SPARK: DISCOVER, BROWSE, EXPLORE

AISSCE

COMPUTER SCIENCE PRACTICAL EXAMINATION 2023-24

DELHI PUBLIC SCHOOL
RUBY PARK, KOLKATA

OBJECTIVE

spark functions as a central hub for storing and organizing diverse forms of information pertaining to library resources. It encompasses key data such as book titles, codes, author names, and availability status. The system actively monitors and updates the status of each book within the library, providing visibility on whether it has been borrowed or is currently accessible for lending.

INPUTS & OUTPUTS

- Module Imports & Database Connections
- CLI Initialization
- Data Storage & Manipulation Function Definition
- User Interaction through the CLI

OPERATION ENVIRONMENT

- **Processor:** Intel i3 and upwards
- **OS:** Windows, Linux, macOS or others
- **Memory:** 1 GB RAM or more
- **Hard Disk Space:** Minimum 3 GB for Database usage
- **Database:** MySQL

FEATURES

- **Comprehensive Information Management:** spark serves as a centralized repository for managing various types of information related to library contents. It includes details such as book titles, codes, author names, and availability status.

- **Book Status Tracking:** spark tracks the status of each book in the library, indicating whether it has been issued or is available for borrowing. This real-time tracking helps both librarians and users in determining the availability of specific books.

- **Efficient Book Issuance:** With spark, librarians can easily issue books to borrowers, keeping track of the transaction details. The system ensures accuracy, reduces manual paperwork, and streamlines the lending process.

- **Returns and Overdue Management:** spark helps in managing book returns, keeping a record of the due dates and reminding borrowers about overdue books. It enables librarians to efficiently handle late returns, fines, and reminders, ensuring smooth operations.

- **Catalog and Search Capabilities:** spark typically provides cataloging functionalities, allowing librarians to create and update book records with relevant information. Users can utilize search features to locate specific books, authors, or genres, enhancing the overall user experience.

SCOPES

- **Efficient Book Management:** spark enables users to store and manage book details as well as customer information.

- **Streamlined Library Data:** This software package facilitates the storage of comprehensive library-related data details.

- **Robust Yearly Operations:** The system is designed to handle recurring yearly operations effectively, even in cases where the database requires periodic maintenance and clearance.

LIMITATIONS

- **Scalability:** spark may face challenges in scaling up to accommodate larger libraries with extensive book collections and a high volume of users. The performance and response times could be affected as the database grows in size.

- **Integration with External Systems:** Integrating spark with other external systems, such as online catalogs or self-checkout systems, may require additional development effort and compatibility considerations.

- **Customization and Flexibility:** The system's level of customization and flexibility may be limited, potentially restricting the ability to tailor the system to specific library requirements or unique workflows.

- **Mobile Access:** As spark does not provide a dedicated mobile application or responsive web interface, users may face limitations when accessing the system on mobile devices. This could impact the convenience and accessibility for users who prefer to use mobile devices for library management tasks.

FUTURE SCOPE

The future scope of the spark includes expanding content with online lectures and additional resources, enhancing the GUI with visuals, continuous development, advanced reporting, a dedicated mobile app, integration with external systems, personalized user profiles, collaboration between libraries, improved accessibility, and integration with learning management systems. These enhancements aim to enrich the user experience, provide comprehensive resources, and meet evolving needs in the educational community.

