Python-Based Library Management System

---

1. **User Interface:**
   - **Command-Line Interface (CLI):**
     - Users should interact with the system through a CLI.
     - The interface should provide clear instructions and prompts to guide users through various operations.
     - Implement menus for different functionalities (e.g., Book Management, Member Management).

2. **User Authentication:**
   - **Registration:**
     - Users should be able to register by providing a username and password.
     - Passwords should be stored securely (consider using hashing techniques).
   - **Login:**
     - Users should log in using their registered credentials.
     - Implement a basic session mechanism to maintain user login status during the session.
   - **Roles:**
     - Define roles such as `Admin` and `Member`.
     - Admins have full access, while members have limited access (e.g., only borrowing books).

3. **Book Management:**
   - **Add New Books:**
     - Admins should be able to add new books to the library.
     - Required details: title, author, genre, ISBN, number of copies.
   - **Update Book Details:**
     - Admins can update details of existing books (e.g., change the number of copies, update title or author).
   - **Delete Books:**
     - Admins can delete a book from the library if it is no longer available.
   - **Search Books:**
     - Implement search functionality where users can search for books by title, author, genre, or ISBN.
     - Display relevant book details in search results.

4. **Member Management:**
   - **Add New Members:**
     - Admins should be able to add new members to the library system.
     - Required details: member name, contact information, membership ID.
   - **Update Member Information:**
     - Admins can update member details (e.g., change contact information).
   - **Delete Members:**
     - Admins can delete a member from the system (e.g., if membership is canceled).

5. **Borrowing and Returning Books:**

- **Borrowing Books:**
  - Members can borrow books from the library.
  - Track borrowed books, including member ID, book ID, borrow date, and due date.
  - Limit the number of books a member can borrow at one time (e.g., 3 books).
- **Returning Books:**
  - Members can return borrowed books.
  - Update the system to mark the book as returned.
  - Calculate and display any late fees (if applicable).

6. **Data Structures and Storage:**
   - **Data Models:**
     - Use classes to represent core entities like `Book`, `Member`, `Library`, and `Transaction`.
     - Each class should include appropriate attributes and methods for the operations they need to perform.
   - **Data Storage:**
     - Use SQLite for persistent storage of data (books, members, transactions).
     - Implement CRUD operations for all entities using SQL queries.

7. **File Handling:**
   - **Export Data:**
     - Implement functionality to export library data (books, members, transactions) to CSV files.
   - **Import Data:**
     - Allow admins to import data from CSV files to populate the library database.
     - Validate the imported data to ensure it meets the system's requirements.

8. **Error Handling:**
   - **Input Validation:**
     - Validate user inputs to prevent errors (e.g., ensure ISBN is a valid format, no negative numbers for book copies).
   - **Exception Handling:**
     - Implement try-except blocks to handle unexpected errors, such as database connection issues or invalid file formats.
   - **User-Friendly Messages:**
     - Provide clear and helpful error messages to guide users when something goes wrong.

9. **Reporting and Analytics (Optional):**
   - **Usage Reports:**
     - Generate reports showing the most borrowed books, active members, and overdue books.
   - **Statistical Analysis:**
     - Provide insights like total number of books, number of books borrowed this month, etc.

10. Final Integration:

- **Modular Design:**
  - Organize the code into modules (e.g., authentication module, book management module) for better maintainability.
- **Deployment:**
  - Provide instructions for deploying the system on a local machine.
- **User Documentation:**
  - Write a user manual explaining how to use the system, including screenshots or examples.

Deliverables:

- **Source Code:**
  - Complete and well-documented Python scripts.
- **Database Schema:**
  - SQL scripts for setting up the necessary tables in the SQLite database.
- **Documentation:**
  - A comprehensive README file with setup instructions, usage guidelines, and system architecture.
- **Demonstration:**
  - A video or presentation showcasing the main features and workflow of the Library Management System.