**Library Management System**

1. **Introduction**:

The library management system project is a database project that focuses on creating a system for managing a library's collections, members, and borrowing activity. The project involves designing and implementing a database to store information about books, authors, publishers, and library members, as well as managing the borrowing and returning of books. The project also typically includes a user-friendly interface for searching the book catalog, generating reports about borrowing activity, and analyzing library usage.

The motivation behind choosing this application as a course project could be that library management systems are commonly used in educational institutions, public libraries, and other organizations that manage large collections of resources. By developing such a system, you will get hands-on experience with different aspects of software development, such as database design, user interface design, and software testing.

The key components of a library management system include a database to store information about resources, members, and borrowing transactions, a user interface to facilitate interactions between librarians and members, and a set of business rules and processes to manage library operations, such as lending, returning, and reserving resources. Other features might include search and browse capabilities, reporting, and analytics tools.

1. **Database details:**

**Assumption:**

* The library management system has a defined set of rules and regulations regarding the borrowing and returning of books.
* Library members have provided accurate personal information during registration.
* The library has a defined system for classifying books into genre.
* The library has a reliable email service for sending email notifications to members.
* The library has a defined system for managing and updating passwords for librarians and members.
* The library has a defined process for updating and maintaining the book catalog and member information.
* The library has a defined process for removing books and members that are no longer needed in the system.
* The data for librarians, members, and books are unique and different. No one has the same personal information.
* The system offers different login methods for librarians and members. Librarians use a User ID and a shared password for system access, while members can create with their own member ID and password. The User ID is identical for both library staff and members and is used by them to log in to their respective library accounts.
* The User ID cannot be changed. Members can change their password, but the librarians cannot do it since they have a default password which is provided by library.

**Description of Library Management System:**

* The library system is designed to offer a wide range of books, with each **book** having a unique ID and a wealth of information, including its title, ISBN, authors, publishers, genre, and availability. To ensure efficient management, a single librarian can oversee many books, and a book can be lent by one member in a certain time.
* The **member** database contains all relevant information about library members, including personal details such as member ID, phone number, first name, last name, and email, borrowing book ID; in addition, record contains the list of book ID’s and will be updated every time member borrow one book. Each member is assigned a unique member ID. Members can borrow at most one book and receive updates on their borrowing status from the librarian, who can send messages regarding issue dates, return status, and due dates. In addition, each member has one account to log into exactly one system.
* The **librarians** themselves are distinguished by unique IDs and names, and phone number, with the ability to manage the borrowing and returning of books, due dates, and overdue notices. They can also login to the library system to manage the books and interact with membership. To keep members informed, librarians can contact them to remind them of due dates, return status, or extensions. In addition, librarians can use their librarian ID and password that was provided by the library to exactly sign in one library management system.
* To ensure security and control, the **system** requires a unique password and unique User ID to grant access to librarians and members. The system can be logged into by many librarians and members simultaneously to manage books and membership.

**ER diagram:**

**Diagram

Description automatically generated**

Dataset was found on this website: <https://www.kaggle.com/datasets/arashnic/book-recommendation-dataset>. The data provided in the dataset is complete and consistent. Of course, I need to create more data to test my model, based on what the dataset has, such that book ID, genre, availability of book, member ID, phone number, email address, etc. The dataset needs to be clean before insert into the tables.

The website of the Leon County Public Library has all the basic functions and features of the Library Management System model that I have described. This system can help the librarians manage and send a notification to members to remind them the due date, extension, or any updating information. Librarians and Members can log into a system which is created for security reason and to manage the information about books and membership.

**The relational schema:**

Members (Member ID, Phone Number, Email, First Name, Last Name, Record, *Book ID*, *Librarian ID,* *User ID*, Due date, Issue date, Return date)

Primary key: member ID

Foreign key: *Librarian ID*, *Book ID, User ID*

Book (Book ID, Title, Available, Genre, ISBN, Publishers, Author, Image)

Primary key: Book ID

System (User ID, Password)

Primary key: User ID

Librarian (Librarian ID, Name, Phone Number, *User ID*)

Primary key: Librarian ID

Foreign key: User ID

**Based on the relational schema, here are some possible functional dependencies:**

Members:

Member ID → Phone Number, Email, First Name, Last Name

Member ID, Book ID → Due date, Issue date, Return Status

Book ID → Librarian ID

Book:

Book ID → Title, Available, Genre, ISBN, Publishers, Author

System:

User ID → Password

Librarian:

Librarian ID → Name, User ID, Password

Librarian ID → Phone Number

These functional dependencies indicate the relationships between the attributes in each table and can be used to help ensure data integrity and consistency. By properly identifying and accommodating functional dependencies, you can design your database to minimize redundancies and inconsistencies and ensure that data is properly stored and updated.

