streamlit run main.py on terminal. This should start the project.

Step by Step How to Use System:

<u>Login:</u>

The first page that loads once the project opens is the login screen. From here the user can click whether to login as a reader or staff. Depending on the response the system will authenticate based on the corresponding SQL table. If the username/password is correct the system will go in two separate ways depending on if the user is a reader or staff. Below are Screenshots of Login Process




Welcome to Library!

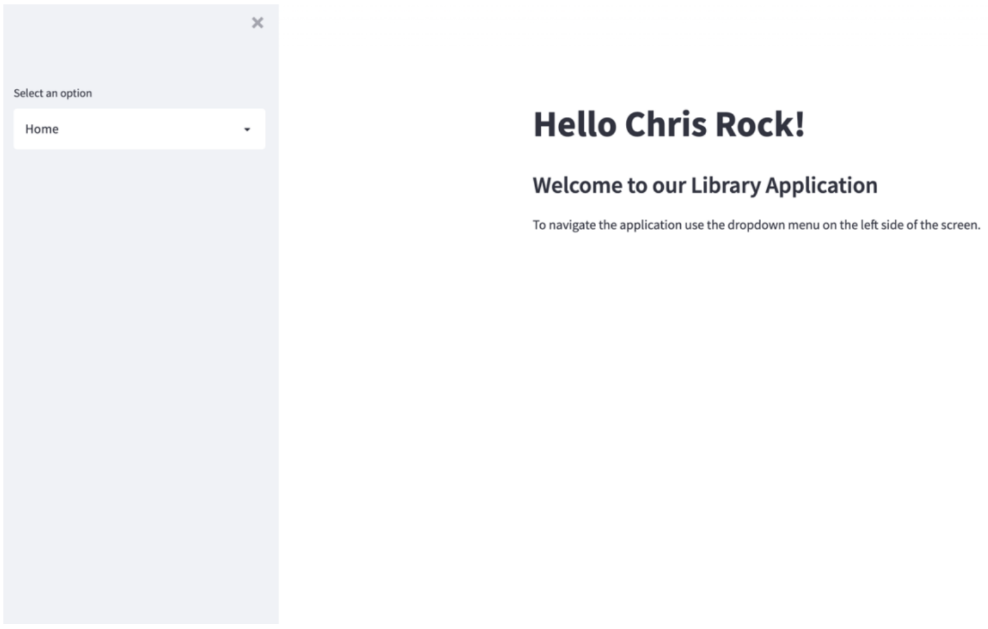
ID: 0

Password: passojf

Incorrect Username or Password

Reader View:

The first page when successfully login in as a reader is the home page. There is a menu on the left side of the screen that can be used to navigate through the various other pages. The name is gotten from the login info.

