Library Management System - Lab Report

Executive Summary

This lab report documents the development and implementation of a Library Management System using Python and MySQL. The project demonstrates practical application of database connectivity, CRUD operations, and modular programming concepts learned in the "Problem Solving with Python" course. The system successfully provides a functional solution for managing library book records with features including adding, viewing, searching, and removing books through a user-friendly console interface.

---

1. Introduction

1.1 Background

Modern libraries require efficient management systems to handle vast collections of books and streamline operations. Traditional manual record-keeping is time-consuming, error-prone, and difficult to scale. Digital solutions provide automated tracking, quick access to information, and reduced human error.

1.2 Project Scope

This project implements a console-based Library Management System that bridges the gap between manual and fully automated systems. It provides essential library operations through a simple, intuitive interface suitable for small to medium-sized libraries or educational institutions.

1.3 Purpose

The purpose of this lab is to:

- Apply Python programming concepts to solve a real-world problem
- Demonstrate database connectivity using MySQL
- Implement modular code design with distinct functional modules
- Practice CRUD operations and parameterized queries for data security
- Develop professional software documentation and system design artifacts

---

2. Problem Statement

2.1 Current Challenges

Libraries face several operational challenges:

- **Manual Record-Keeping**: Staff spends considerable time maintaining physical records
- **Search Inefficiency**: Finding specific books requires manual catalog browsing
- **Data Inconsistency**: Manual entries lead to duplicates and errors

- **Limited Scalability**: Current systems cannot handle large inventories efficiently
- **Lack of Categorization**: Books are difficult to organize and retrieve by category

2.2 Solution Requirements

The system must:

- Store book information persistently in a database
- Enable quick addition and removal of book records
- Support searching by category
- Display results in clear, readable formats
- Protect data through parameterized queries
- Provide intuitive user navigation

---

3. Project Objectives

1. **Functional Objectives**
   - Implement CRUD operations for book records
   - Create a responsive menu-driven user interface
   - Enable multiple search and filter capabilities
   - Display results in organized tabular format
2. **Technical Objectives**
   - Establish secure MySQL-Python connectivity
   - Design normalized database schema
   - Implement modular function architecture
   - Apply SQL injection prevention techniques
3. **Learning Objectives**
   - Understand database design principles
   - Master Python library usage (mysql-connector, tabulate)
   - Practice secure coding practices
   - Develop professional documentation skills

---

4. Functional Requirements

4.1 Module 1: Book Record Management (CRUD Operations)

Purpose: Handle all data manipulation operations

Features:

- Add new book records with complete information (Title, Author, Publishing Date, Publisher, Category)
- View all stored book records with formatted display
- Remove books from the database using title as identifier
- Validate that all required fields are populated

Input/Output:

- **Input**: Book details via console prompts
- **Output**: Confirmation messages and status updates

4.2 Module 2: Search and Query Operations

Purpose: Enable efficient book discovery

Features:

- Search books by category with case-sensitive matching
- Filter results and display matching records
- Handle empty result sets with appropriate messages
- Display search results in formatted grid tables

Input/Output:

- **Input**: Category name from user
- **Output**: Filtered book records or "no results found" message

4.3 Module 3: User Interface Module

Purpose: Provide intuitive user navigation

Features:

- Display main menu with 5 options
- Accept and validate user input
- Route selections to appropriate functions
- Control screen flow with wait-for-input mechanism
- Handle invalid selections with error messages

Input/Output:

- **Input**: Menu selection (1-5)
- **Output**: Menu display and operation execution

