"There is no programming language, no matter how structured, that will prevent programmers from making bad programs." Larry Flon

Project
Library Management System

by Kovid Joshi
Project Manager

&

Shikar Joshi
QC/QA and Publisher

DON BOSCO SCHOOL
PITHORAGARH

Introduction

The LMS Project short for the Library management project is a Program written in `python 3.10.7` language that has a extensive catalog of books that is further extendable to a larger library of books. The Program allows the user to browse the books, by Author, Title, year and ISBN number.

This program is written on `python 3.10.7` version of python and uses custom and built-in libraries from the python. The connection through the MySQL database is possible through the `MySQL-connector-python` or `MySQL-connector` modules downloaded through command line.

Acknowledgment

This Project is a combined effect of me and my project partner. We collectively worked and tested the program for bugs and problems, to fix them for the end user. Our collective efforts have made a program that is able work as it claims to be. Further we thank our Teacher for guiding and correcting us. The software we used to make multiple this Project possible have a great contribution. The document itself is made using LaTeX and further python development took place over the Jet Brains PyCharm 2022.2.3 that itself is a sufficient IDLE for its task. Further I want to thank my colleague to review this code for any bugs and errors.And Further using the powerful MySQL database system and its software MySQL workbench to complete the query task

Overall this was a interesting and comprehensive task to make such a project and we are grateful to get such a opportunity.

Kovid Joshi (Project Manager)

Contents

Part I

System and Feasibility

1 System and Factors of Feasibility

1.1 System Analysis

great Library Management Software is a software that helps a library to manage and list the books in their library. This Project is based on such a problem to solve some problems regarding –

- Listing the books
- Adding the books
- Searching the books

The extensive catalog of books around the world requires a powerful and efficient database system that is maintained and updated regularly by the developers, one such system is MySQL that is fast and powerful. With Integration with python to make the most out of it, this Project is focused in the connectivity of the two.

The LMS comprises of

- Cataloging
- Retrieval
- Adding

as one of the core functionality.

1.2 Feasibility Study

Feasibility of the program can be divided into

Social The Library Management project is developed taking care of the usability. Its main objective of this Program is being a usable utility to the Librarians in the world.

Technical Technically the Program is based on a command line interface and is lightweight. Thanks to python this program is OS independent. With some packages and MySQL installed this program must not cause problems while execution

Financial This program is based on Open Source Code and is free to use.

Part II

Source and Program Structure

2 Source Code

2.1 main.py

The main file is the integration of all the libraries and is the file that will be executed when running the program

```python

import lms.sql_util
import lms.menu
# import getpass

i = 0

# ------------Login------------

while i < 3:
    ask_name = input("Enter your name ").title().strip()
    ask_pass = input("Enter your password ")
    check_data = (ask_name, ask_pass)
    # ---passwords retrieval
    if lms.sql_util.pass_checker(check_data) is False:
        print(" Invalid user, wrong password or name\nplease try again or register as a new user")
        i += 1
        print(f"you have {3 - i if 3 - i != 0 else exit()} tries")
        lms.sql_util.logit(message='Login Failed')
    else:
        break


lms.sql_util.logit(message='Logged in!')

# Body of the program
# noinspection PyUnboundLocalVariable
lms.menu.menu(user=ask_name)
while True:
    ask_option = input(" ==> ").strip().casefold()

    if ask_option in ['browse', '1']:
        # display all the isbn details and the books by them
        lms.sql_util.display(table_name='books')
        lms.sql_util.logit('displaying the books')

    elif ask_option in ['search', 'find', '2']:
        search_options = input("""
        SEARCH mode
        search by -- ISBN(isbn), author(author) or name(name)
        -> """).strip().casefold()

        if search_options in ['isbn', '1']:
            # searching the book using the books ISBN
            ask_isbn = input("Enter the ISBN number of the book ")
            # filtering the input
            if ask_isbn.isnumeric():
                lms.sql_util.search_on_isbn(ask_isbn)
            else:
                print("please enter a valid ISBN number")
            lms.sql_util.logit('searching for a book by its ISBN')

        elif search_options in ['author', '2']:
            # searching using the author name
            ask_author = input("Enter the author to search ").title().strip()

            lms.sql_util.search_on_author(ask_author)
            lms.sql_util.logit("Searching on the basis of author ")

        elif search_options in ['name', 'book name', '3']:
            # searching using the books name
            ask_title = input("Enter the Title of the book ").strip()

            lms.sql_util.search_on_title(ask_title)
            lms.sql_util.logit("searching for a book by title")

    elif ask_option in ['add', 'contribute', 'add books']:
        # adding the books by the user as a contribution
        print("To Add books you have to verify that it's you!")
        verify_user = input("Please enter your name ").strip().title()
        verify_pass = input("verify your password ")
        # using the add_books function of the  sql_util package to
        lms.sql_util.add_books((verify_user, verify_pass))
        lms.sql_util.logit('Adding to the database')

    elif ask_option in ['menu', 'options']:
        # menu
        lms.menu.menu()

    elif ask_option in ['help', 'save me']:
        # help regarding options
        lms.menu.helpme()

    elif ask_option in ['explore', '4']:
        # explore for the library books

        lms.sql_util.explore()

    elif ask_option in ['exit', 'quit', '5', 'close']:
        print("Exiting the program")
        lms.sql_util.logit("Exiting the program ")
        exit()

    elif ask_option in ['version']:
        lms.menu.version()

    else:
        print("I don't recognise that need help type help or menu")

#  using the logit function from lms.log  print(ls.logit("resenting
    by the user"))
```