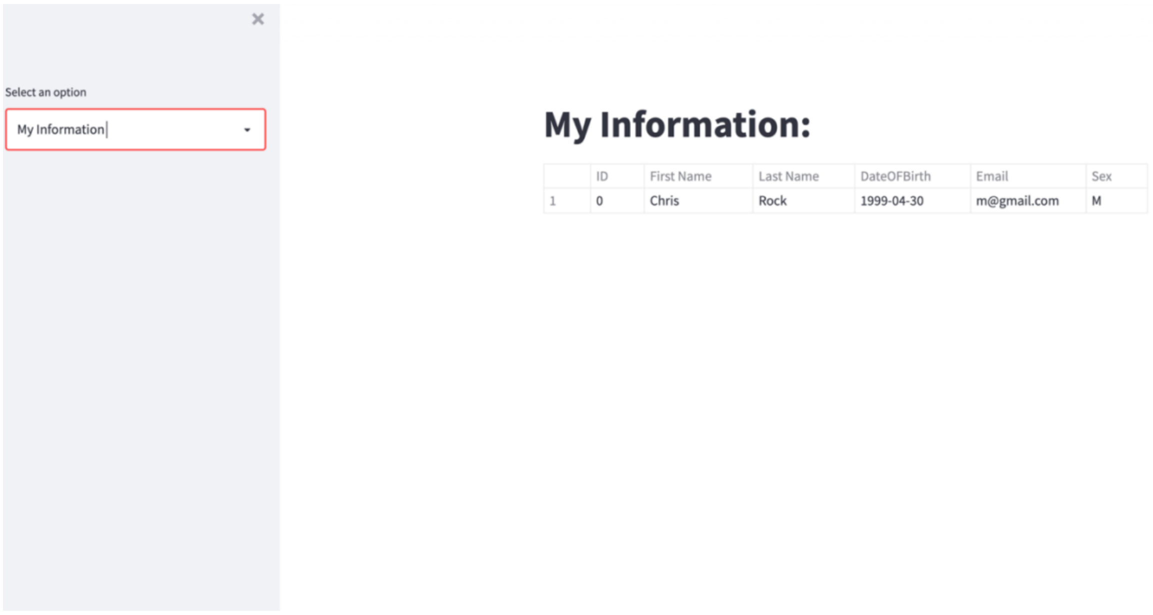


Hello Chris Rock!

Welcome to our Library Application

To navigate the application use the dropdown menu on the left side of the screen.

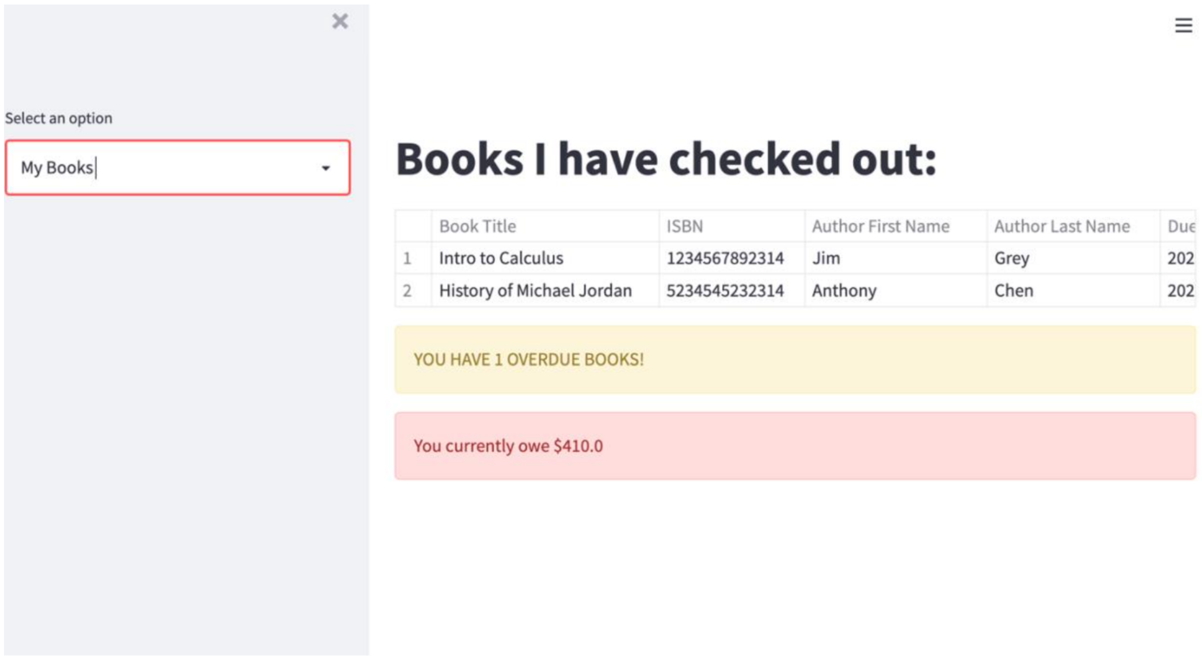
The next page that is available is my information. It delivers information in a table that corresponds to the user that is logged in.



