Lithium Team – Milestone 3: System Design

The library management system application consists of two parts: a Python backend and a React frontend. Since each of the two parts is in a different language, with the backend being in Python and the frontend being in React's custom extension of JavaScript, the two parts are unified using FastAPI to establish a communication layer.

For the backend, Python was chosen because it was easy to write and prototype, the team was familiar with it, and it had multiple libraries relevant to the project. The backend begins with main.py, which creates a FastAPI instance and initializes middleware for the backend and frontend to communicate with each other. It then initializes routers, where each router handles a subpage and its features. Each router defines a set of functions, where each function is associated with a type of HTTP request. When the router receives a specific HTTP request, it runs the function associated with the request.

The init_db.py file is run prior to the backend code, and it initializes the tables in the database. It is idempotent – it detects whether the tables exist while initializing them, and thus does not overwrite them in case of repeated runs. There is also init.py, which obtains user-provided database credentials. These credentials are provided using a dotenv configuration file placed by the user in the application backend directory. This was chosen as a convenient method to provide credentials that would persist across multiple uses of the application. Using these credentials, it initializes a MySQL cursor, which allows Python to run SQL statements that query and modify the database.

The SQL itself consists of queries within each router that run on the tables in the database and extract or modify the data needed for the library management to work. The database tables and their attributes follow the schema outlined in Milestone 1. Two blank tables are initialized, Book_loans and Fines, which manage book loans and fines respectively.

For the frontend, React was chosen because it was a versatile tool that the team was familiar with using. The frontend is structured as four pages: search, borrowers, loans, and fines. The app is called from main.jsx, which calls App.jsx – a file that sets up four subpages, each corresponding to a set of features. The homepage is the Search Books page. This was chosen because the central feature of a library system is the books, and being able to search for specific books in a library is important. The simplicity of the search page's functionality was inspired by Google, whose front page simply displays a search bar.

All the API-related functions, which make HTTP requests, are stored together in api.jsx. This allows FastAPI to handle HTTP requests as methods and to import them into files where they are needed. GET requests obtain data from the backend and send it to the frontend, and POST requests obtain data from the frontend and send it to the backend. These are then received by the routers in the backend, which enable the backend integration.

The frontend also implements additional features to prevent the user from inputting malformed data. First, it designates "required fields" in its data entry forms. These fields are indicated with an asterisk (*). If the user attempts to submit a form without filling in these fields, the browser highlights the first missing field and requests the user to fill it before submitting. Second, the SSN and phone number fields are coded to only accept numeric input.

If a user attempts to enter non-numeric characters into those fields, they will not be displayed. The SSN and phone number are also automatically formatted once entered. Finally, only three books can be checked out by one borrower at once. If more than three books are selected for being checked out, the system only checks out the first three books of the selection and throws an error for the additional books.