UNIVERSITY OF
KARACHI

BSCS-520

ADVANCE SOFTWARE ENGINEERING

PROJECT – FILE (LIBRARY MANAGEMENT SYSTEM)

GROUP MEMBERS:

NAME: MUHAMMAD WAQAS ANSARI
SEAT NO: B21110006095

NAME: HAMZA ABDEALI
SEAT NO: B21110006036

NAME: MUHAMMAD WAHAJ-UL-HAQ
SEAT NO: B21110006097

NAME: MUHAMMAD BILAL NADEEM
SEAT NO: B21110006069

1. Introduction

The **Library Management System** is a web application designed to manage a library's core functionalities. It serves two primary user roles: **Students** who can browse and borrow books, and **Librarians** who manage the library's inventory, track transactions, and impose fines for overdue books or other reasons.

The system uses **Flask** for the web framework, **Jinja2** for templating, and **CSV files** for persistent data storage. The design ensures a clear separation between the data access layer (handling CSV operations), business logic (models), and presentation layer (HTML templates, CSS, JavaScript).

2. Design Patterns Used

Singleton Pattern

The core design pattern employed in this application is the **Singleton pattern**. This pattern ensures that the `Database` class has only one instance throughout the lifetime of the application, providing a single point of access to all CSV data operations.

Key Aspects of Singleton in this Application:

- **Controlled Instance Creation:** The `Database` class overrides the `__new__` method to check if an instance already exists. If not, it creates one; otherwise, it returns the existing instance.
- **Consistency:** All parts of the application access the same `Database` instance, ensuring consistent read/write operations on CSV files and preventing conflicts.
- **Global Access:** The singleton object can be easily accessed from any module within the application, providing centralized management of data persistence.

3. System Architecture & Folder Structure

The system follows a modular architecture:

```
Library-MS-build/
├─ requirements.txt
├─ app.py                             # Main Flask application and route definitions
├─ library/                           # Contains business logic and models
│   ├─ __init__.py                    # Package initializer
│   ├─ book.py                        # Book model definition
│   ├─ database.py                    # Singleton Database class handling CSV operations
│   ├─ librarian.py                   # Librarian model and related methods
│   └─ student.py                     # Student model and related methods
├─ data/                              # CSV files for persistent storage
│   ├─ all_borrows.csv                # Records of active borrowings
│   ├─ all_transactions.csv           # Records of all transactions (borrows/returns)
│   ├─ books.csv                      # List of books
│   ├─ librarian.csv                  # Librarian account details
│   ├─ manual_fines.csv               # Manually added fines by librarians
│   └─ students.csv                   # Student account details
├─ static/
│   ├─ css/
│   │   └─ styles.css                 # CSS styling for the application
│   └─ js/
│       └─ scripts.js                 # JavaScript functions for confirmations and interactions
└─ templates/                         # HTML templates rendered by Flask and Jinja2
    ├─ base.html                      # Base template with dynamic navigation
    ├─ index.html                     # Home page with dynamic messages
    ├─ student_register.html          # Student registration form
    ├─ student_login.html             # Student login form with "Create Account" link
    ├─ student_dashboard.html         # Dashboard for logged-in students
    ├─ student_fines.html             # Displays student's fines
    ├─ librarian_register.html        # Librarian registration form
    ├─ librarian_login.html           # Librarian login form with "Create Account" link
    ├─ librarian_dashboard.html       # Dashboard for logged-in librarians with new links
    ├─ books_list.html                # List of all books (with Yes/No availability)
    ├─ borrow_book.html               # Page for students to borrow books
    ├─ borrowed_books.html            # Page for students to view and return borrowed books
    ├─ librarian_transactions.html    # Page showing transaction history with "Add Fine" links
    ├─ librarian_borrowed_books.html  # Page showing all currently borrowed books with "Add Fine" links
    └─ librarian_add_fine.html        # Form for librarians to add a manual fine
```

Module Responsibilities:

- **app.py:** Defines all Flask routes, handles session management, and orchestrates interactions between templates and the library models.
- **library/database.py:** Implements the Singleton pattern, manages all CRUD operations on CSV files, processes transactions/fines, and handles manual fines.
- **library/book.py:** Data structure for representing a book.
- **library/student.py & library/librarian.py:** Define methods and properties for student and librarian entities, utilizing the `Database` for data operations.
- **Templates:** Render dynamic HTML pages based on user role and actions.
- **Static Files:** Contain CSS for styling and JavaScript for user confirmations.

