A PROJECT REPORT

ON

**LIBRARY**

**MANAGEMENT SYSTEM**

FOR AISSCE 2020 EXAMINATION

[AS A PART OF THE INFORMATICS PRACTICES COURSE (065)]

**SUBMITTED BY: -**

**NAME: JANHAVI AGRAWAL**

**ROLLNO: 24**

UNDER THE GUIDANCE OF: **MR.AMIT SETHI** (COMP.SC)

**CERTIFICATE**

This is to certify that the Project / Dissertation entitled **LIBRARY MANAGEMENT System** is a bonafide work done by **JANHAVI AGRAWAL** of class XII Session 2020-21 in partial fulfilment of CBSE’s AISSCE Examination 2020 and has been carried out under my direct supervision and guidance. This report or a similar report on the topic has not been submitted for any other examination and does not form a part of any other course undergone by the candidate.

………………….. ……………………..

Signature of Student Signature of Teacher/Guide

Name: JANHAVI AGRAWAL Name: MR.AMIT SETHI

Roll No.: 24

**ACKNOWLEDGEMENT**

I, undertook this Project work, as the part of my XII-Informatics Practices course(065). I had tried to apply my best of knowledge and experience, gained during the study and class work experience. However, developing software system is generally a quite complex and time-consuming process. It requires a systematic study, insight vision and professional approach during the design and development. Moreover, the developer always feels the need, the help and good wishes of the people near you, who have considerable experience and idea.

I would like to extend my sincere thanks and gratitude to my teacher

MR. AMIT SETHI, for giving valuable time and moral support to develop this software.

I also feel indebted to my friends for the valuable suggestions during the project work.

JANHAVI AGRAWAL

Class XII - A

**C O N T E N T S**

1. Introduction ------------------------------------------------------------------- (5)

2. Objective & Scope of the Project ------------------------------------- (6)

3. System Implementation -------------------------------------------------- (8)

3.1 The Hardware used: ----------------------------------------------------- (8)

3.2 The Software’s used: ---------------------------------------------------- (8)

4. Event Coding: ----------------------------------------------------------- (9-23)

5. Output: -------------------------------------------------------------------- (24-34)

6. User Manual -------------------------------------------------------------- (35-36)

6.1 How to install: ------------------------------------- (35-36)

6.2 Working with Software: -------------------------- (35-36)

7. References: -----------------------------------------------------------------(37)

**Introduction**

This software project is developed to automate the functionalities of **LIBRARY MANAGEMENT SYSTEM.**

The purpose of the software project is to develop the Management Information System (MIS) to automate the record keeping of products, purchases, stock and sales transactions with a view to enhance the decision making of the functionaries.

A MIS mainly consists of a computerized database, a collection of inter-related tables for a particular subject or purpose like reference of user id in all tables, capable to produce different reports relevant to the user. An application program is tied with the database for easy access and interface to the database. Using Application program (Python) or front-end, we can store, retrieve and manage all information in proper way.

This software, being simple in design and working, does not require much of training to users, and can be used as a powerful tool for automating a **LIBRARY MANAGEMENT SYSTEM.**

During coding and design of the software Project, Python IDLE, as a powerful front-end tool is used for getting Graphical User Interface (GUI) based integrated platform and coding simplicity. As a back-end a powerful, open source RDBMS, MySQL is used as per requirement of the CBSE curriculum of Informatics Practices Course (065).

**Objective & Scope of the Project**

The objective of the software project is to develop a computerized MIS to automate the functions of a LIBRARY MANAGEMENT SYSTEM. This software project is also aimed to enhance the current record keeping system, which will help managers to retrieve the up-to-date information at right time in right shape.

The proposed software system is expected to do the following functionality-

* To provide a user friendly, Graphical User Interface (GUI) based integrated and centralized environment for MIS activities.
* The proposed system should maintain all the records and transactions, and should generate the required reports and information when required.
* To provide graphical and user-friendly interface to interact with a centralized database based on client-server architecture.
* To identify the critical operation procedure and possibilities of simplification using modern IT tools and practices.

In its current scope, the software enables user to retrieve and update the information from centralized database designed with MySQL . This software does not require much training time of the users due to limited functionality and simplicity.

During the development of LIBRARY MANAGEMENT SYSTEM project, Python IDLE, a powerful, open source event-driven form-based development environment is used for modular design and future expandability of the system.

Despite of the best effort of the developer, the following limitations and functional boundaries are visible, which limits the scope of this application software.

**1.** This software can store records and produce reports in pre-designed format in soft copy. There is no facility yet to produce customized reports. Only specified reports are covered.

**2**. There is no provision to calculate profit or loss etc. for the store; however it can be developed easily with the help of adding modules.

**3.** It facilitates manager to record and update only transaction record.

So far as future scope of the project is concerned, firstly it is open to any modular expansion i.e. other modules or functions can be designed and embedded to handle the user need in future. Any part of the software and reports can be modified independently without much effort.

**System Implementation**

**The Hardware used:**

While developing the system, the used hardware are:

PC with Intel Core i5-2400S processor having 4.00 GB RAM, 64 BIT

Operating System, SVGA and other required devices.

**The Software used:**

1. Microsoft Windows® 10 Pro as Operating System.
2. Python 3.8.2 as Front-end Development environment.
3. My SQL 8.0 as Back-end Sever with Database for Testing.
4. Mysql.connector to connect Python module with database.
5. MS-Word 2010 for documentation.
6. PY CHARM community 2020.0

CODING

import mysql.connector  
mydb = mysql.connector.connect(  
 host=**"localhost"**,  
 user=**"root"**,  
 password=**"root"**,  
  
 database=**"library\_management"**)  
mycursor = mydb.cursor()

import Connection  
  
def verifyUser():  
 user\_id = input(**'Please enter your id:'**)  
 password = input(**'Please enter your password:'**)  
 sql = **"SELECT user\_id,role FROM LoginDetails**

**WHERE user\_id=%s**

**AND password=%s"** val = (user\_id,password)  
 Connection.mycursor.execute(sql, val)  
 result = Connection.mycursor.fetchall()  
 return result

import Connection  
from tabulate import tabulate  
  
  
def createUser():  
  
 print(**'-------Register a new user--------'**)  
 print(**''**)  
 user\_name = input(**'Please enter user name:'**)  
 user\_address = input(**'Please enter user address:'**)  
 user\_email = input(**'Please enter user email:'**)  
 user\_phone = input(**'Please enter user phone:'**)  
 try:  
 sql = **"INSERT INTO user**

**(user\_name,user\_address,user\_email,user\_phone) "** \  
 **"VALUES (%s, %s, %s, %s)"** val = (user\_name, user\_address, user\_email, user\_phone)  
 Connection.mycursor.execute(sql, val)  
 rowcount = Connection.mycursor.rowcount  
 if rowcount > 0:  
 sql = **"select BHHHHHHHHHHHuser\_id,user\_name,user\_address,user\_email,**

**user\_phone from user where user\_email='"**

+ user\_email + **"'"** Connection.mycursor.execute(sql)  
 result = Connection.mycursor.fetchall()  
 print(**''**)  
 print(**"Below user details have been**

