AMICUS INTERNATIONAL SCHOOL, BHARUCH

Academic Year: 2023-24

PROJECT REPORT ON

SOFTWARE FOR Library Management

Computer Science Project

Submitted By:

Name: Arghya Kumar

Class: XII-B

Roll No.: _____

Project Guide

Mrs. Arpita Bhoi

PGT (Computer Science)

Certificate

This is to certify that Master **Arghya Kumar** of **Class XII-B (Science)**, has successfully completed the **Computer Science (083)** investigatory project on the topic **Software for Library Management** within the stipulated time frame with sincerity and devotion, under the guidance of Mrs. Arpita Bhoi during the academic year 2023-24 in the partial fulfillment of Practical Examination conducted by CBSE.

Signature of Internal Examiner	Signature of Principal	Signature of External Examiner

Date: _____

Place: _____

Acknowledgement

I extend my heartfelt gratitude to everyone who has contributed to the completion of this Project.

I am sincerely grateful to our principal Mrs. Thresiamma Pappachan and my CS teacher Mrs. Arpita Bhoi. Their unwavering support and guidance have been instrumental in its completion.

Lastly, I express my sincere gratitude to my parents and family for their unwavering support.

To all those mentioned above and many others who have directly or indirectly contributed, your support has been truly invaluable in my growth as a learner.

- Arghya Kumar

Index

Introduction

This application aims to solve the problem of efficient management of library records. It offers a solution for tracking transactions, and managing data about books and patrons in an organised way. The user can efficiently manage the database by creating, updating and removing records. The goal is to facilitate library operations by using technology to manage them conveniently.

The **Library Manager** Application is a command line application (CLI) which allows users to perform certain library operations. The application can be accessed by both the Admin and the Patrons (users/members) of the library.

Users login to the application by providing their respective password (admin) or ID (patron), upon which a list of functions are presented. Users can select their desired function by entering the corresponding number. The application then prompts the user for the necessary inputs, and executes the function. The output is then displayed to the user.

The library administrator can add, update or remove book records, and also update patron information. Furthermore, the admin can issue and return books, as well as view all transactions, with due dates and return status.

The patron can view all books, search for books, and view their own transactions. They can also issue and return books, and view their pending transactions.

Benefits and Limitations

Benefits

- Efficient management of library records
- Simple interface for interacting with the database
- Error-prone and secure

Limitations

- Tedious to use due to menu based CLI - more suitable for a GUI or web app
- Lack of networking
- Limited functionality

Scope of Improvement

- SQL Server can be hosted on a server machine, allowing for Client-Server networking and accessibility.
- Third party modules like `inquirer` can be used to create better CLI interfaces rather than implementing a menu based interface.
- A GUI or web app can be created to make the application more user friendly and accessible.

Technologies Used

This project uses Python as the primary programming language for the frontend application. The project uses an MySQL database for the backend, for efficient data storage and retrieval.

Hardware:

- X86 64-bit processor
- 4GB RAM

Software:

- Operating System: Windows 10
- Python 3.12
- MySQL Server 8.0

Functional Requirements

The following are the functional requirements for a library management system:

Admin Functions

- To list and search books by criteria
- To add, remove and update books
- To view information about patrons
- To issue books and return books
- To view all transactions

Patron Functions

- To list and search books by criteria
- To issue books and return books
- To view self transactions (completed and pending)

Database Design

The database for our application is managed by a SQL Database. It comprises of 3 tables: `Books`, `Patrons`, and `Transactions`. The ERD (Entity Relationship Diagram) shows the various relations and fields in the database (Figure 1).

Books

The `Books` table stores information on books, storing `isbn`, `title`, `author`, `quantity` and `genre`.

- `isbn`: The International Standard Book Number uniquely identifying each book. (Primary Key)
- `title`: The title of the book.
- `author`: The author of the book.
- `quantity`: The number of available copies of the book.

