

## Mini Project Report

*Title: Library Management System*

*Submitted by: Devamritha Santhosh*

### *Abstract*

*The Library Management System is a Python-based mini project designed to efficiently manage book records in a library. It enables users to add, update, delete, and search books while maintaining availability information. The system uses SQLite as its database backend to ensure persistent storage and easy management of data.*

### *Objectives*

1. To automate library book management using Python.
2. To store book details securely using SQLite.
3. To provide functions for adding, updating, deleting, and searching books.
4. To maintain book availability status for easy tracking.

### *Software Requirements*

- Python 3.x
- SQLite3 (Database)
- PyCharm

### *Database Schema*

*The project uses an SQLite database named 'library.db' consisting of two tables:*

```
1. books(book_id INTEGER PRIMARY KEY AUTOINCREMENT, title TEXT, author TEXT, category  
TEXT)  
2. availability(id INTEGER PRIMARY KEY AUTOINCREMENT, title TEXT, available TEXT  
DEFAULT Yes)
```

*The following is the complete Python source code for the Library Management System:*

---

```
import sqlite3
def create_tables():

    with sqlite3.connect("library.db") as connection:
        connection.execute("PRAGMA foreign_keys = ON;")
        cursor = connection.cursor()
        cursor.execute("""
            CREATE TABLE IF NOT EXISTS users(
                user_id INTEGER PRIMARY KEY AUTOINCREMENT,
                username TEXT UNIQUE NOT NULL,
                password TEXT NOT NULL,
                role TEXT CHECK(role IN ('admin','user')) NOT NULL
            )
        """)
        cursor.execute("""
            CREATE TABLE IF NOT EXISTS books(
                book_id INTEGER PRIMARY KEY AUTOINCREMENT,
                title TEXT NOT NULL UNIQUE,
                author TEXT NOT NULL,
                category TEXT
            )
        """)
        cursor.execute("""
            CREATE TABLE IF NOT EXISTS availability(
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                title TEXT NOT NULL,
                available TEXT DEFAULT 'Yes',
                FOREIGN KEY (title) REFERENCES books(title)
            )
        """)
        connection.commit()

def register():
    print("\n Register Account ")
    username = input("Enter username: ").strip()
    password = input("Enter password: ").strip()
    role = input("Enter role (admin/user): ").strip().lower()

    if role not in ("admin", "user"):
        print("Invalid role!\n")
        return

    with sqlite3.connect("library.db") as connection:
        connection.execute("PRAGMA foreign_keys = ON;")
        cursor = connection.cursor()
        if role == "admin":
            cursor.execute("SELECT COUNT(*) FROM users WHERE role='admin'")
            count = cursor.fetchone()[0]
            if count >= 2:
                print("2 Maximum of 2 admins allowed. Cannot register more admins!\n")
                print("Ask an existing admin to delete one admin account first.\n")
```

```

return
try:
    cursor.execute("INSERT INTO users (username, password, role) VALUES (?, ?, ?)",
                  (username, password, role))
    connection.commit()
    print("Account created successfully!\n")
except:
    print("Username already exists!\n")

def login():
    print("\n Login ")
    username = input("Enter username: ").strip()
    password = input("Enter password: ").strip()
    with sqlite3.connect("library.db") as connection:
        cursor = connection.cursor()
        cursor.execute("SELECT role FROM users WHERE username=? AND password=?", (username, password))
        user = cursor.fetchone()
    if not user:
        print("Invalid username or password!\n")
        return None, None
    print(f"\nWelcome! You are logged in as {user[0].upper()}.\\n")
    return username, user[0]

def add_book():
    title = input("Enter Book Title: ")
    author = input("Enter Author Name: ")
    category = input("Enter Category: ")
    with sqlite3.connect("library.db") as connection:
        cursor = connection.cursor()
        cursor.execute("INSERT OR IGNORE INTO books (title, author, category) VALUES (?, ?, ?)", (title, author, category))
        cursor.execute("INSERT OR IGNORE INTO availability (title) VALUES (?)", (title,))
        connection.commit()
    print("Book added successfully!\n")

def update_availability():
    title = input("Enter Book Title to Update: ")
    status = input("Enter Availability (Yes/No): ")
    with sqlite3.connect("library.db") as connection:
        cursor = connection.cursor()
        cursor.execute("UPDATE availability SET available=? WHERE title=?", (status, title))
        connection.commit()
    print("Availability updated!\n")

def view_books():
    with sqlite3.connect("library.db") as connection:
        cursor = connection.cursor()
        cursor.execute("""SELECT books.title, books.author, books.category,
                           availability.available FROM books JOIN availability ON books.title=availability.title""")