**added successfully!"**)  
 print(**''**)  
 print(tabulate(result,  
 headers=[**'User\_id'**, **'User\_name'**, MMMMMMMMMMMMM**'User\_address'**,**'User\_email'**, MNJJJJJJJJJJJ**'User\_phone'**],tablefmt=**'psql'**))  
 user\_id = 1;  
 for row in result:  
 user\_id = row[0]  
  
 sql = **"INSERT INTO LoginDetails XXXXXXXXXXXX(user\_id,password,role)**

**VALUES (%s,%s,%s)"** val = (user\_id, **"password"**, **"user"**)  
 resultcount = Connection.mycursor.execut(sql, val)  
 Connection.mydb.commit()  
 except Exception as e:  
 print(e)  
 return

def displayAllUserDetails():  
 sql = **"SELECT \* FROM User"** Connection.mycursor.execute(sql)  
 result = Connection.mycursor.fetchall()  
 return result  
  
  
def modifyUserDetails():  
 try:  
 userdetails=displayAllUserDetails()  
 print(**''**)  
 print(tabulate(userdetails,  
 headers=[**'user\_id'**, **'user\_name'**, **'user\_address'**, jjjjjjjjjjjjjj**'user\_email'**,**'user\_phone'**],tablefmt=**'psql'**))  
 print(**''**)  
 user\_id = input(**"Enter user id to be edited : "**)  
 print(**''**)  
 sql = **"select \* from User where user\_id="** + user\_id  
 Connection.mycursor.execute(sql)  
 res = Connection.mycursor.fetchall()  
 if len(res) == 0:  
 print(**''**)  
 print(**"Sorry, this user does not exist**

**in Library!"**)  
 return  
 for x in res:  
 print(**''**)

print(tabulate(res,  
 DSSSSSSSSSSSSSSSheaders=[**'user\_id'**,**'user\_name'**,**'user\_address'**, jjjjjjjjjjjjjjjjjj**'user\_email'**,**'user\_phone'**],tablefmt=**'psql'**))  
 print(**""**)  
 headers = [i[0] for i in Connection.mycursor.description]  
 inputs = [str(x) for x in input(**"Enter the comma M separated field names which you want to MMMMMM modify :"**).split(**','**)]  
 print(**''**)  
 print(**"**\n**The values of input are"**, inputs)  
 sql = **"Update User set "** count = 0  
 for field in inputs:  
 for x in headers:  
 if field.upper() == x.upper():  
 count = 1  
 if count == 1:  
 print(**''**)  
 val = input(**"Enter new "** + field + **": "**)  
 if (val.isnumeric()) and

(field.upper()!= **'USER\_PHONE'**):

sql += field + **" = "** + val + **","** else:  
 sql += field + **" = '"** + val + **"',"** count = 0  
 else:  
 print(**''**)  
 print(**"Sorry, the field name "** + field +

**"does not exist!"**)  
 return  
 sql = sql[:-1]  
 sql += **" where user\_id="** + user\_id  
 Connection.mycursor.execute(sql)  
 Connection.mydb.commit()  
 print(**''**)  
 print(**"User details for user id "** + user\_id +

**" has been updated successfully! "**)  
 print(**''**)  
 sql = **"select \* from User where user\_id="** + user\_id  
 Connection.mycursor.execute(sql)  
 res = Connection.mycursor.fetchall()  
 for \_ in res:  
 print(tabulate(res,  
 headers=[**'user\_id’,**U**ser\_name'**,**'user\_address'**, ddddddddddddd **'user\_email'**, **'user\_phone'**],tablefmt=**'psql'**))

print(**""**)  
 except Exception as e:  
 print(e)  
 return

def displayAllAvailableBooks():  
 sql = **"SELECT \* FROM Book where available='Y'"** Connection.mycursor.execute(sql)  
 result = Connection.mycursor.fetchall()  
 return result  
  
  
def displayIssuedBooks(user\_id):  
 sql = **"SELECT u.user\_name,u.user\_phone,u.user\_email,**

**u.user\_address ,b.book\_name,bh.issue\_date,bh.return\_date,**

**"** \**" b.isbn, b.year\_published,b.author,b.publisher from bookhistory bh inner join "** \ **"book b on bh.book\_id=b.book\_id left join user u on u.user\_id=bh.user\_id where bh.approved='Y' and "** \  
**"bh.user\_id="**+str(user\_id)  
 Connection.mycursor.execute(sql)  
 result = Connection.mycursor.fetchall()  
 if len(result)==0:  
 print (**"Sorry no book issued to this user id!"**)  
 print(**"#######################"**)  
 return result  
  
def issueNewBook(user\_id):  
 print(**"Below are the available books:-"**)  
 print(**''**)  
 bookDetails =displayAllAvailableBooks()  
 print(tabulate(bookDetails,  
 headers=[**'Book\_Id'**, **'Book\_Name'**, **'ISBN'**, MMMMMMMMMMMMMMMMMMM**'PublishedkYear'**, **'Author'**, **'Publisher'**,  
 **'Quantity'**, **'Available'**], MMMMMMMMMMMMMMMMMMMtablefmt=**'psql’**))   
 try:  
 print(**''**)  
 book\_id=input(**"enter book\_id to be issued:"**)  
 sql = **"select \* from book where book\_id="** + book\_id  
 Connection.mycursor.execute(sql)  
 res = Connection.mycursor.fetchall()  
 if len(res) == 0:  
 print(**''**)  
 print(**"Sorry, this book does not exist**

**in Library!"**)  
 return

else:  
 sql=**"INSERT INTO MKKKKKKKKKKKKKKKKBookhistory(book\_id,user\_id,approved)**

**values(%s,%s,%s)"** val = (book\_id,user\_id,**"N"**)  
 Connection.mycursor.execute(sql, val)  
 Connection.mydb.commit()  
 rowcount = Connection.mycursor.rowcount  
 if rowcount>0:  
 print(**''**)  
 print(**"Kindly get the book approved by**

**the admin!!"**)  
 else:  
 print(**''**)  
 print(**"Sorry the credentials entered**

**are wrong!!"**)  
 except Exception as e:  
 print(e)  
 return

import Connection  
from tabulate import tabulate  
from datetime import datetime  
from datetime import timedelta  
from datetime import date  
import time  
  
  
  
def createBook():  
  
 print(**'-------Register a new book in Library--------'**)  
 print(**''**)  
 book\_name = input(**'Please enter book name:'**)  
 isbn = input(**'Please enter isbn:'**)  
 year\_published = int(input(**"Please enter publish year:"**))  
 author = input(**"Please enter author name:"**)  
 publisher = input(**"Please enter publisher name:"**)  
 quantity = int(input(**"Please enter books quantity:"**))  
 print(**''**)  
 sql = **"INSERT INTO Book (book\_name, isbn,year\_published,author,publisher,quantity,available) VALUES (%s, %s, %s, "** \ **"%s,%s,%s,%s) "** val = (book\_name, isbn, year\_published, author, publisher,

quantity, **"Y"**)  
 Connection.mycursor.execute(sql, val)  
 rowcount = Connection.mycursor.rowcount  
 Connection.mydb.commit()  
 return rowcount  
  
  
def displayAllBooks():  
 sql = **"SELECT \* FROM Book"** Connection.mycursor.execute(sql)  
 result = Connection.mycursor.fetchall()  
 return result  
  
  
def modifyBook():  
 try:  
 bookDetails = displayAllBooks()  
 print(**''**)  
 print(tabulate(bookDetails,  
 headers=[**'Book\_Id'**, **'Book\_Name'**, **'ISBN'**, MMMMMMMMMMMMMMM**'Published Year'**, **'Author'**, **'Publisher'**, MMMMMMMMMMMMMMM**'Quantity'**,**'Available'**],tablefmt=**'psql'**))  
 print(**''**)  
 book\_id = input(**"Enter book id to be edited : "**)  
 sql = **"select \* from book where book\_id="** + book\_id  
 Connection.mycursor.execute(sql)  
 res = Connection.mycursor.fetchall()  
 if (len(res) == 0):  
 print(**''**)  
 print(**"Sorry, this book does not exist**