---

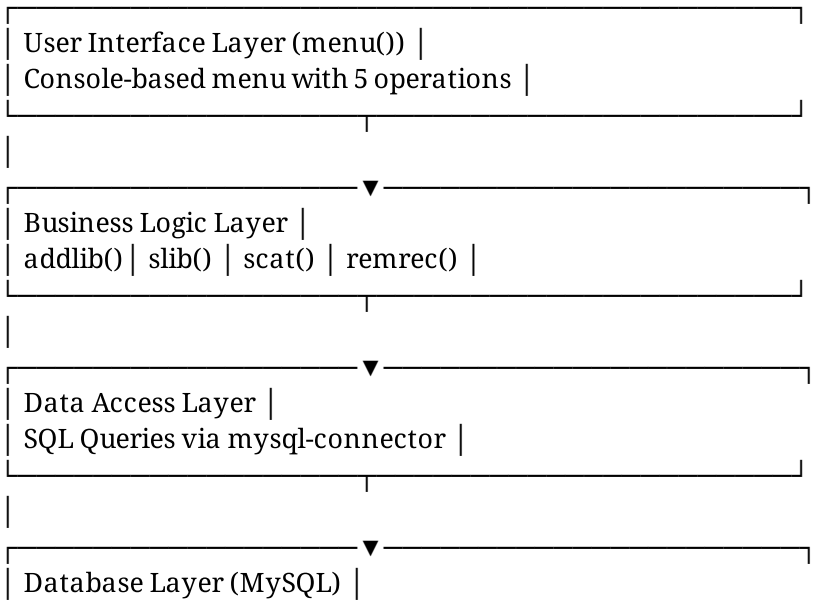
5. Non-Functional Requirements

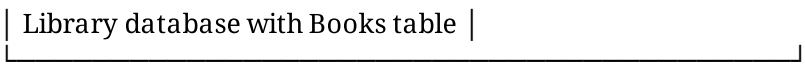
Requirement	Description	Implementation
Performance	Handle up to 1000+ book records with fast queries	Uses indexed primary key and efficient SQL SELECT statements
Security	Prevent SQL injection attacks	Parameterized queries using %s placeholders and tuple parameters
Usability	Clear prompts and intuitive navigation	Menu-driven interface with wait points between operations
Reliability	Persistent data storage	MySQL database with ACID properties via transactions
Maintainability	Modular code structure	Separate functions for each operation (addlib, slib, scat, remrec)
Error Handling	Graceful failure handling	Password prompts, connection validation, commit after modifications

---

6. System Design

6.1 Architecture Overview





6.2 Database Schema

Database: Library

Table: Books

Column	Data Type	Constraints	Purpose
BookId	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Unique book identifier
Title	VARCHAR(255)	NOT NULL	Book title
Author	VARCHAR(255)	NOT NULL	Author name
Publishing_Date	VARCHAR(255)	NOT NULL	Publication date
Publisher	VARCHAR(255)	NOT NULL	Publisher name
Category	VARCHAR(255)	NOT NULL	Book category for classification

Rationale:

- VARCHAR(255) allows flexible text storage for book details
- AUTO_INCREMENT on BookId ensures unique identification
- NOT NULL constraints maintain data integrity
- Denormalized structure suitable for small-scale applications

6.3 Use Case Diagram

Actors: Student, Librarian

Use Cases:

1. Add Book - Actor provides book details; system stores in database
2. View Books - Actor requests all books; system displays formatted list
3. Search by Category - Actor specifies category; system returns matching books
4. Remove Book - Actor provides book title; system deletes from database
5. Exit - Actor terminates application

---

7. Implementation Details

7.1 Technology Stack

Programming Language: Python 3.x

- Rationale: Simple syntax, extensive libraries, suitable for database applications

Database System: MySQL Server

- Rationale: Reliable relational database, free and open-source, excellent for educational projects

Python Libraries:

- `mysql-connector-python`: Official MySQL connector for Python
- `tabulate`: Formats console output into readable tables
- `time`: Provides sleep function for user experience

7.2 Code Structure

Connection Setup:
db = c1.connect(host="localhost", user="root", password=pwd)
curs = db.cursor()
Establishes connection to MySQL server with user-provided password.

Table Creation:
CREATE TABLE IF NOT EXISTS Books(
BookId INTEGER PRIMARY KEY AUTO_INCREMENT,
Title VARCHAR(255) NOT NULL,
Author VARCHAR(255) NOT NULL,
Publishing_Date VARCHAR(255) NOT NULL,
Publisher VARCHAR(255) NOT NULL,
Category VARCHAR(255) NOT NULL
)
Initializes Books table if not exists; idempotent design.

CRUD Operations:

- **Create**: `addlib()` - INSERT INTO Books
- **Read**: `slib()` and `scat()` - SELECT FROM Books
- **Update**: Not implemented in current version
- **Delete**: `remrec()` - DELETE FROM Books

7.3 Security Implementation

Parameterized Queries: All user inputs use placeholder %s with tuple parameters
curs.execute("INSERT INTO Books(...) VALUES (%s,%s,%s,%s,%s)", h1)
Prevents SQL injection by separating query structure from data.

