Library Management System V2

Subject: Modern Application Development - II

Abstract

Digital Library Management System is a multi-user app used by one and only librarian and other general users/students for issuing, buying and managing books. This application allows user or students to request, read, return E-Books. On the other hand, librarian can add new sections/author/books, issue/revoke access book from user and view statistics for the same.

The data flow diagram, database relationship model, the APIs and the wireframe for frontend design were designed before implementing the application code. The APIs were tested using Postman and, similar testing for the frontend was accomplished. Thus, finalizing the Digital Library Management System application.

Technology Stack

- Backend Server: Flask (Python), Celery, Redis
- Frontend Development: VueJS 3, JavaScript, HTML, Bootstrap 5
- Operating System: Linux/Ubuntu

Methodology

- **Data Flow Diagram (DFD)**
  A graphical representation illustrating data flows between User, Librarian and the System is designed as per project statement. The DFDs show the processes (Circles), data stores (Cylinder), and data flows (Arrows) involved. The DFDs are made hierarchically from Level 0 to Level 1 (Level 1 gives detailed User and Librarian data flows with system).
  The Figure 1 (left), depicts CRUD operation on Book/Section/Author for Librarian, along with, Accept/Reject/Revoke Issues and generating Statistics on data models and their usage. In Figure 1 (right), User functionalities include requesting for book issue and returning the same, viewing (reading) and downloading (buying) books, reviewing books. Users also get auto-generated asynchronous E-mails like daily reminder, monthly report, etc. Similar to librarian, users can also view their personal statistics for favourite authors, section distribution for their book issues, etc.

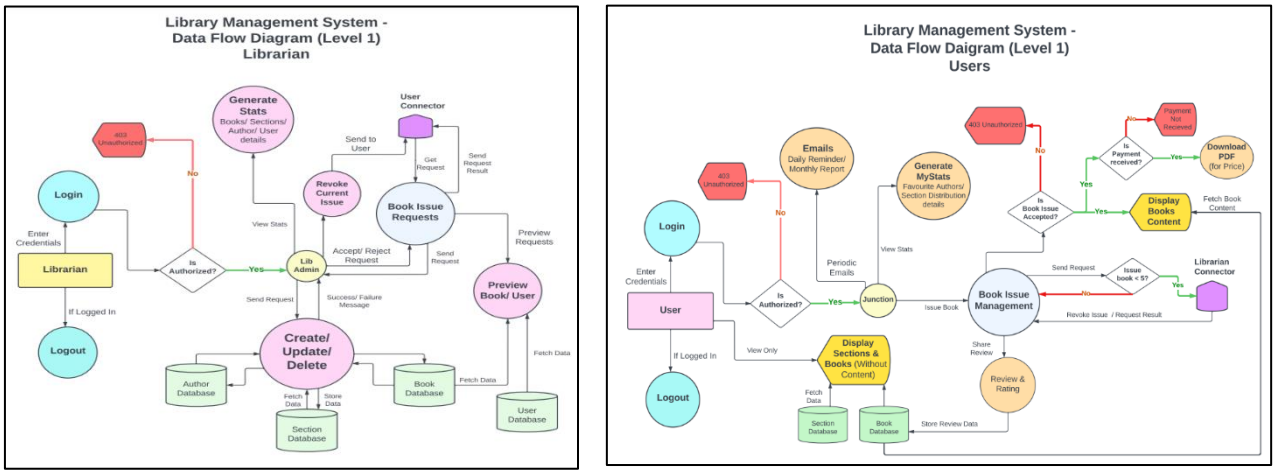

Figure 1: DFD (Level 1) for Librarian and User Functionalities

- **Database Relationship Model**
  Database relationship model is the process of creating a detailed data model for a database and defining the relationship between these models/tables. Designing databases involved defining the structure of the database, like tables, columns, data types, constraints, relationships, and defining primary keys. Minimum redundancy and maximum efficiency (minimum joins required) are set as priority for the database design of the Digital Library Application. Various digital resources are stored in tables like Books, Sections, Authors, Reviews, User and Role. Each of these tables have their own primary keys which are also used as foreign keys in other tables for defining relationships between them. There are multiple relationships few of them are: Book to Section (Many-to-One), Book to User (Many-to-Many), Book to Reviews to User (Many-to-Many). Tables like user_books and user_activity store the books issued by the user thus, can be used to deliver statistics for individual users. The Figure 2, gives detailed logical schema and relationship between tables.

