1. Introduction

a. Project Overview

i. This project will implement a database for a library so that the library can easily manage their borrowing and membership systems, and also keep track of their books and other loanable items

b. Scope

i. The project will consist of databases containing books and all of their attributes, book renters and their attributes, borrowed items and their attributes, and also administrators and their attributes.

c. Glossary

i. Administrator: Someone who has extra privileges, and is generally responsible for managing books, memberships, and can generate reports
ii. Query: A request that is made to the database to generate, update, or delete data from the database
iii. MySQL: A relational database management system used for database operations
iv. Functional Requirements: The operations that the database must be able to perform
v. Non-Functional Requirements: Operational requirements that may add to the performance of the database, but may not be necessary to keep the database operational.
vi. Github: Used for version control, and allows multiple users to work on the same projects at the same time.

2. Stakeholders

a. Administrator: Responsible for managing the system by adding/removing books from the catalog, updating memberships, generating reports, and managing fines and overdue items
b. Librarians: Someone who helps members borrow/return books, updates member information, and maintains day-to-day services at the library
c. Members: People who borrow books and other items from the library

3. Requirements

a. Functional Requirements

Data Deletion. The system will allow safe and controlled deletion of records. To prevent the deletion of important users, the database must only allow users to be deleted if they have no outstanding borrowed books or unpaid fines.

Example Query for deleting a User:
DELETE FROM Users

```
WHERE UserID = X
AND UserID NOT IN (SELECT UserID FROM BorrowedItems WHERE
ReturnDate ReturnDate IS NULL)
AND UserID NOT IN (SELECT UserID FROM Fines WHERE PaidStatus =
‘Unpaid’);
```

Data Entry. The system will support accurate data entries for all major entities. There will be three types of primary entities that can be added. These are User Data, Book Data, and Borrowing Data.

Example Query for adding a new User:

```
INSERT INTO Users (FirstName, LastName, Email, PhoneNumber, UserType,
MembershipStatus, OutstandingFees)
VALUES ('Ethan', 'Doughty', 'ethan.doughty@example.com', '(123) 456-7890',
'Borrower', 'Active’, ‘0.00');
```

Data Retrieval. The system should be able to retrieve requested records quickly and accurately, filtered by certain conditions. For example, the system may be requested to retrieve every book record, or every loan associated with a certain user.

Example Query for retrieving all Book records:

```
SELECT * FROM Books;
```

Data Updates. The system should be able to locate and update the specified records. It should only allow valid modifications to the existing data, rejecting attempted updates that would result in data configurations that violate the imposed constraints (for example, the domain of an attribute or the uniqueness of a primary key). Other requirements might include disallowing an update that would associate more than one person with a book loan. Allowed updates may include changing the name of a user or modifying the author of a book.

Example Query for Updating the name of a User:

```
UPDATE Users
SET FirstName = ‘Bob’
WHERE FirstName = ‘Bobby’;
```

Report Generation. The database should be able to generate reports about the data stored within the database. These reports should be detailed and specific to certain factors that library administrators may want to measure and evaluate.

- **Generate an age group analysis report.** This report will provide a comprehensive analysis of the library's book collection in relation to a particular age group. This report will describe broad trends in areas such

as: genre most often checked out; average time to return a book; average late fees; and more.

Example Query for genre analysis report:

```
SELECT
   Genre,
   COUNT(*) AS TotalCheckouts
FROM Borrowing br
JOIN Books ON BookID = BookID
JOIN Users ON UserID = UserID
GROUP BY Genre
```

Generate a genre analysis report. This report will provide an analysis of what genres are most frequently being checked out from the library. This will allow the library to make decisions like importing more books of a popular genre or allocating less bookshelf space for less popular genres

Example Query for genre analysis report:

```
SELECT
   Genre,
   COUNT(*) AS TotalCheckouts
FROM Borrowing br
JOIN Books ON BookID = BookID
GROUP BY Genre;
```

Generate a collection analysis report. This report will provide a comprehensive analysis of the library's book collection, identifying broad trends in acquisition over the last five years, age of the collection, and more.

Example Query for collection analysis report:

Generate a member engagement report. This report will provide information regarding the engagement of library members. It will show with what frequency members rent books, how many books members rent with each visit, etc.

Example Query for member engagement report:

```
SELECT UserID,
        COUNT(br.BorrowID) AS TotalBorrowings
FROM Users
LEFT JOIN Borrowing ON UserID = UserID
GROUP BY UserID, MemberName
ORDER BY TotalBorrowings DESC;
```

Generate an operational efficiency report. This report should assess the library's operational effectiveness. Analyze key metrics such as book loan and return processing times, overdue book rates, and fine collection trends

Example Query for operational efficiency report:

```
SELECT
        AVG(DATEDIFF(ReturnDate, BorrowDate)) AS AvgLoanDuration
FROM Borrowing
WHERE ReturnDate IS NOT NULL;
```

User Administration. The database must be able to perform operations to add, remove, and update user data as needed.

Add a user. When a person requests a library card to rent books from the library, a new user will be added to the database.

Example Query for adding a new User:

```
INSERT INTO Users (FirstName, LastName, Email, PhoneNumber,
UserType, MembershipStatus, OutstandingFees)
VALUES ('Ethan', 'Doughty', 'ethan.doughty@example.com', '(123)
456-7890', 'Borrower', 'Active', '0.00');
```

Update user's contact and personal information. When a user for example would like to change their email address or phone number, that can be updated in the database if needed.

Example Query for Updating user's contact and personal information:

```
UPDATE Users
SET UserType = 'Standard'
WHERE UserID = 5;
```

Delete a user. When a user would like to rescind their membership, that can be done in the database by deleting the user.

Example Query for adding a new User:

```
DELETE FROM Users
WHERE UserID = 5;
```

b. Data Entities

Books.

BookID, INT (Primary, auto-increment)
Title, VARCHAR(255)

Author, VARCHAR(255)
ISBN, VARCHAR(20) (Unique)
Publication Year, INT (Check for valid year)
Genre, VARCHAR(100)
Availability Status, ENUM('Available', 'Borrowed', 'Reserved')
Total Copies, INT (Default to 1, must be at least 1)

Users.

UserID, INT (Primary, auto-increment)
First Name, VARCHAR(100)
Last Name, VARCHAR(100)
Date of Birth, DATE
Email, VARCHAR(255) (Unique)
Phone Number, VARCHAR(15) (Unique)
User Type, ENUM('standard', 'student', 'senior')
Membership Status, ENUM('Active', 'Inactive', 'Banned') (Default Active)
Outstanding Fees, DECIMAL (Default to 0.00, must be non-negative)

Borrowing.

BorrowID, INT (Primary, auto-increment)
UserID, INT (Foreign key referencing Users)
BookID, INT (Foreign key referencing Books)
Borrow Date, DATE (Default Current Date)
Due Date, DATE
Return Date, DATE (Default NULL)
Overdue Fees, DECIMAL (Default to 0.00, must be non-negative)

c. Non-Functional Requirements

Database operations (retrieval, entry, updates, deletions) should each take less than 5 seconds.
Access to the database should be strictly controlled, with privileges granted on a per-user basis.

d. Hardware Requirements

Laptop Specifications:

CPU: multi-core 64-bit x86 recommended
RAM: 4GB minimum, 8GB or higher recommended
Storage: At least 50GB of free space
OS: Windows, macOS, Linux

e. Software Requirements

Database Management System (DBMS):

- MySQL 8.0 or most recent version
- MySQL Workbench

Development Tools:

- Github(For version control)