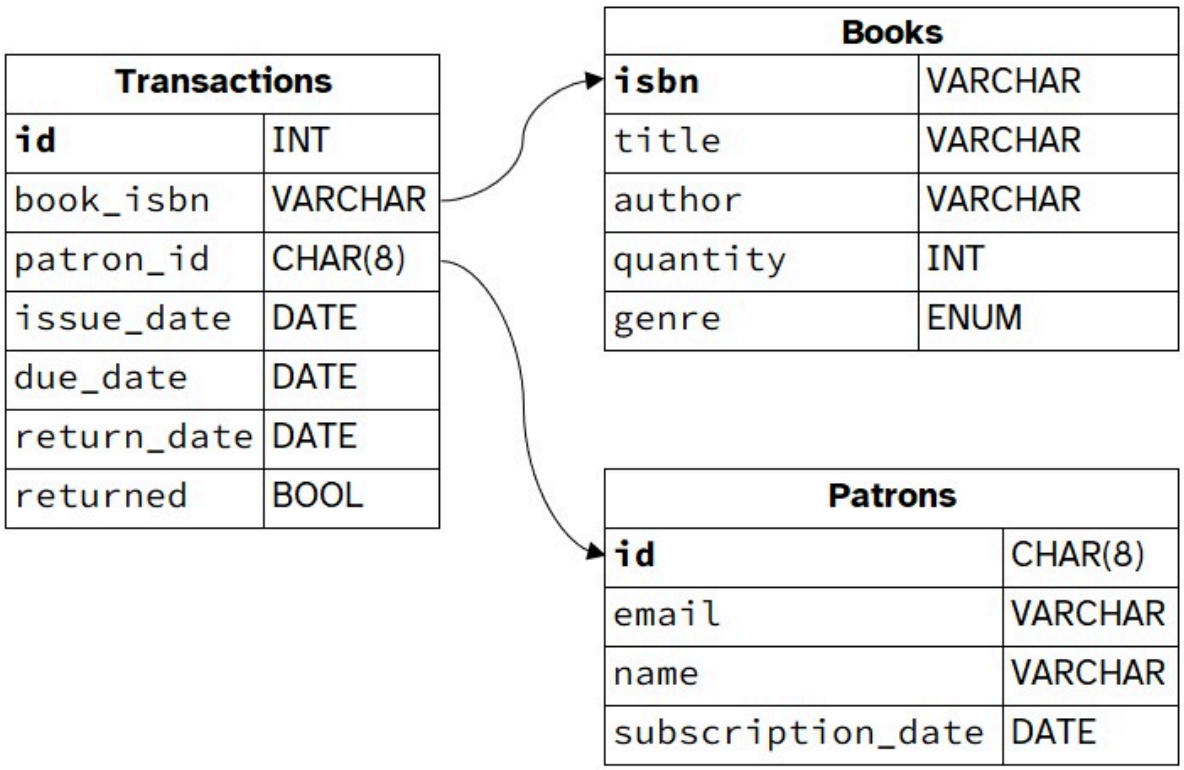


Figure 1: Database Entity Relationship diagram

- `genre`: The genre of the book.

Patrons

The `Patrons` table stores user info, about library patrons who can borrow and return books.

- `id`: A unique identifier for each patron. (Primary Key)
- `email`: The email address of the patron.
- `name`: The name of the patron.
- `subscription_date`: The date when the patron subscribed to the library.

Transactions

The `Transactions` table keeps track of all issuings, with their date of issue and date of return. It helps manage the circulation of books, tracking their status and due dates, and allowing for reporting and analysis.

- `id`: A unique identifier for each transaction. (Primary Key)
- `book_isbn`: A foreign key referencing the “books” table, specifying the ISBN of the book involved.
- `patron_id`: A foreign key referencing the “patrons” table, identifying the patron associated.
- `issue_date`: The date when the book was issued.
- `due_date`: The date when the book is expected to be returned.
- `return_date`: The date when the book was actually returned (if returned).
- `returned`: A boolean indicating whether the book has been returned (default is false).

The database schema is defined in the **`db.sql`** file.

Functional Decomposition

The application is decomposed into smaller components, which include:

- functionality for querying the database
- the main program (driver)

All database queries are isolated inside a separate file, **`database.py`**.

```
# database.py

def IssueBook(isbn, patron_id):
  ...

def ReturnBook(isbn, patron_id):
  ...

def ViewTransactions():
  ...

...
```

Listing 1: Function decomposition of database functions

These functions use the `mysql.connecter` library to connect to the MySQL database and query the database, and also return output and report errors wherever applicable.

The main frontend code is present inside **`main.py`**, which creates a menu based UI (within the terminal) for interacting with these database functions.

```python
# main.py

print("Welcome to the Library Manager!!")
print("(1) ADMIN")
print("(2) USER")
choice = int(input("Enter your choice: "))

if choice == 1:
  # admin
  input("Enter password")
  ...
elif choice == 2:
  # patron
  input("Enter patron ID")
  ...
```

Listing 2: Frontend code for interacting with the database

Inside each conditional, the menu options are listed, and the user is prompted for their choice. At the innermost level, the database functions are called, and the output is shown.

```python
# main.py

while True:
  opt = int(input("\tEnter your choice: "))
  ...
  elif opt == 5: # return book
    ISBN = input("\t\tISBN : ")
    ID = input("\t\tID of the patron: ")
    db.ReturnBook(ISBN, ID)
    print("\t\tBook returned successfully!")
```

Listing 3: Calling database functions in the frontend

Control Flow of the Application

Upon launching the app, two options are presented: utilizing the application as an admin or as a user (patron). The primary functionalities of the app are exclusively accessible to the admin, for evident reasons. On the other hand, the user is limited to performing a few read operations.

The admin operations are:

- Add/Remove/Edit a Book
- Add/Remove/Edit a Patron
- View Books/Patrons
- Search Books
- Issue/Return a Book
- View all Transactions
- View pending Transactions

Most of these functions just perform simple CRUD[1] operations using SQL queries and return the output (if any). The custom functionality which is implemented specifically for this project is the Issue/Return workflow, which is defined based on its **constraints** and **actions.**

Issuing a Book

The `IssueBook()` function takes in two arguments - `isbn` and `patron_id`. The steps of the Issue workflow are:

- Constraints
  1. `isbn` must link to a specific `Book`
  2. `patron_id` must link to a specific `Patron`
  3. `Patron` must not currently have 3 unreturned books

[1]Create, Read, Update, Delete

4. `Patron` must not currently have this book

- Actions
  1. Update `quantity` of `Book` to `quantity - 1`
  2. Create a `Transaction` linking the `Patron` and the `Book`, with the current date as `issue_date`, and with a 7 day time interval, a `return_date`.
  3. Commit to the Database.

Returning a Book

Similar to the Issue Book workflow, the `ReturnBook()` function also takes in arguments - `isbn` and `patron_id`.

- Constraints
  1. `isbn` must link to a specific `Book`
  2. `patron_id` must link to a specific `Patron`
  3. `Patron` must currently have this book - i.e. There should be one `Transaction` such that it is associated with this `Patron` and this `Book`, and has `returned` set to `FALSE`.
- Actions
  1. Update `returned = TRUE` and `return_date` to current date of `Transaction`.
  2. Update `quantity` of `Book` to `quantity + 1`
  3. Commit to the Database.

Modules Used

The following modules are used in the application:

`mysql.connector`

MySQL Connector/Python is a standardized database driver for Python platforms, used to connect to the MySQL database and execute queries.

- `connector.connect()`: Connects to the MySQL database using the specified credentials.
- `connector.cursor()`: Creates a cursor object, which is used to execute queries.
- `cursor.execute()`: Executes the specified SQL query.
- `cursor.fetchall()`: Returns all rows of a query result.
- `cursor.commit()`: Commits the changes to the database.

