SYSTEM DESIGN AND ARCHITECTURE

CS 6360 – Database Design

Bhanu Maneesh Reddy Mannem

bxm220055

Introduction:

The Library Management System (LMS) serves as an essential tool in efficiently managing the bustling activities of libraries in schools and colleges. With a constant influx of readers seeking knowledge, manual tracking of transactions becomes an overwhelming task. This software addresses the challenges faced by librarians by offering a user-friendly graphical interface that interacts seamlessly with a MySQL backend. By streamlining operations such as book searches, loans, borrower management, and fines processing, the LMS significantly enhances the overall management of library resources, providing a more organized and effective solution for librarians.

System Architecture:

The Library Management System (LMS) is a comprehensive solution designed to streamline library operations. The architecture comprises a web-based application running on a local web server, catering to both local library networks and potential internet deployment. The system is built with a frontend developed using HTML, CSS, Bootstrap, JQuery, and JavaScript, while the backend leverages Python with the Django framework. The MySQL database serves as the underlying Database Management System (DBMS), and the system architecture follows the Model-View-Controller (MVC) framework provided by Django.

Frontend(GUI):

The frontend is implemented using Python Django, a web framework known for its simplicity and flexibility. The GUI is designed to be intuitive, ensuring ease of use for librarians. Django's built-in ORM (Object-Relational Mapping) is employed for seamless communication with the MySQL database.

Backend(Database):

MySQL is chosen as the backend database for its reliability and scalability. The database schema is designed to efficiently store and manage information related to books, borrowers, loans, and fines.

Database Schema:

The database schema is normalized to eliminate redundancy and ensure data integrity. The main tables include:

Books: Stores information about each book, including ISBN, title, and availability.

Authors: Contains details about book authors.

Book_Authors: A junction table linking books to authors in a many-to-many relationship.

Borrower: Stores borrower information, including name, SSN, address, and contact details.

Book_Loans: Records book loans, tracking the borrower, due date, and return date.

Fines: Manages fine details, including the amount, payment status, and associated loan.

Functionality:

Graphical User Interface (GUI):

- Developed using Python Django, providing an intuitive and user-friendly experience.
- Ensures ease of access to all system functionalities.
- Responsive design for seamless usability.

Book Search and Availability:

- Utilizes a single text search field for ISBN, title, and author searches.
- Implements case-insensitive substring matching for comprehensive search results.
- Displays ISBN, book title, authors, and availability in search results.

Book Loans:

- Enables book checkout based on selected books or direct input of ISBN.
- Validates borrower limits to ensure a maximum of 3 active loans per borrower.
- Supports checking in books by searching on ISBN, borrower card number, or borrower name.
- Automatically generates due dates and unique loan IDs.

Borrower Management:

- Allows librarians to create new borrower accounts with mandatory details.
- Generates unique card numbers for new borrowers.
- Rejects duplicate borrower creation based on SSN.

Fines:

- Assesses fines at a rate of $0.25/day.
- Provides mechanisms for updating fine details and entering payments.
- Calculates fines for both returned and currently late books.
- Groups fines by borrower card number, facilitating easy management.

Data Initialization:

Baseline data is provided in CSV files, containing information about books, authors, and borrowers. The data is normalized and mapped onto the database schema during the system setup.

Design Decisions and Assumptions:

- 'Book' schema includes an 'Availability' column for efficient book availability tracking.
- ISBN13 information is omitted as it's not specified in the schema or project requirements.
- Assumes both frontend and backend hosted on the same server for streamlined communication.
- Fines assessed based on a daily rate of $0.25.

- Late book fines determined by the difference between due date and return date or the current date.
- Anticipates daily execution of a batch script for accurate fine updates during library closed hours.
- Emphasizes GUI responsiveness over aesthetic design for an intuitive and efficient user interface.

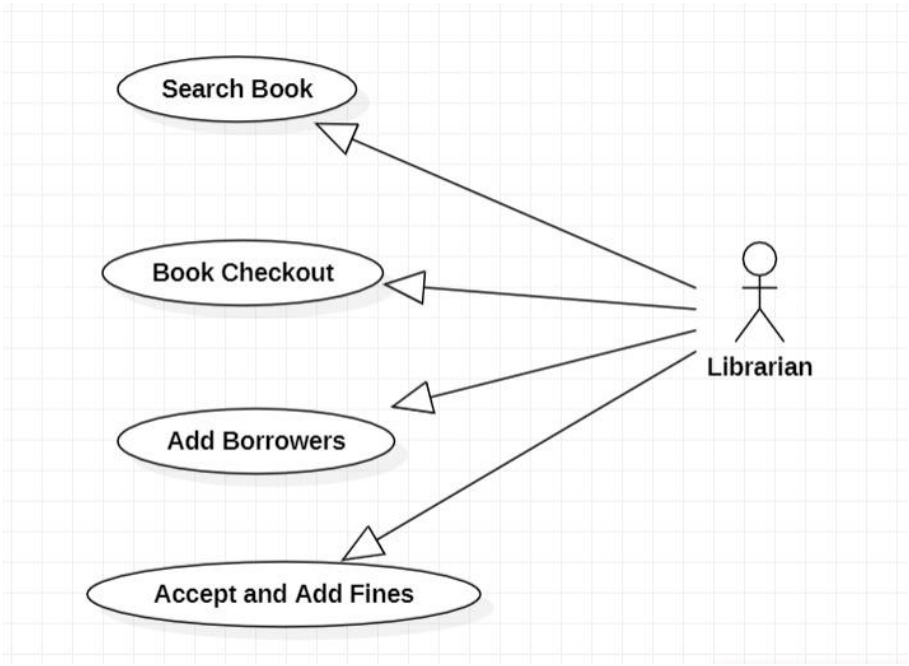

Figure 1 System Design

Conclusion:

The Library Management System offers an efficient and user-friendly solution for librarians to manage library operations seamlessly. The use of Django and MySQL ensures a robust and scalable architecture.