User.py

```
# utility modules
import time
import datetime
from click import clear

# command modules
from Admin import *
from Members import *

# sql connector module
import mysql.connector as sqlcon
mycon = sqlcon.connect(host='localhost', user='root', passwd='', database='lib_mng_db')

data_user = data_book = data_user_book = c_user = c_book = c_user_book = None

def idiot():
    global data_user
    global data_book
    global data_user_book
    global c_user
    global c_book
    global c_user_book
    c_user = mycon.cursor()
    c_user.execute("select * from User_Details")
    data_user = c_user.fetchall()
    c_book = mycon.cursor()
    c_book.execute("select * from Books")
    data_book = c_book.fetchall()
    c_user_book = mycon.cursor()
    c_user_book.execute("select * from User_Books")
    data_user_book = c_user_book.fetchall()

idiot()

users = {i[0]: [i[1], i[3], i[-1]] for i in
         data_user}

while True:

    clear()

    print(
        '''Login
Register

['L' for login, 'R' for register]\n'''
        )

    lr = input()

    if lr.upper() == "L":

        in_uname = input("Enter your username: ")
        in_pass = input("Enter your password: ")

        if in_uname in users:
            if users[in_uname][2] == in_pass:
                USER = {
                    "username": in_uname,
                    "name": users[in_uname][0],
                    "type": users[in_uname][1]
                    }

        if USER == {}:
            print("\nERROR: Incorrect Credentials")
            time.sleep(2)
            continue
```

```
        else:
            if USER['type'] == "Admin":
                print(f'Successfully logged in as Admin {USER["name"]}')
            else:
                print(f'Successfully logged in as Member {USER["name"]}')
            break

    elif lr.upper() == "R":
        in_uname = input("Choose your username: ")
        in_name = input("Enter your name: ")
        in_about = input("Enter a short description about yourself: ")
        in_pass = input("Choose your password: ")
        in_confirmpass = input("Confirm your password: ")

        issue = ""

        if in_pass == in_confirmpass:
            if not in_uname in users:
                USER = {
                    "username": in_uname,
                    "name": in_name,
                    "type": "Member"
                    }
                now = datetime.datetime.now()
                reg_user = mycon.cursor()

                try:
                    reg_user.execute(
                        f'''insert into User_Details (Username, U_Name, About, U_Type,
                        U_Password)
values ('{in_uname}', '{in_name}', '{in_about}', 'Member', '{in_pass}')'''
                        )
                    mycon.commit()
                except:
                    mycon.rollback()
                    USER = {}
                    issue = "ERROR: Changes couldn't be save. Try again later."


            else:
                issue = "ERROR: Username already taken"
        else:
            issue = "ERROR: Passwords do not match"

        if USER == {}:
            print("\n" + issue)
            time.sleep(2)
            continue
        else:
            time.sleep(1)
            print(f'Successfully logged in as Member {USER["name"]}')
            break

clear()

instructions = [
    ["Add users", "Admin"],
    ["Add books", "Admin"],
    ["Update book details", "Admin"],
    ["Issue a book", "Member"],
    ["Return a book", "Member"],
    ["Display book details", "AdminMember"],
    ["Exit", "AdminMember"]
    ]

# this list consists of all the commands the logged in user can execute
valid_instructions = {}
j = 1
for i in instructions:
    if USER["type"] in i[1]:
        valid_instructions[str(j)] = i[0]
        j += 1
```

```
while True:

    for i in valid_instructions:
        print(f'{i} > {valid_instructions[i]}')
    print('\n[Enter respective numbers to execute commands]')
    command = input()

    if valid_instructions[command]:
        cmd = valid_instructions[command]
        idiot()
        if cmd == "Add users":
            add_user(c_user, c_book, c_user_book, mycon)
        elif cmd == "Add books":
            add_book(c_user, c_book, c_user_book, mycon)
        elif cmd == "Update book details":
            update_book(c_user, c_book, c_user_book, mycon, data_book)
        elif cmd == "Issue a book":
            issue_book(c_user, c_book, c_user_book, mycon, data_book, data_user)
        elif cmd == "Return a book":
            return_book(c_user, c_book, c_user_book, mycon, data_book, data_user)
        elif cmd == "Display book details":
            c_book = mycon.cursor()
            c_book.execute("select * from Books")
            data_book = c_book.fetchall()
            display_book(data_book)
        elif cmd == "Exit":
            break
        else:
            print("ERROR: Invalid command")
            time.sleep(2)

    print("\n")
mycon.close()
```

