COMPUTER SCIENCE PROJECT

LIBRARY MANAGEMENT SYSTEM

Biancaa. R

7206

12B

Vidya Mandir Estancia

ACKNOWLEDGEMENTS

I would like to extend my sincere and heartfelt gratitude to all those who have helped me with this project. Without their active guidance, cooperation and support, I wouldn’t have been able to complete and present this project on time.

I would like to thank my partner, Avanthika, without whom this project wouldn’t have been possible at all. I owe a deep sense of gratitude to her for her valuable suggestions, support, help and foresight during the entire phase of our work on this project.

I am particularly grateful for the assistance provided by our Librarian, Mrs. Sai Priya, for her insightful inputs and suggestions in the making of this project.

I would like to express my sincere thanks and gratitude to my Computer Science Teacher, Mrs. Vandana Sivaraj, who gave me this opportunity to increase my knowledge on the topic by doing this project – ‘Library Management System’. I would also like to express my gratitude to the lab teacher, Mrs. Vasantha Meenakshi, and the lab attendant, Mr. Sarathkumar, whose guidance has helped me complete this project without any difficulty.

I would like to thank our school principal, Mrs. Sankari Ravi, and the school management for the support they have offered me and for the help they did by offering their valuable suggestions and guidance for the completion of this project.

I acknowledge with a deep sense of reverence, my gratitude towards my parents, who have offered their invaluable support and encouragement throughout the making of this project.

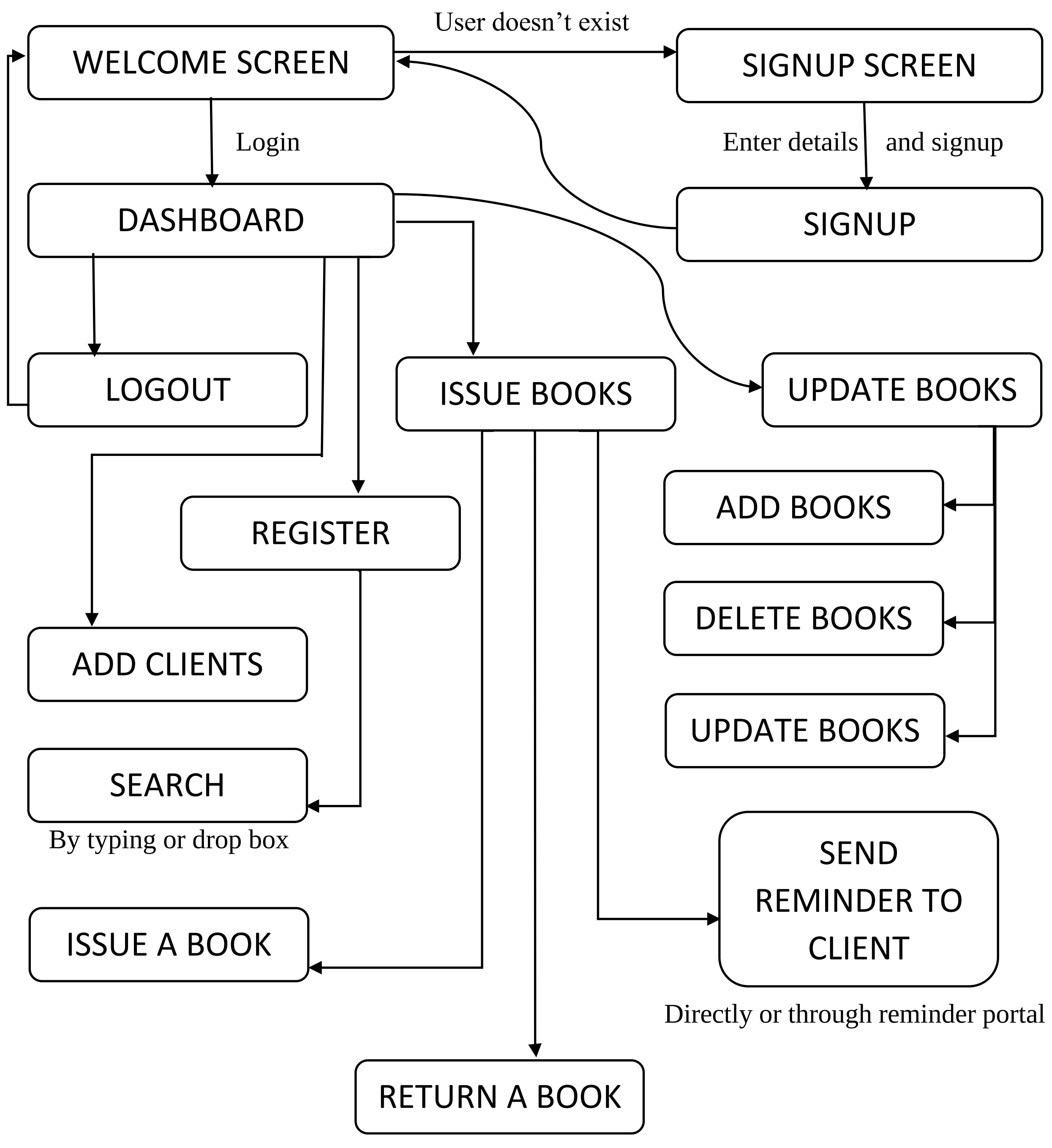
Last but not the least, I would like to thank my classmates, who have helped me clear my doubts and offered their support whenever needed.

INDEX

INTRODUCTION

This project has been titled – 'Library Management System'. Developed using the Python language, using MySQL database for storing records, the PyQt5 module for the GUI and the pywhatkit module for a feature that is used to send reminder messages – searching for books, adding, deleting and updating them, and issuing them to users has been made much easier with our program.

This is how it basically works:



CONCEPTS USED

In this work, concepts of Python Libraries, Modules, Classes, Functions and SQL have been used.

Libraries

There are so many options provided by Python to develop GUI applications and PyQt5 is one of them. It's the Python interface for Qt, one of the most powerful and cross-platform GUI libraries. One can develop an interactive desktop application with so much ease because of the tools provided and their simplicity.

A GUI application consists of Front-end and Back-end. PyQt5 has provided a tool called 'QtDesigner' to design the front-end by drag and drop method so that development can become faster.

Modules

The various modules that have been used are:

- pywhatkit, datetime and time modules for a feature that's used to send reminders to users
- user defined modules db and dbdetails that are used to establish a connection to the database and create tables in it
- modules like QtWidgets, QtGui, uic from PyQt5 used to define different classes used to run the GUI
- the mysql.connector module to perform SQL operations

Functions

Various functions, both user-defined and those from modules have been used to facilitate the working of the program.

SQL Concepts

Various SQL concepts have been used to store records and manipulate them in the database.

Software and Hardware used

Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz 2.80 GHz processor, 64 bit Windows OS with 8.00 GB RAM, Microsoft Visual Studio Code, QtDesigner