4. Module Descriptions

Flask Application (`app.py`)

- **Routing & Session Management:** `app.py` defines all HTTP routes for both student and librarian functionalities. It handles user registration, login, dashboard navigation, and business operations (e.g., borrowing books, adding fines).
- **Dynamic Navigation:** Uses session variables to determine what navigation links to display.
- **Integration with Models:** Instantiates `Student` and `Librarian` objects based on session data and delegates operations to these models and the `Database`.

Models and Data Access

Database Singleton (`library/database.py`)

- **Singleton Pattern Implementation:** Ensures a single instance.
- **CRUD Operations:** Methods to create, read, update, and delete records from CSV files for books, students, librarians, transactions, and fines.
- **Transaction Handling:** Methods for borrowing and returning books record transactions and update the corresponding CSV files.
- **Manual Fines:** Methods to add and retrieve manual fines from `manual_fines.csv`.

- **Additional Retrieval Methods:**
  - `get_all_transactions()`: Retrieves all transaction records.
  - `get_all_borrowed_records()`: Retrieves all active borrow records.

Book Model (`library/book.py`)

- **Data Representation:** Represents a book entity with attributes such as ID, name, author, publisher, publish date, availability, and number of copies.
- **String Representation:** Provides a `__str__` method for readable book information if needed.

Student Model (`library/student.py`)

- **Student Operations:** Methods to view books, borrow and return books, check fines, and deregister.
- **Fines Calculation:** Combines automatic late fees and manual fines to compute total fines.

Librarian Model (`library/librarian.py`)

- **Librarian Operations:** Methods for adding, updating, and removing books. Utilizes `Database` for persistent changes.
- **Delegated Functions:** Most transaction and fine-related operations are handled through Flask routes and the Database.

Templates and Frontend

- **`base.html`:** Base template that includes dynamic navigation links based on session status (student logged in, librarian logged in, or guest).
- **`index.html`:** Home page that displays different welcome messages based on user login status and role.
- **Login Pages (`student_login.html`, `librarian_login.html`):** Include links for users to create an account if they don't already have one.
- **Registration Pages (`student_register.html`, `librarian_register.html`):** Forms for creating new accounts.
- **Dashboard Pages (`student_dashboard.html`, `librarian_dashboard.html`):** Overview pages with navigation specific to each role, including new links for viewing transactions and borrowed books for librarians.
- **Books List (`books_list.html`):** Displays all books with availability shown as "Yes" or "No". For librarians, this page provides options to update or remove books.
- **Borrowing and Returning (`borrow_book.html`, `borrowed_books.html`):** Interfaces for students to borrow and return books.
- **Transactions and Fines:**
  - **`librarian_transactions.html`:** Lists all transactions with "Add Fine" links.
  - **`librarian_borrowed_books.html`:** Lists all currently borrowed books with "Add Fine" links.

    - **`librarian_add_fine.html`:** Form for librarians to add a manual fine.
    - **`student_fines.html`:** Displays a student's total fines.

Static Files

- **`styles.css`:** Provides styling for the application, including layout of navigation, tables, forms, and buttons.
- **`scripts.js`:** JavaScript functions that offer user confirmation dialogs before actions like borrowing, returning, or removing books.

5. Functionalities

Student Features

- **Registration/Login:** Create and access an account.
- **Viewing Books:** List all books with availability indicated as "Yes"/"No".
- **Borrowing Books:** Borrow available books through a dedicated interface.
- **Viewing & Returning Books:** See currently borrowed books and return them.
- **Checking Fines:** View total fines, which include automatic late fees and manual fines added by librarians.
- **De-registration:** Remove account if there are no pending obligations.

Librarian Features

- **Registration/Login:** Create and access an account.
- **Manage Books:** View, add, update, and remove books from the library.
- **View Transactions:** See a history of all borrow/return events, with the option to add fines.
- **View Borrowed Books:** Check all currently borrowed books and add fines where necessary.
- **Add Manual Fines:** Assign fines to students manually based on transaction or borrow information.