Members.py

```
from tabulate import tabulate

# Issue a book
def issue_book(c_user, c_book, c_user_book, mycon, data_book, data_user):
    check = True
    Username = input("Enter username: ")
    Bk_ID = int(input("Enter book ID of book to issue: "))
    for i in data_user:
        if i[0] == Username: break
    else:
        print("Incorrect username. Try again.")
        check = False
    for i in data_book:
        if i[0] == Bk_ID:
            N_o_A_C = i[5]
            break
    else:
        print("Incorrect book ID. Try again.")
        check = False
    if N_o_A_C < 1:
        check = False
    if check:
        c_book.execute(
            "update Books set Number_of_Available_Copies = {} where Book_ID = {}".format(
                N_o_A_C - 1, Bk_ID
            )
        )
        c_user_book.execute(
            "insert into User_Books(Username, Book_ID, Issued_Date, Return_Date) values('{}', {}, curdate(), null)"
            .format(
                Username, Bk_ID
            )
        )
    mycon.commit()
```

```
# Return a book
def return_book(c_user, c_book, c_user_book, mycon, data_book, data_user):
    Username = input("Enter your username: ")
    Bk_ID = int(input("Enter book ID of the book that you want to return: "))
    for i in data_user:
        if i[0] == Username:
            break
    else:
        print("Incorrect username. Try again.")
        check = False
    for i in data_book:
        if i[0] == Bk_ID:
            N_o_A_C = i[5]
            break
    else:
        print("Incorrect book ID. Try again.")
        check = False
    c_book.execute(
        "update Books set Number_of_Available_Copies = {} where Book_ID = {}".format(
            N_o_A_C + 1, Bk_ID
        )
    )
    c_user_book.execute(
        "update User_Books set Return_Date = curdate() where Book_ID = {} and Username = '{}'".format(
            Bk_ID, Username
        )
    )
    mycon.commit()

# View Book Details
def display_book(data_book):
    data_book.insert(0, ["BookID", "Book Title", "Book Author", "Genre", "Total Copies", "Available Copies"])
    print_table(data_book)

def print_table(table):
    headings = table[0]
    data = table[1:]

    table_str = tabulate(data, headers=headings, tablefmt="grid")

    print(table_str)
```

Admin.py

```
from tabulate import tabulate

# Add a member
def add_user(c_user, c_book, c_user_book, mycon):
    Username = input("Enter username: ")
    U_Name = input("Enter your name: ")
    About = input("Enter a few words about yourself: ")
    U_Type = input("Enter Admin/Member: ")
    U_Pswd = input("Set Password: ")
    c_user.execute(
        '''insert into User_Details(Username, U_Name, About, U_Type, Date_of_Joining, U_Password)
        values('{}', '{}', '{}', '{}', curdate(), '{}')'''.format(
            Username, U_Name, About, U_Type, U_Pswd
        )
    )
    mycon.commit()
```

```python
# Add a book
def add_book(c_user, c_book, c_user_book, mycon):
    Bk_ID = input("Enter bookID: ")
    Bk_Title = input("Enter book name: ")
    A_Name = input("Enter author name: ")
    Genre = input("Enter Genre: ")
    N_o_C = input("Enter number of copies of the book: ")
    c_user.execute(
        '''insert into Books(Book_ID, Book_Title, Author_Name, Genre, Number_of_Copies, Number_of_Available_Copies)
        values('{}', '{}', '{}', '{}', {}, {})'''.format(
            Bk_ID, Bk_Title, A_Name, Genre, N_o_C, N_o_C
        )
    )
    mycon.commit()

# Update book count
def update_book(c_user, c_book, c_user_book, mycon, data_book):
    check = True
    Bk_ID = int(input("Enter book ID of the book to update: "))
    choice = input("Number of copies of the book decreases or increases? [d/i]")
    if choice == "d":
        for i in data_book:
            if i[0] == Bk_ID:
                N_o_C = i[4]
                N_o_A_C = i[5]
                break
        else:
            print("Incorrect book ID. Try again.")
            check = False
        if check:
            decrease = int(input("Enter decrease in the number of copies of the book: "))
            if N_o_C - decrease > 0:
                c_book.execute(
                    "update Books set Number_of_Copies = {} where Book_ID = {}".format(
                        N_o_C - decrease, Bk_ID
                    )
                )
            else:
                print(
                    "Invalid decrease in number of copies. Number of copies available = ",
                    N_o_C,
                )
    elif choice == "i":
        for i in data_book:
            if i[0] == Bk_ID:
                N_o_C = i[4]
                N_o_A_C = i[5]
                break
        else:
            print("Incorrect book ID. Try again.")
            check = False
        if check:
            increase = int(input("Enter increase in the number of copies of the book: "))
            c_book.execute(
                "update Books set Number_of_Copies = {}, Number_of_Available_Copies = {} where Book_ID = '{}'".format(
                    N_o_C + increase, N_o_A_C + increase, Bk_ID
                )
            )
    mycon.commit()

# View Book Details
def display_book(data_book):
    data_book.insert(0, ["BookID", "Book Title", "Book Author", "Genre", "Total Copies", "Available Copies"])
    print_table(data_book)

def print_table(table):
    # Assuming the first list is the heading row
    headings = table[0]
    data = table[1:]

    # Use the 'tabulate' function to format the table
    table_str = tabulate(data, headers=headings, tablefmt="grid")

    print(table_str)
```