SOURCE CODE

main.py

```
#importing necessary modules
import pywhatkit, datetime, time
import dbdetails, db
import sys
from PyQt5.uic import loadUi
from PyQt5 import QtWidgets, QtGui
from PyQt5.QtWidgets import *
import mysql.connector

#creating database and new table in mysql
dbuser, dbpass = dbdetails.execute()
db.exec(dbuser,dbpass)

#Connecting to database
mydb = mysql.connector.connect(
    host = "localhost",
    user = dbuser,
    password = dbpass,
    database = "Library"
    )

cursor = mydb.cursor()
cursor = mydb.cursor(buffered=True)

#creating the register class, displays all books and their details
#user can search for records based on any category
class register(QDialog):
    def __init__(self):
        super(register, self).__init__()
        loadUi("table.ui",self)
        self.homeButton.clicked.connect(self.gotodash)
        self.pushButton.clicked.connect(self.gototype)
        self.logout.clicked.connect(self.gotologout)
        self.searchfield.setPlaceholderText("Search..")
        self.searchfield.textChanged.connect(self.gotosearch)
        self.searchButton.clicked.connect(self.gotosearch)
        cursor.execute("SELECT * FROM Register")
        result=cursor.fetchall()
        self.registertable.setColumnCount(len(result[0]))
        self.registertable.setRowCount(0)
        for row_number, row_data in enumerate(result):
            self.registertable.insertRow(row_number)
            for column_number, data in enumerate(row_data):
                self.registertable.setItem(row_number,
column_number, QTableWidgetItem(str(data)))

    def gotosearch(self):
        searchvalue = self.searchfield.text()
        category = self.categoryBox.currentText()
        cursor.execute("SELECT * FROM Register WHERE {} LIKE
'%{}%'".format(category, searchvalue))
        result=cursor.fetchall()
        if result:
            self.label.setText("")
            self.registertable.setColumnCount(len(result[0]))
            self.registertable.setRowCount(0)
            for row_number, row_data in enumerate(result):
                self.registertable.insertRow(row_number)
                for column_number, data in enumerate(row_data):
                    self.registertable.setItem(row_number,
column_number, QTableWidgetItem(str(data)))
        else:
            self.label.setText("No such record found.")
            cursor.execute("SELECT * FROM Register")
            result=cursor.fetchall()
            self.registertable.setColumnCount(len(result[0]))
            self.registertable.setRowCount(0)
            for row_number, row_data in enumerate(result):
                self.registertable.insertRow(row_number)
                for column_number, data in enumerate(row_data):
                    self.registertable.setItem(row_number,
column_number, QTableWidgetItem(str(data)))

    def gototype(self):
        typesearch=TypeSearch()
        widget.addWidget(typesearch)
        widget.setCurrentWidget(typesearch)

    def gotodash(self):
        dashboard = dashBoard()
        widget.addWidget(dashboard)
        widget.setCurrentWidget(dashboard)

    def gotologout(self):
        welcome = WelcomeScreen()
        widget.addWidget(welcome)
        widget.setCurrentWidget(welcome)

#creating typesearch class, to search for records by typing
class TypeSearch(QDialog):
    def __init__(self):
        super(TypeSearch, self).__init__()
        loadUi("searchtype1.ui", self)
        self.tableWidget.setColumnWidth(0,25)
        self.tableWidget.setColumnWidth(1,200)
        self.tableWidget.setColumnWidth(5,150)
        self.searchButton.clicked.connect(self.gotosearching)
        self.backButton.clicked.connect(self.gotoprevious)
        self.homeButton.clicked.connect(self.gotodash)
        self.logout.clicked.connect(self.gotologout)
        self.loaddata()

    def gotoprevious(self):
        reg=register()
        widget.addWidget(reg)
        widget.setCurrentWidget(reg)

    def gotodash(self):
        dashboard = dashBoard()
        widget.addWidget(dashboard)
        widget.setCurrentWidget(dashboard)

    def gotologout(self):
        welcome = WelcomeScreen()
        widget.addWidget(welcome)
        widget.setCurrentWidget(welcome)

    def loaddata(self):
        cursor.execute("SELECT * FROM REGISTER")
        data=cursor.fetchall()
        row=0
        self.tableWidget.setRowCount(len(data))
        for i in data:
            self.tableWidget.setItem(row, 0,
QtWidgets.QTableWidgetItem(str(i[0])))
            self.tableWidget.setItem(row, 1,
QtWidgets.QTableWidgetItem(str(i[1])))
            self.tableWidget.setItem(row, 2,
QtWidgets.QTableWidgetItem(str(i[2])))
            self.tableWidget.setItem(row, 3,
QtWidgets.QTableWidgetItem(str(i[3])))
            self.tableWidget.setItem(row, 4,
QtWidgets.QTableWidgetItem(str(i[4])))
            self.tableWidget.setItem(row, 5,
QtWidgets.QTableWidgetItem(str(i[5])))
            self.tableWidget.setItem(row, 6,
QtWidgets.QTableWidgetItem(str(i[6])))
            self.tableWidget.setItem(row, 7,
QtWidgets.QTableWidgetItem(str(i[7])))
            self.tableWidget.setItem(row, 8,
QtWidgets.QTableWidgetItem(str(i[8])))
```

```
            self.tableWidget.setItem(row, 9,
QtWidgets.QTableWidgetItem(str(i[9])))

            row=row+1

    def gotosearching(self):
        global searchvalue, category
        searchvalue = self.searchfield.text()
        category=self.categoryfield.text()
        try:
            cursor.execute("SELECT * FROM Register WHERE {} LIKE
'%{}%'".format(category, searchvalue))
            data = cursor.fetchall()
            if data:
                search= SearchPage()
                widget.insertWidget(2, search)
                widget.setCurrentIndex(2)
            else:
                self.confirm.setText("No record exists.")
        except mysql.connector.Error:
            self.confirm.setText("Something went wrong. Please try
again after checking all the values.")

class SearchPage(QDialog):
    def __init__(self):
        super(SearchPage,self).__init__()
        loadUi("searchtype2.ui",self)
        self.backButton.clicked.connect(self.gototype)
        self.homeButton.clicked.connect(self.gotodash)
        self.logout.clicked.connect(self.gotologout)
        try:
            cursor.execute("SELECT * FROM Register WHERE {} LIKE
'%{}%'".format(category, searchvalue))
            value=cursor.fetchall()
            row=0
            self.tableWidget.setRowCount(len(value))
            for r in value:
                self.tableWidget.setItem(row, 0,
QtWidgets.QTableWidgetItem(str(r[0])))
                self.tableWidget.setItem(row, 1,
QtWidgets.QTableWidgetItem(str(r[1])))
                self.tableWidget.setItem(row, 2,
QtWidgets.QTableWidgetItem(str(r[2])))
                self.tableWidget.setItem(row, 3,
QtWidgets.QTableWidgetItem(str(r[3])))
                self.tableWidget.setItem(row, 4,
QtWidgets.QTableWidgetItem(str(r[4])))
                self.tableWidget.setItem(row, 5,
QtWidgets.QTableWidgetItem(str(r[5])))
```

```
                self.tableWidget.setItem(row, 6,
QtWidgets.QTableWidgetItem(str(r[6])))
                self.tableWidget.setItem(row, 7,
QtWidgets.QTableWidgetItem(str(r[7])))
                self.tableWidget.setItem(row, 8,
QtWidgets.QTableWidgetItem(str(r[8])))
                self.tableWidget.setItem(row, 9,
QtWidgets.QTableWidgetItem(str(r[9])))
                row=row+1
        except mysql.connector.Error:
            self.confirm.setText("Something went wrong. Please try
again after checking your values.")

    def gototype(self):
        typesearch=TypeSearch()
        widget.addWidget(typesearch)
        widget.setCurrentWidget(typesearch)

    def gotodash(self):
        dashboard = dashBoard()
        widget.addWidget(dashboard)
        widget.setCurrentWidget(dashboard)

    def gotologout(self):
        welcome = WelcomeScreen()
        widget.addWidget(welcome)
        widget.setCurrentWidget(welcome)

#creating issuebooks class, can be used to both issue books or mark
them returned
class IssueBooks(QDialog):
    def __init__(self):
        super(IssueBooks, self).__init__()
        loadUi("issueBooks.ui", self)
        self.homeButton.clicked.connect(self.gotodash)
        self.logout.clicked.connect(self.gotologout)
        self.issuebutton.clicked.connect(self.issueprocess)
        self.returnbutton.clicked.connect(self.returnprocess)
        self.reminderButton.clicked.connect(self.reminderprocess)
        self.tableButton.clicked.connect(self.gotoissuetable)
        self.portalButton.clicked.connect(self.gotoremportal)
        try:
            cursor.execute("SELECT * FROM IssueDetails ORDER BY
Date_issued DESC")
            result = cursor.fetchall()
            if result:
                self.registertable.setColumnCount(len(result[0]))
                self.registertable.setRowCount(0)
                for row_number, row_data in enumerate(result):
                    self.registertable.insertRow(row_number)
                    for column_number, data in enumerate(row_data):
                        self.registertable.setItem(row_number, 
column_number, QTableWidgetItem(str(data)))
        except mysql.connector.Error:
            self.confirm.setText("Something went wrong.")
    #defining the process to issue books
    def issueprocess(self):
        global Rollno
        global Book
        global Author
        global Spno
        Spno = self.spnofield.text()
        Book = self.bookfield.text()
        Author = self.authorfield.text()
        Rollno = self.rollnofield.text()
        if Spno:
            cursor.execute("SELECT * FROM Register WHERE Sp_no = 
{}".format(Spno, ))
            data = cursor.fetchall()

            def new():
                global Rollno
                global Book
                global Author
                global Spno
                #checking for details, retrieving them if not 
already specified
                if not Author:
                    cursor.execute("SELECT Author_Name FROM Register 
WHERE Sp_no = {}".format(Spno, ))
                    a = cursor.fetchone()
                    if a != "NULL":
                        Author = ""
                        for i in a:
                            Author += i
                if not Book:
                    cursor.execute("SELECT Book_Title FROM Register 
WHERE Sp_no = {}".format(Spno, ))
                    a = cursor.fetchone()
                    if a != "NULL":
                        Book = ""
                        for i in a:
                            Book += i
                if Rollno:
                    try:
                        y=
datetime.datetime.now()+datetime.timedelta(days=7)
                        y=str(y)
                        cursor.execute('INSERT INTO
IssueDetails(Sp_no, Book_Title, Author_Name, Roll_no,due)\
                            VALUES ({}, "{}", "{}",
{},"{}")'.format(Spno, Book, Author, Rollno,y))
                        mydb.commit()
                        self.confirm.setText("Book issued!")
                    except mysql.connector.Error as Err:
                        self.confirm.setText("Something went wrong.
Please check the values and try again.")
                        print(Err)
                else:
                    self.confirm.setText("Something went wrong.
Please check the values and try again.")

                cursor.execute("SELECT * FROM IssueDetails ORDER BY
Date_issued DESC")
                result = cursor.fetchall()
                if result:
                    self.registertable.setRowCount(len(result))
                    self.registertable.setColumnCount(len(result[0])
)
                    self.registertable.setRowCount(0)
                    for row_number, row_data in enumerate(result):
                        self.registertable.insertRow(row_number)
                        for column_number, data in
enumerate(row_data):
                        #print(column_number)
                            self.registertable.setItem(
                                row_number, column_number,
QTableWidgetItem(str(data)))
                else:
                    pass
            if data:
                cursor.execute("SELECT Status FROM IssueDetails
WHERE Sp_no = {} ORDER BY Date_issued DESC LIMIT 1".format(Spno, ))
                a = cursor.fetchone()
                if a:
                    status = ""
                    for i in a:
                        status += i

                    else:
                        pass

                    if str(status) == "Borrowed":
                        self.confirm.setText("Book is already
borrowed. Mark returned and try again.")
                    else:
                        new()
                else:
                    new()

            else:
                self.confirm.setText("Book not found")
        else:
            self.confirm.setText("Something went wrong. Please check
the values and try again.")
    #defining the process of returning book
    def returnprocess(self):
        Spno = self.returnedspnofield.text()
        if Spno:
            cursor.execute("SELECT * FROM IssueDetails WHERE Sp_no =
{}".format(Spno, ))
            data = cursor.fetchall()
            if data:
                try:
                    cursor.execute("""UPDATE IssueDetails SET Status
= 'Returned'
                    WHERE Sp_no = {} ORDER BY Date_issued DESC LIMIT
1""".format(Spno,))
                    mydb.commit()
                    self.confirm.setText("Book marked returned!")
                except mysql.connector.Error as Err:
                    self.confirm.setText("Something went wrong.
Please try again.")
                    print(Err)
            else:
                self.confirm.setText("Book doesn't exist or hasn't
been issued")

            cursor.execute("SELECT * FROM IssueDetails ORDER BY
Date_issued DESC")
            result = cursor.fetchall()
            if result:
                self.registertable.setColumnCount(len(result[0]))
                self.registertable.setRowCount(0)
                for row_number, row_data in enumerate(result):
                    self.registertable.insertRow(row_number)
                    for column_number, data in enumerate(row_data):
                        self.registertable.setItem(row_number,
column_number, QTableWidgetItem(str(data)))
            else:
                pass
        else:
            self.confirm.setText("Please enter a value.")

    def reminderprocess(self):
        Rollno = self.rollnofield.text()
        if Rollno:
            cursor.execute("SELECT STATUS FROM IssueDetails WHERE 
Roll_no = {}".format(Rollno,))
            data = cursor.fetchall()
            if data:
                L = []
                for i in data:
                    for j in i:
                        L.append(j)
                if "Borrowed" in L:
                    cursor.execute("SELECT PHONE FROM CLIENTINFO 
WHERE ROLL_NO= {}".format(Rollno))
                    result = cursor.fetchall()
                    if not result:
                        self.confirm.setText("The user does not 
exist.")
                    else:
                        cursor.execute("SELECT NAME FROM CLIENTINFO 
WHERE ROLL_NO= {}".format(Rollno))
                        name = cursor.fetchone()[0]
                        cursor.execute("SELECT DATE_ISSUED FROM 
ISSUEDETAILS WHERE ROLL_NO= {} AND STATUS = 
'Borrowed'".format(Rollno))
                        date = cursor.fetchone()[0]
                        name = str(name)
                        date = str(date)
                        date = date[:10]

                        for i in result:
                            for j in i:
                                j = str(j)
                        pywhatkit.sendwhatmsg_instantly("+91"+j, 
"Hello, this is the Librarian. "+name+", please return the book 
soon. The book was borrowed on "+date+".", tab_close=True)
                        self.confirm.setText("Message successfully 
sent!")
                else:
                    self.confirm.setText("The user has returned all 
books.")
            else:
                self.confirm.setText("The user does not exist or 
hasn't borrowed anything yet.")
        else:
            self.confirm.setText("Enter the roll number to send 
reminder for.")

    def gotoissuetable(self):
        issue = IssueTable()
        widget.addWidget(issue)
        widget.setCurrentWidget(issue)

    def gotoremportal(self):
        rempor = ReminderPortal()
        widget.addWidget(rempor)
        widget.setCurrentWidget(rempor)

    def gotologout(self):
        welcome = WelcomeScreen()
        widget.addWidget(welcome)
        widget.setCurrentWidget(welcome)

    def gotodash(self):
        dashboard = dashBoard()
        widget.addWidget(dashboard)
        widget.setCurrentWidget(dashboard)

#creating the issuetableclass, displays all past issues and returns
of books
class IssueTable(QDialog):
    def __init__(self):
        super(IssueTable, self).__init__()
        loadUi("issuetable.ui",self)
        self.homeButton.clicked.connect(self.gotodash)
        self.logout.clicked.connect(self.gotologout)
        self.searchfield.setPlaceholderText("Search..")
        self.searchfield.textChanged.connect(self.gotosearch)
        self.searchButton.clicked.connect(self.gotosearch)
        cursor.execute("SELECT * FROM IssueDetails")
        result=cursor.fetchall()
        self.registertable.setColumnCount(len(result[0]))
        self.registertable.setRowCount(0)
        for row_number, row_data in enumerate(result):
            self.registertable.insertRow(row_number)
            for column_number, data in enumerate(row_data):
                self.registertable.setItem(row_number,
column_number, QTableWidgetItem(str(data)))

    def gotosearch(self):
        searchvalue1 = self.searchfield.text()
        category1 = self.categoryBox.currentText()
        cursor.execute("SELECT * FROM IssueDetails WHERE {} LIKE
'%{}%'".format(category1, searchvalue1))
        result=cursor.fetchall()
        if result:
            self.label.setText("")
            self.registertable.setColumnCount(len(result[0]))
            self.registertable.setRowCount(0)
            for row_number, row_data in enumerate(result):
                self.registertable.insertRow(row_number)
                for column_number, data in enumerate(row_data):
                        self.registertable.setItem(row_number,
column_number, QTableWidgetItem(str(data)))
        else:
            self.label.setText("No such record found.")
            cursor.execute("SELECT * FROM IssueDetails")
            result=cursor.fetchall()
            self.registertable.setColumnCount(len(result[0]))
            self.registertable.setRowCount(0)
            for row_number, row_data in enumerate(result):
                self.registertable.insertRow(row_number)
                for column_number, data in enumerate(row_data):
                    self.registertable.setItem(row_number,
column_number, QTableWidgetItem(str(data)))

    def gotodash(self):
        dashboard = dashBoard()
        widget.addWidget(dashboard)
        widget.setCurrentWidget(dashboard)

    def gotologout(self):
        welcome = WelcomeScreen()
        widget.addWidget(welcome)
        widget.setCurrentWidget(welcome)

#the reminder portal, used to send reminders to users when they have
to return a book
class ReminderPortal(QDialog):
    def __init__(self):
        super(ReminderPortal, self).__init__()
        loadUi("reminderportal.ui", self)
        self.submitButton.clicked.connect(self.reminderprocess)
        self.homeButton.clicked.connect(self.gotodash)
        self.logout.clicked.connect(self.gotologout)
        self.issuedetails.clicked.connect(self.gotoissuedetails)

    def reminderprocess(self):
        Rollno = self.rollnofield.text()
        if Rollno:
            cursor.execute("SELECT STATUS FROM IssueDetails WHERE
Roll_no = {}".format(Rollno,))
            data = cursor.fetchall()
            if data:
                L = []
                for i in data:
                    for j in i:
                        L.append(j)
                if "Borrowed" not in L:
                    self.confirm.setText("The user has returned all 
books.")
                else:
                    message = self.messagefield.text()
                    if len(message) == 0:
                        message = "Please return your book soon. 
Thankyou!"
                    cursor.execute("SELECT PHONE FROM CLIENTINFO 
WHERE ROLL_NO= {}".format(Rollno))
                    result = cursor.fetchall()
                    if not result:
                        self.confirm.setText("The user does not 
exist.")
                    else:
                        cursor.execute("SELECT NAME FROM CLIENTINFO 
WHERE ROLL_NO= {}".format(Rollno))
                        name = cursor.fetchone()[0]
                        cursor.execute("SELECT DATE_ISSUED FROM 
ISSUEDETAILS WHERE ROLL_NO= {}".format(Rollno))
                        date = cursor.fetchone()[0]
                        name = str(name)
                        date = str(date)
                        date = date[:10]
                        for i in result:
                            for j in i:
                                j = str(j)
                        pywhatkit.sendwhatmsg_instantly("+91"+j, 
"Hello, this is the Librarian. "+name+", "+message+". The book was 
borrowed on "+date+".", tab_close=True)
                        time.sleep(2)
                        self.confirm.setText("Message successfully 
sent!")
            else:
                self.confirm.setText("The user does not exist or 
hasn't borrowed anything yet.")
        else:
            self.confirm.setText("Enter the roll number to send 
reminder for.")

    def gotoissuedetails(self):
        issuebooks = IssueBooks()
        widget.addWidget(issuebooks)
        widget.setCurrentWidget(issuebooks)