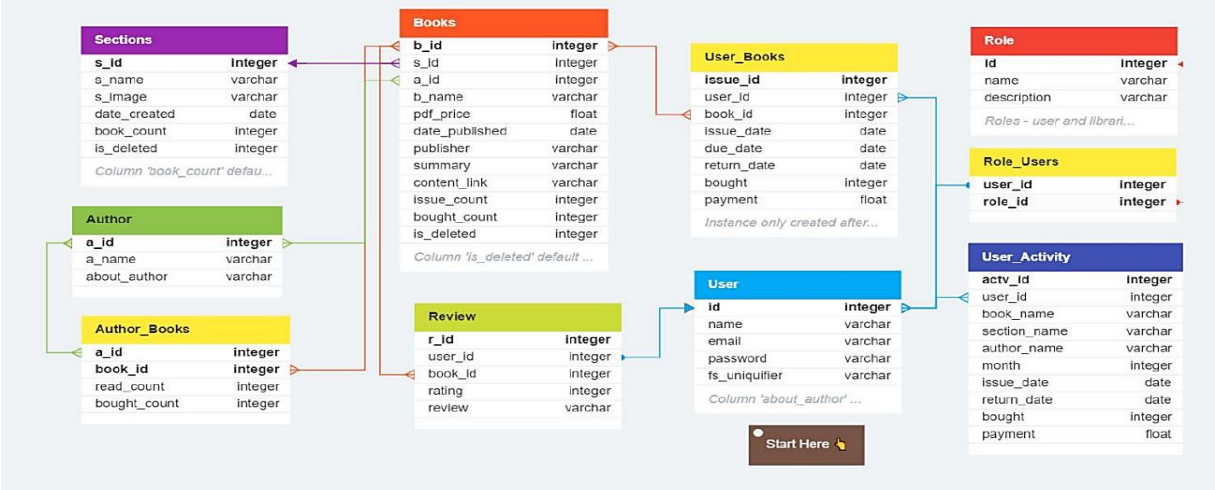

Figure 2: Database Design

- **Application Programming Interface (API)**
  Set of rules and protocols were defined in form API that allows the frontend to communicate with the backend/serve and use its functionalities. APIs are used to define the methods and data formats for the applications to request for and exchange information. For the Digital Library Management System, Representational State Transfer or REST API was used to define the protocols for communication. The interface between the client and server used HTTP methods like GET, POST, PUT and DELETE to perform operations on resources identified by URIs (Uniform Resource Identifiers). For security purpose of APIs, security measures from Flask-Login library in python was used. Security measure involved authorization using Login page developed, Token-Based Authentication and Cross-Origin Resource Sharing (CORS) policy.

- **Frontend Wireframe**
  The look and feel of frontend for the Digital Library was first designed, called as the Frontend wireframe. Firstly, the visual representation or UI were outlined. Following which the basic structure and navigation flow for frontend including hierarchy of pages were defined. In this phase, the URLs to every page in frontend along with the parameters and arguments to pass while traversing through them were decided.

Result & Conclusion

On completion of the designing phase, coding for backend and frontend was initiated. Along with these, asynchronous jobs and tasks using Celery were simultaneously developed. Caching for some common APIs was implement using Redis. Finally, the whole application was tested on Postman (for backend Unit Testing) and on browser for ensuring seamless integration of frontend and backend of the application. Figure 3 are some example of the working application.

Video Link

Click Here to view application demonstration

Reference

- IITM MAD 2 Course Materials
- MAD 2 Live Session Code
- Official Documents for Flask, Celery, Crontab, VueJS, etc.