SAINT GIRI SR. SEC. SCHOOL

COMPUTER SCIENCE

2024-2025

LIBRARY MANAGEMENT

BASED ON CONNECTIVITY OF PYTHON AND MYSQL

SUBMITTED BY

NAME: SHIVANGI
BOARD ROLL NUMBER:
CLASS: XII-B

ACKNOWLEDGEMENT

Apart from the efforts of me, the success of any project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.

I express deep sense of gratitude to almighty God for giving me strength for the successful completion of the project.

I express my heartfelt gratitude to my parents for constant encouragement while carrying out this project.

I gratefully acknowledge the contribution of the individuals who contributed in bringing this project up to this level, who continues to look after me despite my flaws,

I express my deep sense of gratitude to the luminary . Ms. SANYOGITA, The Principal, Saint Giri Sr. Sec. School who has been continuously motivating and extending their helping hand to us.

My sincere thanks to MS. HARSHITA MAHINDRA , PGT Computer Science and IP, A guide, Mentor all the above a friend, who critically reviewed my project and helped in solving each and every problem, occurred during implementation of the project

The guidance and support received from all the members who contributed and who are contributing to this project, was vital for the success of the project.

I am grateful for their constant support and help.

STUDENT SIGNATURE

SAINT GIRI SR. SEC. SCHOOL

COMPUTER SCIENCE

CERTIFICATE

This is to certify that **Shivangi**, a student of class XII has successfully completed her project based on connectivity of Python andMySQL, under the guidance of MS. Harshita Mahindra, PGT Computer Science and IP during the year 2024-25, in fulfillment of computer science practical of conducted by AISSCE, Delhi.

SUBJECT TEACHER

EXTERNAL EXAMINER

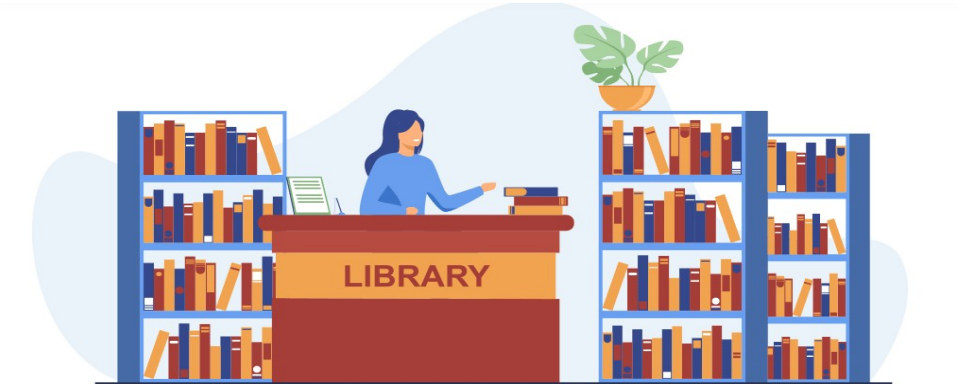
Table of Contents

I am submitting the assignment for a group project on behalf of both members of the group, SHIVANGI AND AASHISH of class XIIth .

It is hereby confirmed that the submission is authorized by both members of the group, We declare that the assignment here submitted is original except for source material explicitly acknowledged/all members of the group have read and checked that all parts of the piece of work, irrespective of whether they are contributed by individual members or all members as a group, here submitted are original except for source material .the submitted soft copy with details listed is identical to the hard copy(ies), if any, which has(have) been / is(are) going to be submitted.

INTRODUCTION:

- The project LIBRARY MANAGEMENT SYSTEM (DIGITAL LIBRARY) includes enrolment of users, adding of books into the library system. The software has the facility to search for news, Wikipedia articles. It includes an authentication facility for admin and user to login into the admin panel and user panel resp. of the system. User can see the books available, details of books issued by the user in the digital library. The Library Management System can be login using a user ID and password. It is accessible either by an admin or user. Only the admin can add, delete and update the data of users and books into the database. The data can be retrieved easily. The interface is very user-friendly. The data are well protected for personal use and makes the data processing very fast. The purpose of the project entitled as “DIGITAL LIBRARY” is to computerize the Front Library Management to develop software which is user friendly, simple, fast, and costeffective. It also has a notes facility where the user can add notes at any point of the program into the database.



LIBRARY MANAGEMENT SYSTEM

Library management system (LMS), is an enterprise resource planning system for a library, used to track enrolled users, available books, books issued and to whom, books returned and it's fines, etc. The purpose of a library management system is to operate a library with efficiency and at reduced costs. The system being entirely automated streamlines all the tasks involved in operations of the library. The library management system software helps in reducing operational costs. Managing a library manually is labour intensive and an immense amount of paperwork is involved. The system saves time for both the user and the librarian. With just a click the user can search for the books available in the library. The librarian can answer queries with ease regarding the availability of books. Adding, removing or editing the database is a simple process. Adding new users or cancelling existing userships can be done with ease. The automated system saves a considerable amount of time as opposed to the manual system.


SOURCE CODE

PYTHON CODE(MAIN FILE)

```python
# Importing necessary libraries
import mysql.connector
import pyfiglet
import datetime as date
import functions as f
import notes as n
import aboutlibrary as abl
import adminwork as ad

#connect to sql
db = mysql.connector.connect(host="localhost",
                             user="root",
                             password="PASSWORD",
                             database="library")
c=db.cursor()

#home menu

def home():
    while True:
        print("==============================================================================")
        print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
        print(pyfiglet.figlet_format("Welcome to",width=1000))
        print(pyfiglet.figlet_format("Digital Library",width=1000))

        print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")

        print("==============================================================================")
        print("--------------------------")
        print("Home")
        print("--------------------------")
        print("1. Admin")
        print("2. User")
        print("3. Exit")
        userChoice = int(input("Enter your Choice to Continue : "))
        print("--------------------------")
         # Handle user choices
        if userChoice == 1:
            authAdmin()
        elif userChoice == 2:
            authUser()
        elif userChoice == 3:
            f.exiting()
        else:
            f.validOption()
            home()

def authAdmin():
    print("--------------------------")
    print("Admin Authentication")
    print("--------------------------")
    adminId = int(input("Enter the Admin's User ID : "))
    adminPassword = input("Enter the Admin's User Password : ")

    #Check if the entered admin ID exists
    c.execute("SELECT password FROM users WHERE userId=%s", (adminId,))
```

```
    result = c.fetchall()
    db.commit()

    # If the entered admin ID is not valid, print an error message
    if len(result) == 0:
        print("--------------------------")
        print("Please enter a valid admin id!")
        print("--------------------------")
        authAdmin()
    else:
     # If the entered password matches the admin's password
        if adminPassword == result[0][0]:
            global USERID
            USERID = adminId
            print("\033[0;35m-------------------------\033[0;0m")
            print("\033[0;36m ADMIN is verified successfully.\033[0;0m")
            print("\033[0;35m-------------------------\033[0;0m")
            admin(USERID)
            # Call the user menu
        else:
            print("Please Enter a Valid Password!")
            print("--------------------------")

def authUser():
    print("----------------------------")
    print("User Authentication")
    print("--------------------------")
    userId = int(input("Enter the User ID : "))
    password = input("Enter the User Password : ")

    #Check if the entered user ID exists
    c.execute("SELECT password FROM users WHERE userId=%s", (userId,))
    result = c.fetchall()
    db.commit()

    # If the entered user ID is not valid, print an error message
    if len(result) == 0:
        print("--------------------------")
        print("Please enter a valid user id!")
        print("--------------------------")
        authUser()
    else:
     # If the entered password matches the user's password
        if password == result[0][0]:
            global USERID
            USERID = userId
            print("\033[0;35m-------------------------\033[0;0m")
            print("\033[0;36mUser is verified successfully.\033[0;0m")
            print("\033[0;35m-------------------------\033[0;0m")
            user(USERID)
             # Call the user menu
        else:
            print("Please Enter a Valid Password!")
            print(" -------------------------")


def admin(USERID):
    print("--------------------------")
    print("ADMIN")
    print("--------------------------")
    print("""
          1. Login into User Panel
          2. Modify User
```

```
            3. Display Users
            4. Search Users
            5. Modify Book
            6. Issue Book
            7. Return Book
            8. Change Admin
            9. Home
            10.Back
            11.Exit
            """)
    userChoice = int(input("Enter your choice to continue : "))
    print("--------------------------")

    #handle user choice
    if userChoice == 1:
            print("You are successfully logged into user's Panel.")
            print("------------------------------")
            user(USERID)
    elif userChoice == 2:
        ad.modifyUser()
        admin(USERID)
    elif userChoice == 3:
        ad.displayUsers()
        menu(USERID)
    elif userChoice ==4:
        ad.searchUsers()
        menu(USERID)
    elif userChoice ==5:
        abl.modifyBook()##
        admin(USERID)
    elif userChoice ==6:
        abl.issueBook()
        admin(USERID)
    elif userChoice ==7:
        abl.returnBook()
        admin(USERID)
    elif userChoice ==8:
        ad.changeadmin()
        menu(USERID)
    elif userChoice == 9:
        home()
    elif userChoice == 10:
        authAdmin()
    elif userChoice == 11:
        f.exiting()
    else:
        f.validOption()


#Function to display the user menu
def user(USERID):
    print("--------------------------")
    print("USER")
    print("--------------------------")
    # Check if the entered user ID exists
    c.execute('SELECT userId FROM users WHERE adminStatus="admin"')
    result = c.fetchall()
    db.commit()

    if result[0][0] ==USERID:
        print("1.   Login into Admin Panel")
        print("2.   About the Library")
        print("3.   Dislpay Books")
```

```
183.          print("4.   Search Books")
184.          print("5.   Issued Book Details")
185.          print("6.   Notes")
186.          print("7.   Home")
187.          print("8.  Back")
188.          print("9.  Exit")
189.          userChoice = int(input("Enter your choice to continue : "))
190.          print("--------------------------")
191.
192.          # Handle user choices
193.          if userChoice == 1:
194.               print("You are successfully logged into Admin's Panel.")
195.               print("-------------------------------------------------")
196.               admin(USERID)
197.          elif userChoice == 2:
198.              abl.aboutLibrary()
199.              userMenu(USERID)
200.          elif userChoice == 3:
201.              abl.displayBooks()
202.              user(USERID)
203.          elif userChoice == 4:
204.              abl.searchBooks()
205.              user(USERID)
206.          elif userChoice == 5:
207.              abl.issuedBooksDetails(USERID)
208.              userMenu(USERID)
209.          elif userChoice == 6:
210.              n.notes(USERID)
211.              addNoteMenu(USERID)
212.          elif userChoice == 7:
213.              home()
214.          elif userChoice == 8:
215.              admin(USERID)
216.          elif userChoice == 9:
217.              f.exiting()
218.          else:
219.              f.validOption()
220.      else:
221.
222.          print("1. About Library")
223.          print("2. Display Books")
224.          print("3. Search Books")
225.          print("4. Issued Books Details")
226.          print("5. Notes")
227.          print("6. Home")
228.          print("7. Back")
229.          print("8. Exit")
230.          userChoice = int(input("Enter your Choice to Continue : "))
231.          print("--------------------------")
232.
233.          # Handle user choices
234.          if userChoice == 1:
235.              abl.aboutLibrary()
236.              userMenu(USERID)
237.          elif userChoice == 2:
238.              abl.displayBooks()
239.              user(USERID)
240.          elif userChoice == 3:
241.              abl.searchBooks()
242.              user(USERID)
243.          elif userChoice == 4:
244.              abl.issuedBooksDetails(USERID)
245.              userMenu(USERID)
```

```
246.         elif userChoice == 5:
247.             n.notes(USERID)
248.             addNoteMenu(USERID)
249.         elif userChoice == 6:
250.             home()
251.         elif userChoice == 7:
252.              userMenu(USERID)
253.         elif userChoice == 8:
254.             f.exiting()
255.         else:
256.             f.validOption()
257.
258. # Function to display the user menu and handle user choices
259. def userMenu(USERID):
260.     print("""1. Add Note
261. 2. Home
262. 3. Back
263. 4. Exit""")
264.     userchoice=int(input("ENTER CHOICE\n"))
265.     print("------------------------------")
266.     if userchoice==1:
267.         n.addnote(USERID)
268.         addNoteMenu(USERID)
269.     elif userchoice==2:
270.         home()
271.     elif userchoice==3:
272.         user(USERID)
273.     elif userchoice==4:
274.         f.exiting()
275.     else:
276.         f.validOption()
277.         userMenu()
278.
279. def addNoteMenu(USERID):
280.     print("""1.HOME
281.          2. BACK
282.          3. User Panel
283.          4. EXIT
284.          """)
285.     userChoice=int(input("Enter your  choice"))
286.
287.     #handle user choice
288.     if userChoice ==1:
289.         home()
290.     elif userChoice ==2:
291.         n.notes(USERID)
292.         addNoteMenu(USERID)
293.     elif userChoice==3:
294.         userMenu(USERID)
295.     elif userChoice==4:
296.         f.exiting()
297.     else:
298.         f.validOption()
299.
300. def menu(USERID):
301.     print("""
302.           1.HOME
303.           2.USERPANEL
304.           3.ADMIN PANEL
305.           4.EXIT""")
306.
307.     userChoice=int(input("Enter your  choice"))
308.     #handle user choice
```

```
    if userChoice ==1:
        home()
    elif userChoice ==2:
        userMenu(USERID)
    elif userChoice==3:
        authAdmin()
    elif userChoice==4:
        f.exiting()
    else:
        f.validOption()


home()