    def gotodash(self):
        dashboard = dashBoard()
        widget.addWidget(dashboard)
        widget.setCurrentWidget(dashboard)
    def gotologout(self):
        welcome = WelcomeScreen()
        widget.addWidget(welcome)
        widget.setCurrentWidget(welcome)

#creating updatemenu class, a menu to navigate to different update
processes
#which are add, delete and update
class UpdateMenu(QDialog):
    def __init__(self):
        super(UpdateMenu, self).__init__()
        loadUi("updateMenu.ui", self)
        self.addbooksbutton.clicked.connect(self.gotoaddbooks)
        self.delbooksbutton.clicked.connect(self.gotodelbooks)
        self.updatebutton.clicked.connect(self.gotoupdatebooks)
        self.homeButton.clicked.connect(self.gotodash)
        self.logout.clicked.connect(self.gotologout)

    def gotologout(self):
        welcome = WelcomeScreen()
        widget.addWidget(welcome)
        widget.setCurrentWidget(welcome)

    def gotodash(self):
        dashboard = dashBoard()
        widget.addWidget(dashboard)
        widget.setCurrentWidget(dashboard)

    def gotoaddbooks(self):
        addbooks = AddBooks()
        widget.addWidget(addbooks)
        widget.setCurrentWidget(addbooks)

    def gotodelbooks(self):
        delbooks = DeleteBooks()
        widget.addWidget(delbooks)
        widget.setCurrentWidget(delbooks)

    def gotoupdatebooks(self):
        updatebooks1 = UpdateBooks1()
        widget.addWidget(updatebooks1)
        widget.setCurrentWidget(updatebooks1)

#creating addbooks class, one can add new records through here
class AddBooks(QDialog):
    def __init__(self):
        super(AddBooks, self).__init__()
        loadUi("addBooks.ui",self)
        self.addButton.clicked.connect(self.takevalue)
        self.homeButton.clicked.connect(self.gotodash)
        self.logout.clicked.connect(self.gotologout)

    def takevalue(self):
        Spno = self.spnofield.text()
        Name = self.namefield.text()
        Author = self.authorfield.text()
        Publisher = self.publisherfield.text()
        Class = self.classfield.text()
        Year = self.yearfield.text()
        Edition = self.editionfield.text()
        Subject = self.subjectfield.text()
        Category = self.categoryfield.text()
        Cost = self.costfield.text()

        if len(Spno)!=0:
            cursor.execute("SELECT * FROM Register WHERE Sp_no =
{}".format(Spno, ))
            data = cursor.fetchall()
            if data:
                self.confirm.setText("Record with same specimen
number already exists.")
            else:
                charvalues = [Author, Edition, Subject, Category]
                for i in charvalues:
                    if len(i) == 0:
                        i = ''
                    else:
                        pass
                query = """INSERT INTO REGISTER (Sp_no, Book_Title,
Author_Name, Publisher, Class, Year_of_Publication, Edition,
Subject, Category, Cost)
                    VALUES ({}, "{}", "{}", "{}", {}, {}, "{}",
"{}", "{}", {})""".format(Spno, Name, Author, Publisher, Class,
Year, Edition, Subject, Category, Cost)
                if not Class:
                    query = """INSERT INTO REGISTER (Sp_no,
Book_Title, Author_Name, Publisher, Class, Year_of_Publication,
Edition, Subject, Category, Cost)
                    VALUES ({}, "{}", "{}", "{}", NULL, {}, "{}",
"{}", "{}", {})""".format(Spno, Name, Author, Publisher, Year,
Edition, Subject, Category, Cost)
                if not Year:
                    query = """INSERT INTO REGISTER (Sp_no,
Book_Title, Author_Name, Publisher, Class, Year_of_Publication,
Edition, Subject, Category, Cost)
```

```
                    VALUES ({}, "{}", "{}", "{}", {}, NULL, "{}",
"{}", "{}", {})""".format(Spno, Name, Author, Publisher, Class,
Edition, Subject, Category, Cost)
                if not Class and not Year:
                    query = """INSERT INTO REGISTER (Sp_no,
Book_Title, Author_Name, Publisher, Class, Year_of_Publication,
Edition, Subject, Category, Cost)
                    VALUES ({}, "{}", "{}", "{}", NULL, NULL, "{}",
"{}", "{}", {})""".format(Spno, Name, Author, Publisher, Edition,
Subject, Category, Cost)
                try:
                    cursor.execute(query)
                    self.confirm.setText("Book added successfully!")
                except mysql.connector.Error as Err:
                    self.confirm.setText("Something went wrong. Try
again after checking all values.")
                mydb.commit()

        elif len(Spno) == 0:
            self.confirm.setText("Something went wrong. Try again
after checking all values.")

    def gotodash(self):
        dashboard = dashBoard()
        widget.addWidget(dashboard)
        widget.setCurrentWidget(dashboard)

    def gotologout(self):
        welcome = WelcomeScreen()
        widget.addWidget(welcome)
        widget.setCurrentWidget(welcome)

#creating deletebooks class, one can delete records through here
class DeleteBooks(QDialog):
    def __init__(self):
        super(DeleteBooks, self).__init__()
        loadUi("deleteBooks.ui",self)
        self.homeButton.clicked.connect(self.gotodash)
        self.searchButton.clicked.connect(self.gotosearch)
        self.logout.clicked.connect(self.gotologout)
        self.confirmbutton.clicked.connect(self.gotosearchfirst)

    def gotosearchfirst(self):
        self.confirm.setText("First search for the book to be
deleted.")

    def gotosearch(self):
        global searchvalue, category
        searchvalue = self.searchfield.text()
        category = self.categoryBox.currentText()
        if searchvalue:
            cursor.execute("SELECT * FROM Register WHERE {} LIKE
'%{}%'".format(category, searchvalue))
            result=cursor.fetchall()
            if not result:
                self.confirm.setText("No record found.")
            else:
                self.confirm.setText("")
                self.registertable.setColumnCount(len(result[0]))
                self.registertable.setRowCount(0)
                for row_number, row_data in enumerate(result):
                    self.registertable.insertRow(row_number)
                    for column_number, data in enumerate(row_data):
                        self.registertable.setItem(row_number,
column_number, QTableWidgetItem(str(data)))
                if len(result)>1:
                    confirmbox = confirmBox()
                    if confirmbox.exec_():
                        self.confirm.setText("Click on proceed.")
                        self.confirmbutton.clicked.connect(self.mult
idelete)
                    else:
                        choose = Choose()
                        if choose.exec_():
                            global spno
                            spno = choose.spnofield.text()
                            self.confirm.setText("Click on
proceed.")
                            self.confirmbutton.clicked.connect(self.
choosedelete)
                        else:
                            pass
                        #self.confirm.setText("Please search using
another value such that only 1 required record is visible.")
                else:
                    self.confirm.setText("Click on proceed.")
                    self.confirmbutton.clicked.connect(self.singlede
lete)
        else:
            self.confirm.setText("Please enter a value.")

    def choosedelete(self):
        try:
            cursor.execute("DELETE FROM REGISTER WHERE Sp_No =
'{}'".format(spno))
            mydb.commit()
            self.confirm.setText("Book deleted successfully")
        except mysql.connector.Error:
            self.confirm.setText("Something went wrong. Please try
again.")

    def singledelete(self):
        try:
            cursor.execute("DELETE FROM REGISTER WHERE {} =
'{}'".format(category, searchvalue))
            mydb.commit()
            self.confirm.setText("Book deleted successfully")
        except mysql.connector.Error:
            self.confirm.setText("Something went wrong. Please try
again.")

    def multidelete(self):
        try:
            cursor.execute("DELETE FROM REGISTER WHERE {} LIKE
'%{}%'".format(category, searchvalue))
            mydb.commit()
            self.confirm.setText("Books deleted successfully")
        except mysql.connector.Error:
            self.confirm.setText("Something went wrong. Please try
again.")

    def gotodash(self):
        dashboard = dashBoard()
        widget.addWidget(dashboard)
        widget.setCurrentWidget(dashboard)

    def gotologout(self):
        welcome = WelcomeScreen()
        widget.addWidget(welcome)
        widget.setCurrentWidget(welcome)

class confirmBox(QDialog):
    def __init__(self):
        super(confirmBox, self).__init__()
        loadUi("confirmbox.ui", self)
        self.okButton.clicked.connect(self.gotook)
        self.cancelButton.clicked.connect(self.gotocancel)

    def gotook(self):
        self.accept()

    def gotocancel(self):
        self.reject()

class Choose(QDialog):
    def __init__(self):
        super(Choose, self).__init__()
        loadUi("choose.ui", self)
        self.okButton.clicked.connect(self.gotook)
        self.cancelButton.clicked.connect(self.gotocancel)

    def gotook(self):
        self.accept()

    def gotocancel(self):
        self.reject()

#creating updatebooks class, one can update records through here
class UpdateBooks1(QDialog):
    def __init__(self):
        super(UpdateBooks1, self).__init__()
        loadUi("updateBooks1.ui", self)
        self.homeButton.clicked.connect(self.gotodash)
        self.searchButton.clicked.connect(self.gotosearch)
        self.logout.clicked.connect(self.gotologout)
        self.confirmbutton.clicked.connect(self.searchfirst)

    def searchfirst(self):
        self.confirm.setText("First search for the book to be
updated.")

#searching for a record to update
    def gotosearch(self):
        global updatesearchvalue, updatecategory
        updatesearchvalue = self.searchfield.text()
        updatecategory = self.categoryBox.currentText()
        if updatesearchvalue:
            cursor.execute("SELECT * FROM Register WHERE {} LIKE
'%{}%'".format(updatecategory, updatesearchvalue))
            result=cursor.fetchall()
            if result:
                self.confirm.setText("")
                self.registertable.setColumnCount(len(result[0]))
                self.registertable.setRowCount(0)
                for row_number, row_data in enumerate(result):
                    self.registertable.insertRow(row_number)
                    for column_number, data in enumerate(row_data):
                        self.registertable.setItem(row_number,
column_number, QTableWidgetItem(str(data)))
                #checking for similar records
                if len(result)>1:
                    choose = Choose()
                    if choose.exec_():
                        updatesearchvalue = choose.spnofield.text()
                        self.confirm.setText("Click on proceed.")
                        self.confirmbutton.clicked.connect(self.upda
teBooks)
                    else:
                        self.confirm.setText("Enter a
value.")
                else:
                    self.confirm.setText("Click on proceed.")
                    self.confirmbutton.clicked.connect(self.updateBo
oks)
            else:
                self.confirm.setText("No record found.")
        else:
            self.confirm.setText("Something went wrong. Please try
again after checking all values.")


    def updateBooks(self):
        updatebooks2 = UpdateBooks2()
        widget.addWidget(updatebooks2)
        widget.setCurrentWidget(updatebooks2)

    def gotodash(self):
        dashboard = dashBoard()
        widget.addWidget(dashboard)
        widget.setCurrentWidget(dashboard)

    def gotologout(self):
        welcome = WelcomeScreen()
        widget.addWidget(welcome)
        widget.setCurrentWidget(welcome)

class UpdateBooks2(QDialog):
    def __init__(self):
        super(UpdateBooks2, self).__init__()
        loadUi("updateBooks2.ui", self)
        self.homeButton.clicked.connect(self.gotodash)
        self.logout.clicked.connect(self.gotologout)
        self.confirmbutton.clicked.connect(self.updateprocess)

#updating the record
    def updateprocess(self):
        try:
            newvalue = self.newrecordfield.text()
            category = self.categoryBox.currentText()
            if not newvalue:
                self.confirm.setText("Please enter the value to
update.")
            else:
                if type(newvalue) is str:
                    cursor.execute("UPDATE Register SET {} = '{}'
WHERE {} = {}".format(category, newvalue, updatecategory,
updatesearchvalue))
                else:
                    cursor.execute("UPDATE Register SET {} = {}
WHERE {} = {}".format(category, newvalue, updatecategory,
updatesearchvalue))
                mydb.commit()
                self.confirm.setText("Record updated successfully!")
                cursor.execute("SELECT * FROM Register WHERE {} LIKE
'%{}%'".format(updatecategory, updatesearchvalue))
                result = cursor.fetchall()
                self.registertable.setColumnCount(len(result[0]))
                self.registertable.setRowCount(0)
                for row_number, row_data in enumerate(result):
                    self.registertable.insertRow(row_number)
                    for column_number, data in enumerate(row_data):
                        self.registertable.setItem(row_number,
column_number, QTableWidgetItem(str(data)))
        except mysql.connector.Error as Err:
            self.confirm.setText("Something went wrong. Try again
after checking values.")

    def gotodash(self):
        dashboard = dashBoard()
        widget.addWidget(dashboard)
        widget.setCurrentWidget(dashboard)

    def gotologout(self):
        welcome = WelcomeScreen()
        widget.addWidget(welcome)
        widget.setCurrentWidget(welcome)

#creating infobox class, displays info about our project
class InfoBox(QDialog):
    def __init__(self):
        super(InfoBox, self).__init__(parent=welcome)
        loadUi("info.ui", self)
        self.okButton.clicked.connect(self.gotoclose)

    def gotoclose(self):
        self.close()

#creating welcomescreen class, the main login screen which shows
first
class WelcomeScreen(QMainWindow):
    #initializing and loading welcome screen
    def __init__(self):
        super(WelcomeScreen, self).__init__()
        loadUi("welcome.ui", self)
        self.infoButton.clicked.connect(self.gotoinfo)
        self.usernamefield.setPlaceholderText("Username")
        self.passwordfield.setPlaceholderText("Password")
        self.passwordfield.setEchoMode(QtWidgets.QLineEdit.Password)
        self.loginbutton.clicked.connect(self.gotodashBoard)
        self.passwordfield.returnPressed.connect(self.gotodashBoard)
        self.newusersignup.clicked.connect(self.gotosignup)

    def gotoinfo(self):
        self.info = InfoBox()
        self.info.setWindowTitle("Info")
        self.info.show()

    def gotosignup(self):
        signup = SignUpScreen()
        widget.addWidget(signup)
        widget.setCurrentWidget(signup)

    #checking user credentials
    def gotodashBoard(self):
        global username
        username = self.usernamefield.text()
        password = self.passwordfield.text()

        if len(username)==0 or len(password)==0:
            self.error.setText("Please fill in all fields")

        else:
            query = "SELECT password FROM login_info WHERE username
= '"+username+"'"
            cursor.execute(query)
            result_pass = cursor.fetchone()

            if result_pass is not None:
                if result_pass[0] == password:
                    print("Successfully logged in.")
                    dashboard = dashBoard()
                    widget.addWidget(dashboard)
                    widget.setCurrentWidget(dashboard)
                else:
                    self.error.setText("Invalid username or
password")
            else:
                self.error.setText("Invalid username or password")

#creating signupscreen class, users signup here
class SignUpScreen(QDialog):
    #initializing and loading signup screen
    def __init__(self):
        super(SignUpScreen, self).__init__()
        loadUi("signup.ui", self)
        self.signupname.setPlaceholderText("Name")
        self.signupuser.setPlaceholderText("Preferred username")
        self.signuppass.setPlaceholderText("Enter password")
        self.confirmpass.setPlaceholderText("Confirm password")
        self.signuppass.setEchoMode(QtWidgets.QLineEdit.Password)
        self.confirmpass.setEchoMode(QtWidgets.QLineEdit.Password)
        self.signupbutton.clicked.connect(self.gotosignup)
        self.backtologin.clicked.connect(self.gotologin)

    #logging out back to welcome screen
    def gotologin(self):
        welcome = WelcomeScreen()
        widget.addWidget(welcome)
        widget.setCurrentWidget(welcome)

    #checking user credentials
    def gotosignup(self):
        global newname, newuser, newpass
        newname = self.signupname.text()
        newuser = self.signupuser.text()
        newpass1 = self.signuppass.text()
        newpass2 = self.confirmpass.text()

        if len(newname)== 0 or len(newuser) == 0 or len(newpass1) == 
0 or len(newpass2) == 0:
            self.signuperror.setText("Please fill in all fields")

        elif newpass1 != newpass2:
            self.signuperror.setText("The passwords do not match")

        else:
            cursor.execute("SELECT * FROM login_info WHERE username 
= '"+newuser+"'")
            data = cursor.fetchall()
            if data:
                self.signuperror.setText("Username already exists")
            else:
                addnewquery = "INSERT IGNORE INTO login_info VALUES\
                    ('"+newname+"','"+newuser+"','"+newpass1+"')"
                cursor.execute(addnewquery)
                mydb.commit()
                self.signuperror.setText("Added new user to 
database. Successful!")

#creating clientscreen class
class ClientScreen(QDialog):
    #initializing and loading client screen
    def __init__(self):
        super(ClientScreen, self).__init__()
        loadUi("addClient.ui", self)
        self.homeButton.clicked.connect(self.gotodash)
        self.addButton.clicked.connect(self.addprocess)
        self.logout.clicked.connect(self.gotologout)
        self.rnofield.setPlaceholderText("Roll number of client")
        self.namefield.setPlaceholderText("Name of the client")
        self.phonefield.setPlaceholderText("Phone number of client")
        self.efield.setPlaceholderText("EmailID of client")

    #adding a new client
    def addprocess(self):
        rno=self.rnofield.text()
        name=self.namefield.text()
        phone=self.phonefield.text()
        email=self.efield.text()
        if not rno or not name or not phone:
            if not rno:
                self.value.setText("Please enter rollno.")
            elif not name:
                self.value.setText("Please enter the name of
client")
            elif not phone:
                self.value.setText("Please enter the phone number of
client")
        else:
            cursor.execute("SELECT * FROM ClientInfo WHERE Roll_no =
{}".format(rno, ))
            data = cursor.fetchall()
            if data:
                self.value.setText("Client already exists.")
            else:
                try:
                    cursor.execute("INSERT INTO CLIENTINFO (Roll_no,
Name, Phone, EmailID)
VALUES({},'{}',{},'{}')".format(rno,name,phone,email))
                    mydb.commit()
                    self.value.setText("Client added successfully")
                except mysql.connector.Error:
                    self.value.setText("Something went wrong. Please
try again after checking all values.")

    def gotodash(self):
        dashboard = dashBoard()
        widget.addWidget(dashboard)
        widget.setCurrentWidget(dashboard)
    def gotologout(self):
        welcome = WelcomeScreen()
        widget.addWidget(welcome)
        widget.setCurrentWidget(welcome)

#creating dashboard class, main dashboard from where one can
navigate
class dashBoard(QDialog):
    #initialising and loading dashboard
    def __init__(self):
        super(dashBoard, self).__init__()
        loadUi("dashboard.ui", self)
        self.logout.clicked.connect(self.gotologout)
        self.updateButton.clicked.connect(self.gotoupdatemenu)
        self.registerButton.clicked.connect(self.gotoRegister)
        self.issueButton.clicked.connect(self.gotoissuebooks)
        self.clientButton.clicked.connect(self.gotoaddclient)
        cursor.execute("SELECT name FROM login_info WHERE username =
'"+username+"'")
        greetname = cursor.fetchone()[0]
        self.name.setText("Hello "+greetname+"!")

    def gotologout(self):
        welcome = WelcomeScreen()
        widget.addWidget(welcome)
        widget.setCurrentWidget(welcome)

    def gotoaddclient(self):
        clientscreen=ClientScreen()
        widget.addWidget(clientscreen)
        widget.setCurrentWidget(clientscreen)

    def gotoRegister(self):
        reg = register()
        widget.addWidget(reg)
        widget.setCurrentWidget(reg)

    def gotoupdatemenu(self):
        updatebooks = UpdateMenu()
        widget.addWidget(updatebooks)
        widget.setCurrentWidget(updatebooks)

    def gotoissuebooks(self):
        issuebooks = IssueBooks()
        widget.addWidget(issuebooks)
        widget.setCurrentWidget(issuebooks)

#main
app = QtWidgets.QApplication(sys.argv)
```

```
widget = QStackedWidget()
widget.setWindowIcon(QtGui.QIcon("icon.png"))
welcome = WelcomeScreen()
signup = SignUpScreen()
widget.addWidget(welcome)
widget.addWidget(signup)
widget.setWindowTitle("VME Library Management")
widget.resize(1400, 750)
widget.show()

try:
    sys.exit(app.exec_())
except:
    print("Exiting..")
```

