Introduction

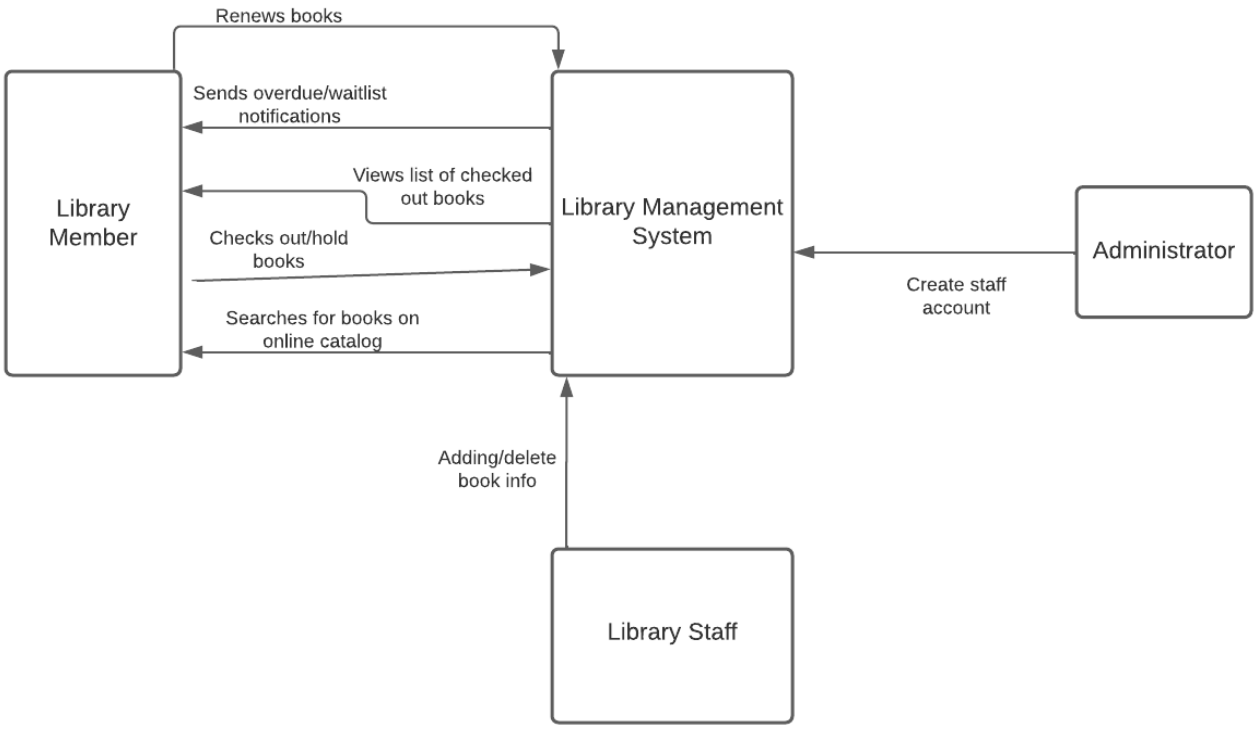
The goal of this project is to gain experience with designing and building a database that interacts with an application. The system should be able to execute different SQL queries and retrieve information from the database and present it to the user. There were different phases for this project from brainstorming system requirements and creating ER diagrams, developing a database schema and creating various SQL query statements, identifying functional dependencies and normalizing the schema, to finally putting everything together and creating an application interface to use the system.

This report will be divided into sections that follow these phases to provide a clear understanding of the project. Additionally, there will be an appendix at the end to download the files relating to our project and view the source code.

System Requirements

The system we chose is a library management system that involves interactions between the library, library staff, and library members. Members can check out books, look up books in the catalog, and place holds on books. Staff members can maintain the data of the status and availability of books. The database will have data on the availability of books, who checked out what, and if anything is overdue.

The context diagram for the system is shown here:



Our interface requirements include:

1. It must be possible for the notification to interface with library member
2. It must be possible for library members and staff to interface with their accounts
3. It must be possible for books being checked out/ reserved in the database to interface with the application that the library staff or members may use.

The functional requirements are as follows:

Add/Remove/Edit Book
Adding, deleting, or modifying a book's information in the system

Check-out Book
Borrowing a book from the library

Check-in Book
Returning a book that has been borrowed back to the library

Search Catalog
Search for books via their title, author name, or ISBN number

Renewing Book
Renew a checked out book, extending the due date

Hold Book
Place a book on hold, can have any number of people waiting on hold list

Create library member
Adding a person's information to the library to issue them a card

Notifications
Sending overdue or waitlist notices to library members when needed

Our non-functional requirements include:

Availability and Accessibility Requirement
The database system and library management application will be available to users at any hour of the day, all year long. The system will be accessible to any standard user operating within the general parameters of application interface usage.