```

MODULE LIBRARY;

```
db = mysql.connector.connect(host="localhost",
                             user="root",
                             password="PASSWORD",
                             database="library")
c=db.cursor()

def aboutLibrary():
    # Retrieve the name of the librarian who is also an admin
    c.execute("SELECT userName FROM users WHERE adminStatus='admin'")
    userName = c.fetchall()

    # Retrieve the total number of books and users in the library
    c.execute("SELECT * FROM books")
    totalBooks = c.fetchall()

    c.execute("SELECT * FROM users")
    totalUsers = c.fetchall()
    db.commit()

    print("-------------------------------------------------")
    print(pyfiglet.figlet_format("About Library",width=1000))
    print("-------------------------------------------------")
    # Display library information
    print("Year of Library's Establishment : ", 2024)
    print("Name of the Librarian : ", userName[0][0])
    print("Total Number of Books Available in the Library : ",
          len(totalBooks))
    print("Total Number of Users Enrolled in the Library : ",
          len(totalUsers))
    print("-------------------------------------------------")

def modifyBook():
    print("--------------------------")
    print("Modify Book")
    print("--------------------------")
```

```
    # Book modification menu
    print("1. Add Book")
    print("2. Delete Book")
    print("3. Update Book Details")
    print("4. Back")
    print("5. Exit")
    userChoice = int(input("Enter your Choice to Continue : "))
    print("--------------------------")

    # User choices handling
    if userChoice == 1:
        addBook()
    elif userChoice == 2:
        deleteBook()
    elif userChoice == 3:
        updateBook()
    elif userChoice == 4:
        pass
    elif userChoice == 5:
        f.exiting()
    else:
        f.validOption()

def deleteBook():
    print("--------------------------")
    print("Delete Book")
    print("--------------------------")
    # Get user input for the book ID to be deleted
    bookId = int(input("Enter the Book ID : "))
    choice = input("Are you sure to delete the Book? (Yes/No) : ")
    print("--------------------------")

    c.execute("SELECT bookId FROM books")
    result = c.fetchall()
    db.commit()

    if choice.lower() in ["yes", "y"]:
        if (bookId,) in result:
            # Execute SQL query to delete the book from the database
            c.execute("DELETE FROM books WHERE bookId=%s", (bookId,))
            db.commit()

            # Notify the user that the book has been deleted successfully
            print("Book deleted Successfully!")
            print("--------------------------")
            modifyBook()
        else:
            print(f'The book of book id "{bookId}" does not available in the digital
library.')
            print("--------------------------")
            modifyBook()
    elif choice.lower() in ["no", "n"]:
        print("--------------------------")
        print("Book Not Deleted!")
        print(" -------------------------")
        modifyBook()
    else:
        f.validOption()

def updateBook():
    print("--------------------------")
    print("Update Book Details")
    print("--------------------------")
```

```
    print("1. Update the Book ID")
    print("2. Update the Book Name")
    print("3. Update the Book Publication Year")
    print("4. Update the Book Author Name")
    print("5. Back")
    print("6. Exit")
    userChoice = int(input("Enter your Choice to Continue : "))
    print("--------------------------")

    c.execute("SELECT bookId FROM books")
    result = c.fetchall()
    db.commit()

    # User choices handling
    if userChoice == 1:
        currentBookId = int(input("Enter the Current Book ID : "))
        newBookId = int(input("Enter the New Book ID : "))

        if (currentBookId,) in result:
        # Execute SQL query to update the Book ID
            c.execute("UPDATE books SET bookId=%s WHERE bookId=%s",
(newBookId,currentBookId))
            db.commit()

            print("Book ID changed Successfully!")
            print(" -------------------------")
            modifyBook()
        else:
            notBook(currentBookId)

    elif userChoice == 2:
        bookId = int(input("Enter the Book ID : "))
        newBookName = input("Enter the New Book Name : ")

        if (bookId,) in result:
            # Execute SQL query to update the Book Name
            c.execute("UPDATE books SET bookName=%s WHERE bookId=%s",(newBookName,
bookId))
            db.commit()

            print("Book Name changed Successfully!")
            print(" -------------------------")
            modifyBook()
        else:
            notBook(bookId)
            updateBook()

    elif userChoice == 3:
        bookId = int(input("Enter the Current Book ID : "))
        newPublicationYear = input("Enter the New Publication Year : ")

        if (bookId,) in result:
            # Execute SQL query to update the Publication Year
            c.execute("UPDATE books SET publicationYear=%s WHERE
bookId=%s",(newPublicationYear, bookId),)
            db.commit()

            print("Book Publication Year changed Successfully!")
            print(" -------------------------")
            modifyBook()

    elif userChoice == 4:
        bookId = int(input("Enter the Current Book ID : "))
```

```
158.         newAuthor = input("Enter the New Author Name : ")
159.
160.         if (bookId,) in result:
161.             # Execute SQL query to update the Author Name
162.             c.execute("UPDATE books SET author=%s WHERE bookId=%s",(newAuthor,
bookId),)
163.             db.commit()
164.
165.             print("Book Author Name changed Successfully!")
166.             print(" -------------------------")
167.             modifyBook()
168.         else:
169.             notBook(bookId)
170.
171.     elif userChoice == 5:
172.         modifyBook()
173.     elif userChoice == 6:
174.         f.exiting()
175.     else:
176.         f.validOption()
177.
178. def notBook(bookId):
179.     print(f'The book of book id "{bookId}" does not available in the digital library.')
180.     print("-----------------------------------------")
181.     updateBook()
182.
183. def issueBook():
184.     print("--------------------------")
185.     print("Issue Book")
186.     print("--------------------------")
187.     bookId = int(input("Enter the Book ID to issue: "))
188.     bookname=input("Enter book_name:")
189.     userId = int(input("Enter the User ID: "))
190.
191.     # Check if the book exists and is available
192.     c.execute("SELECT * FROM books WHERE bookId = %s AND issuestatus = 'not issued'",
(bookId,))
193.     book = c.fetchone()
194.
195.     if book:
196.         # Issue the book to the user
197.         c.execute(
198.     "INSERT INTO issuedbooksdetails (bookId, bookname, userId, issueDate) VALUES (%s,
%s, %s, %s);",
199.     (bookId, bookname, userId, dt.date.today())
200. )
201.         c.execute("UPDATE books SET issueStatus = 'issued' WHERE bookId = %s",
(bookId,))
202.         db.commit()
203.         print(f"Book ID {bookId} has been issued to User ID {userId}.")
204.     else:
205.         print(f"Book ID {bookId} is not available or does not exist.")
206.
207.     print("--------------------------")
208.
209. def returnBook():
210.     print("--------------------------")
211.     print("Return Book")
212.     print("--------------------------")
213.     bookId = int(input("Enter the Book ID to return: "))
214.     userId = int(input("Enter the User ID: "))
215.
216.     # Check if the book is issued to the user
```

```
    c.execute("SELECT issueDate FROM issuedbooksdetails WHERE bookId = %s AND userId = %s", (bookId, userId))
    record = c.fetchone()

    if record:
        issue_date = record[0]
        return_date = dt.date.today()
        days_borrowed = (return_date - issue_date).days
        late_days = max(0, days_borrowed - 14)
        fine = late_days * 5

        # Update the book's status to available
        c.execute("DELETE FROM issuedbooksdetails WHERE bookId = %s AND userId = %s", (bookId, userId))
        c.execute("UPDATE books SET issueStatus = 'not issued' WHERE bookId = %s", (bookId,))
        db.commit()

        print(f"Book ID {bookId} has been returned by User ID {userId}.")
        if fine > 0:
            print(f"You are {late_days} days late. Fine: ₹{fine}")
        else:
            print("No fine incurred.")
    else:
        print(f"No record found for Book ID {bookId} issued to User ID {userId}.")

    print("--------------------------")

def displayBooks():
    print("--------------------------")
    print("Display Books")
    print(" -------------------------")
    # Retrieve all books from the database
    c.execute("SELECT * FROM books ORDER BY bookId")
    result = c.fetchall()
    db.commit()

    # Display books if available, otherwise notify the user
    if result:
        print("Books available in the Digital Library are :")
        print("--------------------------")
        i = 0
        for row in result:
            i += 1
            r = f.length(i)
            print(f"{i}. Book ID : {row[0]}")
            print(" " * r + f"Book Name : {row[1]}")
            print(" " * r + f"Publication Year : {row[2]}")
            print(" " * r + f"Author Name : {row[7]}")
            print(" " * r + f"Issue Status : {row[8]}")
            print(" -------------------------")
    else:
        # Notify the user if no books are found
        print("No books found.")
        print(" -------------------------")

def searchBooks():
    print("--------------------------")
    print("Search Books")
    print("--------------------------")
    print("1. Search by Book ID")
    print("2. Search by Keyword")
    print("3. Back")
```

```
277.     print("4. Exit")
278.     userChoice = int(input("Enter your Choice to Continue : "))
279.     print("--------------------------")
280.
281.     # User choices handling
282.     if userChoice == 1:
283.         searchBooksbyId()
284.     elif userChoice == 2:
285.         searchBooksbyKeyword()
286.     elif userChoice == 3:
287.         pass
288.     elif userChoice == 4:
289.         f.exiting
290.     else:
291.         f.validOption()
292.
293. def searchBooksbyId():
294.     print(" -------------------------")
295.     print("Search Books by Book ID")
296.     print("--------------------------")
297.     # Get user input for Book ID
298.     bookId = int(input("Enter the Book ID to search the Book : "))
299.     print("--------------------------")
300.
301.     # Execute SQL query to retrieve book information by Book ID
302.     c.execute("SELECT * FROM books WHERE bookId=%s", (bookId,))
303.     result = c.fetchall()
304.     db.commit()
305.
306.     # Display search results if books are found, otherwise notify the user
307.     if result:
308.         print(f'Book available in the Digital Library with the Book ID "{bookId}" is
:')
309.         print("--------------------------")
310.         i = 0
311.         for row in result:
312.             i += 1
313.             r = f.length(i)
314.             print(f"{i}. Book ID : {row[0]}")
315.             print(" " * r + f"Book Name : {row[1]}")
316.             print(" " * r + f"Publication Year : {row[2]}")
317.             print(" " * r + f"Author Name : {row[7]}")
318.             print(" " * r + f"Issue Status : {row[8]}")
319.             print(" -------------------------")
320.         searchBooks()
321.     else:
322.         print(f'No book found with the book id "{bookId}".')
323.         print("--------------------------")
324.         searchBooks()
325.
326. def searchBooksbyKeyword():
327.     print("--------------------------")
328.     print("Search Books by Keyword")
329.     print("--------------------------")
330.     # Get user input for keyword
331.     keyword = input("Enter a Keyword to search Books : ")
332.     print("--------------------------")
333.
334.     # Execute SQL query to retrieve books by keyword
335.     c.execute("SELECT * FROM books WHERE bookName LIKE '%{}%' ORDER BY
bookId".format(keyword))
336.     result = c.fetchall()
337.     db.commit()
```

```
338.
339.     # Display search results if books are found, otherwise notify the user
340.     if result:
341.         print(f'Books available in the Digital Library with the Keyword "{keyword}"
are: ')
342.         print("--------------------------")
343.         i = 0
344.         for row in result:
345.             i += 1
346.             r = f.length(i)
347.             print(f"{i}. Book ID : {row[0]}")
348.             print(" " * r + f"Book Name : {row[1]}")
349.             print(" " * r + f"Publication Year : {row[2]}")
350.             print(" " * r + f"Author Name : {row[7]}")
351.             print(" " * r + f"Issue Status : {row[8]}")
352.             print(" -------------------------")
353.         searchBooks()
354.     else:
355.         print(f'No books found with the keyword "{keyword}".')
356.         print("--------------------------")
357.         searchBooks()
358.
359. def issuedBooksDetails(USERID):
360.     print("--------------------------")
361.     print("Issued Books Details")
362.     print("--------------------------")
363.     f.returnPolicy()
364.
365.     c.execute("SELECT * FROM issuedBooksDetails WHERE userId=%s ORDER BY
bookId",(USERID,))
366.     result = c.fetchall()
367.     db.commit()
368.
369.     if result == []:
370.         print("No Books Issued!")
371.         print(" -------------------------")
372.         pass
373.
374.     else:
375.         i = 0
376.         for row in result:
377.             i += 1
378.             r = f.length(i)
379.             print(f"{i}. Book ID : ", row[1])
380.             print(" " * r + "Book Name : ", row[2])
381.             print(" " * r + "Issue Date : ", row[3])
382.             print(" " * r + "Issue Time : ", row[4])
383.             print(" " * r + "Return Date : ", row[5])
384.             print(" " * r + "Return Time : ", row[6])
385.             print(" " * r + "Fine(in Rs.) : ", row[7])
386.             print(" -------------------------")
387.             pass
388.
389. def addBook():
390.     print("--------------------------")
391.     print("Add Book")
392.     print("--------------------------")
393.     # Get user input for book details
394.     try:
395.         bookId = int(input("Enter the Book ID : "))
396.         bookName = input("Enter the Book Name : ")
397.         publicationYear = int(input("Enter the Book Publication Year : "))
398.         author = input("Enter the Book Author Name : ")
```

```
399.         print("--------------------------")
400.
401.         # Check if the book ID already exists
402.         c.execute("SELECT bookId FROM books WHERE bookId = %s", (bookId,))
403.         result = c.fetchone()  # Fetch a single matching row, if any
404.
405.         if result:
406.             print(f'The book with ID "{bookId}" is already available in the digital
library.')
407.             print(" --------------------------")
408.             modifyBook()  # Call the modifyBook function to handle existing books
409.         else:
410.             # Insert the new book into the database
411.             c.execute(
412.                 """INSERT INTO books (bookId, bookName, publicationYear, author)
413.                    VALUES (%s, %s, %s, %s)""",
414.                 (bookId, bookName, publicationYear, author),
415.             )
416.             db.commit()  # Commit changes to the database
417.
418.             print("Book added Successfully!")
419.             print(" --------------------------")
420.             modifyBook()  # Optional: Call modifyBook to handle further actions
421.
422.     except ValueError:
423.         print("Invalid input! Please ensure you enter numeric values for Book ID and
Publication Year.")
424.         print("--------------------------")
425.     except Exception as e:
426.         print(f"An error occurred: {e}")
427.         print("--------------------------")
428.
```

