**DBMS MINI PROJECT**

**Submitted by:**

**SHREE SATHIYAA S.K (312322205164)**

**SIVASANKARI A (312322205170)**

**OF**

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECHNOLOGY**

****

**St. JOSEPH’S COLLEGE OF ENGINEERING**

**(An Autonomous Institution)**

**St. Joseph’s Group of Institutions**

**OMR, Chennai 600 119**

**LIBRARY MANAGEMENT**

ABSTRACT :

The Library Management System is a software application designed to automate

the tasks of a library. It provides a user-friendly interface for librarians to manage the library's collection of books and for patrons to access and borrow books .The system allows librarians to add new books to the library, update existing book information, and remove books that are no longer in circulation. Each book is identified by a unique ID and includes details such as the title, author, and quantity available. Patrons can search for books in the library's catalog, view book details, and borrow books that are available. When a book is borrowed, the system updates the quantity of available copies. Patrons can also return books, which updates the quantity of available copies accordingly. The system includes features for managing user accounts, including adding new users, updating user information, and removing users. It also provides reports and statistics on library usage, such as the number of books borrowed and the most popular books. Overall, the Library Management System is designed to streamline library operations, improve access to library resources, and enhance the overall user experience for both librarians and patrons.

**INTRODUCTION :**

The Library Management System is a Python program designed to automate the management of a library's collection of books. This system provides functionalities for librarians to add, remove, and update book information, as well as for patrons to borrow and return books. It aims to streamline the operations of a library and improve the overall user experience. The system is implemented using object-oriented programming principles, with classes representing the library and books. The main features of the system include adding books to the library, removing books, displaying the library catalog, borrowing books, and returning books. The `Library` class represents the library and contains a dictionary to store book information. It has methods to add a book to the library, remove a book, display the library catalog, borrow a book, and return a book. Each book is identified by a unique ID and includes details such as the title, author, and quantity available. The system also includes an example usage section that demonstrates how to use the `Library` class to manage a library. This section adds books to the library, displays the catalog, borrows books, returns books, and displays the updated catalog after each operation. Overall, the Library Management System provides a simple yet effective solution for managing a library's collection of books using Python programming language. It can be further expanded with additional features such as user management, due date tracking, and more advanced search functionalities.

**CODE :**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Login Page</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div class="login-container">

<h1>Login</h1>

<form action="login.php" method="POST">

<label for="username">Username:</label><br>

<input type="text" id="username" name="username" required><br>

<label for="password">Password:</label><br>

<input type="password" id="password" name="password" required><br>

<button type="submit">Login</button>

</form>

</div>

</body>

</html>

from flask import Flask, request, render\_template, jsonify

import sqlite3

app = Flask(\_name\_)

DATABASE = 'database.db'

def init\_db():

with sqlite3.connect(DATABASE) as db:

cursor = db.cursor()

cursor.execute('''

CREATE TABLE IF NOT EXISTS users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

email TEXT NOT NULL

)

''')

db.commit()

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/users', methods=['GET', 'POST'])

def users():

if request.method == 'GET':

print("GET request received for /users")

db = get\_db()

cursor = db.execute('SELECT \* FROM users')

users = cursor.fetchall()

db.close()

return jsonify(users)

elif request.method == 'POST':

data = request.json

name = data.get('name')

email = data.get('email')

print("POST request received for /users")

print(f"Name: {name}, Email: {email}")

db = get\_db()

db.execute('INSERT INTO users (name, email) VALUES (?, ?)', (name, email))

db.commit()

db.close()

return jsonify({'message': 'User added successfully'})

def get\_db():

db = sqlite3.connect(DATABASE)

return db

if \_name\_ == '\_main\_':

init\_db() # Initialize the database

app.run(debug=True)

class Library:

def \_\_init\_\_(self, name):

self.name = name

self.books = {}

def add\_book(self, book\_id, title, author, quantity):

if book\_id in self.books:

self.books[book\_id]['quantity'] += quantity

else:

self.books[book\_id] = {'title': title, 'author': author, 'quantity': quantity}

def remove\_book(self, book\_id, quantity=1):

if book\_id in self.books:

if self.books[book\_id]['quantity'] <= quantity:

del self.books[book\_id]

else:

self.books[book\_id]['quantity'] -= quantity

else:

print("Book not found.")

def display\_books(self):

print("\nLibrary Catalog:")

for book\_id, details in self.books.items():

print(f"ID: {book\_id}, Title: {details['title']}, Author: {details['author']}, Quantity: {details['quantity']}")

def borrow\_book(self, book\_id):

if book\_id in self.books:

if self.books[book\_id]['quantity'] > 0:

self.books[book\_id]['quantity'] -= 1

print("Book borrowed successfully.")

else:

print("Book not available.")

else:

print("Book not found.")

def return\_book(self, book\_id):

if book\_id in self.books:

self.books[book\_id]['quantity'] += 1

print("Book returned successfully.")

else:

print("Book not found.")

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

library = Library("My Library")

library.add\_book(1, "Python Programming", "John Doe", 5)

library.add\_book(2, "Java Basics", "Jane Smith", 3)

library.add\_book(3, "Data Science for Beginners", "Alice Johnson", 2)

library.display\_books()

library.borrow\_book(1)

library.borrow\_book(2)

library.borrow\_book(1)

library.display\_books()

library.return\_book(1)

library.display\_books()

**OUTPUT :**

****

Library Catalog:

ID: 1, Title: Python Programming, Author: John Doe, Quantity: 5

ID: 2, Title: Java Basics, Author: Jane Smith, Quantity: 3

ID: 3, Title: Data Science for Beginners, Author: Alice Johnson, Quantity: 2

Book borrowed successfully.

Book borrowed successfully.

Book not available.

Library Catalog:

ID: 1, Title: Python Programming, Author: John Doe, Quantity: 3

ID: 2, Title: Java Basics, Author: Jane Smith, Quantity: 2

ID: 3, Title: Data Science for Beginners, Author: Alice Johnson, Quantity: 2

Book returned successfully.

Library Catalog:

ID: 1, Title: Python Programming, Author: John Doe, Quantity: 4

ID: 2, Title: Java Basics, Author: Jane Smith, Quantity: 2

ID: 3, Title: Data Science for Beginners, Author: Alice Johnson, Quantity: 2

**CONCLUSION :**

In conclusion, the Library Management System implemented in Python provides a robust solution for managing a library's collection of books. The system allows librarians to efficiently add, remove, and update book information, as well as track book availability and borrower information. For patrons, the system provides a user-friendly interface to search for books, view book details, and borrow or return books. The system helps streamline library operations, improve access to library resources, and enhance the overall user experience. Future enhancements to the system could include adding features such as user authentication, fine management for overdue books, reservation system for popular books, and integration with online databases for book information. Overall, the Library Management System demonstrates the power and versatility of Python in developing practical and efficient solutions for real-world problems.

\*\*\*\*\*\*