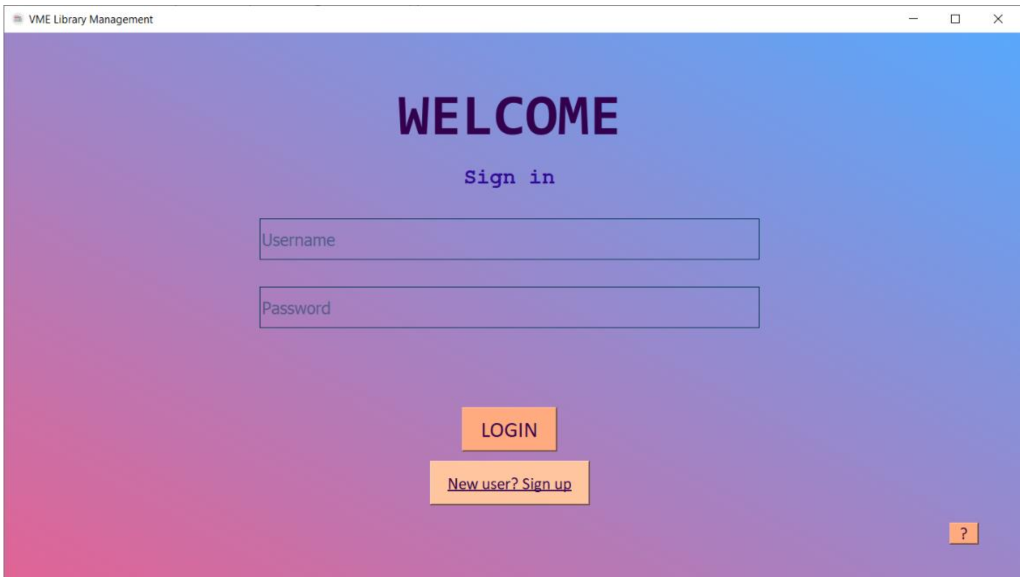
db.py

```
#-----------------------------------------------
#Connecting to server and creating new database
#-----------------------------------------------

import mysql.connector

def exec(dbuser, dbpass):
    #Connecting to server
    mydb = mysql.connector.connect(
                host = "localhost",
                user = dbuser,
                password = dbpass,
                database="library"
            )

    cursor = mydb.cursor()

    #Creating database
    cursor.execute("CREATE DATABASE IF NOT EXISTS Library")
    cursor.execute("USE Library")

    #Creating login table
    cursor.execute("""CREATE TABLE IF NOT EXISTS login_info
    (name VARCHAR(30) NOT NULL,
    username VARCHAR(30) NOT NULL PRIMARY KEY,
    password VARCHAR(30) NOT NULL)""")
    mydb.commit()

    #Creating the book register table
    cursor.execute("""CREATE TABLE IF NOT EXISTS Register
    (Sp_no INT PRIMARY KEY,
    Book_Title VARCHAR(200) NOT NULL,
    Author_Name VARCHAR(100),
    Publisher VARCHAR(100) NOT NULL,
    Class INT,
    Year_Of_Publication INT,
    Edition VARCHAR(20),
    Subject VARCHAR(50),
    Category VARCHAR(50),
    Cost INT NOT NULL)""")
    mydb.commit()
    #Creating the book issues table
    cursor.execute("""CREATE TABLE IF NOT EXISTS IssueDetails
    (Sp_no INT,
    Book_Title VARCHAR(200) NOT NULL,
    Author_Name VARCHAR(100),
    Roll_no INT NOT NULL,
    Date_issued DATETIME DEFAULT CURRENT_TIMESTAMP,
    Status VARCHAR(50) NOT NULL DEFAULT "Borrowed",
    Due VARCHAR(50))""")
    mydb.commit()

    cursor.execute("""CREATE TABLE IF NOT EXISTS ClientInfo
    (Roll_no INT,
    Name VARCHAR(40),
    Phone BIGINT,
    EmailID VARCHAR(50))""")
    mydb.commit()
```