main.py

2.2 SQL utility

This file is used for the utilities in the SQL database and stores a majority of functions

```
"""
mysql user credentials
"""
import os
import yaml
import mysql.connector
import random
import secrets
import json
import time
import string


USER_TABLE = 'lms_users'
BOOKS_TABLE = 'books'
DEBUG_TABLE = 'test_books'
ISSUE_TABLE = 'issue_list'


def main_cnx(user_id='user'):
    """
    function that returns the login connection using the
    cnx_data.yml file
    """
    # changing to the data directory
    if os.path.exists('cnx_data.yml') is False:
        # os.chdir('..')
        os.chdir('data')
    with open('cnx_data.yml') as data_file:
        data = yaml.load(data_file, yaml.SafeLoader)

    cnx = mysql.connector.connect(**data[user_id])
    return cnx


def pass_checker(user_data):
    """
    checking the user input to the registered users
    in the database
    :return: boolean value
    """
    # starting the defined connection using the main_cnx() function
    cnx = main_cnx()

    cursor = cnx.cursor()
    # executing the command using execute statement

    cursor.execute(f'select * from {USER_TABLE}')
    # getting the data in the desired form
    database_data = cursor.fetchall()

    # checking the database from the file data
    if user_data in database_data:
        return True
    else:
        return False


def display(table_name='books'):
    """
    show the books, isbn author from the database
    :param table_name:
    :return:
    """
    # initiating the connection
    cnx = main_cnx()
    cursor = cnx.cursor()

    # executing the sql statement for the data
    cursor.execute(f"select * from {table_name}")

    # printing the data form stored in the cursor
    for lines in cursor:
        print(f'{lines[0]:14} {lines[1]:45}by {lines[2]}')


def search_on_isbn(isbn_number: str):
    """
    searching using the isbn of the book
    :return:
    """
    cnx = main_cnx()
    cursor = cnx.cursor()
    if isbn_number.isnumeric():
        cursor.execute(f"select * from {BOOKS_TABLE} where isbn = {
isbn_number!r}")
        # fetching the data from the database
        data = cursor.fetchall()
        # checking for empty data
        if not data:
            print(f"Sorry no book is found having ISBN {isbn_number}"
)
        else:
            print('Found')
            print(data)
    else:
        print("Please enter a number to search")


def search_on_author(author_name: str):
    """
    searching function using the author name
    :return:
    """

    cnx = main_cnx()
    cursor = cnx.cursor()
    cursor.execute(f"SELECT book_name, published from {BOOKS_TABLE}
where author = {author_name!r}")
    data = cursor.fetchall()
    # printing the data retrieved from database
    # listing of the all the books from the author
    if data:
        print(f"Books by {author_name}")
        print(f"Title {'-'*35}Publishing date")
        for books in data:
            print(f"{books[0]:40} {books[1]:5}")
    else:
        print(f"Author {author_name!r} not found\nPlease check for
```

```python
any typos in the author name and try again")


def search_on_title(book_name: str):
    """
    searching the books in the database using the
    :param book_name:
    :return:
    """

    cnx = main_cnx()
    cursor = cnx.cursor()
    cursor.execute(f"SELECT book_name, published, author from {BOOKS_TABLE} where book_name like {book_name+'%'!r}")
    data = cursor.fetchall()
    if data:
        print("Found")
        for books in data:
            print(f"{books[0]:40} {books[1]}, by {books[2]}")

        return True
    else:
        print(f"Not Found with title {book_name!r}")
        return False


def add_books(verify_user):
    """
    Adding the books by the user as a contribution to the project database
    helping it to grow to a more vast book library
    :param verify_user:
    :return:
    """
    if pass_checker(verify_user) is False:
        print("Sorry the credentials are wrong")
    else:
        cnx = main_cnx()
        # making the cursor
        cursor = cnx.cursor()
        # asking the details of the books by the valid user
        while True:
            try:
                print("Enter the following details of the book exit to leave \n")
                ask_isbn = input("Enter the isbn number ").strip().casefold()
                if ask_isbn in ['exit', 'quit']:
                    break
                ask_book_name = input("Enter the book name ").strip()
                ask_author = input(f"Enter the Author of the book {ask_book_name!r} ").title().strip()
                ask_year = input("Enter the year of publishing ")
                # if no exception occurs break the loop
                # ------tmp------##
                cursor.execute(f"insert into {DEBUG_TABLE} values ({ask_isbn!r}, {ask_book_name!r}, {ask_author!r},"
                               f" {ask_year})")
                # executing the changes to the table
                cnx.commit()
                print("*Successfully* added the book to the library thanks for the contribution \n"
                      "help this project to grow.\n")

            except (mysql.connector.errors.DatabaseError, mysql.connector.errors.InterfaceError):
                print(f" {'*'*9}SORRY! there was an error, sorry for the inconvenience {'*'*9}")
                print(f"{'*'*9}Please enter a number value for the publishing year{'*'*9}")


def explore():
    """
    exploring the data
    :return:
    """

    cnx = main_cnx()

    cursor = cnx.cursor()

    # getting data for the author
    cursor.execute(f"select author from {BOOKS_TABLE}")
    author = cursor.fetchall()

    # getting the number of books in the database
    cursor.execute(f'select count(*) from {BOOKS_TABLE}')
    times = cursor.fetchall()

    # getting the old books in database
    cursor.execute(f'select book_name, author from {BOOKS_TABLE} where published < 2000 ')
    old = cursor.fetchall()

    # processing the retried values
    classic_time = random.randint(0, len(old) - 1)
    random_author = author[random.randint(0, len(author) - 1)][0]
```

~~~python
    classic_book = old[classic_time][0]
    classic_author = old[classic_time][1]
    total_books = times[0][0]

    print(fr"""
    +{'-' * 30}LIBRARY MANAGEMENT SYSTEM{'-' * 30}+
    |{" "*85}|
    |    Read `By Authors like{" "*61}|
    |    {random_author}{" "*(91 - (8 + 1 + len(random_author)))}|
    |    ```````  Total books in library {total_books} ```````{" "*(91- (49+len(str(total_books))))}|
    |    ~Time less classics{" "*63}|
    |    {classic_book}      by' {classic_author}{" "*(91 - (17+1+len(classic_author)+len(classic_book)))}|
    |{" "*85}|
    +{'-' * 30}{'*' * 25}{'-' * 30}+
        """)


def logit(message=''):
    """
    logging the events happened in the LMS
    :param message:
    :return:
    """

    if os.path.exists('logfile.log') is False:
        with open('logfile.log', 'x') as new_file:
            pass

    number_id = ''.join(secrets.choice(string.digits) for _ in range(5))
    log_data = [time.asctime(time.localtime()), number_id, message]

    with open('logfile.log', 'a') as log_file:
        json.dump(log_data, log_file)
        log_file.write('\n')

    return number_id
~~~