MODULE NOTES

```
 1. import mysql.connector
 2. import functions as f
 3.
 4. #connect to sql
 5. db = mysql.connector.connect(host="localhost",
 6.                               user="root",
 7.                               password="PASSWORD",
 8.                               database="library")
 9. c=db.cursor()
10.
11. # Function to manage notes
12.
13. def notes(USERID):
14.     print("--------------------------")
15.     print("Notes")
16.     print("--------------------------")
17.     # Display menu options
18.     print("""
19.           1.Modify Note
20.           2.Display Notes
21.           3.Search Notes
```

```
22.          4.TO add notes menu
23.          5.exit """)
24.    userChoice = int(input("Enter your Choice to Continue : "))
25.    print("--------------------------")
26.
27.    if userChoice == 1:
28.       modifyNote(USERID)
29.    elif userChoice == 2:
30.       displayNotes(USERID)
31.    elif userChoice==3:
32.       searchNotes(USERID)
33.    elif userChoice==4:
34.       pass
35.    elif userChoice==5:
36.       f.exiting()
37.    else:
38.       f.validOption()
39.
40. def addnote(USERID):
41.    print("--------------------------")
42.    print("Add Note")
43.    print("--------------------------")
44.    # Get note details from the user
45.    noteNumber = int(input("Enter the Note Number : "))
46.    print("--------------------------")
47.
48.    c.execute("SELECT noteNumber FROM notes where userId=%s", (USERID,))
49.    result = c.fetchall()
50.    db.commit()
51.
52.    if (noteNumber,) in result:
53.       print( f'The note of note number "{noteNumber}" is already exists in the digital
library.')
54.       print("--------------------------")
55.
56.
57.    else:
58.       noteTitle =str(input("Enter the Note Title : "))
59.       noteDescription = str(input("Enter the Note Description : "))
60.       print("--------------------------")
61.       # Execute SQL query to insert the note into the database
62.       c.execute("""INSERT INTO notes (userId, noteNumber, noteTitle, noteDescription,
updateDate, updateTime)
63.                VALUES ({}, {}, '{}', '{}', CURRENT_DATE,
CURRENT_TIME)""".format(USERID, noteNumber, noteTitle, noteDescription))
64.       db.commit()
65.    print(f'The note of note number "{noteNumber}" is added successfully.')
66.    print("--------------------------")
67.
68. def modifyNote(USERID):
69.    print(" -------------------------")
70.    print("Modify Notes")
71.    print("--------------------------")
72.    # Display menu options
73.    print("""
74.          1.Delete Note
75.          2.Update note
76.          3.Back to menu
77.          4.Exit""")
78.    userChoice = int(input("Enter your Choice to Continue : "))
79.    print("--------------------------")
80.    if userChoice==1:
81.       deleteNote(USERID)
```

```
82.     elif userChoice == 2:
83.        updateNotes(USERID)
84.     elif userChoice==3:
85.        notes(USERID)
86.     elif userChoice==4:
87.        f.exiting()
88.     else:
89.        f.validOption()
90.
91.
92. def searchNotes(USERID):
93.     print(" -------------------------")
94.     print("Search Notes")
95.     print("-------------------------")
96.     # Display search options
97.     print("1. Search by Note Number")
98.     print("2. Search by Keyword")
99.     print("3. Home")
100.    print("4. Back")
101.    print("5. Exit")
102.     # Get user choice
103.    userChoice = int(input("Enter your Choice to Continue : "))
104.    print("-------------------------")
105.
106.     # Handle user choices
107.    if userChoice == 1:
108.       searchNotesbynoteNumber(USERID)
109.    elif userChoice == 2:
110.       searchNotesbyKeyword(USERID)
111.    elif userChoice == 3:
112.       notes(USERID)
113.    elif userChoice == 4:
114.       modifyNote(USERID)
115.    elif userChoice == 5:
116.       f.exiting()
117.    else:
118.       f.validOption()
119.
120. def searchNotesbynoteNumber(USERID):
121.    # Get the note number to search
122.     noteNumber = int(input("Enter the Note Number to search the Note : "))
123.
124.     # Execute SQL query to fetch notes with the given note number
125.     c.execute("SELECT * FROM notes WHERE notenumber=%s", (noteNumber,))
126.     result = c.fetchall()
127.     db.commit()
128.
129.     # Check if notes are found
130.     if result:
131.         print(f'Note available in the Digital Library with the Note Number
"{noteNumber}" is: ')
132.         i = 0
133.         for row in result:
134.             i += 1
135.             r = f.length(i)
136.             print(f"{i}. Note Number : {row[1]}")
137.             print(" " * r + f"Note Title : {row[2]}")
138.             print(" " * r + f"Note Description : {row[3]}")
139.             print(" -------------------------")
140.         searchNotes(USERID)
141.     else:
142.         # If no notes are found with the given note number
143.         print(f'No note found with the note number "{noteNumber}".')
```

```
144.         print("--------------------------")
145.         searchNotes(USERID)
146.
147. def searchNotesbyKeyword(USERID):
148.     print("--------------------------")
149.     print("Search Notes by Keyword")
150.     print("--------------------------")
151.
152.      # Get keyword from user
153.     keyword = input("Enter a Keyword to search Notes : ")
154.
155.      # Execute SQL query to fetch notes with the given keyword in the title
156.     c.execute("SELECT * FROM notes WHERE noteTitle LIKE '%{}%' ORDER BY noteNumber
".format(keyword))
157.     result = c.fetchall()
158.     db.commit()
159.
160.      # Check if notes are found
161.     if result:
162.        print(f'Notes available in the Digital Library with the Keyword "{keyword}" are:
')
163.        i = 0
164.        for row in result:
165.           i += 1
166.           r = f.length(i)
167.           print(f"{i}. Note Number : {row[1]}")
168.           print(" " * r + f"Note Title : {row[2]}")
169.           print(" " * r + f"Note Description : {row[3]}")
170.           print("--------------------------")
171.           searchNotes(USERID)
172.     else:
173.        # If no notes are found with the given keyword
174.        print(f'No notes found with the keyword "{keyword}".')
175.        print("--------------------------")
176.        searchNotes(USERID)
177.
178. def displayNotes(USERID):
179.     # Fetch all notes from the database
180.     c.execute("SELECT * FROM notes ORDER BY noteNumber")
181.     result = c.fetchall()
182.     db.commit()
183.
184.     # Check if there are notes available
185.     if result:
186.       print(f"Notes available in the Digital Library are :")
187.       i = 0
188.       for row in result:
189.           i += 1
190.           r = f.length(i)
191.           print(f"{i}. Note Number : {row[1]}")
192.           print(" " * r + f"Note Title : {row[2]}")
193.           print(" " * r + f"Note Description : {row[3]}")
194.           print(" " * r + f"Update Date : {row[4]}")
195.           print(" " * r + f"Update Time : {row[5]}")
196.           print("--------------------------")
197.           searchNotes(USERID)
198.     else:
199.       # If no notes are found
200.        print("No notes found.")
201.        print(" -------------------------")
202.        searchNotes(USERID)
203.
204. def deleteNote(USERID):
```

```
205.     print("--------------------------")
206.     print("DELETE Notes")
207.     print("--------------------------")
208.     # Get note number to be deleted from the user
209.     noteNumber = int(input("Enter the Note Number to Delete the Note : "))
210.     choice = input("Are you sure to delete the Note? (Yes/No) : ")
211.     print("--------------------------")
212. 
213.     c.execute("SELECT noteNumber FROM notes where userId=%s", (USERID,))
214.     result = c.fetchall()
215.     db.commit()
216. 
217.     if choice.lower() in ["yes", "y","Y","Yes"]:
218.        if (noteNumber,) in result:
219.     # Execute SQL query to delete the note from the database
220.           c.execute("delete FROM notes WHERE userId=%s and noteNumber=%s",
221.                     (USERID, noteNumber),)
222.           db.commit()
223. 
224.           print(f'The note of note number "{noteNumber}" is deleted successfully.')
225.           print("--------------------------")
226.           modifyNote(USERID)
227. 
228.        else:
229.           print(
230.               f'The note of note number "{noteNumber}" does not exists in the digital
library.')
231.           print("--------------------------")
232.           modifyNote(USERID)
233. 
234.     elif choice.lower() in ["no", "n","N","No"]:
235.        print("--------------------------")
236.        print("Note Not Deleted!")
237.        print("--------------------------")
238.        modifyNote(USERID)
239. 
240.     else:
241.        f.validOption()
242. 
243. def notNote(noteNumber):
244.     print(f'The note of note number "{noteNumber}" does not exists in the digital
library.' )
245.     print("--------------------------")
246. 
247. 
248. def updateNotes(USERID):
249.     print(" ------------------------")
250.     print("Update Notes")
251.     print("--------------------------")
252.      # Display update options
253.     print("1. Update the Note Number")
254.     print("2. Update the Note Title")
255.     print("3. Update the Note Description")
256.     print("4. Back")
257.     print("5. Exit")
258.      # Get user choice
259.     userChoice = int(input("Enter your Choice to Continue : "))
260.     print("--------------------------")
261. 
262.     c.execute("SELECT noteNumber FROM notes where userId=%s", (USERID,))
263.     result = c.fetchall()
264.     db.commit()
265. 
```

```
266.     # Handle user choices
267.    if userChoice == 1:
268.       # Update Note Number
269.       currentNoteNumber = int(input("Enter the Current Note Number : "))
270.       newNoteNumber = int(input("Enter the New Note Number : "))
271.
272.       if (currentNoteNumber,) in result:
273.          # Update date and time
274.          c.execute("update notes set updateDate=CURRENT_DATE where userId=%s and
noteNumber=%s", (USERID, currentNoteNumber),)
275.          c.execute("update notes set updateTime=CURRENT_TIME where userId=%s and
noteNumber=%s", (USERID, currentNoteNumber),)
276.          #Update Note Number
277.          c.execute("update notes set noteNumber=%s where userId=%s and
noteNumber=%s",(newNoteNumber, USERID, currentNoteNumber),)
278.          db.commit()
279.
280.          print("Note Number changed Successfully!")
281.          print("--------------------------")
282.          updateNotes(USERID)
283.       else:#1502
284.          notNote(currentNoteNumber)
285.          updateNotes(USERID)
286.
287.    elif userChoice == 2:
288.         # Update Note Title
289.       noteNumber = int(input("Enter the Current Note Number : "))
290.       newTitle = input("Enter the New Note Title : ")
291.
292.       if (noteNumber,) in result:
293.             # Update date and time
294.          c.execute("update notes set updateDate=CURRENT_DATE where userId=%s and
noteNumber=%s",(USERID, noteNumber),)
295.          c.execute("update notes set updateTime=CURRENT_TIME where userId=%s and
noteNumber=%s",(USERID, noteNumber),)
296.          # Update Note Title
297.          c.execute("update notes set noteTitle=%s where userId=%s and
noteNumber=%s",(newTitle, USERID, noteNumber),)
298.          db.commit()
299.
300.          print("Note Title changed Successfully!")
301.          print("--------------------------")
302.          updateNotes(USERID)
303.       else:
304.          notNote(noteNumber)
305.          updateNotes(USERID)
306.
307.    elif userChoice == 3:
308.      # Update Note Description
309.       noteNumber = int(input("Enter the Current Note Number : "))
310.       newDescription = input("Enter the New Note Description : ")
311.       if (noteNumber,) in result:
312.       # Update date and time
313.          c.execute("update notes set updateDate=CURRENT_DATE where userId=%s and
noteNumber=%s",(USERID, noteNumber),)
314.          c.execute("update notes set updateTime=CURRENT_TIME where userId=%s and
noteNumber=%s",(USERID, noteNumber),)
315.          # Update Note Description
316.          c.execute("update notes set noteDescription=%s where userId=%s and
noteNumber=%s",(newDescription, USERID, noteNumber),)
317.          db.commit()
318.
319.          print("Note Description changed successfully!")
```

```
            print("-------------------------")
            updateNotes(USERID)
        else:
            notNote(noteNumber)
            updateNotes(USERID)

    elif userChoice == 4:
        modifyNote(USERID)
    elif userChoice == 5:
        f.exiting()
    else:
        f.validOption()