For the Book table, the dependency Book ID → Title, Available, Genre, ISBN, Publishers, Author does not violate BCNF or 3NF.

For the System table, User ID → Password does not violate BCNF or 3NF. For the Librarian table, the dependency Librarian ID → Name does not violate BCNF or 3NF.

Therefore, the schema is in both BCNF and 3NF.

For the Members table, the dependencies Member ID → Phone Number, Email, First Name, Last Name and Member ID, Book ID → Due date, Issue date, Return date do not violate BCNF or 3NF.

All tables were created are good, and they do not have any functional dependencies that violate BCNF or 3NF.

1. **Functionality details**:

**Coding**: https://github.com/techwithtim/Flask-Web-App-Tutorial/blob/main/website/templates/base.html

There are some functions that I would include in my project:

* **Basic Functions**:
* Insert records to the database.
* Search the database and list or print returned results: searching the information about book and membership.
* Show a few different interesting queries on the database.
* How many books are currently available in the library?
* What is the name of the book that was checked out the greatest number of times in the last year?
* Which books have been checked out by a member with the ID 123456?
* Which books are currently available and belong to the “sports” genre?
* Who is the author of the book with the ISBN number 123456?
* What is the name of the member who checked out the book with the title “Contact”?
* One of the queries must involve join of multiple tables, and one must be an aggregate query:
* Join query: Find the names and email addresses of all members who have expired books currently, along with the titles of the books they have expired.
* Aggregate query: Find the total number of available books, grouped by genre.
* Show how to update records: update information for new books, new members, and new librarian, book return, etc. Also, members can update their password, and librarians can update the return status, due date, and issue date for borrowers.
* Show how to delete records: delete the book, librarian information or membership if it does not exist anymore or is expired.
* **Advanced Functions**:

To enhance the user experience and increase the functionality of the Library Management System, these unique features can make the website become more useful and outstanding from others (Choose one of these to implement):

* *Book recommendation*: Provide a feature for library members to receive book recommendations based on their reading history and preferences.
* *Overdue reminders*: Send automated reminders to members when their checked-out items are approaching their due date. (**I implemented this**)
* *Late Fees Calculator*: This function will calculate the late fees for members who have returned books past the due date. To implement this, you can use a simple formula based on the number of days late and the late fee rate, which can be stored in the database.
* *Event calendar*: Add a calendar feature to display information about upcoming library events and programs.
* *Email Notifications for Book Availability*: This function will send email notifications to members who have placed a hold on a book that has become available. To implement this, you can use an email service, such as Gmail, to send the emails and a task scheduling tool to schedule the notifications.

1. **Implementation details**

In this project, **MySQL** was used as the relational database management system, and **Flask** (Python) was used to establish connections between tables and execute queries. To create a web interface, **HTML** will be used to design the user interface and create the simple layout of the website.

The Web interface was simply designed like a menu which contains choices for member and librarian. The Home page was created for everyone. This means, Members and Librarians will use the same home page to log in. If they successfully log in, they will be led to a page which is for only members or librarians based on their role. The base page for each one will have its own features. For example, the base page for librarians has features for only librarians, such as, search books, update book, delete book, create new book, create new librarian, delete membership, etc. For members, they can borrow a book, return book, search a book, update information, look up information, etc.

The **front-end Web interface** interacts with the **backend database** through requests and responses. When a user interacts with the web-interface (HTML), such as creating, delete, update information, log in, borrowing book, returning book, etc., the web interface will send a request to the backend server (Flask). Then the backend server will process the request, retrieves, and solves data from the database, then sends a response back to front-end web interface.

Here is a general plan for implementing some important functions:

1. How to log into the system?

* Verify the User ID and Password by comparing them to the data on System\_Table. If both match the information in System\_Table, and User ID is the same to Member ID in Member Table or Librarian Id in Librarian Table, the user will be led to the page which is for him/her.

1. "Which books have been checked out by a member with the Member ID 123456789?"

* Join the Member and Book tables on the book ID column to get the list of books that have been borrowed by any member.
* Filter the joined table by the member ID to get the list of books borrowed by the specific member.
* Return the list of book titles and other relevant information, such as author or genre.

1. "Which books are currently available and belong to the 'sports' genre?"

* Call the book table by the book ID column to get the list of all books.
* Filter the table by the book status and genre columns to get the list of available books in the sports genre.
* Return the list of book titles and other relevant information, such as author or book ID.

1. "Who is the author of the book with the ISBN number 123456789?"

* Join the book and author tables on the author ID column to get the list of all books and their author information.
* Filter the joined table by the ISBN column to get the author of the specific book.
* Return the author's name.