My Information:

	ID	First Name	Last Name	DateOFBirth	Email	Sex
1	0	Chris	Rock	1999-04-30	m@gmail.com	M

The next page that is available through the navigation is my books. From here a user can simply read the books they have checked out, how many books they have overdue, and how much they owe the library.



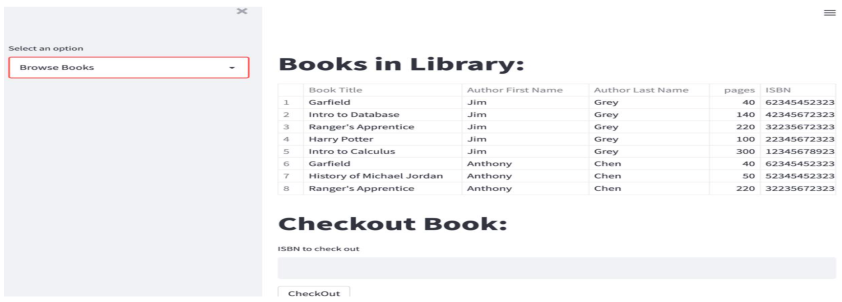
Books I have checked out:

	Book Title	ISBN	Author First Name	Author Last Name	Due
1	Intro to Calculus	1234567892314	Jim	Grey	202
2	History of Michael Jordan	5234545232314	Anthony	Chen	202

YOU HAVE 1 OVERDUE BOOKS!

You currently owe $410.0

The browse books page shows allows the user to view all the books. To check out a book you have to write the ISBN number to checkout on the input field. Once clicked if the book is available to be checkout it will be added to the books. If not it will give a message that the book can't be checked out.

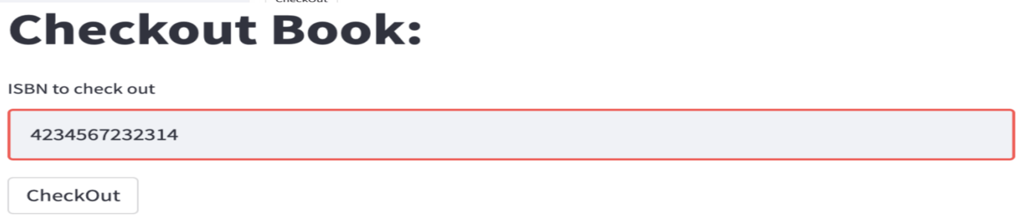


Books in Library:

	Book Title	Author First Name	Author Last Name	pages	ISBN
1	Garfield	Jim	Grey	40	62345452323
2	Intro to Database	Jim	Grey	140	42345672323
3	Ranger's Apprentice	Jim	Grey	220	32235672323
4	Harry Potter	Jim	Grey	100	22345672323
5	Intro to Calculus	Jim	Grey	300	12345678923
6	Garfield	Anthony	Chen	40	62345452323
7	History of Michael Jordan	Anthony	Chen	50	52345452323
8	Ranger's Apprentice	Anthony	Chen	220	32235672323

Checkout Book:

ISBN to check out

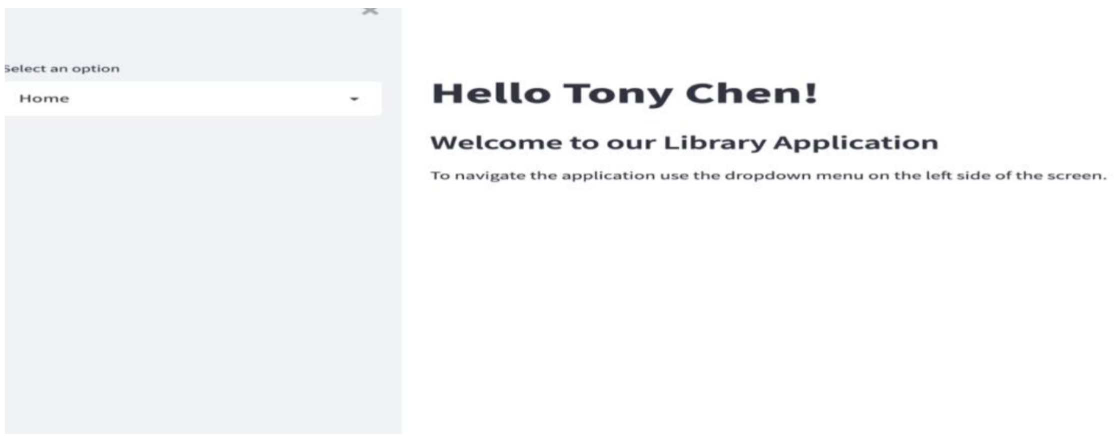


Checkout Book:

ISBN to check out

4234567232314

Staff View:
The first page when successfully login in as a staff is the home page. There is a menu on the left side of the screen that can be used to navigate through the various other pages. The name is gotten from the login info.

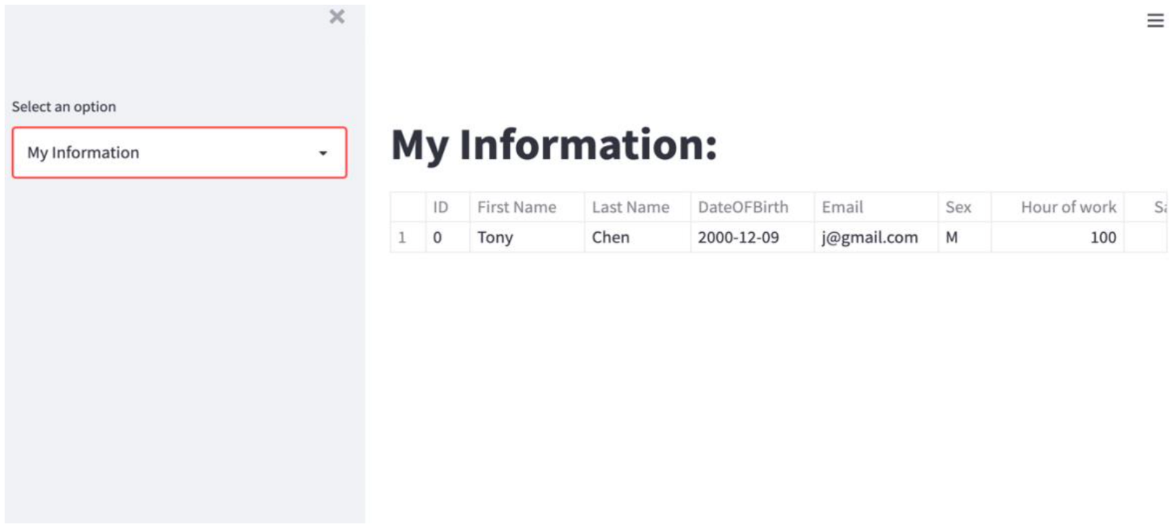


Hello Tony Chen!

Welcome to our Library Application

To navigate the application use the dropdown menu on the left side of the screen.