```

MODULE FUNCTIONS;

```
import pyfiglet

#returnbook#issuebook
#improvise
#Function to calculate the length of a given integer after converting it to a string
def length(i, padding=2):
    """
    Calculate the length of the string representation of an integer with optional
padding.

    :param i: The integer to measure.
    :param padding: Additional padding to add to the length. Default is 2.
    :return: Length of the integer as a string with padding.
    """
    try:
        s = str(i)
        return len(s) + padding
    except Exception as e:
        print(f"Error in length calculation: {e}")
        return 0

# Function to display a message for an invalid option
def validOption():
    print("----------------------------------------")
    print("Please enter a valid option!!!!")
    print("----------------------------------------")

# Function to handle program exit
def exiting():
    """
    Handle program exit with a confirmation prompt.
    """
    confirm = input("Are you sure you want to exit? (yes/no): ").strip().lower()
    if confirm in ["yes", "y"]:
        print("----------------------------------------")
        print(pyfiglet.figlet_format("Exiting the program."))
        print(pyfiglet.figlet_format("Thank  You!"))
        print("----------------------------------------")
```

```
        exit()
    else:
        print("Returning to the program.")

#function to display return policy statement
def returnPolicy():

    print("Return Policy : ")
    print("The issued book should be returned within 14 days(2 weeks).")
    print("""If the user kept the issued book for more than 14 days,
then the user have to pay ₹5 as fine for each extra day
the user kept the issued book.""")

    print("--------------------------")