6. User Flows & Interfaces

Student Flow

1. **Home Page:** Dynamic welcome message changes if logged in.
2. **Login/Registration:** Students can log in or create an account via login pages.
3. **Dashboard:** After login, the student can navigate to view books, borrow, return, check fines, or deregister.
4. **View Books:** Displays a list of all books; availability shown as Yes/No.
5. **Borrow Books:** Page for selecting and borrowing available books.
6. **View Borrowed Books:** List currently borrowed books with options to return them.
7. **Check Fines:** Displays total fines.

8. **Deregister:** Account removal if conditions are met.

Librarian Flow

1. **Home Page:** Dynamic welcome message changes if logged in.
2. **Login/Registration:** Librarians log in or create an account.
3. **Dashboard:** Access to book management, transactions, and borrowed books views.
4. **Manage Books:** Update library inventory.
5. **View Transactions:** Browse transaction history and add fines.
6. **View Borrowed Books:** See active borrowings and add fines.
7. **Add Fine:** Input fines for students manually.
8. **Logout:** End session when finished.

7. Running the Application

Prerequisites:

- Python 3.x installed
- Flask installed (`pip install flask`)

Steps:

1. Set up the folder structure as described.
2. Ensure all necessary CSV files exist in the `data` folder (the application will create missing files).
3. Run the application:

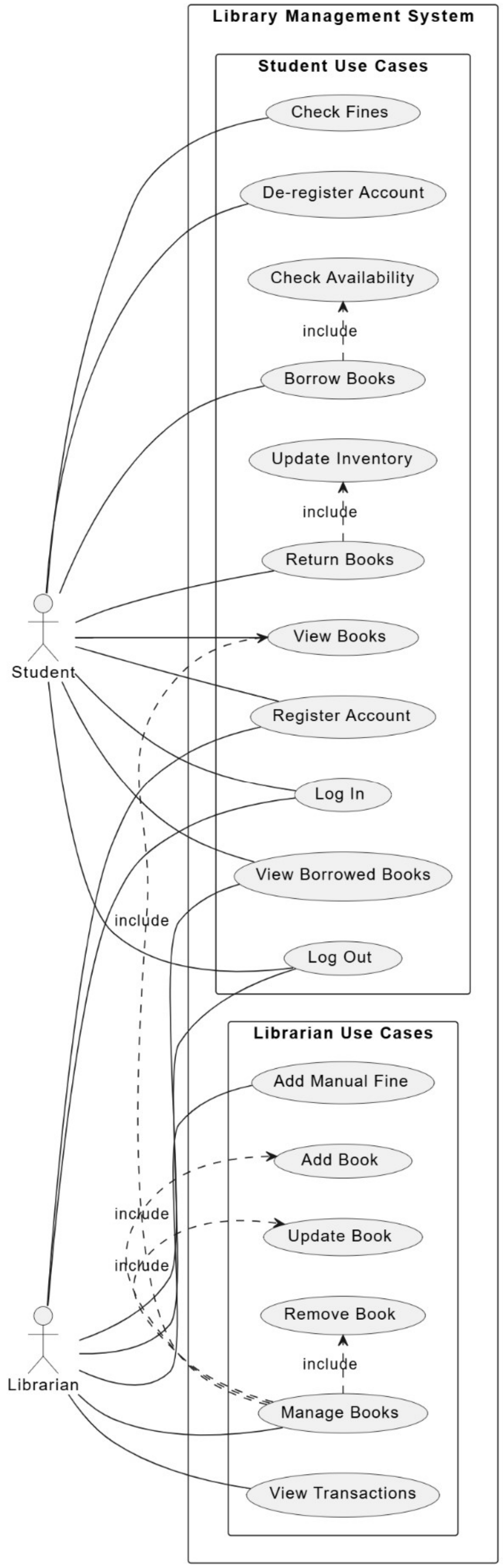
   ```
   python app.py
   ```

4. Open a browser and navigate to http://127.0.0.1:5000/.

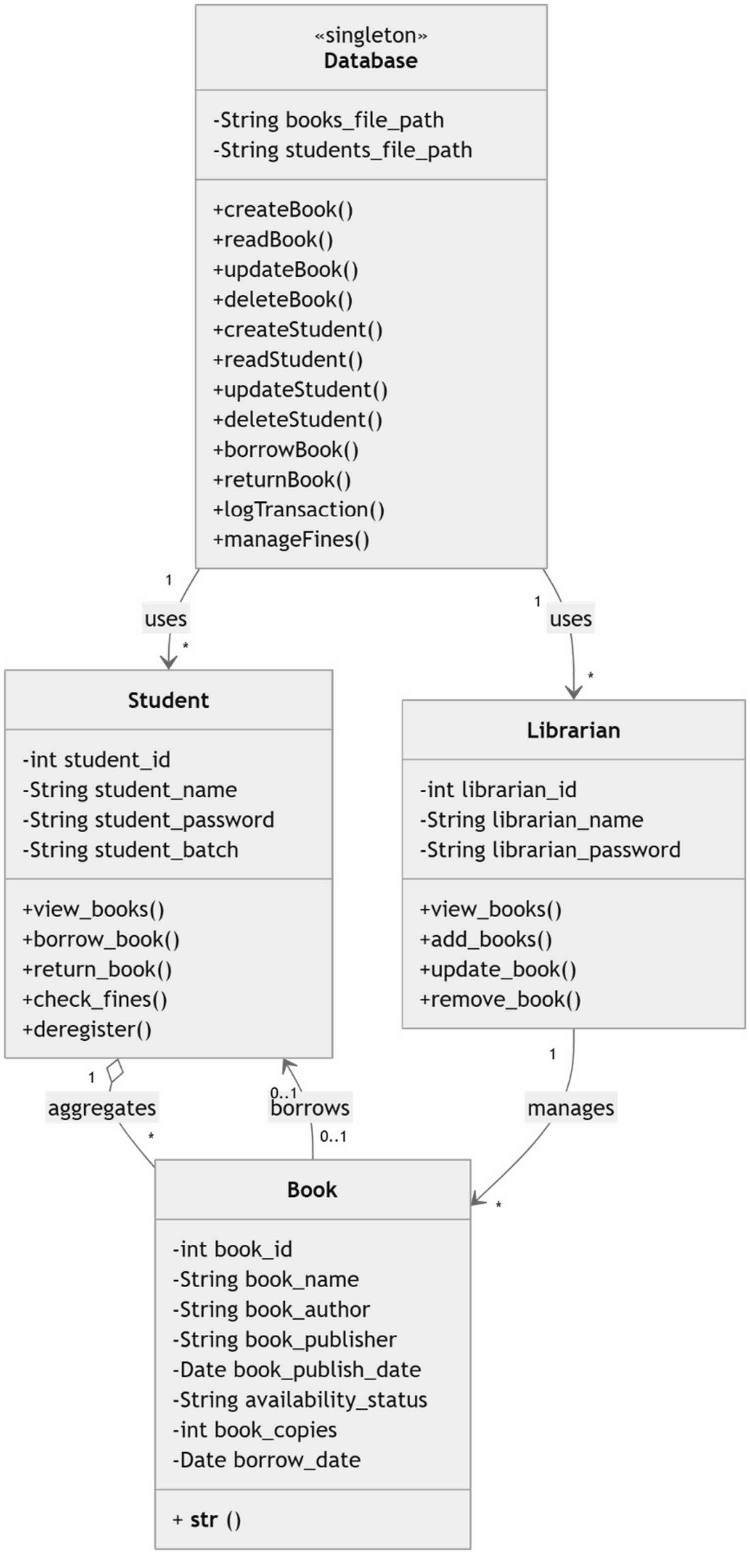
8. Conclusion

The Library Management System is designed with modularity, clarity, and ease of use in mind. Utilizing the **Singleton** pattern for database operations ensures consistent data management throughout the app. The system provides dynamic navigation and user-specific functionalities, catering to both students and librarians through a straightforward web interface. By following this documentation, developers and administrators can understand the architecture, functionalities, and user flows of the system, facilitating maintenance, further development, and effective usage of the application.