**in Library!"**)  
 return  
 for x in res:  
 print(**''**)  
 print(tabulate(res,  
 headers=[**'Book\_Id'**, **'Book\_Name'**, **'ISBN'**,**'Year\_published'**,**'Author'**, **'Publisher'**,**'Quantity'**, **'Available'**],tablefmt=**'psql'**))  
 print(**""**)  
 headers = [i[0] for i in Connection.mycursor.description]  
 print(**''**)  
 inputs = [str(x) for x in input(**"Enter the comma separated field names which you want to modify :"**).split(**','**)]  
 print(**''**)  
 print(**"**\n**The values of input are"**, inputs)  
 sql = **"Update book set "** count = 0  
 for field in inputs:  
 for x in headers:  
 if (field.upper() == x.upper()):  
 count = 1  
 if (count == 1):  
 val = input(**"Enter new "** + field + **": "**)  
 if ((val.isnumeric()) and

(field.upper() != **'ISBN'**)):  
 sql += field + **" = "** + val + **","** else:  
 sql += field + **" = '"** + val + **"',"** count = 0  
 else:  
 print(**''**)  
 print(**"Sorry, the field name "** + field +

**"does not exist!"**)  
 return  
 sql = sql[:-1]  
 sql += **" where book\_id="** + book\_id  
 Connection.mycursor.execute(sql)  
 Connection.mydb.commit()  
 print(**''**)  
 print(**"Book details for book id "** + book\_id +

**" has been updated successfully! "**)

print(**''**)  
 sql = **"select \* from book where book\_id="** + book\_id  
 Connection.mycursor.execute(sql)  
 res = Connection.mycursor.fetchall()  
 for x in res:  
 print(**''**)  
 print(tabulate(res,  
 headers=[**'Book\_Id'**, **'Book\_Name'**, MMMMMMMMMMMMMMMMM**'ISBN'**,**'Year\_published'**,**'Author'**, KKKKKKKKKKKKKKKKK**'Publisher'**, **'Quantity'**, KKKKKKKKKKKKKKKKK**'Available'**],tablefmt=**'psql'**))  
 print(**""**)  
 except Exception as e:  
 print(e)  
 return

def removeBook():  
 try:  
 bookDetails = displayAllBooks()  
 print(tabulate(bookDetails,  
 headers=[**'Book\_Id'**, **'Book\_Name'**, **'ISBN'**, **'Published Year'**, **'Author'**, **'Publisher'**, **'Quantity'**,**'Available'**], tablefmt=**'psql'**))  
 print(**''**)  
 book\_id = input(**"Enter book id to be deleted : "**)  
 print(**''**)  
 sql = **"select \* from book where book\_id="**+book\_id  
 Connection.mycursor.execute(sql)  
 res = Connection.mycursor.fetchall()  
 if(len(res)==0):  
 print(**''**)  
 print(**"Sorry, this book does not exist**

**in Library!"**)  
 return  
 sql=**"select**

**u.user\_id,u.user\_name,u.user\_email,u.user\_phone,**

**bh.book\_id, b.book\_name,bh.issue\_date,"** \  
 **"bh.return\_date,bh.approved from bookhistory**

**bh inner join**

**user u on bh.user\_id=u.user\_id left "** \  
 **"join book b on b.book\_id=bh.book\_id**

**where bh.book\_id="**+book\_id  
  
 Connection.mycursor.execute(sql)  
 res= Connection.mycursor.fetchall()  
  
 if (len(res)==0):  
 sql2=**"delete from book where book\_id="**+book\_id  
 Connection.mycursor.execute(sql2)  
 Connection.mydb.commit()  
 if ( Connection.mycursor.rowcount>0):  
 print(**"Book has been successfully removed!"**)  
 print(**''**)  
 bookDetails =displayAllBooks()  
 print(tabulate(bookDetails,  
 headers=[**'Book\_Id'**, **'Book\_Name'**, **'ISBN'**,

**'Published Year'**, **'Author'**, **'Publisher'**, skkkkkkkkkkkkkkkkk **'Quantity'**,**'Available'**], tablefmt=**'psql'**))  
 else:  
 for row in res:  
 if (row[8]==**"Y"**):  
 currentDate= time.strptime KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK(datetime.now().strftime

(**'%Y- %m-%d'**), **"%Y-%m-%d"**)  
 returnDate = time.strptime(row[7].strftime

(**'%Y-%m-%d'**), **"%Y-%m-%d"**)  
 if (returnDate>=currentDate):  
 print(**''**)  
 print(**"Sorry,this book is**

**already issued**

**to "**+row[1]+**" with return date ,LK "**+str(row[7]))  
 print(**''**)  
 print(**"Kindly delete after**

**the "**+str(row[7])+**" !"**)  
 print(**' '**)  
 print(**"Below are the issued**

**book details:-"**)  
 print(**''**)  
 print(tabulate (res,  
 llllllllllllllllllheaders=[**'User\_id'**,**'User\_name'**,**'user\_email'**, MMMMMMMMMMMMMMMMMM**'user\_phone'**,**'book\_id'**,**'book\_name'**,

,,,,,,,,,,,,,,,,,,,**'book\_issue\_date'**,**'book\_return\_date'**,

,K,KKKKKKKKKKKKKKKK**'book\_approved'**],tablefmt=**'psql'**))  
   
 else:  
 print(**"below are the issued**

**book details:-"**)  
 print(tabulate(res,  
 MMMMMMMMMMMMMMMMMMheaders=[**'User\_id'**,**'User\_name'**,**'user\_email'**, MKKKKMMMM**'user\_phone'**,**'book\_id','book\_name'**,**'book\_issue\_date'**, MMMMMMMMM**'book\_approved'**], tablefmt=**'psql'**))  
  
 print(**"This book was issued to"**

+ row[1] +

**"but return date has expired"**)  
 bookDeleteFromHistoryAndBook(book\_id)  
 else:  
 bookDeleteFromHistoryAndBook(book\_id)  
 return  
 except Exception as e:  
 print(e)  
def bookDeleteFromHistoryAndBook(book\_id):  
 print(**""**)  
 sql3=**"delete from BookHistory where book\_id="**+book\_id  
 Connection.mycursor.execute(sql3)  
 sql4=**"delete from book where book\_id="**+book\_id  
 Connection.mycursor.execute(sql4)  
 Connection.mydb.commit()  
 print(**"Book has been removed successfully!"**)  
 bookDetails = displayAllBooks()  
 print(tabulate(bookDetails,  
 headers=[**'Book\_Id'**, **'Book\_Name'**, **'ISBN'**,