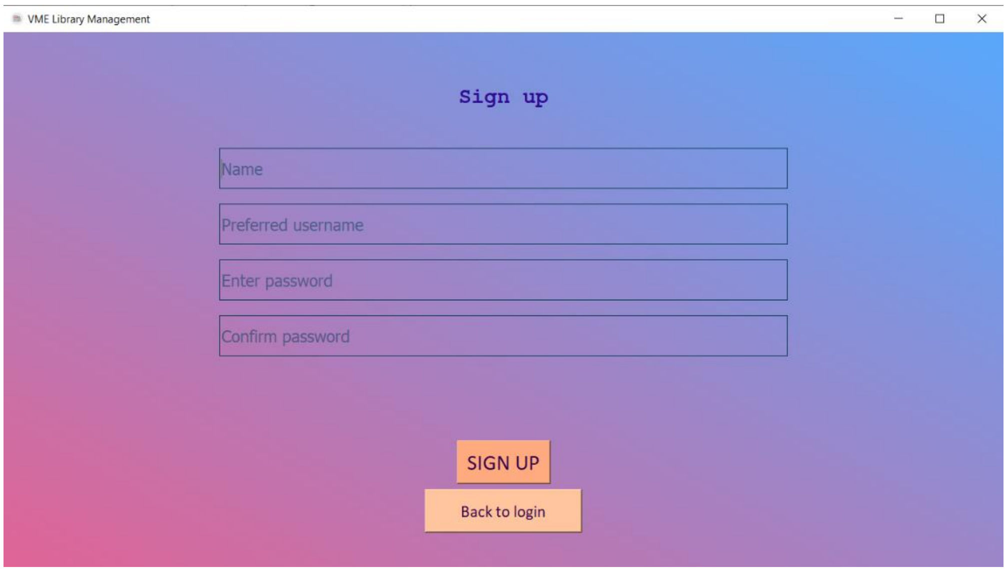
dbdetails.py

```
#----------------------------------------------------------
#Set your mysql username and password to create connections
#----------------------------------------------------------

def execute():

    #Enter user credentials from database of your computer
    #download mysql if it doesn't exist on your pc :)

    dbUSER = "root" #your mysql user name here
    dbPASS = "" #your mysql user password here
    return dbUSER, dbPASS
```