`tabulate`

Tabulate is a Python library used for printing tabular data in the terminal.

- `tabulate()`: Prints the specified data in a tabular format.

`tkinter.simpledialog`

Used for creating simple modal dialogs for user input.

- `simpledialog.askstring()`: Creates a dialog box with a text field, and returns the input.

User Defined Functions

In **`database.py`**, different functions for performing database operations are defined.

- `AddBook()`: Adds a new book to the database.
- `RemoveBook()`: Removes a book from the database.
- `SearchBookByISBN()`, `SearchBookByTitle()`, `SearchBookByAuthor()`, `SearchBookByGenre()`: Searches for a book by the specified criteria.
- `EditBook()`: Edits the details of a book. (quantity)
- `IssueBook()`: Issues a book to a patron.
- `ReturnBook()`: Returns a book from a patron.
- `AddPatron()`: Adds a new patron to the database.
- `RemovePatron()`: Removes a patron from the database.
- `EditPatron()`: Edits the details of a patron.
- `SearchPatronByID()`, `SearchPatronByName()`: Searches for a patron by the specified criteria.
- `ViewTransactions()`: Views all transactions.
- `ViewPendingTransactions()`: Views all pending transactions.
- `ViewBooks()`: Views all books.
- `ViewPatrons()`: Views all patrons.

Additionally, there are some helper functions for common tasks to avoid code repetition.

- `ValidateISBN()`: Checks if the given ISBN is valid by using check digits.
- `ValidateDate()`: Checks if the given date is in valid format.
- `TrySQLCommand()` : Executes the given SQL command and returns the output along with column headers.
- `showtable()`: Displays the given data in a tabular format.
- `triminput()`: Takes user input and removes leading and trailing whitespaces.
- `numinput()`: Takes valid numerical input from the user.