**'Published Year'**, **'Author'**, **'Publisher'**, MMMMMMMMMMMMMMMMMM**'Quantity'**,**'Available'**], tablefmt=**'psql'**))

def displayIssuedBooks():  
 sql = **"select aaaaaaaaaau.user\_id,u.user\_name,u.user\_email,u.user\_phone,**

**bh.book\_id,"** \  
 **"b.book\_name,bh.issue\_date,bh.return\_date,**

**bh.approved from bookhistory bh inner join user u**

**"** \ **"on bh.user\_id=u.user\_id left join book b**

**on b.book\_id=bh.book\_id where bh.approved='Y'"** Connection.mycursor.execute(sql)  
 result = Connection.mycursor.fetchall()  
 if (len(result)==0):  
 print(**''**)  
 print(**"Sorry, Currently no books are issued!"**)  
 else:  
 print(**"Below are the issued book details:-"**)  
 print(**''**)  
 print(tabulate(result,  
 headers=[**'User\_id'**, **'User\_name'**, MNJJJJJJJJJJJJJJJJJJ**'User\_email'**,**'User\_phone'**, **'Book\_id'**,  
 **'Book\_name'**,**'Book\_Issue\_date'**, MMMMMMMMMMMMMMMMMMMM**'Book\_Return\_Date'**,**'Book\_Approved'**], MKKKKKKKKKKKKKKKKKKKKtablefmt=**'psql'**))  
  
  
def displayUnissuedBooks():  
 sql = **"select mmmmmmmmmmmu.user\_id,u.user\_name,u.user\_email,u.user\_phone,**

**bh.book\_id,"** \  
 **"b.book\_name,bh.issue\_date,bh.return\_date,**

**bh.approved from bookhistory bh inner join user u**

**"** \**"on bh.user\_id=u.user\_id left join book b**

**on b.book\_id=bh.book\_id where bh.approved='N'"** Connection.mycursor.execute(sql)  
 result = Connection.mycursor.fetchall()  
 if (len(result)==0):  
 print(**"Sorry, no pending books are to be issued!"**)  
 else:  
 print(**"Below books are pending to be issued:-"**)  
 print(**''**)  
 print(tabulate(result,  
 headers=[**'User\_id'**, **'User\_name'**, **'User\_email'**,**'User\_phone'**, **'Book\_id’,'Book\_name'**,  
 **'Book\_Issue\_date'**, **'Book\_Return\_Date'**,  
**'Book\_Approved'**], tablefmt=**'psql'**))  
  
def displaybookhistoryrecords():  
 sql = **"SELECT \* FROM Bookhistory"** Connection.mycursor.execute(sql)  
 result = Connection.mycursor.fetchall()  
 return result  
  
def approveUserRequest():  
 try:  
 sql = **"select kkkkkkkkkkkkkkku.user\_id,u.user\_name,bh.book\_id,b.book\_name,**

**bh.issue\_date,**

**bh.return\_date,bh.approved "** \  
 **"from bookhistory bh left join book b on "** \  
 **"bh.book\_id=b.book\_id left join "** \  
 **"user u on bh.user\_id=u.user\_id where**

**exists(select \* from bookhistory**

**where approved='N')"** Connection.mycursor.execute(sql)  
 res = Connection.mycursor.fetchall()  
 if (len(res)==0):  
 print(**''**)

print(**"All the books are currently approved in**

**the records!"**)  
 print(**''**)  
 displayIssuedBooks()  
 return  
 print(tabulate(res,  
 kkkkkkkkkkkkkkheaders=[**'user\_id'**,**'user\_name'**,**'book\_id'**,

**'book\_name'**,

**'issue\_date'**,**'return\_date'**,**'approved'**], kkkkkkkkkkkkkktablefmt=**'psql'**))  
 print(**""**)  
 book\_id=input(**"Please enter book id which is to**

**be approved: "**)  
 user\_id=input(**"Please enter user id for which the**

**book is to be approved: "**)

sql=**"select \* from bookhistory**

**where book\_id="**+book\_id+**"**

**and user\_id="**+user\_id  
 Connection.mycursor.execute(sql)  
 res = Connection.mycursor.fetchall()  
 if (len(res) == 0):  
 print(**''**)  
 print(**"Sorry, the combination of this book id**

**and user id**

**does not exist in the library!"**)  
 return  
 else:  
 issueDate = datetime.now().strftime(**'%Y-%m-%d'**)  
 returnDate = (datetime.now() +

timedelta(days=12)).strftime

(**'%Y-%m-%d'**)

sqlUpdate = **"Update bookhistory set**

**issue\_date= STR\_TO\_DATE**

**('"** + issueDate + **"', '%Y-%m-%d'), "** \  
 **"return\_date= STR\_TO\_DATE**

**('"** + returnDate +

**"','%Y-%m-%d')"** + **" ,approved='Y' "** \  
 **"where book\_id="** + book\_id +

**" and user\_id="** + user\_id  
 Connection.mycursor.execute(sqlUpdate)  
 if (Connection.mycursor.rowcount>0):  
 print(**''**)  
 print(**"Book with book\_id="**+book\_id+**" has**

**been issued to**

**the user with user\_id="**+user\_id)  
 displayIssuedBooks()  
 else:  
 print(**''**)  
 print(**"Sorry, this book**

**with book\_id="**+book\_id+**" could**

**not be issued "**)  
 return  
 except Exception as e:  
 print(e)  
 return

**# MAIN MENU PROGRAM**

from tabulate import tabulate  
import User  
import UserVerification  
import Book  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
 print(**''**)  
 print(**"------WELCOME TO THE 24/7 LIBRARY--------"**)  
 print(**''**)  
 print(**'-------Library Management System---------'**)  
 print(**''**)  
 ch = **"Y"** while ch == **"Y"**:  
 print(**'Login'**)  
 result = UserVerification.verifyUser()  
 if (len(result)==0):  
 print(**''**)  
 print(**"Incorrect user\_id or password. Please**

**try again!"**)  
 print(**"#############################"**)  
 print(**''**)

else:  
 for row in result:  
 user\_id=row[0]  
 role=row[1]  
 if(role == **"admin"**):  
 print(**''**)  
 print(**"Below is list of menus you can work on :-"**)  
 print(**''**)  
 while ch == **"Y"**:  
 print(**"1. Display all book details"**)  
 print(**"2. Display all issued book details"**)  
 print(**"3. Display unissued book details"**)  
 print(**"4. Register a new book in library"**)  
 print(**"5. Modify book details"**)  
 print(**"6. Remove book"**)  
 print(**"7. Register User Details"**)  
 print(**"8. Display all user details"**)  
 print(**"9. Modify User details"**)  
 print(**"10. Approve the book requested by user"**)  
 print(**"11. Logout"**)  
 print(**"12. Exit"**)  
 print(**''**)  
 choice = int(input(**"Enter your choice:"**))

if(choice == 1):  
 print(**''**)  
 bookDetails = Book.displayAllBooks()  
 print(tabulate(bookDetails,

headers=[**'Book\_Id'**, mmmmmmmmmmmmmmmmmmm**'Book\_Name'**,**'ISBN'**,**'Published JJJJJJJJJJJJJJJJJJJJYear'**,**'Author'**,**'Publisher'**,**'Quantity'**,