```

MODULE ADMINWORK;

```

import functions as f
import mysql.connector
db = mysql.connector.connect(host="localhost",
                             user="root",
                             password="PASSWORD",
                             database="library")
c=db.cursor()

def modifyUser():
    print("--------------------------")
    print("Modify User")
    print("--------------------------")
    # Display user modification options
    print("1. Add User")
    print("2. Delete User")
    print("3. Update User Details")
    print("4. Back to admin")
    print("5. Exit")
    # Get user choice
    userChoice = int(input("Enter your Choice to Continue : "))
    print("--------------------------")

    # User choices handling
    if userChoice == 1:
        # Add a new user
        addUser()
    elif userChoice == 2:
        # Delete a user
        deleteUser()
    elif userChoice == 3:
        # Update user details
        updateUser()
    elif userChoice == 4:
```

```
34.         # Return to the previous menu
35.         pass
36.     elif userChoice == 5:
37.         # Exit the program
38.         f.exiting()
39.     else:
40.         f.validOption()
41.
42. def addUser():
43.     print("--------------------------")
44.     print("Add User")
45.     print("--------------------------")
46.     # Get user input for user details
47.     userId = int(input("Enter the User ID : "))
48.     userName = input("Enter the User Name : ")
49.     userPhoneNumber = input("Enter the User Phone Number : ")
50.     userEmailId = input("Enter the User Email ID : ")
51.     password = input("Enter the User Password : ")
52.     print("--------------------------")
53.
54.     c.execute("SELECT userId FROM users")
55.     result = c.fetchall()
56.     db.commit()
57.
58.     if (userId,) in result:
59.         print(f'The user of user number "{userId}" is already enrolled in the digital
library.')
60.         print("--------------------------")
61.         pass
62.     else:
63.         # Execute SQL query to insert the new user into the database
64.         c.execute("INSERT INTO users (userId, userName, phoneNumber, emailId,password)
"
65.                   "VALUES (%s, %s, %s, %s, %s)",(userId, userName, userPhoneNumber,
userEmailId, password),)
66.         db.commit()
67.
68.         # Notify the user that the user has been added successfully
69.         print("--------------------------")
70.         print("User added successfully!")
71.         print("--------------------------")
72.         modifyUser()
73.
74. def deleteUser():
75.     print("--------------------------")
76.     print("Delete User")
77.     print("--------------------------")
78.     # Get user input for the user ID to be deleted
79.     userId = int(input("Enter the User ID : "))
80.     choice = input("Are you sure to delete the User? (Yes/No) : ")
81.
82.     c.execute("SELECT userId FROM users")
83.     result = c.fetchall()
84.     db.commit()
85.
86.     if choice.lower() in ["yes", "y"]:
87.         if (userId,) in result:
88.             c.execute("DELETE FROM users WHERE userId=%s", (userId,))
89.             db.commit()
90.
91.             # Notify the user that the user has been deleted successfully
92.             print("User deleted successfully!")
93.             print("--------------------------")
```

```
            modifyUser()
        else:
            print(f'The user of user id "{userId}" does not enrolled in the digital library.')
            print("--------------------------")
            modifyUser()
    elif choice.lower() in ["no", "n"]:
        print("--------------------------")
        print("User Not Deleted!")
        print("--------------------------")
        modifyUser()
    else:
        f.validOption()

def updateUser():
    print("--------------------------")
    print("Update User Details")
    print("--------------------------")
    # Display user update options
    print("1. Update the User ID")
    print("2. Update the User Name")
    print("3. Update the User Phone Number")
    print("4. Update the User Email ID")
    print("5. Update the User Password")
    print("6. Back")
    print("7. Exit")
    # Get user choice
    userChoice = int(input("Enter your Choice to Continue : "))
    print("--------------------------")

    c.execute("SELECT userId FROM users")
    result = c.fetchall()
    db.commit()

    if userChoice == 1:
        # Update user ID
        currentUserId = int(input("Enter the Current User ID : "))
        newUserId = int(input("Enter the New User ID : "))

        if (currentUserId,) in result:
            c.execute("update users set userId=%s where userId=%s", (newUserId,currentUserId))
            db.commit()

            print("User ID changed Successfully!")
            print(" --------------------------")
            modifyUser()
        else:
            notUser(currentUserId)

    elif userChoice == 2:
        # Update user name
        userId = int(input("Enter the User ID : "))
        newUserName = input("Enter the New User Name : ")

        if (userId,) in result:
            c.execute("update users set userName=%s where userId=%s",(newUserName, userId))
            db.commit()

            print("User Name changed Successfully!")
            print(" --------------------------")
            modifyUser()
```

```
154.         else:
155.             notUser(userId)
156.
157.     elif userChoice == 3:
158.         # Update user phone number
159.         userId = int(input("Enter the Current User ID : "))
160.         newPhoneNumber = input("Enter the New Phone Number : ")
161.
162.         if (userId,) in result:
163.             c.execute("update users set phoneNumber=%s where
userId=%s",(newPhoneNumber, userId),)
164.             db.commit()
165.
166.             print("User Phone Number changed Successfully!")
167.             print(" -------------------------")
168.             modifyUser()
169.         else:
170.             notUser(userId)
171.
172.     elif userChoice == 4:
173.         # Update user email ID
174.         userId = int(input("Enter the Current User ID : "))
175.         newEmailId = input("Enter the New Email ID : ")
176.
177.         if (userId,) in result:
178.             c.execute("update users set emailId=%s where userId=%s",
(newEmailId,userId))
179.             db.commit()
180.
181.             print("User Email ID changed Successfully!")
182.             print(" -------------------------")
183.             modifyUser()
184.         else:
185.             notUser(userId)
186.
187.     elif userChoice == 5:
188.         # Update user password
189.         userId = int(input("Enter the Current User ID : "))
190.         newPassword = input("Enter the New Password : ")
191.         if (userId,) in result:
192.             c.execute("update users set password=%s where userId=%s",(newPassword,
userId))
193.             db.commit()
194.
195.             print("User Password changed Successfully!")
196.             print(" -------------------------")
197.             modifyUser()
198.         else:
199.             notUser(userId)
200.
201.     elif userChoice == 6:
202.         # Go back to the previous menu
203.         modifyUser()
204.     elif userChoice == 7:
205.         # Exit the program
206.         f.exiting()
207.     else:
208.         f.validOption()
209.
210. def notUser(userId):
211.     print(f'The user of user id "{userId}" does not enrolled in the digital library.')
212.     print("-------------------------")
213.     modifyUser()
```

```
214.
215. def displayUsers():
216.     print("--------------------------")
217.     print("Display Users")
218.     print(" -------------------------")
219.     # Fetch all users from the database
220.     c.execute("SELECT * FROM users ORDER BY userId")
221.     result = c.fetchall()
222.     db.commit()
223.
224.     if result:
225.         # Display user information
226.         print("Users enrolled in the Digital Library are :")
227.         i = 0
228.         for row in result:
229.             i += 1
230.             r = f.length(i)
231.             print(f"{i}. User ID : {row[0]}")
232.             print(" " * r + f"User Name : {row[1]}")
233.             print(" " * r + f"Phone Number : {row[2]}")
234.             print(" " * r + f"Email ID : {row[3]}")
235.             print(" " * r + f"Admin Status : {row[5]}")
236.             print(" -------------------------")
237.             pass
238.
239.     else:
240.         print("No users found.")
241.         print(" -------------------------")
242.         pass
243.


244. # Function to search users
245. def searchUsers():
246.     print("--------------------------")
247.     print("Search Users")
248.     print("--------------------------")
249.     # User search menu
250.     print("1. Search by User ID")
251.     print("2. Search by Keyword")
252.     print("3. Back")
253.     print("4. Exit")
254.     userChoice = int(input("Enter your Choice to Continue : "))
255.     print("--------------------------")
256.
257.     # User choices handling
258.     if userChoice == 1:
259.         searchUsersbyId()
260.     elif userChoice == 2:
261.         searchUsersbyKeyword()
262.     elif userChoice == 3:
263.         pass
264.     elif userChoice == 4:
265.         f.exiting()
266.     else:
267.         f.validOption()
268.
269. def searchUsersbyId():
270.     userId = int(input("Enter User ID to search: "))
271.     c.execute("SELECT * FROM users WHERE userId=%s", (userId,))
272.     result = c.fetchall()
273.     db.commit()
274.
```

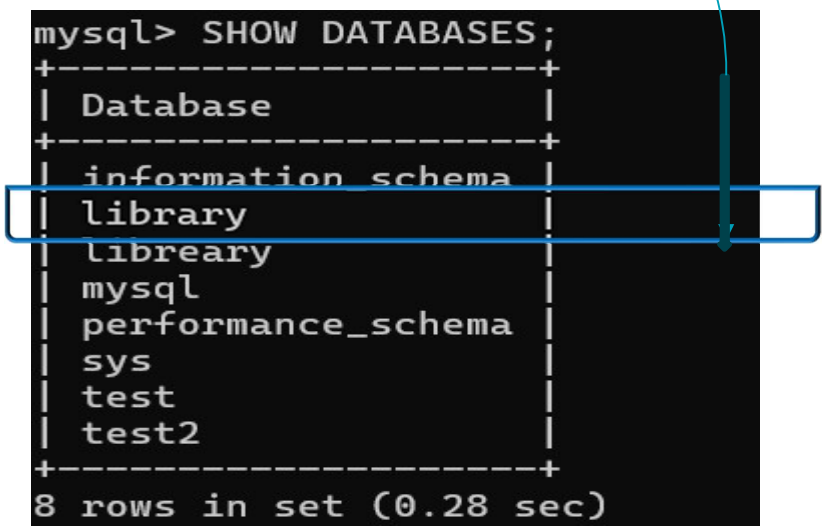
```
    if result:
        print(f"User with ID {userId} found:")
        for row in result:
            print(f"User ID: {row[0]}, Name: {row[1]}, Phone: {row[2]}, Email:
{row[3]}, Admin Status: {row[5]}")
    else:
        print(f"No user found with ID {userId}.")

def searchUsersbyKeyword():
    keyword = input("Enter a keyword to search (in name or email): ")
    c.execute("SELECT * FROM users WHERE userName LIKE %s OR emailId LIKE %s",
(f"%{keyword}%", f"%{keyword}%"))
    result = c.fetchall()
    db.commit()

    if result:
        print(f"Users matching '{keyword}':")
        for row in result:
            print(f"User ID: {row[0]}, Name: {row[1]}, Phone: {row[2]}, Email:
{row[3]}, Admin Status: {row[5]}")
    else:
        print(f"No users found with keyword '{keyword}'.")

def changeadmin():
    print("--------------------------")
    print("Change Admin")
    print("--------------------------")
    # Get new admin's ID and password from the user
    newAdminId = int(input("Enter the New Admin's User ID : "))
    newAdminPassword = input("Enter the New Admin's Password : ")
    choice = input("Are you sure to change the Admin? (Yes/No) : ")
    print("--------------------------")

    # Check if the entered user ID exists
    c.execute("SELECT password FROM users WHERE userId=%s", (newAdminId,))
    result = c.fetchall()
    db.commit()

    # Check the user's choice to proceed or cancel
    if choice.lower() in ["yes", "y"]:
        # If the user ID is not valid, print an error message
        if len(result) == 0:
            print("Please enter a valid user id!")
        else:
            # If the entered password matches the user's password
            if newAdminPassword == result[0][0]:
                # Update admin status for all users
                c.execute("UPDATE users SET adminStatus='not admin' WHERE adminStatus
='admin'")
                c.execute("UPDATE users SET adminStatus='admin' WHERE userId
=%s",(newAdminId,),)
                db.commit()

                print("Admin Changed Successfully!")
                print(" -------------------------")
                pass

            else:
                print("Please enter a valid password!")
    elif choice.lower() in ["no", "n"]:
        print("Admin Not Changed!")
        print("--------------------------")
        pass
```

```
333.     else:
334.         f.validOption()
335.
```