Accuracy Requirement
The database system and library management application will update quickly with respect to real time updates in order to avoid concurrency issues. The accuracy of the system will be maintained with respect to real world data.

Reliability Requirement
The database system and library management application will be completely reliable. This is necessary so that losses or damages are avoided. For example, we will know exactly who has which books at all times.

Usability Requirement
The library management application will reflect normal and standard website operating procedures such that real world and regular users will easily navigate the library management application. The application will be capable of interfacing with regular operating systems and browsers.

Performance and Response Time Requirement
The library management application will update quickly enough for changes in the real world system to be reflected in the database and application such that users will have an acceptable experience with using the application. These updates to the database and application will take less than one minute to process and become reflected in the application and user experience. The general performance requirement is such that the user and developer experience of the database and database application is one of practical utility and efficient usage.

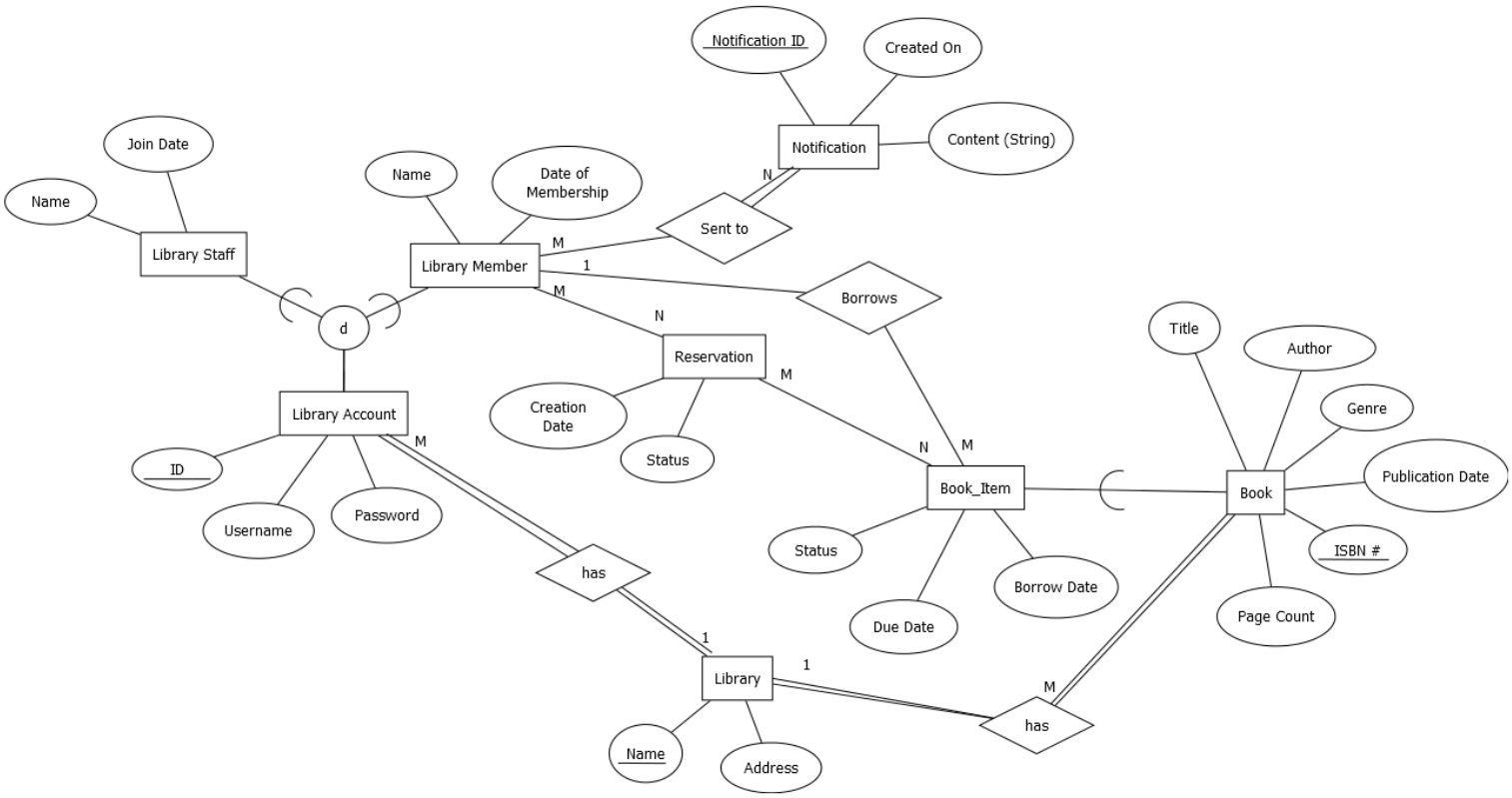
Security and Privacy Requirement
We guarantee that user information will be completely secure from outside attack and investigation. Our users' data, privacy, and interactions with our database and application will be completely private information. For example, the database operators will have available to them only the necessary information about which individual has which book. An individual’s information will be completely secured.