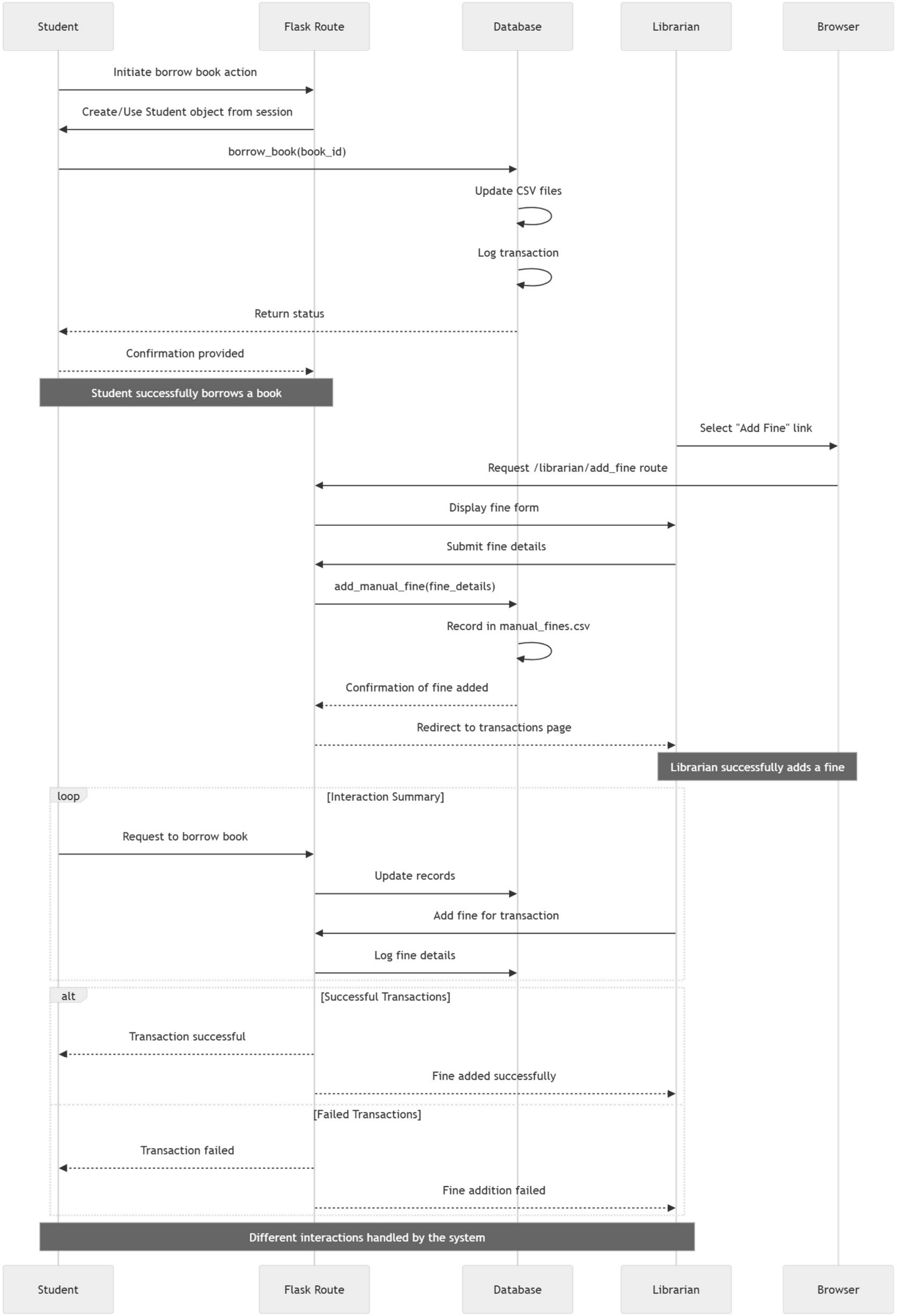
USE CASE DIAGRAM



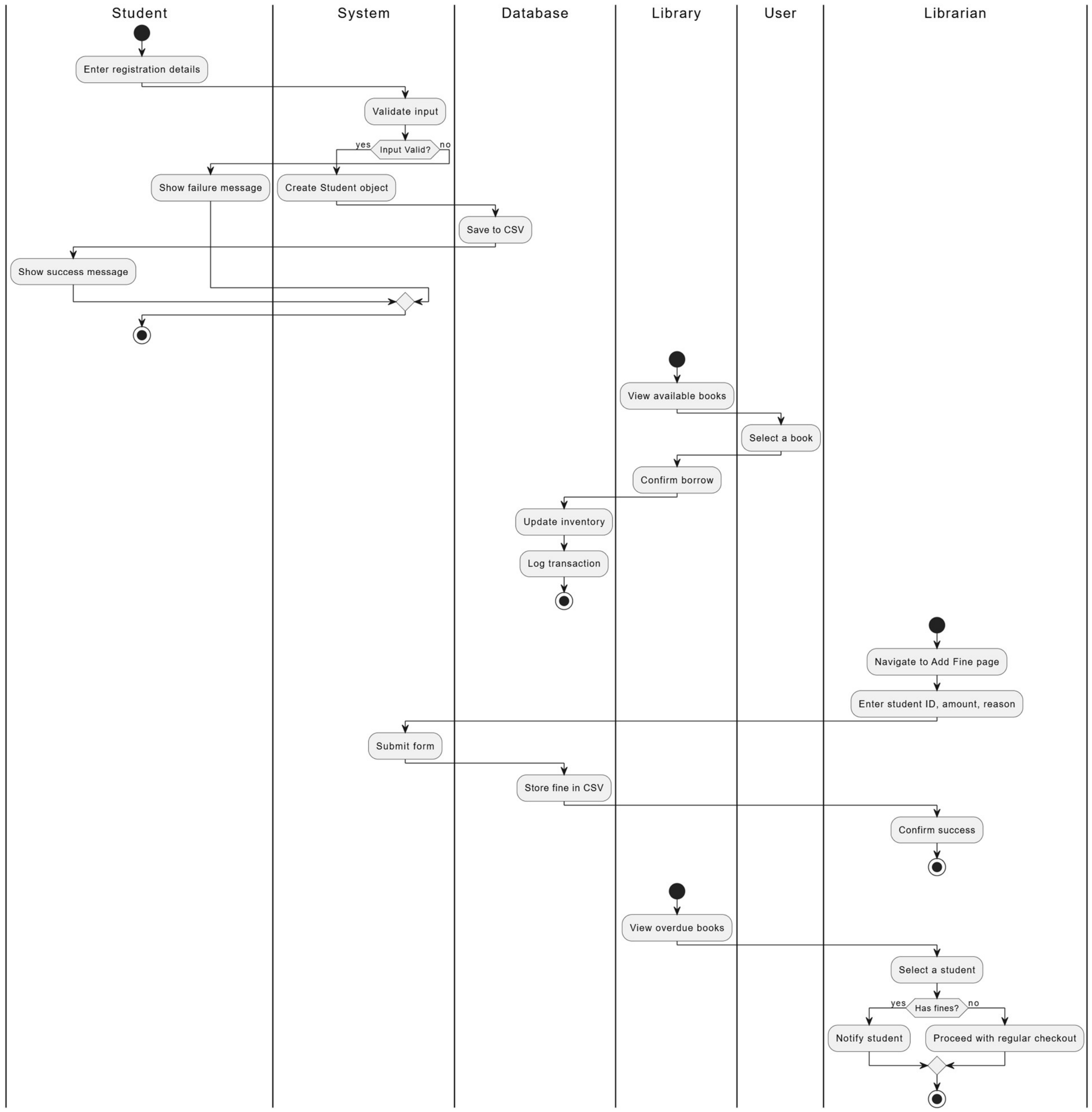
CLASS DIAGRAM



SEQUENCE DIAGRAM



ACTIVITY DIAGRAM


Student
System
Database
Library
User
Librarian
Enter registration details
Validate input
yes
Input Valid?
no
Show failure message
Create Student object
Save to CSV
Show success message
View available books
Select a book
Confirm borrow
Update inventory
Log transaction
Navigate to Add Fine page
Enter student ID, amount, reason
Submit form
Store fine in CSV
Confirm success
View overdue books
Select a student
yes
Has fines?
no
Notify student
Proceed with regular checkout