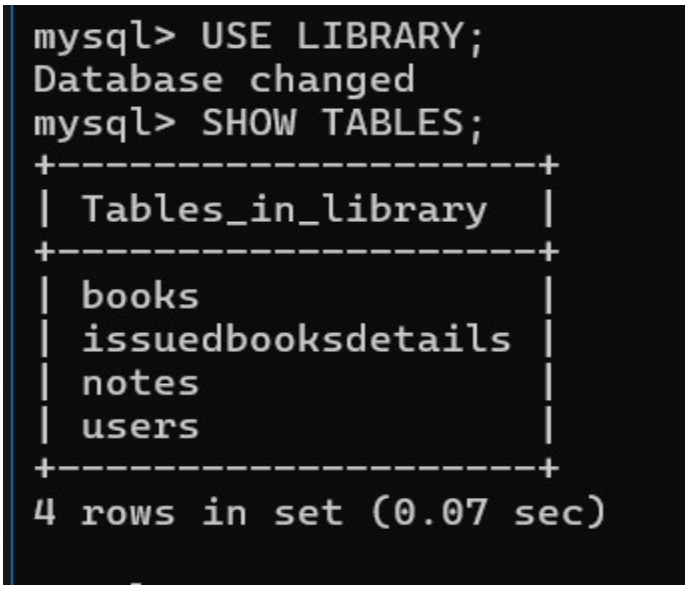

MySQL

MYSQL INITIALS;

LIBRARY MODULE:



TABLES;



BOOKS TABLE;

```
mysql> DESC BOOKS;
+---------------+------------+------+-----+-----------+-------+
| Field         | Type       | Null | Key | Default   | Extra |
+---------------+------------+------+-----+-----------+-------+
| bookId        | int        | NO   | PRI | NULL      |       |
| bookName      | varchar(50) | NO   |     | NULL      |       |
| publicationYear | int        | YES  |     | NULL      |       |
| issueDate     | date       | YES  |     | NULL      |       |
| issueTime     | time       | YES  |     | NULL      |       |
| returnDate    | date       | YES  |     | NULL      |       |
| returnTime    | time       | YES  |     | NULL      |       |
| author        | varchar(40) | YES  |     | NULL      |       |
| issueStatus   | varchar(10) | NO   |     | not issued |       |
| issueduserid  | int        | YES  | MUL | NULL      |       |
+---------------+------------+------+-----+-----------+-------+
10 rows in set (0.06 sec)

mysql> SELECT*FROM BOOKS;
+-------+-------------+---------------+-----------+-----------+-----------+-----------+------------+-------------+-------------+
| bookId | bookName    | publicationYear | issueDate | issueTime | returnDate | returnTime | author     | issueStatus | issueduserid |
+-------+-------------+---------------+-----------+-----------+-----------+-----------+------------+-------------+-------------+
|   1011 | HARRYPOTTER |          2013 | NULL      | NULL      | NULL      | NULL      | j.k.Rowling | not issued  |        NULL |
|   1017 | PERCYJACKSON |          2015 | 2024-12-13 | 02:56:35 | NULL      | NULL      | RICKRIORDAN | issued      |        1114 |
|   1105 | JAMESBOND   |          2017 | NULL      | NULL      | NULL      | NULL      | IANFLEMING | not issued  |        NULL |
|   1125 | pythonbasics |          2019 | NULL      | NULL      | NULL      | NULL      | S.K.       | issued      |        1113 |
|   1655 | ontheroad   |          2021 | NULL      | NULL      | NULL      | NULL      | j.k.       | issued      |        1111 |
|  11067 | SHERLOCKHOLMES |          2017 | NULL      | NULL      | NULL      | NULL      | A.C.DOYLE  | not issued  |        NULL |
+-------+-------------+---------------+-----------+-----------+-----------+-----------+------------+-------------+-------------+
6 rows in set (0.04 sec)
```

USERS TABLE;

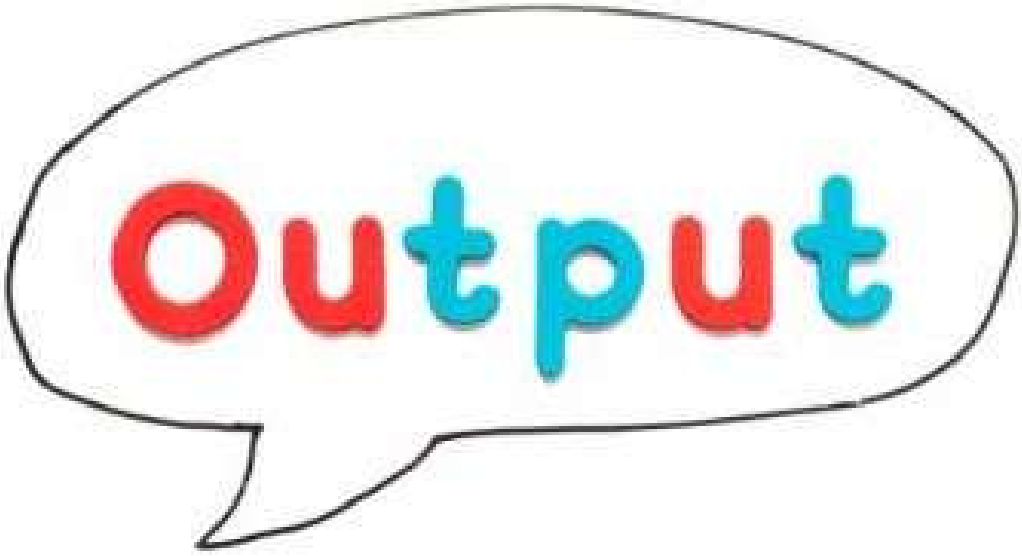
```
mysql> DESC USERS;
+-------------+-------------+------+-----+-----------+-------+
| Field       | Type        | Null | Key | Default   | Extra |
+-------------+-------------+------+-----+-----------+-------+
| userid      | int         | NO   | PRI | NULL      |       |
| userName    | varchar(50) | NO   |     | NULL      |       |
| phoneNumber | varchar(13) | YES  |     | NULL      |       |
| emailId     | varchar(40) | NO   |     | NULL      |       |
| password    | varchar(40) | NO   |     | NULL      |       |
| adminstatus | varchar(9)  | NO   |     | not admin |       |
+-------------+-------------+------+-----+-----------+-------+
6 rows in set (0.04 sec)

mysql> SELECT*FROM USERS;
+--------+----------+-------------+----------------+-------------+-------------+
| userid | userName | phoneNumber | emailId        | password    | adminstatus |
+--------+----------+-------------+----------------+-------------+-------------+
|   1111 | SHIV     | 987654321   | abcd@gmail.com | shiv@1111   | not admin   |
|   1112 | Shankar  | 9876342521  | efgh@gmail.com | shankar@1112 | admin       |
|   1113 | Asif     | 9823764321  | ijkl@gmail.com | asif@1113   | not admin   |
|   1114 | Farida   | 987698321   | mnop@gmail.com | farida@1114 | not admin   |
|   1115 | Sita     | 9823554321  | qrst@gmail.com | sita@1115   | not admin   |
|   1116 | Anna     | 987234321   | uvwx@gmail.com | anna@1116   | not admin   |
+--------+----------+-------------+----------------+-------------+-------------+
6 rows in set (0.05 sec)
```

NOTES TABLE;

```
+----------------+---------------+------+-----+---------+-------+
| Field          | Type          | Null | Key | Default | Extra |
+----------------+---------------+------+-----+---------+-------+
| userId         | int           | NO   | MUL | NULL    |       |
| notenumber     | int           | NO   | PRI | NULL    |       |
| noteTitle      | varchar(50)   | YES  |     | NULL    |       |
| noteDescription| varchar(10000)| YES  |     | NULL    |       |
| updateDate     | date          | YES  |     | NULL    |       |
| updateTime     | time          | YES  |     | NULL    |       |
+----------------+---------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql> SELECT*FROM NOTES;
+--------+------------+---------------------+---------------------------------------------------------------------------
| userId | notenumber | noteTitle           | noteDescription

                                                                                   | updateDate | updateTime |
+--------+------------+---------------------+---------------------------------------------------------------------------
|   1113 |          1 | 4Dimensions         | 4-dimensional spacetime: three-dimensional space of length, width, and height, plus time

                                                                                   | 2024-12-12 | 23:43:08   |
|   1111 |         10 | ADVANCE PYTHON TOPICS | Python language is a very versatile language and it is used in many technical fields. Some fields
 require only basic knowledge of Python but some fields require you to know advanced Python such as Data Science, Artificial Intelligence and Rob
otics.Pre-requisites: Before starting Advanced Python, one should have studied basic Python. | 2024-12-12 | 23:31:55   |
+--------+------------+---------------------+---------------------------------------------------------------------------
```

ISSUED BOOKS DETAILS TABLE;

```
mysql>  DESC ISSUEDBOOKSDETAILS;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| userid     | int         | NO   | MUL | NULL    |       |
| bookid     | int         | NO   | MUL | NULL    |       |
| bookname   | varchar(50) | NO   |     | NULL    |       |
| issueDate  | date        | YES  |     | NULL    |       |
| issuetime  | time        | YES  |     | NULL    |       |
| returnDate | date        | YES  |     | NULL    |       |
| returntime | time        | YES  |     | NULL    |       |
| fineInRs   | int         | NO   |     | 0       |       |
+------------+-------------+------+-----+---------+-------+
8 rows in set (0.04 sec)

mysql>  SELECT*FROM ISSUEDBOOKSDETAILS;
+--------+--------+---------------+-----------+-----------+------------+------------+----------+
| userid | bookid | bookname      | issueDate | issuetime | returnDate | returntime | fineInRs |
+--------+--------+---------------+-----------+-----------+------------+------------+----------+
|   1114 |   1017 | PERCYJACKSON  | NULL      | NULL      | NULL       | NULL       |        0 |
|   1113 |   1125 | PYTHONBASICS  | NULL      | NULL      | NULL       | NULL       |        0 |
|   1111 |   1655 | ONTHEROAD     | NULL      | NULL      | NULL       | NULL       |        0 |
+--------+--------+---------------+-----------+-----------+------------+------------+----------+
```


Output

PYTHON OUTPUT;

HOME()

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
__        __   _                                      _
\ \      / /__| | ___ ___  _ __ ___   ___  | |_ ___
 \ \ /\ / / _ \ |/ __/ _ \| '_ ` _ \ / _ \ | __/ _ \
  \ V  V /  __/ | (_| (_) | | | | | |  __/ | || (_) |
   \_/\_/ \___|_|\___\___/|_| |_| |_|\___|  \__\___/