Maintainability Requirement

Changes, such as new users or password updates are verified regularly with respect to real time and real world operations. Users are regularly informed about information pertaining to their interactions with the library, database and application, such as overdue books, reservation status, and availability queries.

Conceptual Design of the Database

The ER model for our database is shown below:



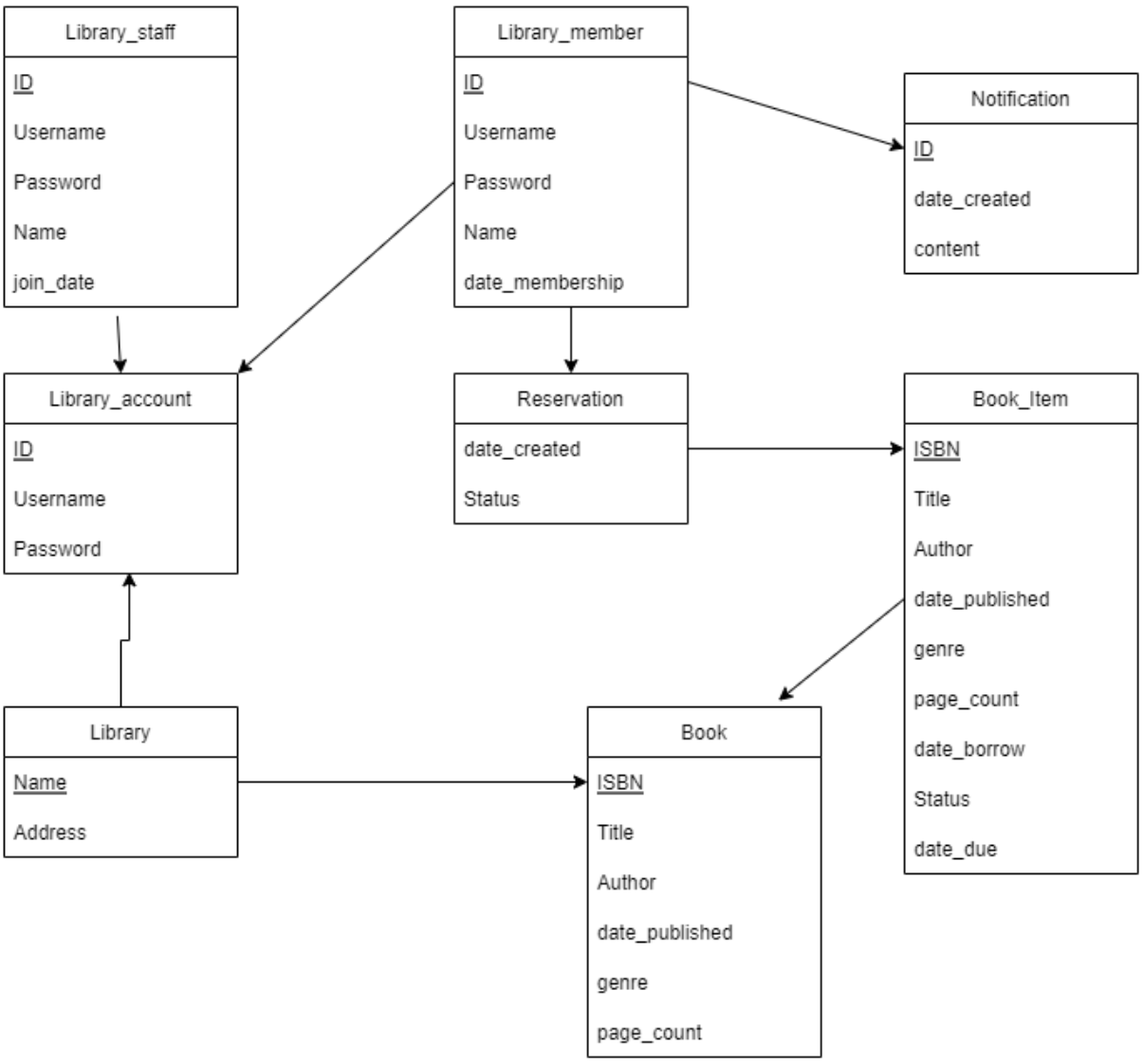
For this model, we have the following business rules:

1. If a Library Member has a book checked out, their library account cannot be deleted until all of their checked-out books are returned. (reject/ ignore)

2. If a Library Member has a book checked out, their account can be updated without a change to the status of their checked-out book (reject/ ignore)
3. If a library member has a reservation, and they delete their account, then the reservation is also deleted(CASCADE)
4. If a library member has a reservation, their account can be updated without a change to their reservation (reject/ ignore)
5. If a Library Member has a Reservation and their account is deleted, then the corresponding Reservation is Cascaded
6. If a Library Member has a Reservation and their account is updated, then the corresponding Reservation is unchanged (reject/ ignore).