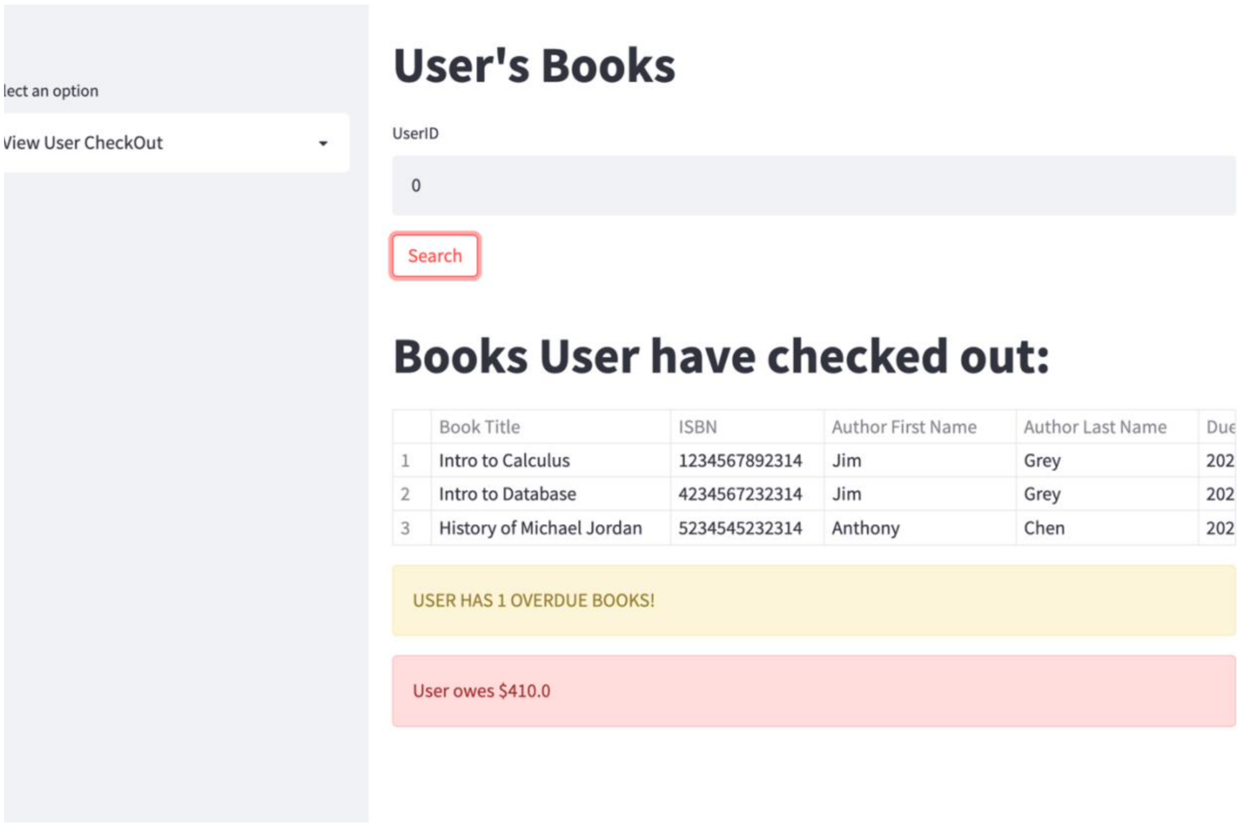
The next page that is available is my information. It delivers information in a table that corresponds to the user that is logged in.



My Information:

	ID	First Name	Last Name	DateOFBirth	Email	Sex	Hour of work	S
1	0	Tony	Chen	2000-12-09	j@gmail.com	M	100	

The next page that is available is view user checkout. The user has to enter a reader ID. From there the librarian can see what books the user has checked out, whether there are any late books, and how much money the user owes.