 ____  _       _ _                 _  _     _ _
|  _ \(_) __ _(_) |_ __ _| | |   (_) |__  _ __ __ _ _ __ _   _
| | | | |/ _` | | __/ _` | | |   | | '_ \| '__/ _` | '__| | | |
| |_| | | (_| | | || (_| | | |___| | |_) | | | (_| | |  | |_| |
|____/|_|\__, |_|\__\__,_|_| |_____|_|_.__/|_|  \__,_|_|   \__, |
         |___/                                               |___/

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
==========================================================================
--------------------------
Home
--------------------------
1. Admin
2. User
3. Exit
Enter your Choice to Continue : 1
--------------------------
```

ADMIN AUTHENTICATION;

```
--------------------------
Admin Authentication
--------------------------
Enter the Admin's User ID : 1116
Enter the Admin's User Password : anna@1116
--------------------------
 ADMIN is verified successfully.
--------------------------
--------------------------
USER
--------------------------

          1. Login into User Panel
          2. Modify User
          3. Display Users
          4. Search Users
          5. Modify Book
          6. Issue Book
          7. Return Book
          8. Change Admin
          9. Home
          10.Back
          11.Exit

Enter your choice to continue : 1
--------------------------
```

ADMIN TO USER PANEL;

```
Enter your choice to continue : 1
-------------------------
You are successfully logged into user's Panel.
-----------------------------
-------------------------
USER
-------------------------
1.   Login into Admin Panel
2.   About the Library
3.   Dislpay Books
4.   Search Books
5.   Issued Book Details
6.   Notes
7.   Home
8.  Back
9.  Exit
Enter your choice to continue : 6
-------------------------
```

NOTES MENU;

```
Notes
-------------------------

         1.Modify Note
         2.Display Notes
         3.Search Notes
         4.TO add notes menu
         5.exit
Enter your Choice to Continue : 1
-------------------------
-------------------------
Modify Notes
-------------------------

         1.Delete Note
         2.Update note
         3.Back to menu
         4.Exit
```

DISPLAY NOTE;

```
Enter your Choice to Continue : 2
------------------------
Notes available in the Digital Library are :
1. Note Number : 1
   Note Title : 4Dimensions
   Note Description : 4-dimensional spacetime: three-dimensional space of length, width, and height, plus time
   Update Date : 2024-12-12
   Update Time : 23:43:08
------------------------
```

SEARCH NOTE;

```
-------------------------
Search Notes
-------------------------
1. Search by Note Number
2. Search by Keyword
3. Home
4. Back
5. Exit
Enter your Choice to Continue : 1
-------------------------
Enter the Note Number to search the Note : 11
Note available in the Digital Library with the Note Number "11" is:
1. Note Number : 11
   Note Title : HI, GOOD MORNING
   Note Description : hope, u will appreciate our efforts we put in this program.
-------------------------
-------------------------
Search Notes
-------------------------
1. Search by Note Number
2. Search by Keyword
3. Home
4. Back
5. Exit
Enter your Choice to Continue : 2
-------------------------
-------------------------
Search Notes by Keyword
-------------------------
Enter a Keyword to search Notes : ADVANCE PYTHON TOPICS
Notes available in the Digital Library with the Keyword "ADVANCE PYTHON TOPICS" are:
1. Note Number : 10
   Note Title : ADVANCE PYTHON TOPICS
```

MODIFY NOTE;

```
-------------------------
Modify Notes
-------------------------

         1.Delete Note
         2.Update note
         3.Back to menu
         4.Exit
Enter your Choice to Continue : 2
-------------------------
-------------------------
Update Notes
-------------------------
1. Update the Note Number
2. Update the Note Title
3. Update the Note Description
4. Back
5. Exit
Enter your Choice to Continue : 1
-------------------------
Enter the Current Note Number : 11
Enter the New Note Number : 15
Note Number changed Successfully!
-------------------------
-------------------------
Update Notes
-------------------------
1. Update the Note Number
2. Update the Note Title
3. Update the Note Description
4. Back
5. Exit
```

USER AUTHENTICATION;

```
|____/|_|\__, |_|\__\__,_|_| |_____|_|_.__/|_|  \__,_|_|
         |___/

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
==========================================================================
-------------------------
Home
-------------------------
1. Admin
2. User
3. Exit
Enter your Choice to Continue : 2
-------------------------
-------------------------
User Authentication
-------------------------
Enter the User ID : 1111
Enter the User Password : shiv@1111
-------------------------
User is verified successfully.
-------------------------
-------------------------
USER
-------------------------
1. About Library
2. Display Books
3. Search Books
4. Issued Books Details
5. Notes
6. Home
7. Back
8. Exit
Enter your Choice to Continue : 5
```

ABOUT LIBRARY

```
--------------------------------------------------

    _    _                 _     _     _ _
   / \  | |__   ___  _   _| |_  | |   (_) |__  _ __ __ _ _ __ _   _
  / _ \ | '_ \ / _ \| | | | __| | |   | | '_ \| '__/ _` | '__| | | |
 / ___ \| |_) | (_) | |_| | |_  | |___| | |_) | | | (_| | |  | |_| |
/_/   \_\_.__/ \___/ \__,_|\__| |_____|_|_.__/|_|  \__,_|_|   \__, |
                                                              |___/

--------------------------------------------------
Year of Library's Establishment :  2024
Name of the Librarian :  Anna
Total Number of Books Available in the Library :  7
Total Number of Users Enrolled in the Library :  7
--------------------------------------------------
1. Add Note
2. Home
3. Back
4. Exit
ENTER CHOICE
3
-------------------------------
```

ISSUED BOOKS DETAILS;

```
Enter your Choice to Continue : 4
--------------------------
--------------------------
Issued Books Details
--------------------------
Return Policy :
The issued book should be returned within 14 days(2 weeks).
If the user kept the issued book for more than 14 days,
then the user have to pay ₹5 as fine for each extra day
the user kept the issued book.
--------------------------
1. Book ID :  1655
   Book Name :  ONTHEROAD
   Issue Date :  None
   Issue Time :  None
   Return Date :  None
   Return Time :  None
   Fine(in Rs.) :  0
--------------------------
1. Add Note
2. Home
3. Back
4. Exit
ENTER CHOICE
```

DISPLAY BOOKS;

```
Enter your Choice to Continue : 2
-------------------------
-------------------------
Display Books
-------------------------
Books available in the Digital Library are :
-------------------------
1. Book ID : 1011
   Book Name : HARRYPOTTER
   Publication Year : 2013
   Author Name : j.k.Rowling
   Issue Status : issued
-------------------------
2. Book ID : 1017
   Book Name : PERCYJACKSON
   Publication Year : 2015
   Author Name : RICKRIORDAN
   Issue Status : issued
-------------------------
3. Book ID : 1105
   Book Name : JAMESBOND
   Publication Year : 2017
   Author Name : IANFLEMING
   Issue Status : not issued
-------------------------
4. Book ID : 1125
   Book Name : pythonbasics
   Publication Year : 2019
   Author Name : S.K.
   Issue Status : issued
-------------------------
```

ADD NOTE;

```
1. Add Note
2. Home
3. Back
4. Exit
ENTER CHOICE
1
------------------------------
--------------------------
Add Note
--------------------------
Enter the Note Number : 19
--------------------------
Enter the Note Title : hi,
Enter the Note Description : hope you like the programme
--------------------------
The note of note number "19" is added successfully.
--------------------------
```

SEARCH BOOK;

```
--------------------------
Search Books
--------------------------
1. Search by Book ID
2. Search by Keyword
3. Back
4. Exit
Enter your Choice to Continue : 1
--------------------------
--------------------------
Search Books by Book ID
--------------------------
Enter the Book ID to search the Book : 1017
--------------------------
Book available in the Digital Library with the Book ID "1017" is :
--------------------------
1. Book ID : 1017
   Book Name : PERCYJACKSON
   Publication Year : 2015
   Author Name : RICKRIORDAN
   Issue Status : issued
--------------------------
```

BOOKS DATABASE MODIFICATION (BY ADMIN);

```
--------------------------
--------------------------
Modify Book
--------------------------
1. Add Book
2. Delete Book
3. Update Book Details
4. Back
5. Exit
Enter your Choice to Continue : 1
--------------------------
--------------------------
Add Book
--------------------------
Enter the Book ID : 11234
Enter the Book Name : chemistry
Enter the Book Publication Year : 2024
Enter the Book Author Name : arihant
--------------------------
Book added Successfully!
--------------------------
```

```
Enter your Choice to Continue : 2
-------------------------
-------------------------
Delete Book
-------------------------
Enter the Book ID : 34
Are you sure to delete the Book? (Yes/No) : y
-------------------------
Book deleted Successfully!
-------------------------
```

UPDATE BOOK;(BY ADMIN)

```
-------------------------
Update Book Details
-------------------------
1. Update the Book ID
2. Update the Book Name
3. Update the Book Publication Year
4. Update the Book Author Name
5. Back
6. Exit
Enter your Choice to Continue : 1
-------------------------
Enter the Current Book ID : 11234
Enter the New Book ID : 12344
Book ID changed Successfully!
-------------------------
Enter your Choice to Continue : 2
-------------------------
Enter the Book ID : 12344
Enter the New Book Name : Physics
Book Name changed Successfully!
-------------------------
Enter your Choice to Continue : 3
-------------------------
Enter the Current Book ID : 12344
Enter the New Publication Year : 2023
Book Publication Year changed Successfully!
Enter your Choice to Continue : 4
-------------------------
Enter the Current Book ID : 12344
Enter the New Author Name : oswal
Book Author Name changed Successfully!
-------------------------
```

DISPLAY USERS TO ADMIN;

```
          1. Login into User Panel
          2. Modify User
          3. Display Users
          4. Search Users
          5. Modify Book
          6. Issue Book
          7. Return Book
          8. Change Admin
          9. Home
          10.Back
          11.Exit

Enter your choice to continue : 3
--------------------------
--------------------------
Display Users
--------------------------
Users enrolled in the Digital Library are :
1. User ID : 1111
   User Name : SHIV
   Phone Number : 987654321
   Email ID : abcd@gmail.com
   Admin Status : not admin
--------------------------
2. User ID : 1112
   User Name : Shankar
   Phone Number : 9876342521
   Email ID : efgh@gmail.com
   Admin Status : not admin
--------------------------
```

MODIFY USERS BY ADMIN(ADDING)