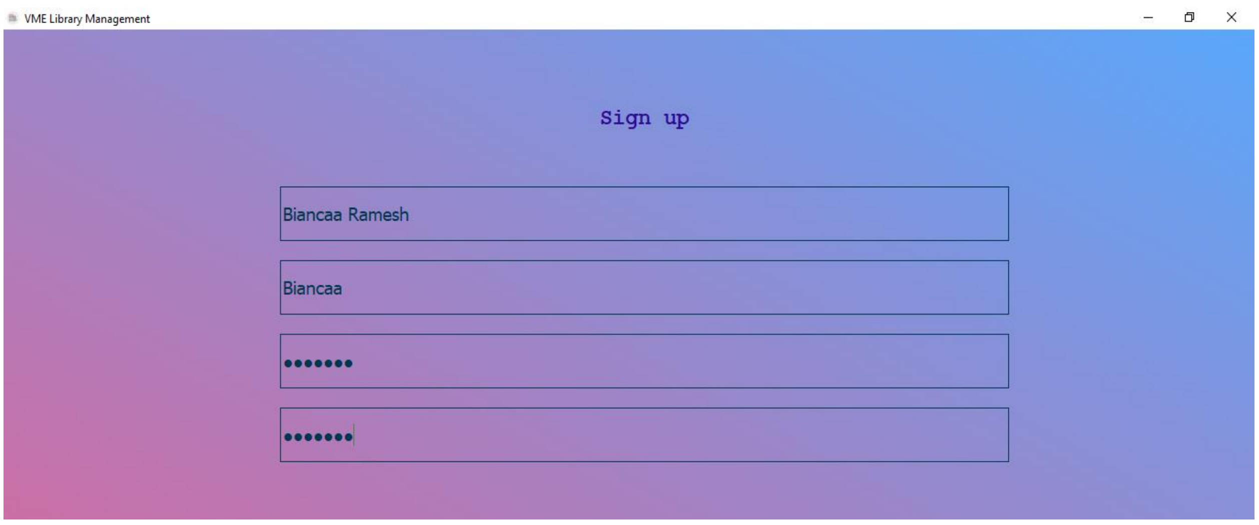
OUTPUT



The Welcome Screen



The Signup Screen



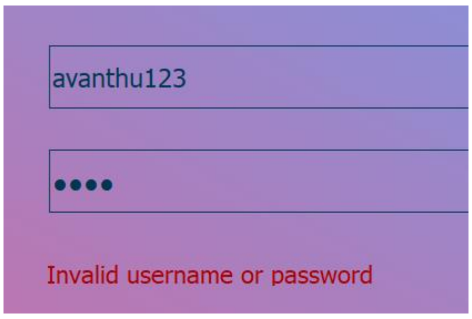
Biancaa Ramesh

Biancaa

•••••••

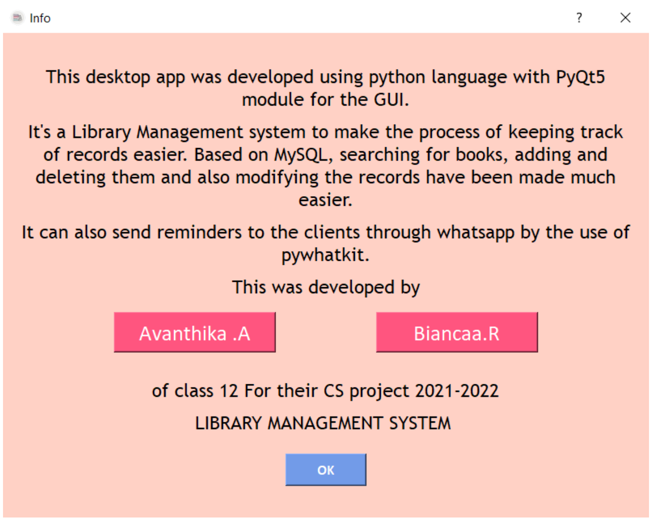
•••••••

When user enters wrong details


avanthu123

●●●●

Invalid username or password


Info

This desktop app was developed using python language with PyQt5 module for the GUI.

It's a Library Management system to make the process of keeping track of records easier. Based on MySQL, searching for books, adding and deleting them and also modifying the records have been made much easier.

It can also send reminders to the clients through whatsapp by the use of pywhatkit.

This was developed by

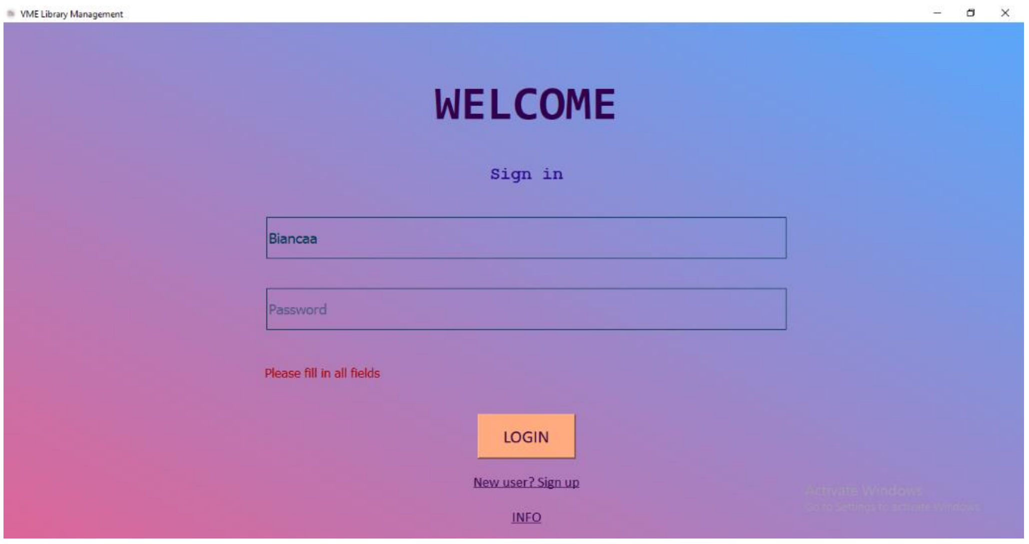
Avanthika .A   Biancaa.R

of class 12 For their CS project 2021-2022

LIBRARY MANAGEMENT SYSTEM

When user clicks on the '?'

