Library Management System

Course: CS4347.008 Database Systems

Group: Phosphorus (Oluwawemimo J Jayeoba, Gleb Klepko, Nicole Nnenna Urum Eke, Pavan Arani, Vikyatha Komandla)

Project: Library Management System (Milestone 2 & 3)

This application is a Library Management System designed for librarians to manage books, borrowers, loans, and fines. It features a backend logic layer connecting to a MySQL database (for Milestone #2) and a PyQt6 graphical user interface (for Milestone #3).

1. System Requirements

- **Programming Language:** Python 3.10 or higher.
- **Database:** MySQL Server 8.0 or higher.
- **Operating System:** Windows/macOS/Linux.

2. Dependencies & Third-Party Modules

The application relies on the following third-party Python packages:

- `mysql-connector-python` (v8.2.0): MySQL database driver.
- `python-dotenv` (v1.0.0): Environment variable management.
- `PyQt6` (v6.6.1): GUI Framework.

Installation

From the project root directory, install the dependencies using pip:

```
pip install -r requirements.txt
```

3. Database Configuration

Before running the application, you must set up the MySQL database.

Step 1: Initialize the Schema

1. Open your MySQL client
2. Create a database named `LIBMS` (or use the provided `Libms_schema.sql` file if available in your artifacts).

3. Ensure the tables `BOOK`, `AUTHOR`, `BOOK_AUTHOR`, `BORROWER`, `LOAN`, and `FINE` exist according to the project schema.

Step 2: Environment Variables

Create a file named `.env` in the **root** directory of the project. Add your MySQL credentials to this file:

```
MYSQL_HOST=localhost
MYSQL_USER=your_username
MYSQL_PASS=your_password
MYSQL_DB=LIBMS
```

Step 3: Import Data (Optional)

If you wish to populate the database with the provided CSV data, use the scripts in the `normalization` folder.

1. Navigate to the project root.
2. Run the import script:

```
python normalization/scripts/import-to-mysql.py
```

(Follow the interactive prompts instruction to select CSV files and target tables)

4. How to Run the Application

The project is structured to run as a package. **All commands must be run from the project root directory.**

Milestone 2: Backend Logic (Command Line Interface)

You can test specific service modules directly via the command line to verify management logic.

(If python command does not work, try python3)

1. Book Search

Search for books by Title, ISBN, or Author.

```
# Syntax: python -m app.services.book_search "<query>"
python -m app.services.book_search "William"
```

2. Borrower Management

Create new borrowers or search for existing ones.

```
# Create a new borrower
# Syntax: create <Name> <SSN> <Address> [Fname] [Lname] [Email] [Phone]
python -m app.services.borrower_manager create "John Doe" "123-45-6789" "123 Main St"

# Search for a borrower
python -m app.services.borrower_manager search "John"
```

3. Loan Management

Checkout and check-in books.

```
# Checkout a book
# Syntax: checkout <ISBN> <CARD_ID>
python -m app.services.loan_manager checkout 0440234743 ID000001

# Use Search function to find LOAN_ID
# Syntax: python -m app.services.loan_manager search <CARD_ID>/<ISBN>/<Name>
python -m app.services.loan_manager search ID000001

# Check-in a book
# Syntax: checkin <LOAN_ID>
python -m app.services.loan_manager checkin 1
```

4. Fines Management

Update daily fines and view unpaid reports.

```
# Update/Calculate fines (Simulates daily batch process)
python -m app.services.fine update

# View report of all unpaid fines
python -m app.services.fine view-unpaid
```

Milestone 3: Graphical User Interface (GUI)

This is the main entry point for the application.

```
python libms.py
```

5. Project Directory Structure

```
project_root/
├── .env                    <-- Database credentials
├── libms.py                <-- GUI Entry point
├── requirements.txt        <-- Python dependencies
├── app/                    <-- Application base Folder
│   ├── db/
│   │   └── database.py     <-- DB Connection logic
│   ├── services/
│   │   ├── book_search.py
│   │   ├── borrower_manager.py
│   │   ├── fine.py
│   │   └── loan_manager.py
│   └── ui/
│       └── gui.py          <-- PyQt6 Interface
└── normalization/          <-- Data cleaning, SQL schema and import scripts
```

6. Troubleshooting

- **'python' command not found:** If running `python` yields an error or command not found, try using `python3` instead (this is common on macOS and Linux systems).
- **ModuleNotFoundError:** Ensure you are running commands from the project **root** directory and using `python -m app.services...` for CLI tools so that imports resolve correctly.
- **Database Connection Failed:** Double-check the `.env` file exists and contains the correct credentials for your local MySQL server.