Source Code

main.py

```
from tkinter import simpledialog
import time
import database as db
from tabulate import tabulate

def ValidateDate(date):
  # yyyy-mm-dd
  try:
    time.strptime(date, "%Y-%m-%d")
    return True
  except ValueError:
    return False

def showtable(data, headers):
  print(tabulate(data, headers=columns, tablefmt="pretty"))

def triminput(*args, **kwargs):
  return input(*args, **kwargs).strip()

def numinput(*args, **kwargs):
  while True:
    try:
      return int(input(*args, **kwargs))
    except ValueError:
      print("Invalid input! Please enter a number.")

print("Welcome to the Library Manager!!")
print()
print("ACCESS TO : ")
print("(1) ADMIN")
print("(2) USER")
print()
choice = numinput("Enter your choice: ")
print()

if choice >= 1 and choice <= 2:
  if choice == 1:
    pwd = simpledialog.askstring("Password", "Enter your password:",
show="*")
    if pwd == "admin123123" or pwd == "libraryroot123":
      print("Recognised as admin....")
      print("Accessing administrative functions.....")
      print()
      time.sleep(2)
      while True:
        print("\t(1) Book functions")
        print("\t(2) Patron functions")
        print("\t(3) Transactions and Returns")
        print("\t(4) Exit")

        acc = numinput("\tPLEASE ENTER THE OPTION NUMBER : ")
        if acc >= 1 and acc <= 4:
```

```
52            if acc == 1:
53              print("\tAccessing........")
54              time.sleep(2)
55
56              while True:  #
57                print("\t\t**** BOOK FUNCTIONS ****")
58                print("\t\t OPTIONS : ")
59                print("\t\t(1) Add a book")
60                print("\t\t(2) Remove a book")
61                print("\t\t(3) Update book quantity")
62                print("\t\t(4) Issue a Book")
63                print("\t\t(5) Return a Book")
64                print("\t\t(6) Search a Book")
65                print("\t\t(7) View all books")
66                print("\t\t(8) Go back to main menu")
67                print("\t\t(9) Exit")
68
69                opt = numinput("\t\tPLEASE ENTER THE OPTION NUMBER : ")
70                if opt >= 1 and opt <= 9:
71                  if opt == 1:
72                    nISBN = triminput("\t\tISBN : ")
73                    nTITLE = triminput("\t\tTITLE : ")
74                    nAUTHOR = triminput("\t\tAUTHOR : ")
75                    nGENRE = triminput("\t\tGENRE (Fiction/Non-Fiction): ")
76                    Quantity = numinput("\t\tQUANTITY : ")
77                    out = db.AddBook(Quantity, nTITLE, nAUTHOR, nISBN,
    nGENRE)
78                    if out != 1:
79                      print("\t\tBook added successfully!")
80                  elif opt == 2:
81                    ISBN = triminput("\t\tISBN : ")
82                    out = db.RemoveBook(ISBN)
83                    if out != 1:
84                      print("\t\tBook removed successfully!")
85                  elif opt == 3:
86                    ISBN = triminput("\t\tISBN : ")
87                    out = db.EditBook(ISBN)
88                    if out != 1:
89                      print("\t\tBook updated successfully!")
90                  elif opt == 4:
91                    ISBN = triminput("\t\tISBN : ")
92                    ID = triminput("\t\tID of the patron: ")
93                    out = db.IssueBook(ISBN, ID)
94                    if out != 1:
95                      print("\t\tBook issued successfully!")
96                  elif opt == 5:
97                    ISBN = triminput("\t\tISBN : ")
98                    ID = triminput("\t\tID of the patron: ")
99                    out = db.ReturnBook(ISBN, ID)
100                   if out != 1:
101                     print("\t\tBook returned successfully!")
102                 elif opt == 6:
103                   while True:
104                     print("\t\t\t OPTIONS :")
105                     print("\t\t\t(1) By ISBN number")
106                     print("\t\t\t(2) By Author")
```

```
                    print("\t\t\t(3) By Title")
                    print("\t\t\t(4) By Genre")
                    print("\t\t\t(5) Return to previous menu")
                    print("\t\t\t(6) Exit")

                    x = numinput("\t\t\tPLEASE ENTER THE OPTION NUMBER :
    ")
                    if x >= 1 and x <= 6:
                      if x == 1:
                        ISBN = triminput("\t\t\t\tISBN : ")
                        res = db.SearchBookByISBN(ISBN)
                        if res != 1:
                          data, columns = res
                          showtable(data, headers=columns)
                        else:
                          print("\t\t\t\tBook not found!")

                      elif x == 2:
                        Author = triminput("\t\t\t\tAuthor : ")
                        res = db.SearchBookByAuthor(Author)
                        if res != 1:
                          data, columns = res
                          showtable(data, headers=columns)
                        else:
                          print("\t\t\t\tBook not found!")

                      elif x == 3:
                        Title = triminput("\t\t\t\tTitle : ")
                        res = db.SearchBookByTitle(Title)
                        if res != 1:
                          data, columns = res
                          showtable(data, headers=columns)
                        else:
                          print("\t\t\t\tBook not found!")
                          break
                      elif x == 4:
                        Genre = triminput("\t\t\t\tGenre : ")
                        res = db.SearchBookByGenre(Genre)
                        if res != 1:
                          data, columns = res
                          showtable(data, headers=columns)
                        else:
                          print("\t\t\t\tBook not found!")
                          break
                      elif x == 5:
                        break
                      elif x == 6:
                        exit()
                    else:
                      print("\t\t\tIncorrect option entered!")
                  elif opt == 7:
                    db.ViewBooks()
                  elif opt == 8:
                    break
                  elif opt == 9:
                    exit()
```

```
                else:
                  print("\t\tIncorrect option entered!")
            elif acc == 2:
              print("\tAccessing........")
              time.sleep(2)

              while True:
                print("\t\t**** PATRON FUNCTIONS ****")
                print("\t\t OPTIONS : ")
                print("\t\t(1) Add Patron")
                print("\t\t(2) Remove Patron")
                print("\t\t(3) Update Patron")
                print("\t\t(4) Search Patron")
                print("\t\t(5) View Patrons")
                print("\t\t(6) Go back to main menu")
                print("\t\t(7) Exit")
                opt = numinput("\t\tPLEASE ENTER THE OPTION NUMBER : ")
                if opt >= 1 and opt <= 7:
                  if opt == 1:
                    ID = triminput("\t\tID : ")
                    Email = triminput("\t\tEmail : ")
                    Patron_Name = triminput("\t\tPatron Name : ")
                    Subcription_Date = triminput("\t\tEnter Date(YYYY-MM-
    DD) : ")
                    if not ValidateDate(Subcription_Date):
                      print("\t\t\tIncorrect date entered!")
                      break
                    db.AddPatron(ID, Email, Patron_Name, Subcription_Date)