If user leaves a field empty:


VME Library Management

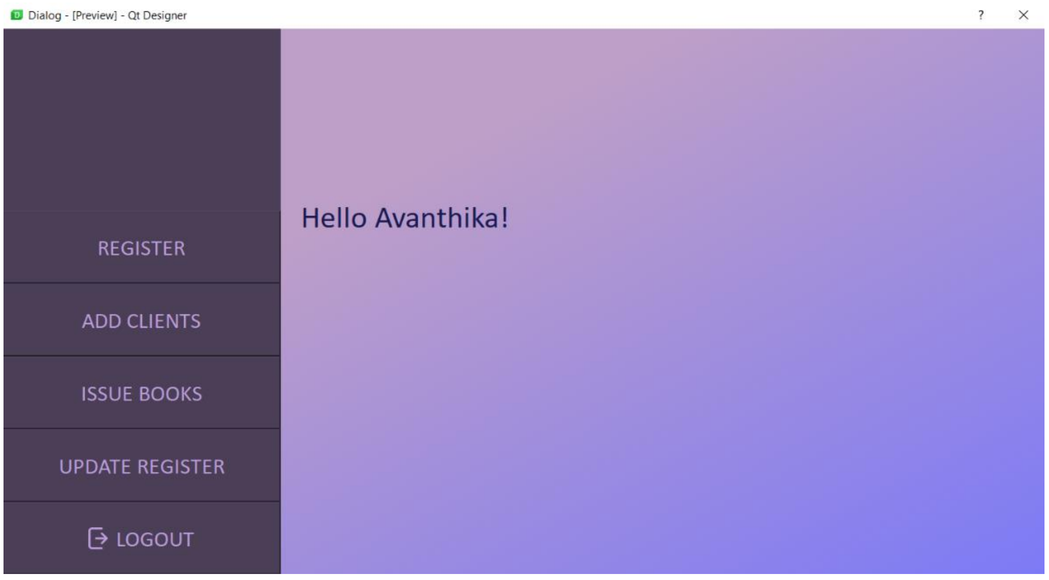
WELCOME

Sign in

Biancaa

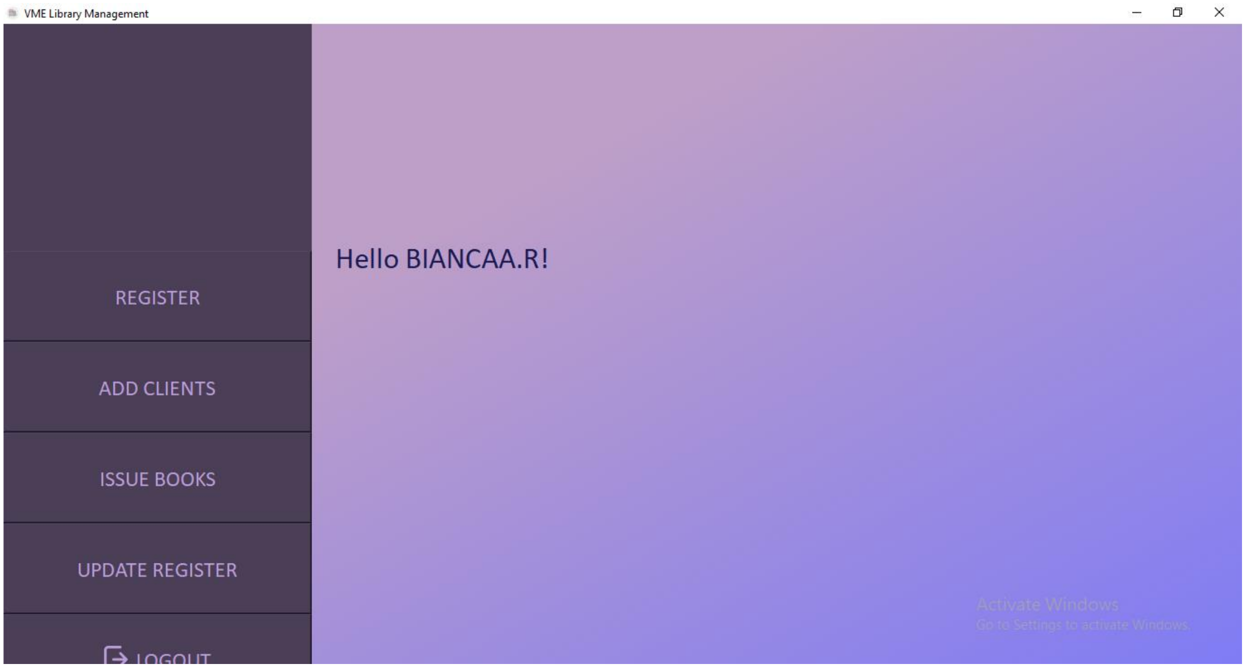
Password

Please fill in all fields

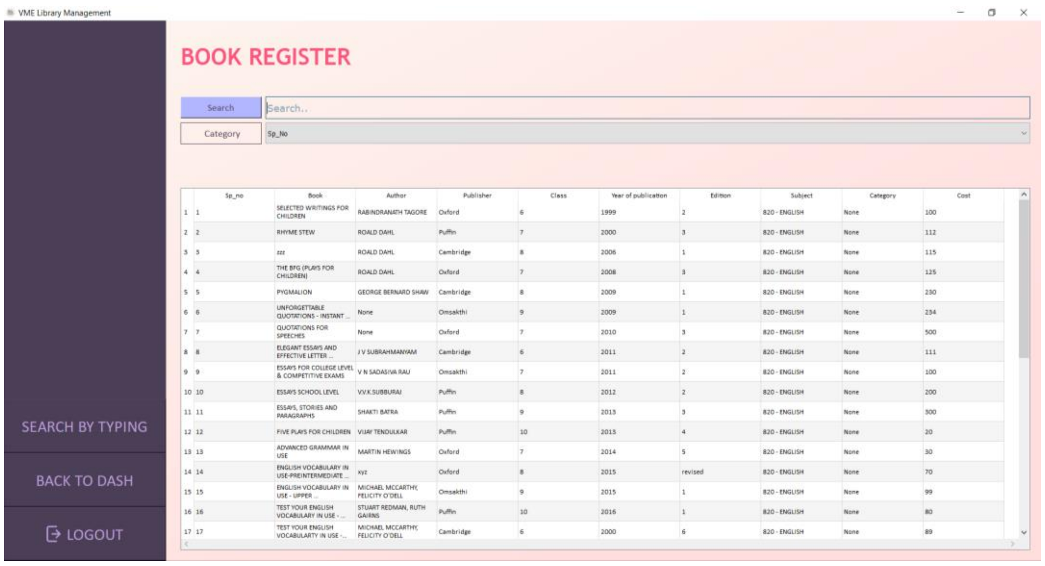

Dialog - [Preview] - Qt Designer

Hello Avanthika!

The Dash Board (It customises itself for each user)



Hello BIANCAA.R!



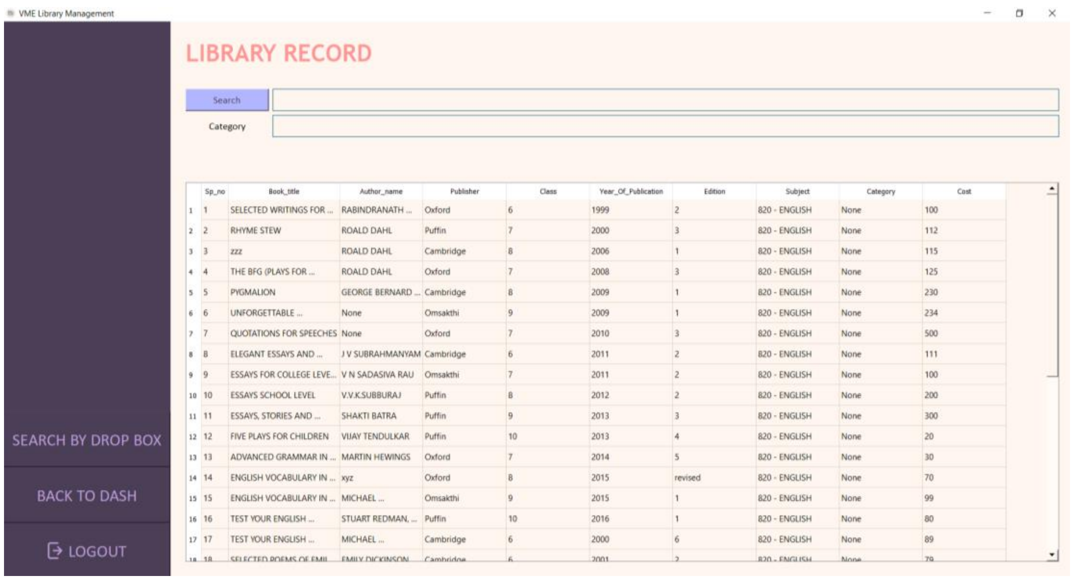
BOOK REGISTER

Search..