lect an option

View User CheckOut

User's Books

UserID

0

Search

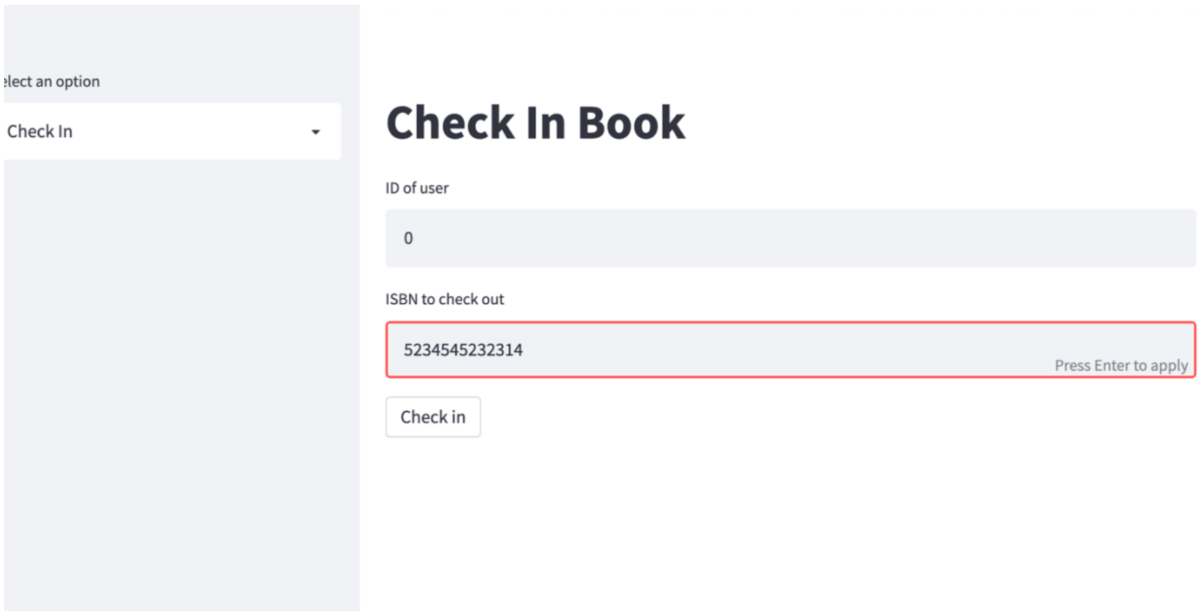
Books User have checked out:

	Book Title	ISBN	Author First Name	Author Last Name	Due
1	Intro to Calculus	1234567892314	Jim	Grey	202
2	Intro to Database	4234567232314	Jim	Grey	202
3	History of Michael Jordan	5234545232314	Anthony	Chen	202

USER HAS 1 OVERDUE BOOKS!

User owes $410.0

The next page that is available is check in book. The librarian has to enter a user id and an ISBN and click check in. If that book is checked out we will then remove that book from the books checked out by that specific reader.


lect an option

Check In

Check In Book

ID of user

0

ISBN to check out

5234545232314

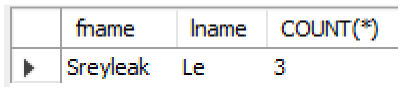
Press Enter to apply

Check in

7. Additional Queries and Views

List of the reader that who has check out two or more book:

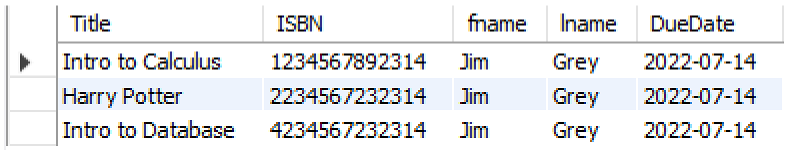
```
SELECT fname, lname, COUNT(*)
FROM Reader, CheckedOut
WHERE Reader.reader_ID = CheckedOut.reader_ID
GROUP BY fname, lname
HAVING COUNT(*) > 2;
```



fname	lname	COUNT(*)
Sreyleak	Le	3

List the overdue book:

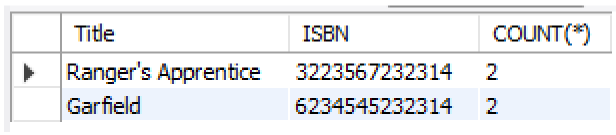
```
SELECT Book.Title, Book.ISBN, Author.fname, Author.lname, CheckedOut.DueDate
FROM Reader, Book, CheckedOut, Writes, Author
WHERE Book.ISBN = CheckedOut.ISBN AND Author.Author_ID = Writes.Author_ID
AND Reader.Reader_ID = CheckedOut.Reader_ID AND  Book.ISBN = Writes.ISBN and
Reader.Reader_ID;
```



Title	ISBN	fname	lname	DueDate
Intro to Calculus	1234567892314	Jim	Grey	2022-07-14
Harry Potter	2234567232314	Jim	Grey	2022-07-14
Intro to Database	4234567232314	Jim	Grey	2022-07-14

List titles, ISBN, and name of book with multiple authors:

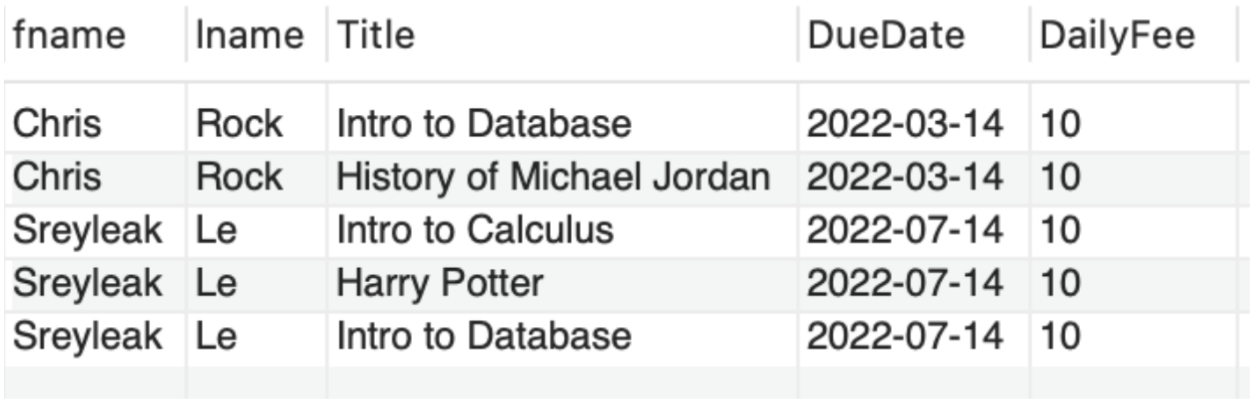
```
SELECT Title, book.ISBN, COUNT(*)
FROM book, writes
WHERE book.ISBN = writes.ISBN
GROUP BY book.ISBN
HAVING COUNT(*) > 1;
```



Title	ISBN	COUNT(*)
Ranger's Apprentice	3223567232314	2
Garfield	6234545232314	2

view for staff to see basic reader information

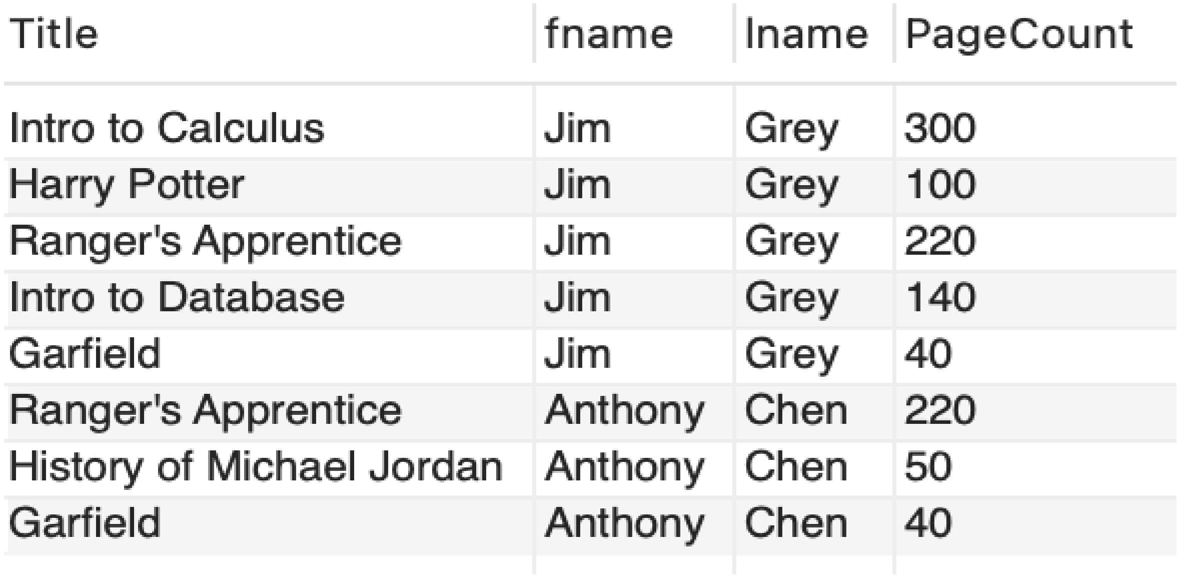
```
CREATE VIEW ReaderInfoView
AS      SELECT fname, lname, Title, DueDate, DailyFee
        FROM Book, Reader, checkedOut
        WHERE Book.ISBN = checkedOut.ISBN and Reader.Reader_ID =
checkedOut.Reader_ID;
```