Logical Database Schema

Database schema:



To construct the schema, we used the following SQL commands:

```
CREATE TABLE Book (
ISBN CHAR(13) NOT NULL,
Title VARCHAR(256),
Author VARCHAR(256),
Date_Published VARCHAR(256),
Genre VARCHAR(256),
Pages INT,
PRIMARY KEY (ISBN)
);

CREATE TABLE Book_Item (
   ISBN CHAR(13) NOT NULL,
   Title VARCHAR(256),
   Author VARCHAR(256),
   Date_Published VARCHAR(256),
   Genre VARCHAR(256),
   Pages INT,
   Date_Borrow VARCHAR(256),
   Borrow_Status boolean,
   Return_Date VARCHAR(256),
   FOREIGN KEY (ISBN) REFERENCES BOOK(ISBN)
);

CREATE TABLE Library (
   Name VARCHAR(256) NOT NULL,
   Address VARCHAR(256),
   PRIMARY KEY (Name)
);

CREATE TABLE Library_Account (
   ID VARCHAR(256) NOT NULL,
   Username VARCHAR(256) Not Null,
   Password VARCHAR(256),
   PRIMARY KEY (ID)
);

CREATE TABLE Library_Staff (
   ID VARCHAR(256) NOT NULL,
   Username VARCHAR(256) Not Null,
   Password VARCHAR(256),
```

```
    Name VARCHAR(256),
    Join_Date VARCHAR(256),
    FOREIGN KEY (ID) REFERENCES Library_Account (ID)
);
CREATE TABLE Library_Member (
    ID VARCHAR(256) NOT NULL,
    Username VARCHAR(256) Not Null,
    Password VARCHAR(256),
    Name VARCHAR(256),
    Join_Date VARCHAR(256),
    FOREIGN KEY (ID) REFERENCES Library_Account (ID)
);

CREATE TABLE Notification(
    ID VARCHAR(256) NOT NULL,
    Date_Created VARCHAR(256),
    Content VARCHAR(256),
    PRIMARY KEY (ID)
);

CREATE TABLE Reservation(
    Date_Created VARCHAR(256),
    Reservation_Status VARCHAR(256)
);
```

The expected database operations include:

- Inserting, updating, or deleting books to the library
- Creating, updating, or deleting a library member account
- Updating the status of a book when checked out, put on hold, or returned
- Searching for books based on their title, author name, genre…

For this system, the data volume is based on the number of books in the library as well as library users including members and staff, so there could be hundreds of thousands of records.

Functional Dependencies & Database Normalization

The functional dependencies that exist in our schema are:

Library_Account
{ID} → {Username}
{Username}→{Password}

Library_Staff
{ID} →{Username}

{ID}→{Name}
{ID}→{Join_Date}
{Username}→{Password}

Library_Member
{ID} →{Username}
{ID}→{Name}
{ID}→{Join_Date}
{Username}→{Password}

Notification
{ID} →{Date Created}
{ID} →{Content}

Reservation
{ID} →{Date Created}
{ID} →{Status}

Book_Item
{ISBN} → {Title}
{ISBN} → {Author}
{ISBN} → {Date_Published}
{ISBN} → {Genre}
{ISBN} → {Page_Count}
{ISBN} → {Date_Borrow}
{ISBN} → {Status}
{ISBN} → {Date_due}

Book
{ISBN} → {Title}
{ISBN} → {Author}
{ISBN} → {Date_Published}
{ISBN} → {Genre}
{ISBN} → {Page_Count}

Library
{Name} → {Address}

For the normalization process:

1NF
Remove redundant attributes from Book_Item and just have a foreign key (ISBN) that points to ISBN in Book

Book_Item
ISBN, date_borrow, borrow_status, return_date

The genre attribute in book can hold multiple values so separate into two tables:

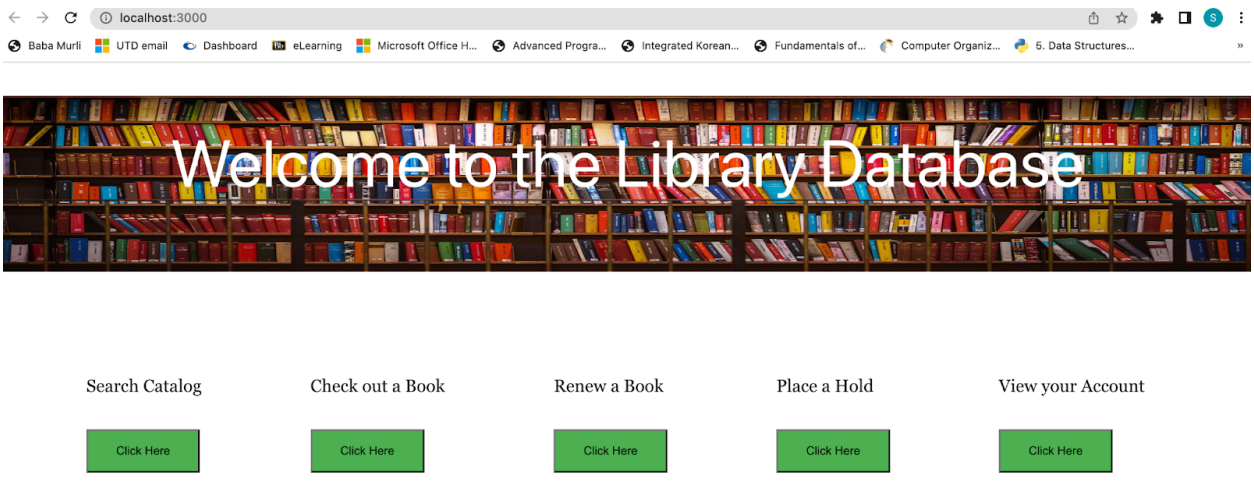
Book
ISBN, Title, Author, date_published, page_count

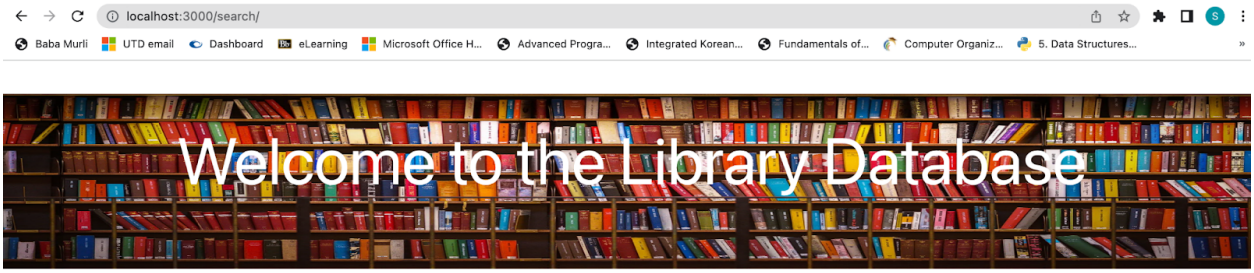
Book_genre
ISBN, genre

The Database System

Our system is built using mysql, nodejs, and reactjs. The whole application is named "Library System" and there are two folders within named "frontend" and "api". Both of these can be run with the command "npm start" on the terminal. A window will open in the browser displaying the user interface which will have five buttons with five functions. The user can click through these buttons to access the library database. The database is currently on a local system. When the user clicks on one of the five options, a request is sent to the api which processes the request and queries the database to retrieve the information which is then sent back to the user.

A few screenshots of the application are below:





Please enter your member ID:

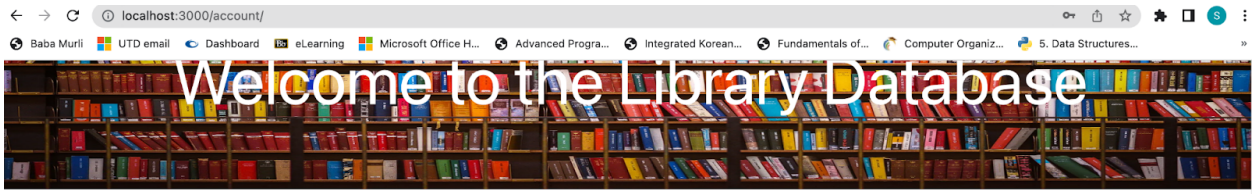
123 Submit

Search for Books:

sisters grimm Submit

Books:

Sisters Grimm
Michael Buckley
Childrens Fantasy
Check Out



Please enter your member ID:

123

Please enter your password:

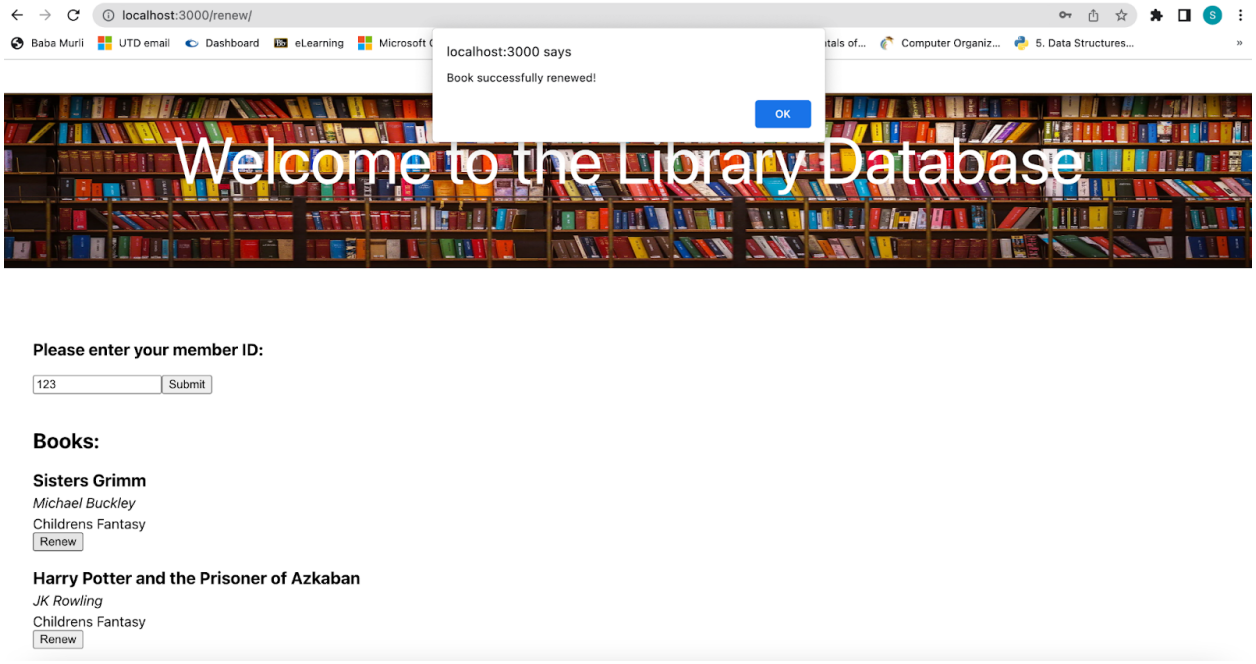
••••••••

Login

Your Checked Out Books:

Sisters Grimm
Michael Buckley
Childrens Fantasy

Harry Potter and the Prisoner of Azkaban
JK Rowling
Childrens Fantasy



Suggestions on Database Tuning

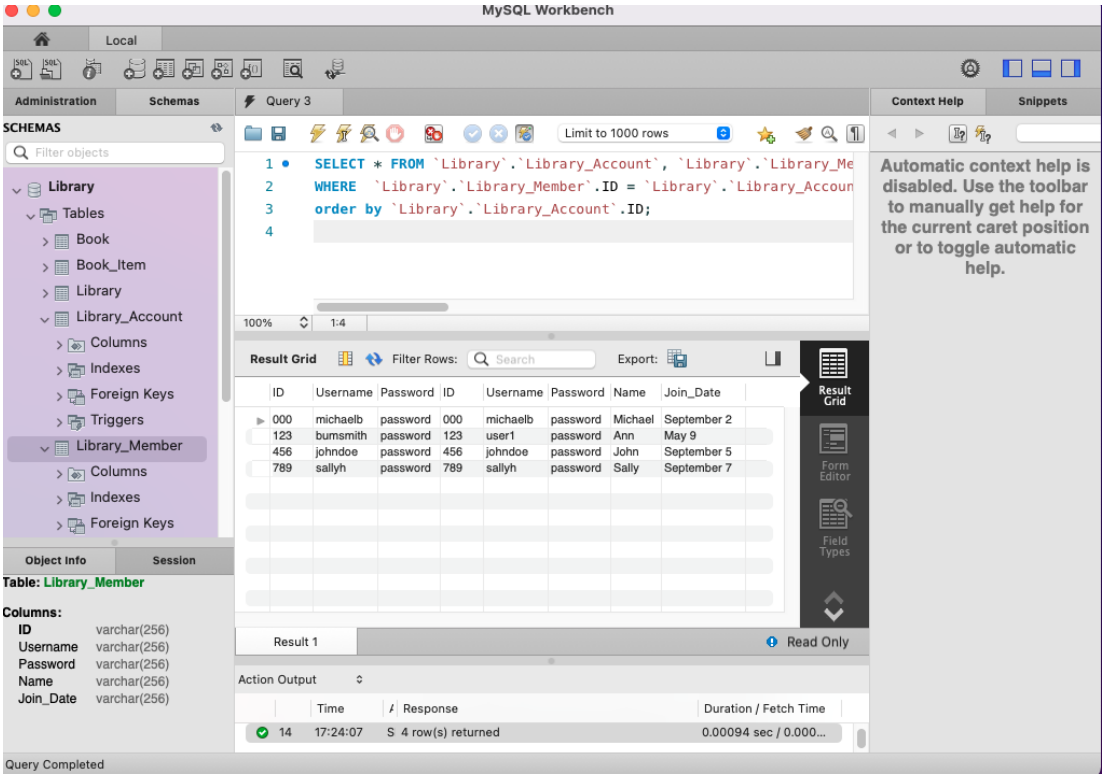
There are several key elements of the database which would benefit from tuning. Specific components of the forign key and primary key relations may be updated such that more complex database operations are possible. This would be a result of updating the schema of the database to include a more feasible model of relations.