```
--------------------------
Modify User
--------------------------
1. Add User
2. Delete User
3. Update User Details
4. Back to admin
5. Exit
Enter your Choice to Continue : 1
--------------------------
--------------------------
Add User
--------------------------
Enter the User ID : 2508
Enter the User Name : Shivangi
Enter the User Phone Number : 324678689
Enter the User Email ID : shiva@2508
Enter the User Password : shiv@2508
--------------------------
--------------------------
User added successfully!
--------------------------
```

SEARCH USER;

```
Enter your choice to continue : 4
-------------------------
-------------------------
Search Users
-------------------------
1. Search by User ID
2. Search by Keyword
3. Back
4. Exit
Enter your Choice to Continue : 1
-------------------------
Enter User ID to search: 1111
User with ID 1111 found:
User ID: 1111, Name: SHIV, Phone: 987654321, Email: abcd@gmail.com, Admin Status: not admin
```

RETURN BOOK;

```
Enter your choice to continue : 7
-------------------------
-------------------------
Return Book
-------------------------
Enter the Book ID to return: 1125
Enter the User ID: 1113
Book ID 1125 has been returned by User ID 1113.
No fine incurred.
-------------------------
```

ISSUE BOOK;

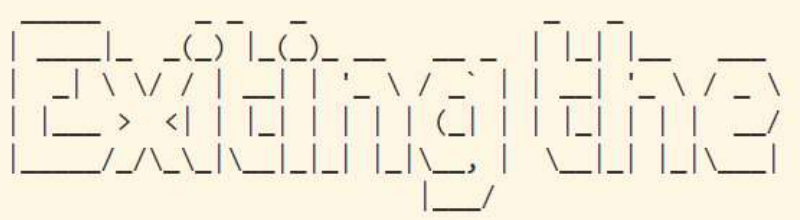
```
Enter your choice to continue : 6
-------------------------
-------------------------
Issue Book
-------------------------
Enter the Book ID to issue: 1125
Enter book_name:pythonbasics
Enter the User ID: 1116
Book ID 1125 has been issued to User ID 1116.
-------------------------
```

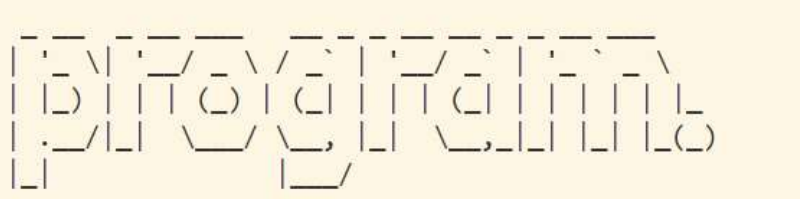
CHANGING ADMIN;

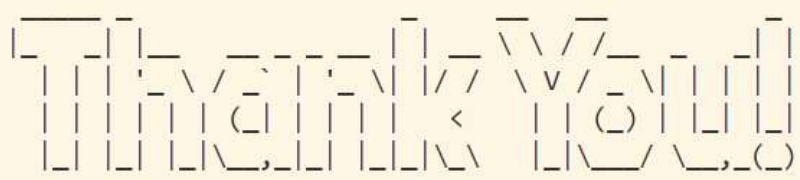
```
Enter your choice to continue : 8
-------------------------
-------------------------
Change Admin
-------------------------
Enter the New Admin's User ID : 1112
Enter the New Admin's Password : shankar@1112
Are you sure to change the Admin? (Yes/No) : y
-------------------------
Admin Changed Successfully!
-------------------------
```

EXITING THE PROGRAM

```
          11.Exit

Enter your choice to continue : 11
--------------------------
Are you sure you want to exit? (yes/no): y
--------------------------------------
```







```
--------------------------------------
```

MYSQL AFTER CHANGES;

BOOKS TABLE

```
mysql> select* from books;
+--------+--------------+-----------------+-----------+-----------+------------+------------+-------------+-------------+-------------+
| bookId | bookName     | publicationYear | issueDate | issueTime | returnDate | returnTime | author      | issueStatus | issueduserid |
+--------+--------------+-----------------+-----------+-----------+------------+------------+-------------+-------------+-------------+
|   1011 | HARRYPOTTER  |            2013 | 2024-12-13 | 15:23:13  | NULL       | NULL       | j.k.Rowling | issued      |        1112 |
|   1017 | PERCYJACKSON |            2015 | 2024-12-13 | 02:56:35  | NULL       | NULL       | RICKRIORDAN | issued      |        1114 |
|   1105 | JAMESBOND    |            2017 | NULL      | NULL      | NULL       | NULL       | IANFLEMING  | not issued  |        NULL |
|   1125 | pythonbasics |            2019 | 2024-11-29 | NULL      | NULL       | NULL       | S.K.        | issued      |        1113 |
|   1655 | ontheroad    |            2021 | NULL      | NULL      | NULL       | NULL       | j.K.        | issued      |        1111 |
|  11067 | SHERLOCKHOLMES |          2017 | NULL      | NULL      | NULL       | NULL       | A.C.DOYLE   | not issued  |        NULL |
|  12344 | Physics      |            2023 | NULL      | NULL      | NULL       | NULL       | oswal       | not issued  |        NULL |
+--------+--------------+-----------------+-----------+-----------+------------+------------+-------------+-------------+-------------+
```

USERS TABLE

```
mysql> select*from users;
+--------+----------+-------------+----------------+-------------+-------------+
| userid | userName | phoneNumber | emailId        | password    | adminstatus |
+--------+----------+-------------+----------------+-------------+-------------+
|   1111 | SHIV     | 987654321   | abcd@gmail.com | shiv@1111   | not admin   |
|   1112 | Shankar  | 9876342521  | efgh@gmail.com | shankar@1112 | admin      |
|   1113 | Asif     | 9823764321  | ijkl@gmail.com | asif@1113   | not admin   |
|   1114 | Farida   | 987698321   | mnop@gmail.com | farida@1114 | not admin   |
|   1115 | Sita     | 7786399973  | qrst@gmail.com | sita@1115   | not admin   |
|   1116 | Anna     | 987234321   | uvwx@gmail.com | anna@1116   | not admin   |
|   1117 | shubham  | 7654636785  | shubh@1117     | shubham@1117 | not admin  |
|   2508 | Shivangi | 324678689   | shiva@2508     | shiv@2508   | not admin   |
+--------+----------+-------------+----------------+-------------+-------------+
```

ISSUED BOOKS DETAILS

```
mysql> select*from issuedbooksdetails;
+--------+--------+--------------+------------+-----------+------------+------------+----------+
| userid | bookid | bookname     | issueDate  | issuetime | returnDate | returntime | fineInRs |
+--------+--------+--------------+------------+-----------+------------+------------+----------+
|   1114 |   1017 | PERCYJACKSON | NULL       | NULL      | NULL       | NULL       |        0 |
|   1111 |   1655 | ONTHEROAD    | NULL       | NULL      | NULL       | NULL       |        0 |
|   1116 |   1125 | pythonbasics | 2024-12-13 | NULL      | NULL       | NULL       |        0 |
+--------+--------+--------------+------------+-----------+------------+------------+----------+
```

NOTES TABLE

```
mysql> select* from notes;
+--------+------------+----------------------+------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------+------------+------------+
| userId | notenumber | noteTitle            | noteDescription


                     | updateDate | updateTime |
+--------+------------+----------------------+------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------+------------+------------+
|   1113 |          1 | 4Dimensions          | 4-dimensional spacetime: three-dimensional space of length, width, and height, plus time

                     | 2024-12-12 | 23:43:08   |
|   1111 |         10 | ADVANCE PYTHON TOPICS | Python language is a very versatile language and it is used in many technical fields. Some fields
 require only basic knowledge of Python but some fields require you to know advanced Python such as Data Science, Artificial Intelligence and Rob
otics.Pre-requisites: Before starting Advanced Python, one should have studied basic Python. | 2024-12-12 | 23:31:55   |
|   1111 |         15 | HI, GOOD MORNING     | hope, u will appreciate our efforts we put in this program.


                     | 2024-12-13 | 22:01:34   |
|   1111 |         19 | hi,                  | hope you like the programme


                     | 2024-12-13 | 22:09:50   |
+--------+------------+----------------------+------------------------------------------------------------------------------------------------------
```

TESTING

Software Testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate. Software Testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

It can also be stated as the process of validating and verifying that a software program/application/product meets the business and technical requirements that guided its design and development, so that it works as expected and can be implemented with the same characteristics. Software Testing, depending on the testing method employed, can be implemented at any time in the development process, however the most test effort is employed after the requirements have been defined and coding process has been completed.

TESTING METHODS

Software testing methods are traditionally divided into black box testing and white box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

BLACK BOX TESTING

Black box testing treats the software as a "black box," without any knowledge of internal implementation. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification based testing.

SPECIFICATION-BASED TESTING

Specification-based testing aims to test the functionality of software according to the applicable requirements.[16] Thus, the tester inputs data into, and only sees the output from, the test object. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behaviour), either "is" or "is not" the same as the expected value specified in the test case. Specification based testing is necessary, but it is insufficient to guard against certain risks

ADVANTAGES AND DISADVANTAGES

The black box tester has no "bonds" with the code, and a tester's perception is very simple: a code must have bugs. Using the principle, "Ask and you shall receive," black box testers find bugs where programmers don't. But, on the other hand, black box testing has been said to be "like a walk in a dark labyrinth without a flashlight," because the tester doesn't know how the software being tested was actually constructed.

That's why there are situations when (1) a black box tester writes many test cases to check something that can be tested by only one test case, and/or (2) some parts of the back end are not tested at all. Therefore, black box testing has the advantage of "an unaffiliated opinion," on the one hand, and the disadvantage of "blind exploring," on the other.

WHITE BOX TESTING

White box testing, by contrast to black box testing, is when the tester has access to the internal data structures and algorithms (and the code that implement these)

TYPE OF WHITE BOX

- api testing - testing of the application using public and private apis.
- code coverage - creating tests to satisfy some criteria of code coverage.For example, the test designer can create tests to cause all statements in the program to be executed at least once.
- fault injection methods.
- mutation testing methods.
- static testing - white box testing includes all static testing.

CODE COMPLETENESS EVALUATION

White box testing methods can also be used to evaluate the completeness of a test suite that was created with black box testing methods. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested.

Two common forms of code coverage are:

- ***Function Coverage: Which reports on functions executed***
- ***Statement Coverage: Which reports on the number of lines executed to complete the test***.

Bibliography

WEBSITES

- www.google.com
- www.w3school.com
- www.geekforgeeks.com
- www.python.org
- www.wikipedia.com
- www.youtube.com
- www.microsoft.com
- www.cbse.nic.in

BOOKS

- CLASS 12 SUMITA ARORA PYTHON
- CLASS 12 PREETI ARORA PYTHON
- CLASS 12 NCERT