                    print("\t\tPatron added successfully!")
                  elif opt == 2:
                    ID = triminput("\t\tID : ")
                    db.RemovePatron(ID)
                    print("\t\tPatron removed successfully!")
                  elif opt == 3:
                    ID = triminput("\t\tID : ")
                    db.EditPatron(ID)
                    print("\t\tPatron updated successfully!")
                  elif opt == 4:
                    while True:
                      print("\t\t\t OPTIONS :")
                      print("\t\t\t(1) By ID")
                      print("\t\t\t(2) By Name")
                      print("\t\t\t(3) Return to previous menu")
                      print("\t\t\t(4) Exit")
                      x = numinput("\t\t\tPLEASE ENTER THE OPTION NUMBER:
    ")
                      if x >= 1 and x <= 4:
                        if x == 1:
                          ID = triminput("\t\t\tID : ")
                          res = db.SearchPatronByID(ID)
                          if res != 1:
                            data, columns = res
                            showtable(data, headers=columns)
                          else:
                            print("\t\t\t\tPatron not found!")
                            break
```

```
                        elif x == 2:
                          Name = triminput("\t\t\tName : ")
                          res = db.SearchPatronByName(Name)
                          if res != 1:
                            data, columns = res
                            showtable(data, headers=columns)
                          else:
                            print("\t\t\t\tPatron not found!")
                            break
                        elif x == 3:
                          break
                        elif x == 4:
                          exit()
                      else:
                        print("Incorrect option entered!")
                  elif opt == 5:
                    db.ViewPatrons()
                  elif opt == 6:
                    break
                  elif opt == 7:
                    exit()
                else:
                  print("Incorrect option entered!")
            elif acc == 3:
              print("\tAccessing........")
              time.sleep(2)

              while True:
                print("\t\t**** TRANSACTIONS ****")
                print("\t\t OPTIONS : ")
                print("\t\t(1) View Transactions")
                print("\t\t(2) View all pending returns")
                print("\t\t(3) Return to previous menu")
                print("\t\t(4) Exit")

                opt = numinput("\t\tPLEASE ENTER THE OPTION NUMBER : ")
                if opt >= 1 and opt <= 4:
                  if opt == 1:
                    db.ViewTransactions()
                  elif opt == 2:
                    db.ViewTransactionsPending()
                  elif opt == 3:
                    break
                  elif opt == 4:
                    exit()
                else:
                  print("Incorrect option entered!")
            elif acc == 4:
              exit()
          else:
            print("Incorrect option entered!")
      else:
        print("Wrong Password !!")
        exit()
    elif choice == 2:
      ID = simpledialog.askstring("ID", "Enter your Patron ID:")
```

```
272     if db.SearchPatronByID(ID) == 1:
273       print("Patron ID not found !!")
274       exit()
275     else:
276       print("Recognised as our registered patron....")
277       print("Accessing patron functions.....")
278       print()
279       time.sleep(2)
280       while True:
281         print("\tOPTIONS :")
282         print("\t(1) Search a book")
283         print("\t(2) View all books")
284         print("\t(3) View my issued books")
285         print("\t(4) Issue a book")
286         print("\t(5) Return a book")
287         print("\t(6) Exit")
288         acc = numinput("\tPLEASE ENTER THE OPTION NUMBER : ")
289         if acc >= 1 and acc <= 6:
290           if acc == 1:
291             while True:
292               print("\t\t OPTIONS :")
293               print("\t\t(1) By ISBN number")
294               print("\t\t(2) By Author")
295               print("\t\t(3) By Title")
296               print("\t\t(4) By Genre")
297               print("\t\t(5) Return to previous menu")
298               print("\t\t(6) Exit")
299               x = numinput("\t\t\tPLEASE ENTER THE OPTION NUMBER : ")
300               if x >= 1 and x <= 6:
301                 if x == 1:
302                   ISBN = triminput("\t\t\tISBN : ")
303                   res = db.SearchBookByISBN(ISBN)
304                   if res != 1:
305                     data, columns = res
306                     showtable(data, headers=columns)
307                   else:
308                     print("\t\t\t\tBook not found!")
309                     break
310                 elif x == 2:
311                   Author = triminput("\t\t\tAuthor : ")
312                   res = db.SearchBookByAuthor(Author)
313                   if res != 1:
314                     data, columns = res
315                     showtable(data, headers=columns)
316                   else:
317                     print("\t\t\t\tBook not found!")
318                     break
319                 elif x == 3:
320                   Title = triminput("\t\t\tTitle : ")
321                   res = db.SearchBookByTitle(Title)
322                   if res != 1:
323                     data, columns = res
324                     showtable(data, headers=columns)
325                   else:
326                     print("\t\t\t\tBook not found!")
327                     break
```

```
                elif x == 4:
                  Genre = triminput("\t\t\tGenre : ")
                  res = db.SearchBookByGenre(Genre)
                  if res != 1:
                    data, columns = res
                    showtable(data, headers=columns)
                  else:
                    db.SearchBookByAuthor(Author)
                elif x == 5:
                  break
                elif x == 6:
                  exit()
            elif acc == 2:
              db.ViewBooks()
            elif acc == 3:
              query = """SELECT b.title "Book", p.name "Issued by",
t.issue_date "Issued On", t.due_date "Due Date",
                              IFNULL(t.return_date, \'Not Returned\')
\'Returned On\'
                         FROM transactions t
                         JOIN books b ON t.book_isbn = b.isbn
                         JOIN patrons p ON p.id = t.patron_id
                         WHERE t.patron_id = %s"""
              res = db.Query(query, (ID, ))
              if res != 1:
                data, columns = res
                print(
                  tabulate(data,
                           headers=columns,
                           tablefmt="pretty",
                           stralign="center"))
              else:
                print("\t\t\t\tNo books issued!")
            elif acc == 4:
              ISBN = triminput("\t\tISBN : ")
              out = db.IssueBook(ISBN, ID)
              if out != 1:
                print("\t\tBook issued successfully!")
            elif acc == 5:
              ISBN = triminput("\t\tISBN : ")
              out = db.ReturnBook(ISBN, ID)
              if out != 1:
                print("\t\tBook returned successfully!")
            elif acc == 6:
              exit()
          else:
            print("Incorrect option entered!")
else:
  print("Incorrect option entered!")
```