sql_util.py

2.3 Menu and Help

Menu file stores the menus and helps

~~~python
"""
menu, options and help for the file:main.py
~~~

```
"""


def menu(user=''):
    print(f"""
   +{'-'*60}+
   |              Library Management System                    |
   | Hi {user}{" "*(65-(1+8+len(user)))}|
   |     1.Browse books (browse)                               |
   |     2.Search for the book (find)                          |
   |     3.Add Books (add)                                     |
   |     4.Explore (explore)                                   |
   |     5.exit (exit)                                         |
   +{'-'*60}+
   | For help enter help, for version information enter version |
   +{'-'*60}+
    """)


def helpme():
    print("""
    USER HELP

    *browse*
    Browse helps the user to browse the extensive catalog of books
   from
    the LMS database.

    Search
    search comprises of the multiple type of search in the books
   database
    this options has 3 sub options inside it
        1.ISBN search
        2.Author search
        3.Search by Title of the Book

    *add*
    Add is a option for people who want to add data to the database
   for making
    new books in the library catalog

    *help*
    gets you here

    *explore*
    get the some great recommendations from the some of the best
   authors and books
    in the library

```

```
  for version type version
    """)


def version():
    print("""
    version information '0.5' 'Bloodymary'
    """)
```

menu.py

3 Data Flow of the Program

Data flow diagram for the program is –

3.1 System Design

3.2 SQL Database Structure

Database structure can be described as

3.3 Program Dependency tree

Part III
Post Updates

4 Future Updates

The following program like the rest of the programs are not prefect. The following program can be improved in feature and security.

- This program is vulnerable to a SQL injection where a hacker can inject a SQL to alter, delete, view and do all sorts of things with the SQL database. The solution of this problem is that the given program takes a filtered input of the things from the users side.
- The program can be made online rather than running the SQL locally by setting up a server that can act as a universal server where database can be accessed and data can be retrieved
- Searching using the regular expression can improve the query result and can make it more useful in searching over the words from the database or files.
- further advanced commands to link tables in MySQL can improve the overall functionality of the program and can truly bring the concept of foreign key to work.