**'Available'**],tablefmt=**'psql'**))  
 elif(choice == 2):  
 print(**''**)  
 Book.displayIssuedBooks()  
 elif (choice == 3):  
 print(**''**)  
 Book.displayUnissuedBooks()  
 elif(choice == 4):  
 print(**''**)  
 bookCreateCount = Book.createBook()  
 if(bookCreateCount >0):  
 print(**"Book has been registered successfully"**)  
 print(**''**)  
 bookDetails = Book.displayAllBooks()  
 print(tabulate(bookDetails,  
 headers=[**'Book\_Id'**, **'Book\_Name'**, mmmmmmmmmmmmmmmmmmmmmmm**'ISBN'**,**'Published Year'**, **'Author'**, mnnnnnnnnnnnnmnnmmnnnnn**'Publisher'**, **'Quantity'**, **'Available'**],  
 tablefmt=**'psql'**))  
 elif (choice == 5):  
 print(**''**)  
 Book.modifyBook()  
 elif (choice == 6):  
 print(**''**)  
 Book.removeBook()  
 elif (choice == 7):  
 print(**''**)  
 User.createUser();  
 elif (choice == 8):  
 print(**''**)  
 userdetails=User.displayAllUserDetails()  
 print(tabulate(userdetails,  
 headers=[**'user\_id'**, **'user\_name'**, **'user\_address'**,**'user\_email'**, **'user\_phone'**], tablefmt=**'psql'**))  
 elif (choice == 9):  
 print(**''**)  
 User.modifyUserDetails()  
 elif (choice == 10):  
 print(**''**)  
 Book.approveUserRequest()  
 elif (choice == 11):  
 print(**''**)  
 print(**"You have logged out from your account!"**)  
 print(**"##########################"**)  
 print(**''**)  
 break  
 else:  
 print(**''**)  
 print(**"THANK YOU FOR VISITING 24/7 LIBRARY"**)  
 print(**''**)  
 print(**" -----XXXX-----XXXX----XXX-----"**)  
 print(**''**)  
 exit(0)  
 print(**''**)  
 ch = input(**"Do You Want to Continue <y/n>:"**)  
 print(**''**)  
 print(**"Below is list of menus you can work on :-"**)  
 print(**''**)  
 ch = ch.upper()  
 if(ch==**"N"**):  
 print(**''**)  
 print(**"THANK YOU FOR VISITING 24/7 LIBRARY"**)  
 print(**''**)  
 print(**" -----XXXX-----XXXX----XXX-----"**)  
 print(**''**)  
 break  
elif(role == **"user"**):  
 while ch == **"Y"**:  
 print(**''**)  
 print(**"Below is list of menus you can work on :-"**)  
 print(**''**)  
 print(**"1. Display Available Book Details"**)  
 print(**"2. Check Issued Book Details"**)  
 print(**"3. Request to issue a new book"**)  
 print(**"4. Logout"**)  
 print(**"5. Exit"**)  
 print(**''**)  
 choice = int(input(**"Enter your choice:"**))  
 if(choice == 1):  
 print(**''**)  
 bookDetails=User.displayAllAvailableBooks()  
 print(tabulate(bookDetails,  
 headers=[**'Book\_Id'**, **'Book\_Name'**, **'ISBN'**, **'Published Year'**, **'Author'**, **'Publisher'**,  
 **'Quantity'**, **'Available'**], tablefmt=**'psql'**))  
 elif(choice == 2):  
 print(**''**)  
 issuedBooks=User.displayIssuedBooks(user\_id)  
 if (len(issuedBooks)>0):  
 print(tabulate(issuedBooks,  
 MKKKKKKKKKKKKKKKKKKKheaders=[**'user\_Name'**,**'user\_phone'**,

**'user\_email','user\_address'**,**'book\_name'**,**'issue\_date'**,

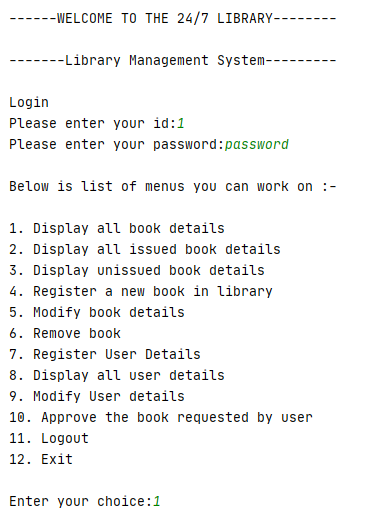
**'return\_date’,** **"** \**" 'isbn'**, **'year\_published'**, **'author'**,

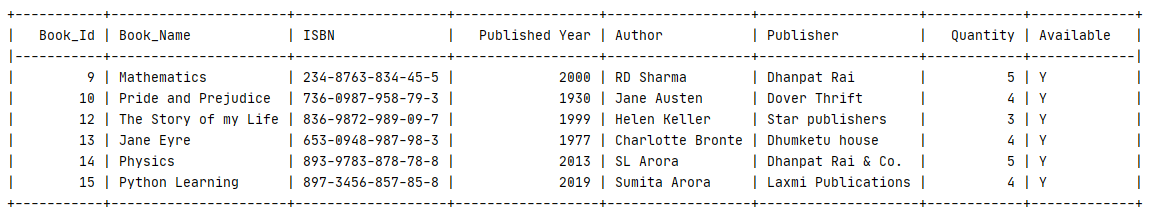
**'publisher'**], tablefmt=**'psql'**))  
 elif(choice == 3):  
 print(**''**)  
 issuebook=User.issueNewBook(user\_id)  
 elif (choice == 4):  
 print(**''**)  
 print(**"You have logged out from your account!"**)  
 print(**"##########################"**)  
 print(**''**)  
 break

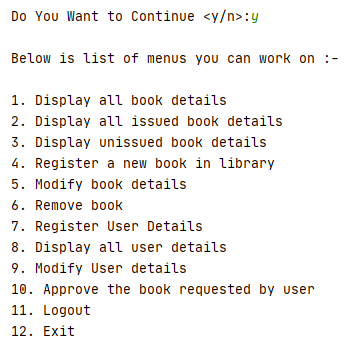
else:  
 exit(0)  
 ch = input(**"Do You Want to Continue <y/n>:"**)  
 print(**''**)  
 ch = ch.upper()  
 if(ch==**"N"**):  
 print(**''**)  
 print(**"THANK YOU FOR VISITING 24/7 LIBRARY"**)  
 print(**''**)  
 print(**" -----XXXX-----XXXX----XXX-----"**)  
 print(**''**)  
 break  
else:  
 print(**'User id and password are incorrect, please retry!'**)  
 print(**'###################################'**)