Other tuning may be to include more operations and views such that the database model could better operate in connection to an application fit for real world use.

Another tune may be in relation to the username and password system. It could be beneficial to include a type of encryption in relation to the application such that the user's data could be better protected.

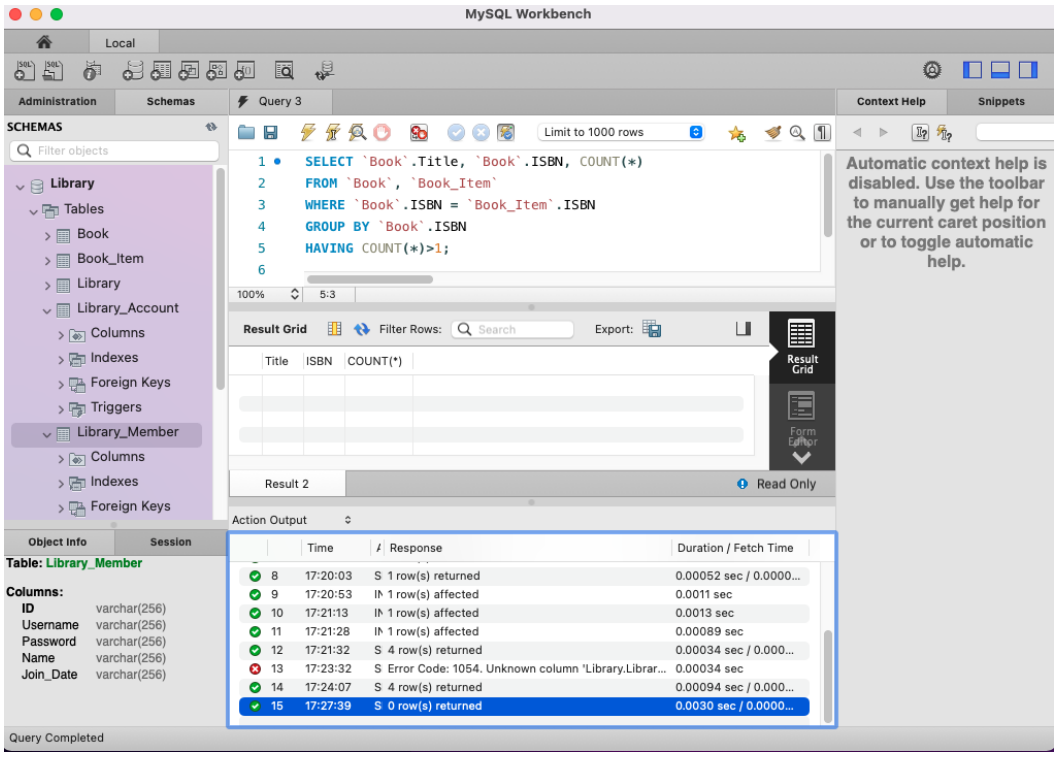
Additional Queries and Views

```
1. SELECT * FROM library.library_account, library.library_member
WHERE  library.library_member.ID = library.library_account.ID
ORDER BY library.library_account.ID;
```

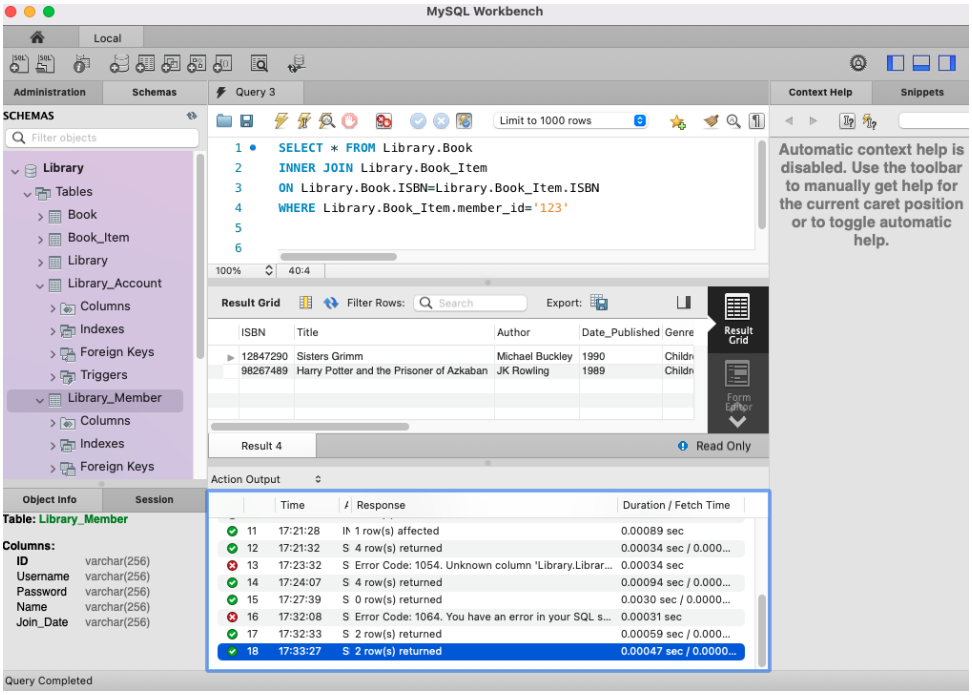



```
2. CREATE VIEW BookView
AS SELECT *
FROM Book
WHERE pages > (
      SELECT AVG(pages)
      FROM Book_Item);
```

```
3. SELECT Book.Title, Book.ISBN, COUNT(*)
FROM Book, Book_Item
WHERE Book.ISBN = Book_Item.ISBN
GROUP BY Book.ISBN
HAVING COUNT(*)>1;
```



