# Library Management System

**Course:** CS4347.008 Database Systems

**Project:** Library Management System Milestone #3

This document provides instructions for building, configuring, and running the Library Management System GUI. The application is built using **Python** (PyQt6) and interacts with a **MySQL** backend.

## 1. System Environment

- **Programming Language:** Python 3.10 or higher.
- **Database:** MySQL Server 8.0 or higher.
- **Operating System:** Windows, macOS, or Linux.
- **Interpreter:** Standard CPython interpreter.

## 2. Dependencies

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

- `mysql-connector-python` (v8.2.0): Database connectivity.
- `python-dotenv` (v1.0.0): Environment configuration.
- `PyQt6` (v6.6.1): GUI framework.

### Installation

Open your terminal in the **project root directory** and run:

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

## 3. Database Configuration

### Step 1: Initialize Schema

1. Open your MySQL interface (Workbench, CLI, etc.).
2. Run the provided `Libms_schema.sql` script to create the `LIBMS` database and tables.

### Step 2: Environment Variables

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

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

### Step 3: Import Data (Optional)

To populate the database with the provided CSV data:

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

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

## 4. Running the Application (GUI)

To launch the Graphical User Interface:

1. Ensure you are in the **project root directory** in your terminal.
2. Run the following command:

```
python libms.py
```

*Note: On macOS/Linux, if `python` fails, try `python3 libms.py`.*

## 5. Application Features

- **Books Tab:** Search for books by ISBN, Title, or Author.
    - **Checkout:** Select an available book to check it out.
    - **Check In:** Select a checked-out book to return it.
- **Users Tab:** Search for borrowers.
    - **Create User:** Add new borrowers (Card IDs are auto-generated).
    - **View Fines:** See detailed fine history for a specific user.
- **Fines Menu (Top Bar):**
    - **Update Fines:** Triggers the daily batch process to calculate new fines.
    - **View All Unpaid:** Displays a master list of all borrowers owing money.

# 6. Testing the Fines Functionality

Fines are calculated based on time passing. Since we cannot wait 15 days during a demo, you must **manipulate the database** to simulate an overdue book.

**Step 1: Checkout a Book** Use the GUI to search for a book and check it out to a user.

**Step 2: Find the Loan ID** In the GUI "Books" tab, search for that book again. Click on it. The details panel on the right will display the **Loan ID**.

**Step 3: Travel Back in Time (SQL)** Open your MySQL client and run this SQL command to force the book to be overdue (change `<LOAN_ID>` to the number you found in Step 2):

```
USE LIBMS;
-- Set the due date to 5 days ago or how ever much you want
UPDATE LOAN
SET Date_due = DATE_SUB(CURDATE(), INTERVAL 5 DAY)
WHERE Loan_id = <LOAN_ID>;
```

**Step 4: Update & Verify Fines (GUI)**

1. Return to the GUI.
2. In the top menu bar, click **Fines** -> **Update Fines**.
3. A popup will confirm how many fines were created/updated.
4. Go to **Fines** -> **View All Unpaid Fines** to see the debt registered against the borrower.

# 7. Troubleshooting

- **`zsh: command not found: python`** : On macOS, Python 3 is often mapped to `python3`. Try running:

    ```
    python3 libms.py
    ```

- **`ModuleNotFoundError: No module named 'app'`** : You are likely running the script from inside the `app/` folder. You **must** run the command from the main **project root** folder so Python understands the directory structure.
- **Database Connection Failed**: Ensure your `.env` file is in the root directory and contains the correct password for your local MySQL server.