Setup.sql

```sql
DROP database if exists lib_mng_db;
CREATE database lib_mng_db;
USE lib_mng_db;

DROP table if exists User_Details;
CREATE TABLE User_Details
(   Username    varchar(30) primary key,
    U_Name  varchar(30) not null,
    About   varchar(100),
    U_Type  varchar(6)  not null,
    Date_of_joining date,
    U_Password  varchar(30) not null
);

DROP table if exists Books;
CREATE TABLE Books
(   Book_ID integer primary key,
    Book_Title  varchar(100)    not null,
    Author_Name varchar(100),
    Genre   varchar(50),
    Number_of_Copies    integer not null    default(0),
    Number_of_Available_Copies  integer not null    default(0)
);

DROP table if exists User_Books;
CREATE TABLE User_Books
(   User_Book_ID    integer   auto_increment    primary key,
    Username    varchar(30) not null references User_Details(Username),
    Book_ID integer not null references Books(Book_ID),
    Issued_Date date    not null,
    Return_date date    default(null)
);


insert into User_Details(Username, U_Name, About, U_Type, Date_of_Joining, U_Password)
values("Admin001", "Shanta Mukherjee", "Head Librarian",
"Admin", curdate(), "HopscotchLib");

alter table User_Books auto_increment = 100;
```

Spark Command Line Interface:

[commands executed as a user logged in as a member]

```
Login
Register

['L' for login, 'R' for register]

L
Enter your username: shuvam
Enter your password: papaya


1 > Issue a book
2 > Return a book
3 > Display book details
4 > Exit

[Enter respective numbers to execute commands]
1
Enter username: shuvam
Enter book ID of book to issue: 2


1 > Issue a book
2 > Return a book
3 > Display book details
4 > Exit

[Enter respective numbers to execute commands]
2
Enter your username: shuvam
Enter book ID of the book that you want to return: 2



1 > Issue a book
2 > Return a book
3 > Display book details
4 > Exit

[Enter respective numbers to execute commands]
3
+----------+-------------------------+---------------+---------+-----------------+--------------------+
|   BookID | Book Title              | Book Author   | Genre   |   Total Copies |   Available Copies |
+==========+=========================+===============+=========+=================+====================+
|        1 | Concepts of Physics     | HC Verma      | Physics |               9 |                  9 |
+----------+-------------------------+---------------+---------+-----------------+--------------------+
|        2 | Goblet Of Fire          | JK Rowling    | Fiction |              20 |                 19 |
+----------+-------------------------+---------------+---------+-----------------+--------------------+
|        3 | How to Design an F1 Car | Adrian Newey  | Design  |               5 |                  5 |
+----------+-------------------------+---------------+---------+-----------------+--------------------+
```

Spark Command Line Interface:

[commands executed as a user logged in as an admin]

```
Login
Register

['L' for login, 'R' for register]

L
Enter your username: Admin001
Enter your password: HopscotchLib

1 > Add users
2 > Add books
3 > Update book details
4 > Display book details
5 > Exit

[Enter respective numbers to execute commands]
1
Enter username: jakePeralta
Enter your name: Jacob Peralta
Enter a few words about yourself: best human/genius of the NYPD
Enter Admin/Member: Member
Set Password: diehard

1 > Add users
2 > Add books
3 > Update book details
4 > Display book details
5 > Exit

[Enter respective numbers to execute commands]
2
Enter bookID: 4
Enter book name: Deathly Hallows
Enter author name: JK Rowling
Enter Genre: Fiction
Enter number of copies of the book: 25

1 > Add users
2 > Add books
3 > Update book details
4 > Display book details
5 > Exit

[Enter respective numbers to execute commands]
3
Enter book ID of the book to update: 4
Number of copies of the book decreases or increases? [d/i]i
Enter increase in the number of copies of the book: 2
```

```
1 > Add users
2 > Add books
3 > Update book details
4 > Display book details
5 > Exit

[Enter respective numbers to execute commands]
4
```

BookID	Book Title	Book Author	Genre	Total Copies	Available Copies
1	Concepts of Physics	HC Verma	Physics	9	9
2	Goblet Of Fire	JK Rowling	Fiction	20	20
3	How to Design an F1 Car	Adrian Newey	Design	5	5
4	Deathly Hallows	JK Rowling	Fiction	27	27