```

```

cursor = connection.cursor()

if choice == '1':
    term = input("Enter Book Title: ")
    cursor.execute("SELECT * FROM books WHERE title LIKE ?", ('%' + term + '%'))
elif choice == '2':
    term = input("Enter Author Name: ")
    cursor.execute("SELECT * FROM books WHERE author LIKE ?", ('%' + term + '%'))
elif choice == '3':
    term = input("Enter Category: ")
    cursor.execute("SELECT * FROM books WHERE category LIKE ?", ('%' + term + '%'))
else:
    print("Invalid choice!\n")
    return

results = cursor.fetchall()

if results:
    print("\nSearch Results:")
    for book in results:

        cursor.execute("SELECT available FROM availability WHERE title=?", (book[1],))
        status = cursor.fetchone()
        available = status[0] if status else "No"
        print(f"Title: {book[1]} | Author: {book[2]} | Category: {book[3]} | Available: {available}")
        print()
    else:
        print("No books found matching your search.\n")

def delete_admin(current_admin):
    with sqlite3.connect("library.db") as connection:
        cursor = connection.cursor()
        cursor.execute("SELECT username FROM users WHERE role='admin' AND username!=?", (current_admin,))
        admins = cursor.fetchall()
        if not admins:
            print("No other admin available to delete.\n")
            return
        print("Existing Admins:")
        for a in admins:
            print(" ", a[0])
        target = input("Enter admin username to delete: ")
        cursor.execute("DELETE FROM users WHERE username=? AND role='admin'", (target,))
        connection.commit()
        print("Admin deleted successfully!\n")

def admin_menu(username):

```

```

while True:
    print(""\nAdmin Menu
1. Add Book
2. 2. Update Availability
3. 3. View All Books
4. 4. Delete Another Admin
5. 5. Logout""")
    choice = input("Enter choice: ")
    if choice == '1':
        add_book()
    elif choice == '2':
        update_availability()
    elif choice == '3':
        view_books()
    elif choice == '4':
        delete_admin(username)
    elif choice == '5':
        print("Logged out.\n")
        break
    else:
        print("Invalid choice.\n")

def user_menu():
    while True:
        print(""\nUser Menu
1. View All Books
2. 2. Search Books
3. 3. Logout""")
        choice = input("Enter choice: ")
        if choice == '1':
            view_books()
        elif choice == '2':
            search_books()
        elif choice == '3':
            print("Logged out.\n")
            break
        else:
            print("Invalid choice.\n")

def main():
    create_tables()
    while True:
        print(""\n Library System
1. Register
2. 2. Login
3. 3. Exit""")
        choice = input("Enter choice: ")
        if choice == '1':
            register()
        elif choice == '2':
            username, role = login()
            if role == 'admin':
                admin_menu(username)
            elif role == 'user':
                user_menu()
        elif choice == '3':

```

```
        print("Exiting.")  
        break  
    else:  
        print("Invalid choice!\n")  
  
main()
```

```

C:\Users\devan_v\PycharmProjects\PythonProject\venv\Scripts\python.exe C:\Users\devan_v\PycharmProjects\PythonProject\

Library System
1. Register
2. Login
3. Exit
Enter choice: 2

Login
Enter username: anu
Enter password: anu

Welcome You are logged in as ADMIN.

--- Admin Menu ---
1. Add Book
2. Update Availability
3. View All Books
4. Delete Another Admin
5. Logout
Enter choice: 3

Book List
Title: ente sathyashana pareekshnangal | Author: mahatma gandhi | Category: biography | Available: Yes
Title: agni chirakukal | Author: apj abdul kalam | Category: biography | Available: Yes

--- Admin Menu ---
1. Add Book
2. Update Availability
3. View All Books
4. Delete Another Admin
5. Logout
Enter choice: 2

Enter Book Title to Update: ente sathyashana pareekshnangal
Enter Availability (Yes/No): yes
Availability updated!

--- Admin Menu ---
1. Add Book
2. Update Availability
3. View All Books
ject 1 Library

```

```

Library System
1. Register
2. Login
3. Exit
Enter choice: 2

Login
Enter username: deva
Enter password: deva

Welcome You are logged in as USER.

User Menu
1. View All Books
2. Search Books
3. Logout
Enter choice: 2

Search by:
1. Title
2. Author
3. Category
Enter choice: 1
Enter Book Title: ente sathyashana pareekshnangal

Search Results
Title: ente sathyashana pareekshnangal | Author: mahatma gandhi | Category: biography | Available: yes

```

### Explanation

1. The program connects to an SQLite database named 'library.db' and creates two tables if they do not exist.
2. The 'add\_book' function inserts new book details and updates availability.
3. The 'update\_book' function allows modification of author, category, and availability status.
4. The 'delete\_book' function removes a book from both tables.
5. The 'search\_book' function retrieves book details and availability.
6. The 'main' function acts as a menu-driven interface for user interaction.

## Output

When executed, the program displays a menu-driven interface allowing the user to manage books. Each operation prints success messages or error alerts in the console. The database retains all changes permanently.

## Conclusion

The Library Management System successfully automates basic library functions such as adding, updating, deleting, and searching for books. Using SQLite ensures reliable data storage and retrieval with minimal setup.