database.py

```
import mysql.connector
from tabulate import tabulate

```

```
4  db = mysql.connector.connect(user="root", host="localhost",
   passwd="pass", database="library")
5  cursor = db.cursor()
6  cursor.execute("USE library")
7
8  # Try to execute SQL command
9  def TrySQLCommand(query, values=None):
10   cursor.execute(query, values)
11   result = cursor.fetchall()
12   if cursor.rowcount == 0:
13     raise ValueError("No matching records found.")
14   return result, [desc[0] for desc in cursor.description]
15
16 # check for valid ISBN number
17 def ValidateISBN(isbn):
18   # Remove hyphens and spaces
19   isbn = isbn.replace("-", "").replace(" ", "")
20
21   # ISBN-10
22   if len(isbn) == 10:
23     if not isbn[:-1].isdigit():
24       return False
25     if isbn[-1].upper() == "X":
26       isbn_sum = (sum(int(digit) * (i + 1) for i, digit in
   enumerate(isbn[:-1])) + 10)
27     else:
28       isbn_sum = sum(int(digit) * (i + 1) for i, digit in
   enumerate(isbn))
29     return isbn if isbn_sum % 11 == 0 else False
30
31   # ISBN-13
32   elif len(isbn) == 13:
33     if not isbn.isdigit():
34       return False
35     isbn_sum = sum(int(digit) * (1 if i % 2 == 0 else 3) for i, digit in
   enumerate(isbn))
36     return isbn if isbn_sum % 10 == 0 else False
37
38   return False
39
40 # Edit Books Table Functions
41 # Add Books
42 def AddBook(nQuantity, nTITLE, nAUTHOR, nISBN, nGenre):
43   vISBN = ValidateISBN(nISBN)
44   if vISBN == False:
45     print("INVALID ISBN NUMBER!!")
46     return 1
47   try:
48     newbooks = "INSERT INTO books (quantity, title, author, isbn, genre)
   VALUES (%s, %s, %s, %s, %s)"
49     cursor.execute(newbooks, (nQuantity, nTITLE, nAUTHOR, vISBN, nGenre))
50     db.commit()
51   except ValueError:
52     db.rollback()
53     return 1
54
```

```
# Remove Books
def RemoveBook(ISBN):
  vISBN = ValidateISBN(ISBN)
  if vISBN == False:
    print("INVALID ISBN NUMBER!!")
    return 1
  try:
    deletebook = "DELETE FROM books WHERE isbn= %s;"
    cursor.execute(deletebook, (vISBN, ))
    db.commit()
  except ValueError:
    print("ISBN number not found. Check and Try Again!")
    db.rollback()
    return 1

# Searching Books
# By ISBN
def SearchBookByISBN(ISBN):
  try:
    SearchBook = "SELECT * FROM books WHERE isbn= %s;"
    return TrySQLCommand(SearchBook, (ISBN, ))
  except ValueError:
    db.rollback()
    return 1

# By Author
def SearchBookByAuthor(Author):
  try:
    SearchBook = "SELECT * FROM books WHERE author LIKE %s"
    return TrySQLCommand(SearchBook, ("%" + Author + "%", ))
  except ValueError:
    db.rollback()
    return 1

# By Title
def SearchBookByTitle(Title):
  try:
    SearchBook = "SELECT * FROM books WHERE title LIKE %s"
    return TrySQLCommand(SearchBook, ("%" + Title + "%", ))
  except ValueError:
    db.rollback()
    return 1

# By Genre
def SearchBookByGenre(Genre):
  try:
    SearchBook = "SELECT * FROM books WHERE genre LIKE %s"
    return TrySQLCommand(SearchBook, (Genre, ))
  except ValueError:
    db.rollback()
    return 1

def EditBook(ISBN):
  val = SearchBookByISBN(ISBN)
  if val == 1:
    print("ISBN number not found. Check and Try Again!")
```

```
    return 1
  Search, columns = val

  if Search == 1:
    return 1
  else:
    print(tabulate(Search, headers=columns, tablefmt="pretty"))

    quantity = input("ENTER THE NEW QUANTITY OF BOOK AVAILABLE : ")
    query = "UPDATE books SET quantity= %s WHERE isbn=%s"
    cursor.execute(query, (quantity, ISBN))
    db.commit()

# Issue Books to Patron and updating the return status
def IssueBook(ISBN, ID):
  if SearchBookByISBN(ISBN) == 1:
    print("ISBN number not found. Check and Try Again!")
    return 1
  if SearchPatronByID(ID) == 1:
    print("Patron ID not found. Check and Try Again!")
    return 1
  # check if patron has less than 3 unreturned books
  check1 = ("SELECT COUNT(*) FROM transactions WHERE patron_id = %s AND
returned IS FALSE")
  cursor.execute(check1, (ID, ))
  if cursor.fetchone()[0] >= 3:
    print("PATRON HAS ALREADY ISSUED 3 BOOKS! \nPLEASE RETURN A BOOK AND
TRY AGAIN!")
    return 1

  # check if patron currently has this book
  check2 = "SELECT * FROM transactions WHERE patron_id = %s AND
book_isbn = %s AND returned IS FALSE"
  cursor.execute(check2, (ID, ISBN))
  cursor.fetchall()
  if cursor.rowcount > 0:
    print("PATRON HAS ALREADY ISSUED THIS BOOK!")
    return 1

  query = "UPDATE books SET quantity = quantity - 1 WHERE isbn = %s AND
quantity >= 1"
  cursor.execute(query, (ISBN, ))
  if cursor.rowcount == 1:
    try:
      issue = "INSERT INTO transactions (book_isbn, patron_id,
issue_date, due_date) VALUES (%s, %s, CURDATE(), DATE(CURDATE() + 7))"
      cursor.execute(issue, (ISBN, ID))
      db.commit()
      # print("Book issued successfully!")
    except:
      db.rollback()
      print("Book issue failed!")
      return 1
  else:
    db.rollback()
```

```
161     print("BOOK ISSUE COULD NOT BE COMPLETED AS BOOK OUT OF STOCK!
    \nPLEASE CHOOSE ANOTHER BOOK AND TRY AGAIN!")
162     return 1
163
164 def ReturnBook(ISBN, ID):
165   if ValidateISBN(ISBN) == False:
166     print("INVALID ISBN NUMBER!!")
167     return 1
168   if SearchBookByISBN(ISBN) == 1:
169     print("ISBN number not found. Check and Try Again!")
170     return 1
171   if SearchPatronByID(ID) == 1:
172     print("Patron ID not found. Check and Try Again!")
173     return 1
174   trans = "UPDATE transactions SET return_date = CURDATE(), returned =
    TRUE WHERE book_isbn = %s AND patron_id = %s AND return_date IS NULL AND
    returned = FALSE"
175   cursor.execute(trans, (ISBN, ID))
176   if cursor.rowcount == 0:
177     print("No book issued to this patron with this ISBN number!")
178     return 1
179   db.commit()
180
181   query = "UPDATE books SET quantity = quantity + 1 WHERE ISBN = %s"
182   cursor.execute(query, (ISBN, ))
183   db.commit()
184
185 def AddPatron(ID, Email, Patron_Name, Subcription_Date):
186   newpatron = ("INSERT INTO patrons (id, email, name, subscription_date)
    VALUES (%s,%s,%s,%s)")
187   cursor.execute(newpatron, (ID, Email, Patron_Name, Subcription_Date))
188   db.commit()
189
190 def RemovePatron(ID):
191   try:
192     deletepatron = "DELETE FROM patrons WHERE id= %s;"
193     cursor.execute(deletepatron, (ID, ))
194     db.commit()
195   except ValueError:
196     db.rollback()
197     return 1
198
199 def EditPatron(ID):
200   val = SearchPatronByID(ID)
201   if val == 1:
202     return 1
203   else:
204     search, headers = val
205     print(tabulate(search, headers=["ID", "Email", "Name", "Subscription
    Date"], tablefmt="pretty"))
206     print("OPTIONS : ")
207     print("(1) ---> Patron ID number")
208     print("(2) ---> Patron Email")
209     print("(3) ---> Patron Name")
210     print("(4) ---> Renew Patron Subcription Date")
211     print("(5) ---> Remove Patron")
```

```
212     option = int(input("Enter OPTION NUMBER of what you want to edit :
    "))
213     if option >= 1 and option <= 5:
214       if option == 1:
215         id = input("ENTER THE NEW 8-DIGIT UNIQUE ID OF PATRON : ")
216         if ID != id and len(id) == 8:
217           query = "UPDATE patrons SET id= %s WHERE id=%s"
218           cursor.execute(query, (id, ID))
219           db.commit()
220         else:
221           print("Failed to edit! Try again!")
222           db.rollback()
223           return 1
224       elif option == 2:
225         Email = input("ENTER THE NEW EMAIL OF PATRON : ")
226         if "@" in Email and "." in Email:
227           cursor.execute("UPDATE patrons SET email= %s WHERE id= %s",
    (Email, ID))
228           db.commit()
229         else:
230           print("Enter a VALID EMAIL and Try Again!")
231           db.rollback()
232           return 1
233       elif option == 3:
234         Name = input("ENTER THE NEW NAME OF PATRON : ")
235         cursor.execute("UPDATE patrons SET name= %s WHERE id= %s",
    (Name, ID))
236         db.commit()
237       elif option == 4:
238         query = "UPDATE patrons SET subscription_date=DATE(NOW()) WHERE
    id=%s"
239         cursor.execute(query, (ID, ))
240         db.commit()
241       elif option == 5:
242         RemovePatron(ID)
243
244 def SearchPatronByID(ID):
245   try:
246     SearchPatron = "SELECT * FROM patrons WHERE id= %s"
247     return TrySQLCommand(SearchPatron, (ID, ))
248   except ValueError:
249     db.rollback()
250     return 1
251
252 def SearchPatronByName(Patron_Name):
253   try:
254     SearchPatron = "SELECT * FROM patrons WHERE name LIKE %s"
255     return TrySQLCommand(SearchPatron, ("%" + Patron_Name + "%", ))
256   except ValueError:
257     db.rollback()
258     return 1
259
260 def ViewTransactions():
261   query = """SELECT t.id AS "ID", b.title AS "Book", p.name AS "Issued
    By", t.issue_date AS "Issued On", t.due_date AS "Due On",
    IFNULL(t.return_date, "Not Returned") AS "Returned On"
```

```
262                 FROM transactions t, books b, patrons p
263                 WHERE t.book_isbn = b.isbn
264                   AND t.patron_id = p.id
265                 ORDER BY t.issue_date, t.book_isbn;"""
266     cursor.execute(query)
267     columns = [desc[0] for desc in cursor.description]
268     data = cursor.fetchall()
269     print(tabulate(data, headers=columns, tablefmt="pretty"))
270 
271 def ViewTransactionsPending():
272     query = """SELECT t.id "ID", b.title "Book", p.name "Issued By",
    t.issue_date "Issued On", t.due_date "Due On", IFNULL(t.return_date,
    "Not Returned") "Returned On"
273                 FROM transactions t, books b, patrons p
274                 WHERE t.book_isbn = b.isbn
275                   AND t.patron_id = p.id
276                   AND t.returned = FALSE
277                 ORDER BY t.issue_date, t.book_isbn;"""
278     cursor.execute(query)
279     columns = [desc[0] for desc in cursor.description]
280     data = cursor.fetchall()
281     print(tabulate(data, headers=columns, tablefmt="pretty"))
282 
283 def ViewPatrons():
284     query = """SELECT patrons.id "ID", name "Name", email "Email ID",
    subscription_date "Subscribed on", COUNT(transactions.id) "Unreturned
    Books"
285                 FROM patrons
286                 LEFT JOIN transactions ON patrons.id =
    transactions.patron_id AND transactions.returned = false
287                 GROUP BY patrons.id, email, name, subscription_date;"""
288     cursor.execute(query)
289     columns = [desc[0] for desc in cursor.description]
290     data = cursor.fetchall()
291     print(tabulate(data, headers=columns, tablefmt="pretty"))
292 
293 def ViewBooks():
294     query = "SELECT * FROM books"
295     cursor.execute(query)
296     columns = [desc[0] for desc in cursor.description]
297     data = cursor.fetchall()
298     print(tabulate(data, headers=columns, tablefmt="pretty"))
299 
300 def Query(query, values=None):
301     cursor.execute(query, values)
302     data = cursor.fetchall()
303     return data, [desc[0] for desc in cursor.description]
```