1. "What is the name of the member who is borrowing the book with the title 'I am Sky'?"

* Join the Book, and Member tables on the book ID and member ID columns to get the list of all records and their member and book information.
* Filter the joined table by the book title to get the specific record for that book.
* Return the member's name.

1. "Find the names, due dates, and email addresses of all members who have overdue books currently, along with the titles of the books they have checked out."

* Join the Book, and Member tables on the book ID and member ID columns to get the list of all records and their member and book information.
* Filter the joined table by the book status column to get the list of checked out books.
* Return the list of member names, email addresses, and book information.

1. "Display the borrowed record for a Member ID."

* Join the Member and Book tables on the book ID column to get the list of all checkout records and their book information.
* Filter the joined table by the member ID to get the list of checkout records for the specific member.
* Return the list of book titles and other relevant information, such as checkout date or due date.

1. “Find the total number of available books, grouped by genre.” This is an aggregate query.

* Counting books that are available by genre in Book Table.
* Then displaying the list of number of books corresponding to each genre
  + - Maybe take advantage of this to develop an advanced function “Book Recommendation.”

1. *"Display the overdue list of Member ID."*

* Compare Return date to Current date, and check Return Status on Member tables.
* If Due Date <= Current date AND Return Status = ‘No’, the information of members who satisfy this will be displayed.

For the ***advanced function* “Overdue Reminder,”** I will follow these general steps:

* + Set up a background task or scheduler that runs daily to check for overdue records.
  + Retrieve all records that are overdue (i.e., their due date is less than the current date) by taking advantage of function "*Display the overdue list of Member ID."*
  + For each overdue record, retrieve the member's contact information (e.g., email, phone number, name, and book ID, etc.).
  + Send an automated reminder to the member via email or SMS, providing details about the overdue items and the associated late fees.

1. **Experiences**

This project helped me develop important skills such as database design, data modeling, and data management. I learned how to create an ER diagram to visualize the relationships between the entities in the library management system and how to translate that into a SQL schema.

Furthermore, I faced some challenges while working on this project, such as designing a user-friendly interface for librarians and members to interact with the system. I had to ensure that the system was easy to navigate and that the search feature was efficient enough to return relevant results. To overcome these challenges, I had to adapt my knowledge of HTML and Flask to create a visually appealing and responsive website that meets the requirements of the project.

In the future, I could extend this project to include more advanced features such as machine learning algorithms to make personalized book recommendations for members based on their borrowing history. Additionally, I could implement a system for online book reservations and payment processing to improve the library's services. These enhancements would require a deeper understanding of programming languages and frameworks such as Python and Flask, which I could learn through further study and practice. Overall, this project provided me with a solid foundation for further development and improvement of library management systems.

1. **References**
2. "Schedule Documentation - Examples". Schedule. Retrieved April 19, 2023. <https://schedule.readthedocs.io/en/stable/examples.html>.
3. Code Speedy. Sending Emails using SMTP and MIME in Python. Retrieved September 17, 2021, from <https://www.codespeedy.com/sending-emails-using-smtp-and-mime-in-python/>
4. Data Flair. Library Management System Python Project – Download Source Code. Retrieved September 17, 2021, from <https://data-flair.training/blogs/library-management-system-python-project/>
5. Kaggle. (n.d.). Book Recommendation Dataset. Retrieved September 17, 2021, from <https://www.kaggle.com/arashnic/book-recommendation-dataset>
6. Matellio. How to Develop Library Management System? Retrieved September 17, 2021, from <https://www.matellio.com/blog/how-to-develop-library-management-system/>
7. PyBites. How to Send an Email with Python. Retrieved September 17, 2021, from <https://pybit.es/articles/python-MIME/>
8. Python documentation. email.mime – Creating Email and MIME Objects. Retrieved September 17, 2021, from <https://docs.python.org/3/library/email.mime.html>
9. Python documentation. Examples: Email Examples. Retrieved September 17, 2021, from <https://python.readthedocs.io/fr/hack-in-language/library/email-examples.html>
10. Python Geeks. Python Library Management System Project. Retrieved September 17, 2021, from <https://pythongeeks.org/python-library-management-system-project/>
11. Tech With Tim. Flask Web App Tutorial. Retrieved September 17, 2021, from <https://github.com/techwithtim/Flask-Web-App-Tutorial/blob/main/website/templates/base.html>
12. Váradi, Zoltán. "Flask API: How to Return Response but Continue Execution." Medium, May 3, 2021. <https://zoltan-varadi.medium.com/flask-api-how-to-return-response-but-continue-execution-828da40881e7>.