fname	lname	Title	DueDate	DailyFee
Chris	Rock	Intro to Database	2022-03-14	10
Chris	Rock	History of Michael Jordan	2022-03-14	10
Sreyleak	Le	Intro to Calculus	2022-07-14	10
Sreyleak	Le	Harry Potter	2022-07-14	10
Sreyleak	Le	Intro to Database	2022-07-14	10

-- view for stripped view basic info of books

```
CREATE VIEW booksView
AS SELECT Title, fname, lname, PageCount
        FROM Book, Author, Writes
        WHERE Book.ISBN = Writes.ISBN and Writes.Author_ID = Author.Author_ID;
```



Title	fname	lname	PageCount
Intro to Calculus	Jim	Grey	300
Harry Potter	Jim	Grey	100
Ranger's Apprentice	Jim	Grey	220
Intro to Database	Jim	Grey	140
Garfield	Jim	Grey	40
Ranger's Apprentice	Anthony	Chen	220
History of Michael Jordan	Anthony	Chen	50
Garfield	Anthony	Chen	40

8. User Application Interface

Our user interface is set up using python. The users are first directed to login and on the login menu they choose one of 2 login paths to login as a reader or librarian. The subsequent homepage and the Navigation bar to the left differ based on the path the user chooses for login. The login type is determined in the code using a boolean state variable which identifies whether to run the librarian or reader homepage.

Reader:

Homepage: A welcome text is generated with the users first can last names queried using the ID given during login.

My Information: This screen shows a table of the users information this is queried using the ID given during login.

View My Books: In this screen, we show a table of book information for the booked currently checked out by the user. We also calculate the number of books past the due date as well as the balance owed on these books

Show list of books: This page shows a list of all books in the library database in a tabular format to the reader.

Checkout book: This page takes input from the reader where they enter an ISBN number. This ISBN number is used to verify that a copy of this book exists and the ISBN number is used to add the book to the readers checked out books table.

Librarian:

Homepage: Query database with user id and return fname and name as a full name

Information screen: Query database with user id and return all rows except password

User's books: Query database with user id and return the joined table of checked out books according to user id, author according to written, and isbn according to books, these tables all joined will return all books the user has checked out with the relevant book information as well as who wrote it.

Check-in books: Give the system an ISBN and a User Id and it will delete the row(if it exists) in the CheckedOut table, thereby checking the book back into the system.

9. Conclusion and Possible Future Work

Overall the System works fairly well although the base design is not perfect it is very workable and expandable for further systems which in todays software world is usually a requirement. This project has had a couple of issues during its creation mainly scheduling meetings for all of us to collaborate on what is being done as well as some minor misunderstanding about the intended scope of this project. However, despite all that the system works well and is quite functional. In the future, work may be done to increase security on passwords within system, possibly adding AES 128bit encryption, and allowing the staff to add books to redistribute some responsibilities away from the database administrators. Perhaps when the Library catalogue grows large enough a seach function could also be added to enhance the library experience as well.

10. References

Pages:
https://gist.github.com/Cawinchan/2f9c4196f89746e58c9416a5bd0a7dfc#file-page_helper_function-py

Database: https://dev.mysql.com/doc/

Front End: https://docs.streamlit.io/

11. Appendix

https://github.com/Lrogan/DatabaseProject