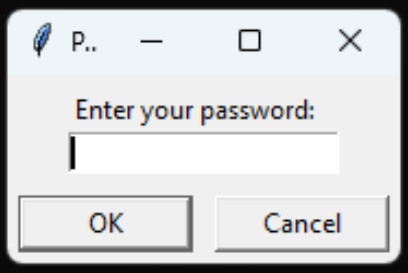
db.sql

```
1  DROP DATABASE IF EXISTS library;
2  
3  CREATE DATABASE library;
4  
5  USE library;
6  
```

```sql
CREATE TABLE books (
    isbn VARCHAR(13) NOT NULL PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    author VARCHAR(50) NOT NULL,
    quantity INT NOT NULL,
    genre ENUM("Fiction", "Non-Fiction") NOT NULL
);

CREATE TABLE patrons (
    id CHAR(8) NOT NULL PRIMARY KEY,
    email VARCHAR(50) NOT NULL,
    name VARCHAR(30) NOT NULL,
    subscription_date DATE NOT NULL,
    CONSTRAINT id_length CHECK (LENGTH(TRIM(id)) = 8),
    CONSTRAINT valid_email CHECK (email LIKE '%@%.%')
);

CREATE TABLE transactions (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    book_isbn VARCHAR(13) NOT NULL,
    patron_id CHAR(8) NOT NULL,
    issue_date DATE NOT NULL,
    due_date DATE NOT NULL,
    return_date DATE,
    returned BOOLEAN NOT NULL DEFAULT FALSE,
    FOREIGN KEY (book_isbn) REFERENCES books(isbn) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (patron_id) REFERENCES patrons(id) ON DELETE CASCADE ON UPDATE CASCADE
);
```