MySQL Database Tables:

```
mysql> use lib_mng_db;
Database changed


mysql> show tables;
+----------------------+
| Tables_in_lib_mng_db |
+----------------------+
| Books                |
| User_Books           |
| User_Details         |
+----------------------+
3 rows in set (0.00 sec)


mysql> desc Books;
+---------------------------+-------------+------+-----+---------+-------------------+
| Field                     | Type        | Null | Key | Default | Extra             |
+---------------------------+-------------+------+-----+---------+-------------------+
| Book_ID                   | int         | NO   | PRI | NULL    |                   |
| Book_Title                | varchar(30) | NO   |     | NULL    |                   |
| Author_Name               | varchar(30) | YES  |     | NULL    |                   |
| Genre                     | varchar(20) | YES  |     | NULL    |                   |
| Number_of_Copies          | int         | NO   |     | 0       | DEFAULT_GENERATED |
| Number_of_Available_Copies | int         | NO   |     | 0       | DEFAULT_GENERATED |
+---------------------------+-------------+------+-----+---------+-------------------+
6 rows in set (0.00 sec)


mysql> select * from Books;
+---------+-------------------------+--------------+---------+------------------+---------------------------+
| Book_ID | Book_Title              | Author_Name  | Genre   | Number_of_Copies | Number_of_Available_Copies |
+---------+-------------------------+--------------+---------+------------------+---------------------------+
|       1 | Concepts of Physics     | HC Verma     | Physics |                9 |                         9 |
|       2 | Goblet Of Fire          | JK Rowling   | Fiction |               20 |                        20 |
|       3 | How to Design an F1 Car | Adrian Newey | Design  |                5 |                         5 |
|       4 | Deathly Hallows         | JK Rowling   | Fiction |               27 |                        27 |
+---------+-------------------------+--------------+---------+------------------+---------------------------+
4 rows in set (0.01 sec)
```

```
mysql> desc User_Books;
+--------------+-------------+------+-----+---------+-------------------+
| Field        | Type        | Null | Key | Default | Extra             |
+--------------+-------------+------+-----+---------+-------------------+
| User_Book_ID | int         | NO   | PRI | NULL    | auto_increment    |
| Username     | varchar(30) | NO   |     | NULL    |                   |
| Book_ID      | int         | NO   |     | NULL    |                   |
| Issued_Date  | date        | NO   |     | NULL    |                   |
| Return_date  | date        | YES  |     | NULL    | DEFAULT_GENERATED |
+--------------+-------------+------+-----+---------+-------------------+
5 rows in set (0.00 sec)


mysql> select * from User_Books;
+--------------+-------------+---------+-------------+-------------+
| User_Book_ID | Username    | Book_ID | Issued_Date | Return_date |
+--------------+-------------+---------+-------------+-------------+
|          100 | shuvam      |       2 | 2023-11-16  | 2023-11-16  |
|          101 | shuvam      |       2 | 2023-11-16  | 2023-11-16  |
|          102 | jakePeralta |       3 | 2023-11-16  | NULL        |
+--------------+-------------+---------+-------------+-------------+
3 rows in set (0.00 sec)



mysql> desc User_Details;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| Username        | varchar(30)  | NO   | PRI | NULL    |       |
| U_Name          | varchar(30)  | NO   |     | NULL    |       |
| About           | varchar(100) | YES  |     | NULL    |       |
| U_Type          | varchar(6)   | NO   |     | NULL    |       |
| Date_of_joining | date         | YES  |     | NULL    |       |
| U_Password      | varchar(30)  | NO   |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
6 rows in set (0.01 sec)


mysql> select * from User_Details;
+-------------+------------------+---------------------------------+--------+-----------------+--------------+
| Username    | U_Name           | About                           | U_Type | Date_of_joining | U_Password   |
+-------------+------------------+---------------------------------+--------+-----------------+--------------+
| A           | AAH              | HAA                             | Admin  | 2023-11-15      |              |
| Admin001    | Shanta Mukherjee | Head Librarian                  | Admin  | 2023-11-07      | HopscotchLib |
| jakePeralta | Jacob Peralta    | best human/genius of the NYPD   | Member | 2023-11-16      | diehard      |
| LandoNorris | Lando Norris     | omzzz                           | Member | 2023-11-15      | racewin      |
| shuvam      | Shuvam Mitra     | ami ki jani                     | Member | NULL            | papaya       |
+-------------+------------------+---------------------------------+--------+-----------------+--------------+
5 rows in set (0.00 sec)
```

BIBLIOGRAPHY

Source Code: https://github.com/SohamDeep2026/Spark

MySQL Connector/Python Developer Guide: https://dev.mysql.com/doc/connector-python/en/

Python Package Index - PyPI: https://pypi.org/