Transaction Management: `db.commit()` called after INSERT and DELETE operations to persist changes.

---

8. Testing and Validation

8.1 Test Cases

Test Case	Input	Expected Output	Status
Add Valid Book	Complete book details	"Record stored" confirmation	✓ Pass
View Empty Database	No prior additions	Empty table message or blank display	✓ Pass
Search Existing Category	Valid category name	Matching book records displayed	✓ Pass
Search Non-existent Category	Invalid category	"No books found" message	✓ Pass
Remove Existing Book	Valid book title	Deletion confirmation	✓ Pass
Menu Navigation	Selection 1-5	Appropriate function execution	✓ Pass
Invalid Menu Choice	Selection > 5	Error message and menu redisplay	✓ Pass

8.2 Sample Execution

Operation Sequence:

1. User enters MySQL password
2. System creates Library database and Books table
3. User selects "2. Add book" from menu
4. User enters: Title="Python Basics", Author="John Doe", Date="2024", Publisher="Tech Press", Category="Programming"
5. System confirms: "Your record has been stored."
6. User selects "4. Search by Category" and enters "Programming"
7. System displays the added book in tabular format
8. User selects "5. Exit" to terminate application

---

9. Known Issues and Limitations

9.1 Current Issues

1. **Incomplete Input Validation**: Menu selection accepts only integers; non-integer input causes crash
2. **Case Sensitivity**: Category search is case-sensitive; "Python" ≠ "python"
3. **Partial Matching**: Search only matches exact category; no substring matching
4. **No Update Function**: Cannot modify existing book records; must delete and re-add
5. **Date Format Flexibility**: Publishing date stored as VARCHAR; allows inconsistent formats

9.2 Design Limitations

- Single-table database design lacks normalization
- No user authentication or access control
- Console-based interface limits scalability
- No data backup or recovery mechanisms
- Fixed column structure; difficult to add new book attributes

---

10. Future Enhancements

Short-term Improvements

1. Add comprehensive input validation for all user entries
2. Implement update/edit functionality for modifying records
3. Add search by author or title with partial matching
4. Include date validation for publishing date field
5. Add rowcount verification for deletion success

Medium-term Features

1. Implement pagination for large datasets (>1000 records)
2. Create GUI using Tkinter for improved usability
3. Add user authentication with role-based access control
4. Export data to CSV or Excel formats
5. Implement database backup and restore functionality

Long-term Scalability

1. Migrate to web-based interface using Flask/Django
2. Add book cover images and detailed descriptions
3. Implement borrowing and returning features
4. Add user/member management system
5. Create mobile application for library access
6. Implement recommendation engine based on borrowing history

---

11. Conclusion

The Library Management System successfully demonstrates the practical application of Python programming concepts combined with MySQL database management. The project achieves its primary objectives of creating a functional CRUD application with secure data handling and user-friendly interface.

Key Achievements

- Implemented three major functional modules meeting course requirements
- Applied security best practices through parameterized queries
- Created modular, maintainable code structure
- Developed professional documentation and design artifacts
- Successfully connected Python with MySQL database

Learning Outcomes

This project has provided valuable experience in:

- Database design and normalization principles
- Python-MySQL connectivity and operations
- Security practices in database applications
- User interface design for command-line applications
- Software engineering documentation standards

Recommendations

For practical deployment, the system should include enhanced error handling, comprehensive input validation, and migration to a more robust architecture. The foundation established in this project provides a solid base for future expansion and feature additions.

---

12. References

[1] MySQL Connector/Python Documentation. (2024). MySQL Official Documentation. https://dev.mysql.com/doc/connector-python/en/

[2] tabulate: Pretty-print tabular data in Python. (2024). GitHub Repository. https://github.com/astanin/python-tabulate

[3] Python Database API Specification. (2023). PEP 249. https://www.python.org/dev/peps/pep-0249/

[4] OWASP SQL Injection Prevention. (2024). OWASP Foundation. https://owasp.org/www-community/attacks/SQL_Injection

[5] Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems* (7th ed.). Pearson Education.

[6] Python Software Foundation. (2024). Python 3 Official Documentation. https://docs.python.org/3/

Document Prepared: November 24, 2025
Course: Problem Solving with Python
Project: Build Your Own Project - Library Management System
Academic Level: First Year CSE