# Library Management System - Project Part 2: Library Database Requirements

**EECS447: Database Systems**

**Team Name:** ASYNCLIB
**Team Members:** Nick, Ashton, Cole, Sean, Yadhunath

**Professor:** Hossein Saiedian
**Semester:** Fall 2025

## Revisions

Version: 1.4

| Version | Date | Author | Description |
|---|---|---|---|
| 1.4 | 09/23/25 | Nick, Ashton, Cole, Sean, Yadhunath | Made the initial draft, assigned roles, created the document template. |
| 1.5 | 09/26/25 | Nick, Ashton, Cole, Sean, Yadhunath | Worked on finalizing the requirements and performed final clean-up for submission. |
| | | | |

## *Project Overview*

- This project is a Library management system (LMS) that is designed to take in a detailed amount of information about books and be able to manage memberships and clientele along with profiles and account management. The database demonstrates critical vast wealth of information into an easily readable and navigable interface for a client's simplicity.

---

***Scope***

- This project will develop a resource management system for books and magazines, storing details such as title, author, ISBN or issue number, publication data, genre, and availability. It will also include a client and membership management system with unique IDs, contact information, membership categories (regular, student, senior), and borrowing restrictions. A tracking system will manage borrowed, returned, and reserved media with timestamps, client details, and rules for borrowing limits and late return fees. A user interface will enable staff to check out and return items, add new resources, manage accounts, calculate fines, and view availability and client profiles.

---

***Glossary***

- **PostgreSQL: Adds objects-oriented features to sql, such as inheritance and custom data types along with additional features**
- **Flask: Python based web framework**
- **React: JavaScript library for building user interfaces**

---

***Stakeholders (order does not matter)***

1. Librarians and Associate(library staff)
2. Clients(members)
    - Regular, Student, and Senior memberships
3. IT staff and Admins
4. Developers(Us developers)
5. Professor and TA(Saiedian and Sophia)

---

***Functional Requirements***

*These are functions that will be included in the database at minimum*

*User Management*

- Data user information: Unique profile IDs, membership level, contact information, name, age.
- Membership types(Age based): Junior, regular, student, senior.
- Borrowing limits correspond with membership types.

*Storage information*

- Data: book ISBN, issue number, publication date, genre, availability.
- Magazines: unique ID, number, publication date, genre, availability.
- Digital Media: Unique ID, publication date, director, genre, availability.

*Interface functionality*

- User borrowing and returning processes.
- Paying late fees.
- Track date of return, late fees, check out date, and renewal dates.
- Borrowing history tracking.
- Automate calculating fees and fee reports.

*Access*

- Library admins have admin access.
- Developers have admin and full access.
- Clients have restricted access.

---

# *Database Requirements*

## *Hardware Requirements*

### EECS Server Environment (Primary Deployment)

- **Processor**: Multi-core x86_64 architecture
- **Memory**: Minimum 2GB RAM allocated for database operations
- **Storage**: Minimum 10GB available disk space for:
    - Database files (~500MB estimated for project data)
    - Transaction logs and backups
    - System overhead
- **Network**: Stable network connection for remote access

### Local Development Environment (Alternative)

- **Processor**:
    - Intel Core i3 or AMD equivalent (minimum)

    - Intel Core i5 or better (recommended)
- **Memory**:
    - 4GB RAM (minimum)
    - 8GB RAM (recommended for optimal performance)
- **Storage**:
    - 50GB available disk space (minimum)
    - SSD recommended for better I/O performance
- **Network**: Internet connection for software downloads and GitHub access

## *Software Requirements*

### Backend

- PostgreSQL
- Flask (Python)
    - Connects to database to run queries

### Frontend

- React (Javascript)
    - For a simple UI for the users to interact with the backend and the database.

* - indicates that an attribute can be NULL

## *ITEMS (Every book/movie/magazine in the database)*

| item_id | type | title | publication_date | publisher | availability | held_by* |
|---|---|---|---|---|---|---|
| INT | VARCHAR / INT | VARCHAR(128) | DATETIME | VARCHAR(32) | BOOL | INT |
| **Primary key** | Whether it is a book or magazine | Title of work | Can be a full date and time or just a year | - | Is it in stock? | *Foreign key referencing clid - ON DELETE SET NULL* |

## *BOOKS*

| item_id | authorfirst* | authorlast | ISBN | genre |
|---|---|---|---|---|
| INT | VARCHAR(32) | VARCHAR(32) | INT | String array |
| *Foreign key ref. item_id* | - | - | - | - |

### *MEDIA*

| item_id | directorfirst | directorlast | genre |
|---|---|---|---|
| INT | VARCHAR(32) | VARCHAR(32) | String array |
| *Foreign key ref. item_id* | - | - | - |

### *MAGAZINES*

| item_id | issue_number |
|---|---|
| INT | INT |
| *Foreign key ref. item_id* | |

### *CLIENTS*

| clid | firstname | lastname | phone | email | member_type | account_status |
|---|---|---|---|---|---|---|
| INT | VARCHAR(32) | VARCHAR(32) | INT? | VARCHAR(64) | INT | VARCHAR(8) |
| **Primary key** | - | - | - | - | *Foreign key ref. member_type_id - ON DELETE SET NULL* | - |

### *MEBERSHIPS*

| member_type_id | member_type_name | borrowing_limit | late_fee_rate | checkout_length |
|---|---|---|---|---|
| INT | VARCHAR(16) | INT | FLOAT | INT |
| **Primary key** | - | - | Daily rate of late fee | Days before late |

### *RESERVATIONS*

| item_id | reserved_by | date_of_reservation |
|---|---|---|
| INT | INT | DATETIME |
| *Foreign key ref. item_id* | *Foreign key ref. clid* | |

[Database spreadsheet]