4. SELECT * FROM Library.Book INNER JOIN Library.Book_Item ON Library.Book.ISBN=Library.Book_Item.ISBN WHERE Library.Book_Item.member_id=?



```
5. CREATE VIEW AccountInfoView
AS SELECT Library_Account.ID, Library_Member.ID
FROM Library_Account
INNER JOIN Library_Member ON Library_Account.ID = Library_Member.ID
HAVING COUNT(*) >= 1;
```

User Application Interface

We built our application interface using React.js and Node.js. The frontend built using React first displays the home page where the user can select from five options. These are:

1. Searching for books in the catalog
2. Checking out books
3. Renewing a book
4. Placing a hold on a book
5. Viewing all their checked out books + due dates

For the first function, the user inputs a book title and a list of books in the library with that title or with similar titles will display onto the screen along with info about their author and genre. The SQL query for this is:

```
`SELECT * FROM Library.Book WHERE Title LIKE '%${title}%'`
```

For the second function, the user enters their member ID and enters the book title and then clicks on the “Check out” button on the book they wish to check out. This request will then be processed to check if the book is available or not. If it is available, a new Book_item record will be created in the database. The user will get an alert on if the book was successfully checked out or not. The SQL queries for this are:

```
`SELECT * FROM Library.Book_Item WHERE ISBN='${isbn}'`
```

And

```
`INSERT INTO Library.Book_Item VALUES ('${isbn}','${new Date}','${new
Date(+new Date + 12096e5)}','${memberid}')`
```

For the third function, the user enters their member id and can see a list of books that are checked out by them. The list shows the book title, author, and genre. They can then choose to renew any book by clicking the renew button. The SQL queries for this are:

```
`SELECT * FROM Library.Book INNER JOIN Library.Book_Item ON
Library.Book.ISBN=Library.Book_Item.ISBN WHERE
Library.Book_Item.member_id='${memberid}'`
```

And

```
`UPDATE Library.Book_Item
 SET Return_Date='${new Date(+new Date + 12096e5)}'
 WHERE ISBN='${isbn}' AND member_id='${memberid}'`
```

For the fourth function, the user can click a button to place a hold on a book. But a hold can only be placed if there are less than three people on the hold list. The system will first check this and then place the hold. The SQL queries for this are:

```
`SELECT COUNT(*)
 FROM Library.Reservations
 WHERE ISBN='${isbn}'
 GROUP BY ISBN`
```

And

```
`INSERT INTO Library.Reservations
 VALUES ('${isbn}', '${memberid}', '${new Date()}')`
```

For the fifth function, the user can log in and see their account info such as the books they have checked out or placed a hold on. The SQL query for this is:

```
`SELECT * FROM Library.Library_Member WHERE ID='${memberid}' AND
Password='${password}'`
```

Conclusions and Future Work

This project taught us the process of building a system from start to finish from the first step of defining the system and designing the database to the last step of building a user interface for it. We also gained experience in writing SQL queries and testing them. We used new tools like Mysql Workbench and learned how to connect to the database using a programming language.

For the future, we would make the application more realistic by including various assertions into our SQL queries to prevent users from committing some actions. We could also incorporate more interactions in this system like including an ebook third party provider where library members can not only check out physical books but also ebooks. This would require new

tables and relationships and more complex queries. We could also improve the user interface to make it easier to navigate.

References

"Design a Library Management System - Grokking the Object Oriented Design Interview." *Educative*, https://www.educative.io/courses/grokking-the-object-oriented-design-interview/RMlM3NgjAyR.

Elmasri, Ramez, and Sham Navathe. *Fundamentals of Database Systems*. Pearson, 2020.