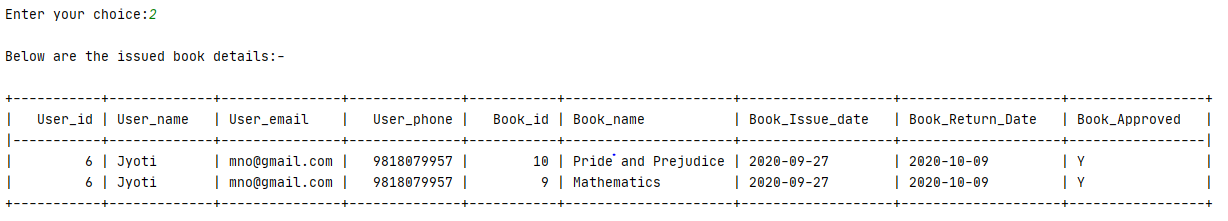
OUTPUT

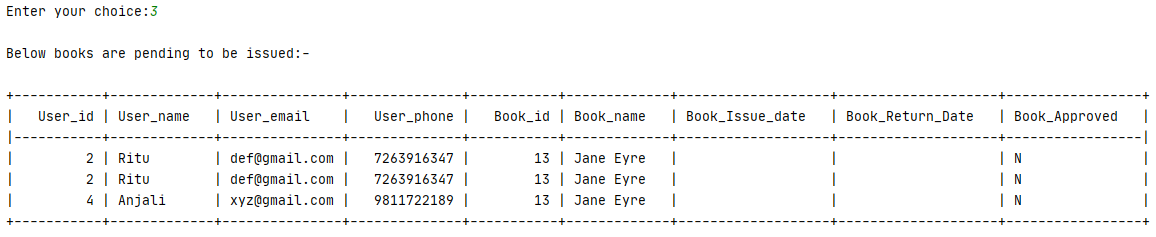
**FOR ADMIN:**

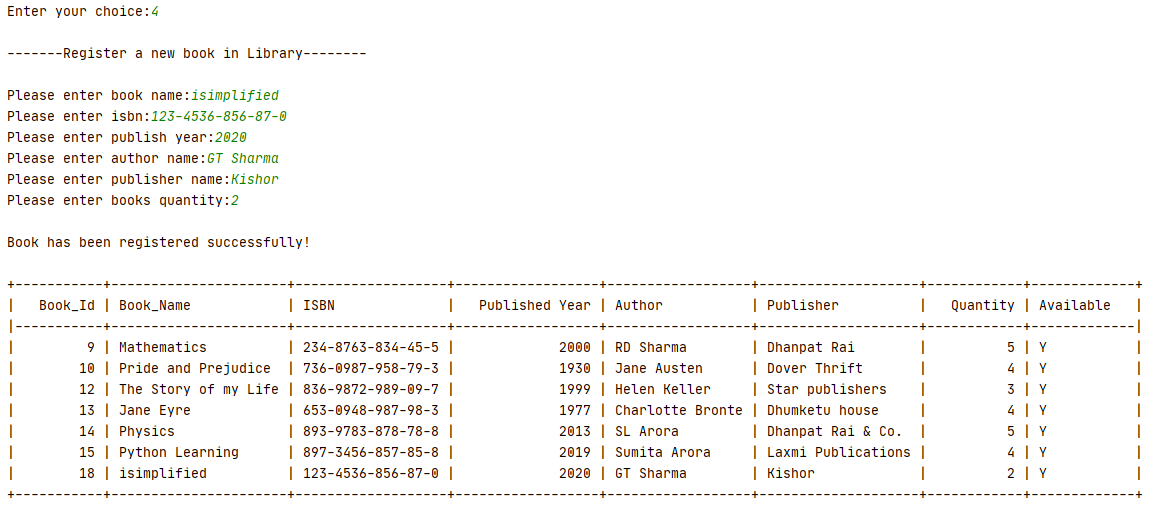


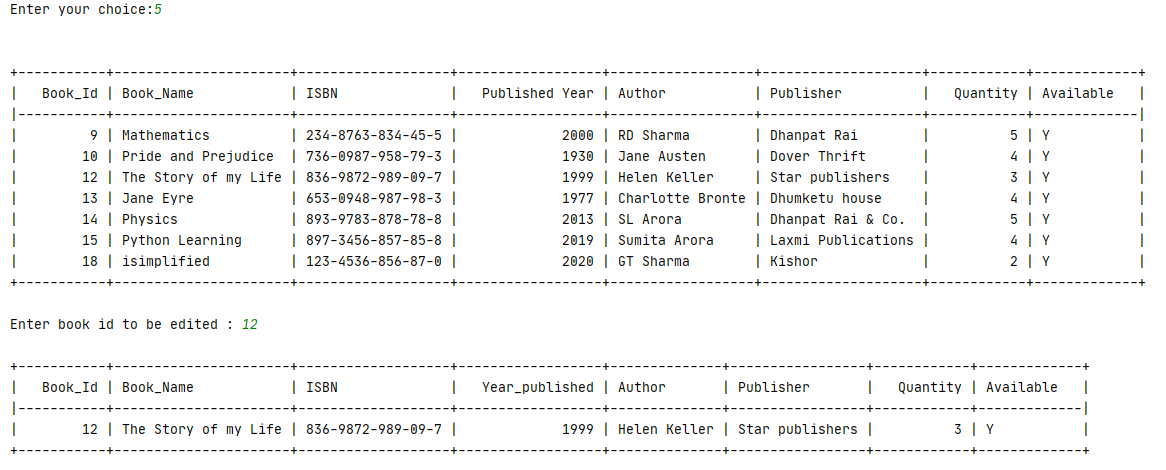


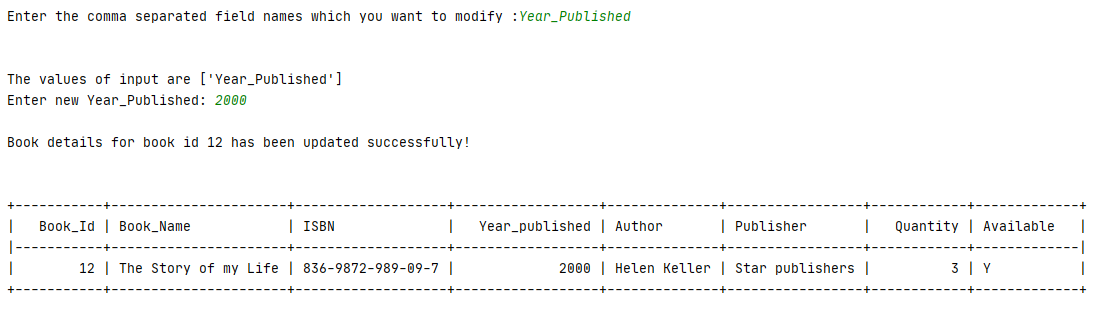


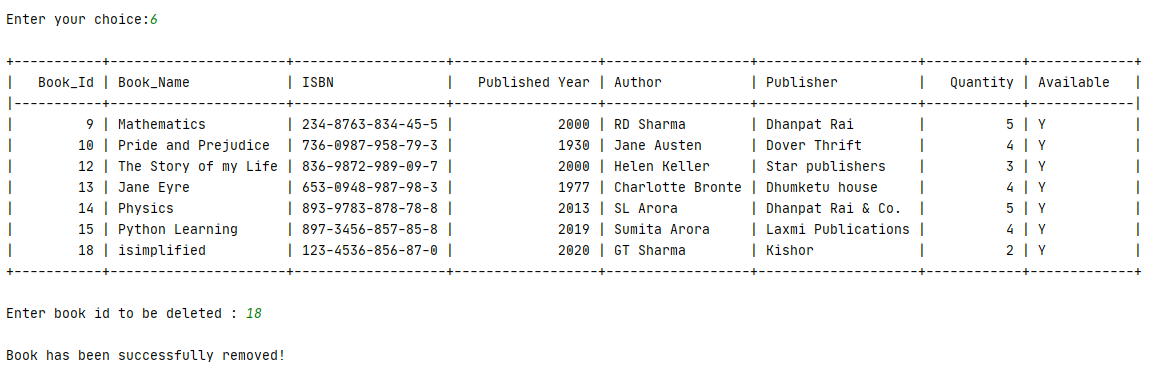


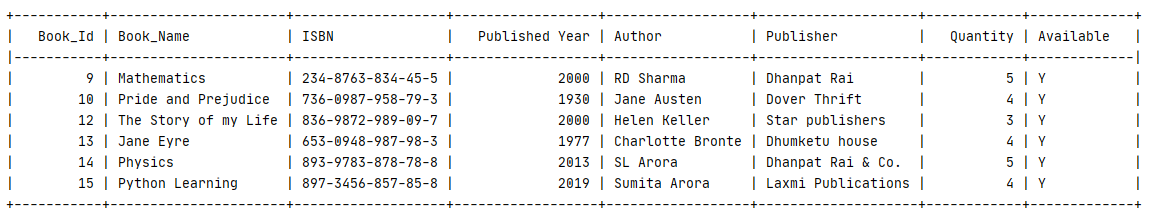


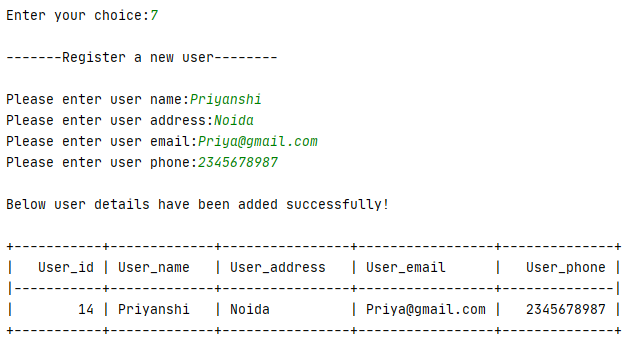


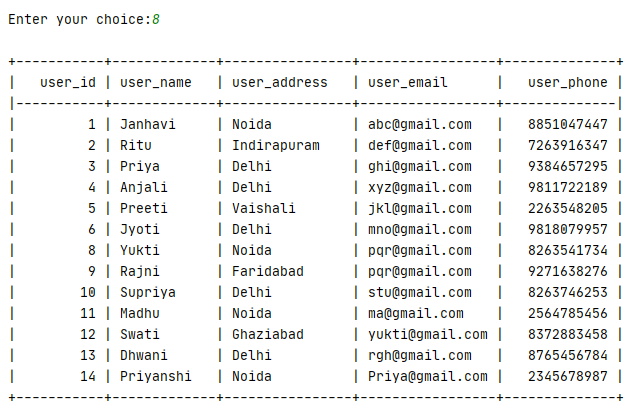


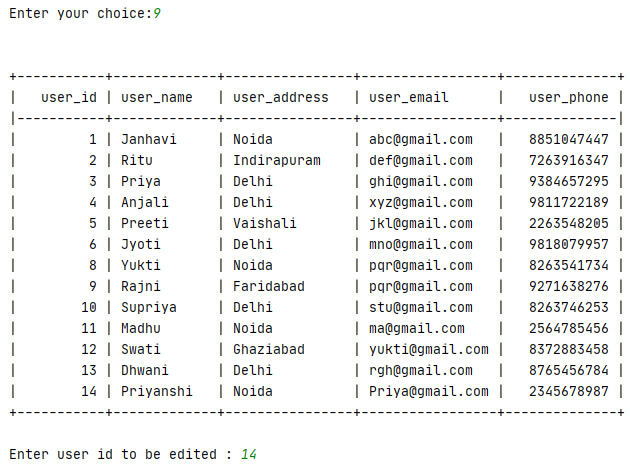


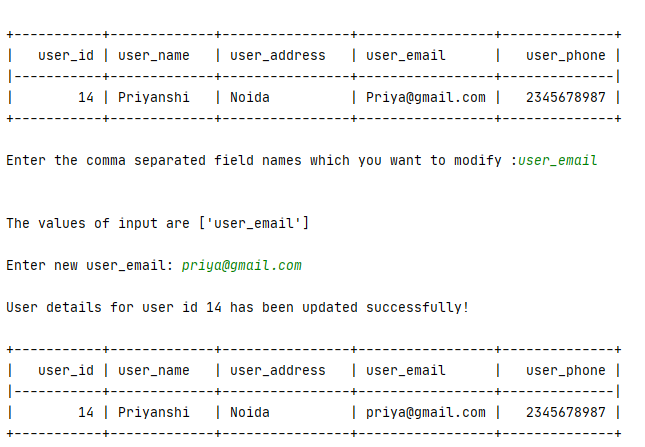


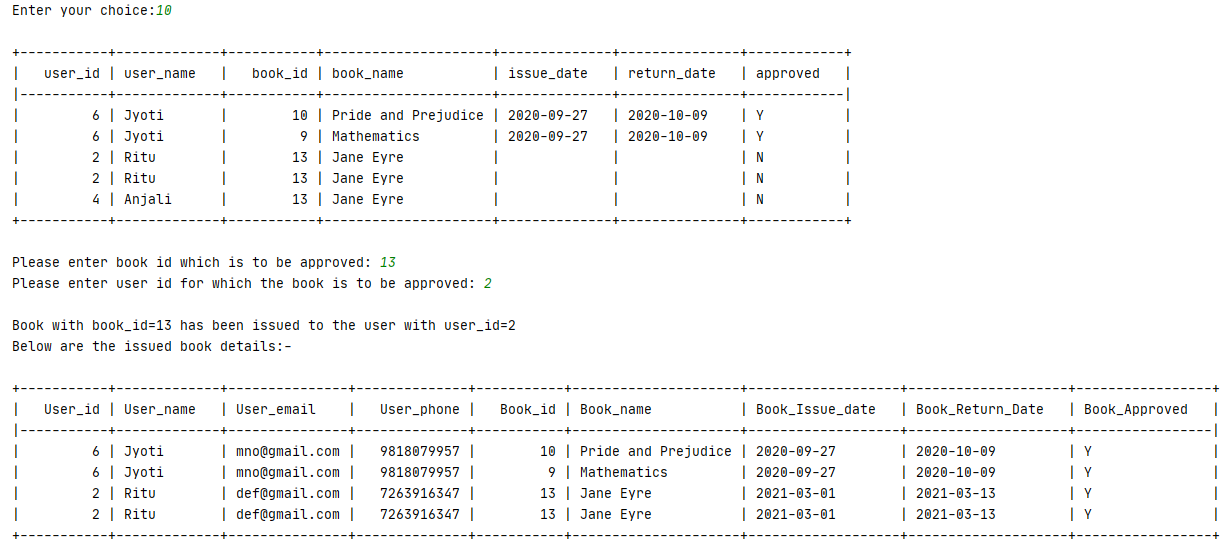


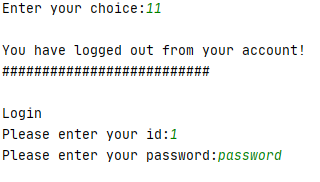


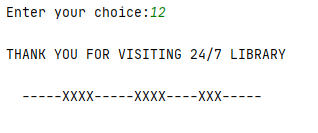




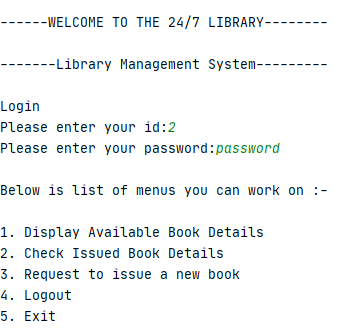


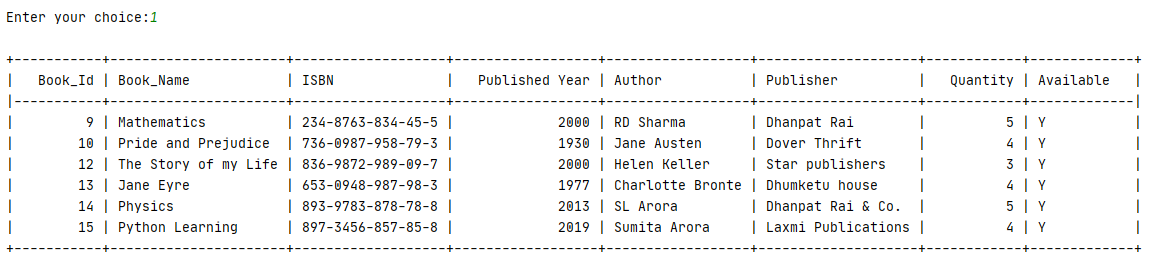


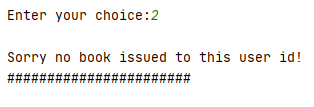


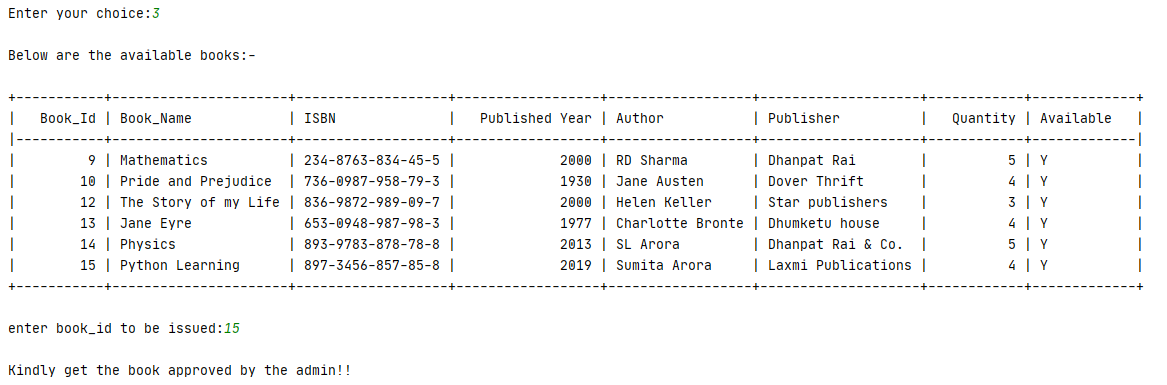


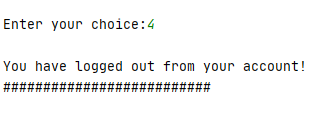
**FOR USER:**

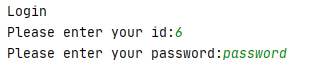


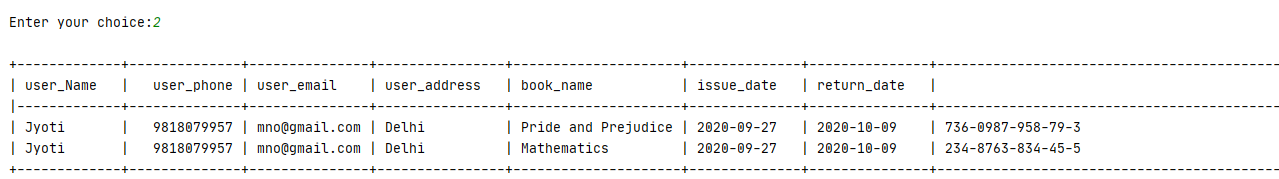












**User Manual**

**How to Install Software:**

Hardware Requirement-

¨ Intel Pentium/Celeron or similar processor based PC at Client/Server end.

¨ 128 MB RAM and 4GB HDD space (for Database) is desirable.

¨ Standard I/O devices like Keyboard and Mouse etc.