Output

```
python src/main.py
Welcome to the Library Manager!!

ACCESS TO :
(1) ADMIN
(2) USER

Enter your choice: 1
```




```
Enter your choice: 1

Recognised as admin....
Accessing administrative functions.....

        (1) Book functions
        (2) Patron functions
        (3) Transactions and Returns
        (4) Exit
        PLEASE ENTER THE OPTION NUMBER :
```

```
**** BOOK FUNCTIONS ****
 OPTIONS :
(1) Add a book
(2) Remove a book
(3) Update a book
(4) Issue a Book
(5) Return a Book
(6) Search a Book
(7) View all books
(8) Go back to main menu
(9) Exit
PLEASE ENTER THE OPTION NUMBER : 6
         OPTIONS :
        (1) By ISBN number
        (2) By Author
        (3) By Title
        (4) By Genre
        (5) Return to previous menu
        (6) Exit
        PLEASE ENTER THE OPTION NUMBER : 2
                Author : Lee
+---------------+-----------------------+------------+----------+---------+
|     isbn      |         title         |   author   | quantity |  genre  |
+---------------+-----------------------+------------+----------+---------+
| 9780099549482 | To Kill A Mockingbird | Harper Lee |    4     | Fiction |
+---------------+-----------------------+------------+----------+---------+
```

```
**** BOOK FUNCTIONS ****
 OPTIONS :
(1) Add a book
(2) Remove a book
(3) Update a book
(4) Issue a Book
(5) Return a Book
(6) Search a Book
(7) View all books
(8) Go back to main menu
(9) Exit
PLEASE ENTER THE OPTION NUMBER : 7
+---------------+-----------------------------------+-----------------+----------+---------+
|     isbn      |               title               |      author     | quantity |  genre  |
+---------------+-----------------------------------+-----------------+----------+---------+
| 9780007525508 |             The Hobbit            |  J.R.R. Tolkien |    3     | Fiction |
| 9780099549482 |        To Kill A Mockingbird      |    Harper Lee   |    4     | Fiction |
| 9780140430721 |         Pride and Prejudice       |   Jane Austen   |    19    | Fiction |
| 9780439362139 | Harry Potter and the Sorcerer's Stone |  J.K. Rowling  |    5     | Fiction |
| 9780743297332 |         The Sun Also Rises        | Ernest Hemingway |    4     | Fiction |
+---------------+-----------------------------------+-----------------+----------+---------+
```

```
              **** TRANSACTIONS ****
               OPTIONS :
              (1) View Transactions
              (2) View all pending returns
              (3) Return to previous menu
              (4) Exit
              PLEASE ENTER THE OPTION NUMBER : 1
+----+--------------------------------------+-------------+------------+------------+--------------+
| ID |                 Book                 |  Issued By  | Issued On  |   Due On   | Returned On  |
+----+--------------------------------------+-------------+------------+------------+--------------+
| 2  |         To Kill A Mockingbird        | Jane Smith  | 2023-08-02 | 2023-08-09 |  2023-08-14  |
| 5  |         To Kill A Mockingbird        |  John Doe   | 2023-08-05 | 2023-08-12 | Not Returned |
| 7  |         To Kill A Mockingbird        | Jane Smith  | 2023-08-13 | 2023-08-20 | Not Returned |
| 8  |          Pride and Prejudice         |  John Doe   | 2023-08-13 | 2023-08-20 | Not Returned |
| 9  | Harry Potter and the Sorcerer's Stone |  John Doe   | 2023-08-13 | 2023-08-20 | Not Returned |
| 1  |          The Sun Also Rises          |  John Doe   | 2023-08-01 | 2023-08-08 |  2023-08-13  |
| 4  |          The Sun Also Rises          | Emily Brown | 2023-08-04 | 2023-08-11 |  2023-08-10  |
+----+--------------------------------------+-------------+------------+------------+--------------+
```

```
Recognised as our registered patron....
Accessing patron functions.....

        OPTIONS :
        (1) Search a book
        (2) View all books
        (3) View my issued books
        (4) Issue a book
        (5) Return a book
        (6) Exit
        PLEASE ENTER THE OPTION NUMBER :
```

```
       OPTIONS :
       (1) Search a book
       (2) View all books
       (3) View my issued books
       (4) Issue a book
       (5) Return a book
       (6) Exit
       PLEASE ENTER THE OPTION NUMBER : 3
+-----------------------------+----------------+------------+------------+--------------+
|             Book            |    Issued by   | Issued On  |  Due Date  | Returned On  |
+-----------------------------+----------------+------------+------------+--------------+
| Their Eyes Were Watching God | Michael Johnson | 2023-08-03 | 2023-08-17 | Not Returned |
+-----------------------------+----------------+------------+------------+--------------+
```

Bibliography

[1] “MySQL 8.1 Reference”. [Online]. Available: https://dev.mysql.com/doc/refman/8.1/en/

[2] “MySQL Connector/Python Developer's Guide”. [Online]. Available: https://dev.mysql.com/doc/connector-python/en/

[3] “tabulate”, Python Package Index (PyPI). [Online]. Available: https://pypi.org/project/tabulate/

[4] “tkinter.simpledialog”, The Python Standard Library Documentation. [Online]. Available: https://docs.python.org/3/library/dialog.html

[5] “time”, The Python Standard Library Documentation. [Online]. Available: https://docs.python.org/3/library/time.html