¨ Printer is needed for hard-copy reports.

¨ Local Area Network (LAN) is required for Client-Server Installation

Software Requirement-

¨ Windows 2000/XP OS is desirable.

¨ NetBeans Ver 5.1 or higher should be installed with JDK and JVM.

¨ MySQL Ver 6.1 with Library Database must be present at machine.

**Database Installation-**

The software project is distributed with a backup copy of a Database named LIBRARY with required tables. Some dummy records are present in the tables for testing purposes, which can be deleted before inserting real data. The project is shipped with manav.SQL file which installs a database and tables in the computer system.

Note: The PC must have MySQL server with user (root) and password (h). If root password is any other password, it can be changed by running MySQL Server Instance Configure Wizard.

Start 4Program 4 MySQL 4MySQL Server 4MySQL Server Instance Config Wizard

Provide current password of root and new password as “h”, this will change the root password.

To install a MySQL database from a dump file (faisal.sql), simply follow the following steps.

Step 1: Copy the manav.sql file in C:\Program files\Mysql\M

Step 2: Open MySQL and type the following command to create the database named travel agency.

mysql> create database library;

Step 3: Open Command Window (Start 4Run 4 cmd)

Step 4: Go to the following folder using CD command of DOS.

C:\Program files\Mysql\MySql server 5.1\Bin>

Step 5: type the following command on above prompt -

C:….\bin>mysql -u root - janhavi library <janhavi.sql>

This will create a library database with required tables.

**References**

In order to work on this project titled – LIBRARY MANAGEMENT SYSTEM, the following books and literature are referred by me during the various phases of development of the project:

(1) The Complete Reference Python 3.7 [-by Shildit]

(2) My SQL, Black Book [-by Steven Holzner]

(3) Understanding SQL– Gruber

(4) <http://www.mysql.org/>

(5) http://www.python.org/

(6) On-line Help of Python ®

(7) Informatics Practices for class XII [-by Sumita Arora]

(8) Together with Informatics Practices

(9) Various Websites of Discussion Forum and software development activities.

Other than the above-mentioned books, the suggestions and supervision of my teacher and my classmates also helped me to develop this software project.