Category: Sp_No

	Sp_no	Book	Author	Publisher	Class	Year of publication	Edition	Subject	Category	Cost
1	1	SELECTED WRITINGS FOR CHILDREN	RABINDRANATH TAGORE	Oxford	6	1999	2	820 - ENGLISH	None	100
2	2	RHYME STEW	ROALD DAHL	Puffin	7	2000	3	820 - ENGLISH	None	112
3	3	zzz	ROALD DAHL	Cambridge	8	2006	1	820 - ENGLISH	None	115
4	4	THE BFG (PLAYS FOR CHILDREN)	ROALD DAHL	Oxford	7	2008	3	820 - ENGLISH	None	125
5	5	PYGMALION	GEORGE BERNARD SHAW	Cambridge	8	2009	1	820 - ENGLISH	None	230
6	6	UNFORGETTABLE QUOTATIONS - INSTANT ...	None	Omsakthi	9	2009	1	820 - ENGLISH	None	234
7	7	QUOTATIONS FOR SPEECHES	None	Oxford	7	2010	3	820 - ENGLISH	None	500
8	8	ELEGANT ESSAYS AND EFFECTIVE LETTER ...	J V SUBRAHMANYAM	Cambridge	6	2011	2	820 - ENGLISH	None	111
9	9	ESSAYS FOR COLLEGE LEVEL & COMPETITIVE EXAMS	V N SADASIVA RAU	Omsakthi	7	2011	2	820 - ENGLISH	None	100
10	10	ESSAYS SCHOOL LEVEL	V.V.K.SUBBURAJ	Puffin	8	2012	2	820 - ENGLISH	None	200
11	11	ESSAYS, STORIES AND PARAGRAPHS	SHAKTI BATRA	Puffin	9	2013	3	820 - ENGLISH	None	300
12	12	FIVE PLAYS FOR CHILDREN	VIJAY TENDULKAR	Puffin	10	2013	4	820 - ENGLISH	None	20
13	13	ADVANCED GRAMMAR IN USE	MARTIN HEWINGS	Oxford	7	2014	5	820 - ENGLISH	None	30
14	14	ENGLISH VOCABULARY IN USE-PREINTERMEDIATE ...	xyz	Oxford	8	2015	revised	820 - ENGLISH	None	70
15	15	ENGLISH VOCABULARY IN USE - UPPER ...	MICHAEL MCCARTHY, FELICITY O'DELL	Omsakthi	9	2015	1	820 - ENGLISH	None	99
16	16	TEST YOUR ENGLISH VOCABULARY IN USE - ...	STUART REDMAN, RUTH GAIRNS	Puffin	10	2016	1	820 - ENGLISH	None	80
17	17	TEST YOUR ENGLISH VOCABULARTY IN USE -...	MICHAEL MCCARTHY, FELICITY O'DELL	Cambridge	6	2000	6	820 - ENGLISH	None	89

The Register and search

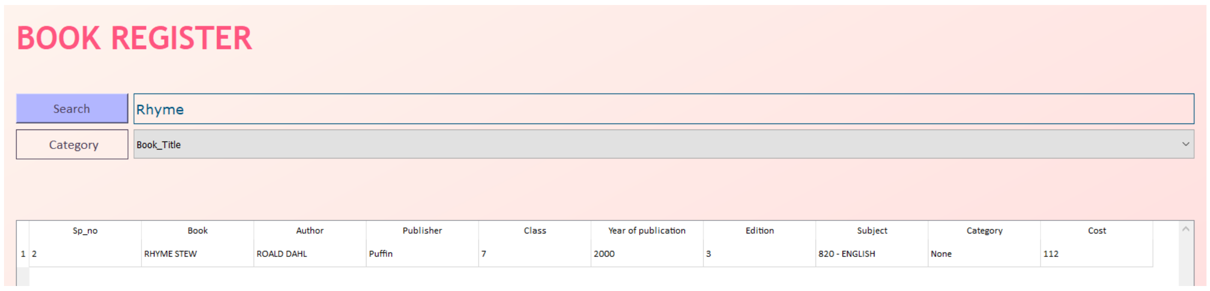


LIBRARY RECORD

	Sp_no	Book_title	Author_name	Publisher	Class	Year_Of_Publication	Edition	Subject	Category	Cost
1	1	SELECTED WRITINGS FOR ...	RABINDRANATH ...	Oxford	6	1999	2	820 - ENGLISH	None	100
2	2	RHYME STEW	ROALD DAHL	Puffin	7	2000	3	820 - ENGLISH	None	112
3	3	zzz	ROALD DAHL	Cambridge	8	2006	1	820 - ENGLISH	None	115
4	4	THE BFG (PLAYS FOR ...	ROALD DAHL	Oxford	7	2008	3	820 - ENGLISH	None	125
5	5	PYGMALION	GEORGE BERNARD ...	Cambridge	8	2009	1	820 - ENGLISH	None	230
6	6	UNFORGETTABLE ...	None	Omsakthi	9	2009	1	820 - ENGLISH	None	234
7	7	QUOTATIONS FOR SPEECHES	None	Oxford	7	2010	3	820 - ENGLISH	None	500
8	8	ELEGANT ESSAYS AND ...	J V SUBRAHMANYAM	Cambridge	6	2011	2	820 - ENGLISH	None	111
9	9	ESSAYS FOR COLLEGE LEVE...	V N SADASIVA RAU	Omsakthi	7	2011	2	820 - ENGLISH	None	100
10	10	ESSAYS SCHOOL LEVEL	V.V.K.SUBBURAJ	Puffin	8	2012	2	820 - ENGLISH	None	200
11	11	ESSAYS, STORIES AND ...	SHAKTI BATRA	Puffin	9	2013	3	820 - ENGLISH	None	300
12	12	FIVE PLAYS FOR CHILDREN	VIJAY TENDULKAR	Puffin	10	2013	4	820 - ENGLISH	None	20
13	13	ADVANCED GRAMMAR IN ...	MARTIN HEWINGS	Oxford	7	2014	5	820 - ENGLISH	None	30
14	14	ENGLISH VOCABULARY IN ...	xyz	Oxford	8	2015	revised	820 - ENGLISH	None	70
15	15	ENGLISH VOCABULARY IN ...	MICHAEL ...	Omsakthi	9	2015	1	820 - ENGLISH	None	99
16	16	TEST YOUR ENGLISH ...	STUART REDMAN, ...	Puffin	10	2016	1	820 - ENGLISH	None	80
17	17	TEST YOUR ENGLISH ...	MICHAEL ...	Cambridge	6	2000	6	820 - ENGLISH	None	89
18	18	SELECTED POEMS OF EMIL	EMILY DICKINSON	Cambridge	6	2001	2	820 - ENGLISH	None	79

Search by typing

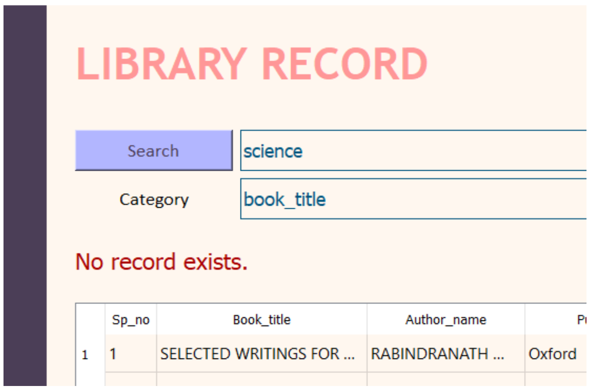
Searching for a book


BOOK REGISTER

Search: Rhyme

Category: Book_Title

	Sp_no	Book	Author	Publisher	Class	Year of publication	Edition	Subject	Category	Cost
1	2	RHYME STEW	ROALD DAHL	Puffin	7	2000	3	820 - ENGLISH	None	112

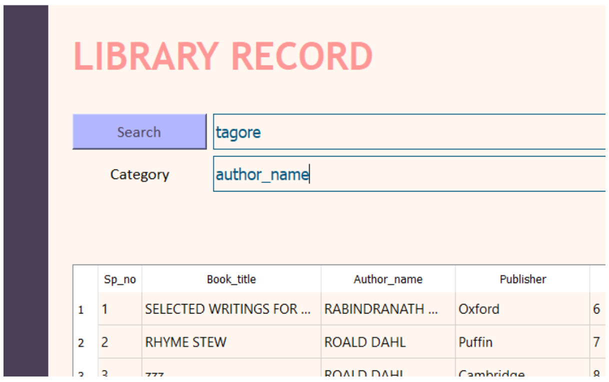

LIBRARY RECORD

Search: science

Category: book_title

No record exists.

	Sp_no	Book_title	Author_name	Pu
1	1	SELECTED WRITINGS FOR ...	RABINDRANATH ...	Oxford

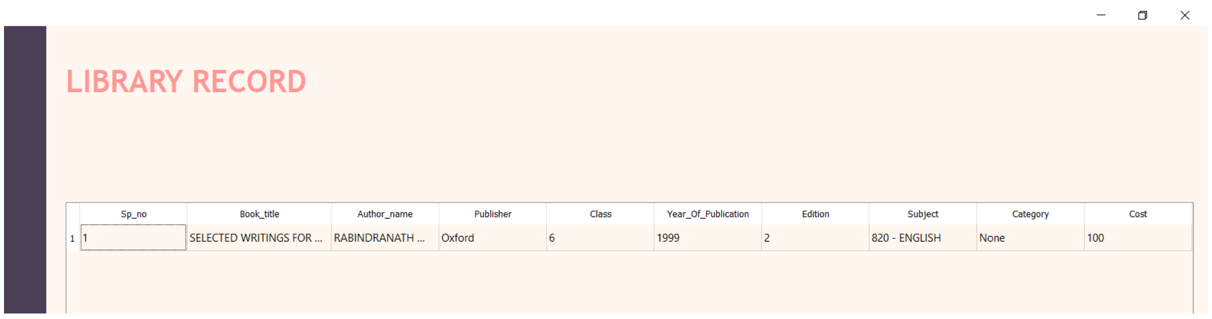

LIBRARY RECORD

Search: tagore

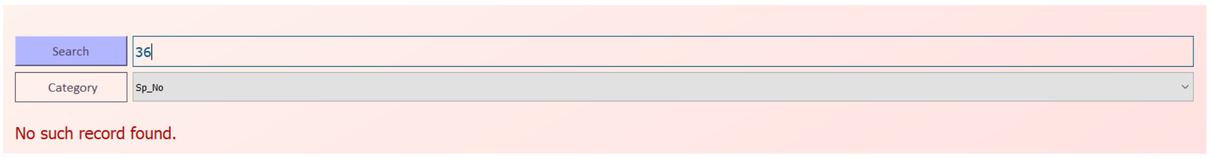
Category: author_name

	Sp_no	Book_title	Author_name	Publisher	
1	1	SELECTED WRITINGS FOR ...	RABINDRANATH ...	Oxford	6
2	2	RHYME STEW	ROALD DAHL	Puffin	7
3	3	zzz	ROALD DAHL	Cambridge	8

Record displayed if exists, if not, error message shown

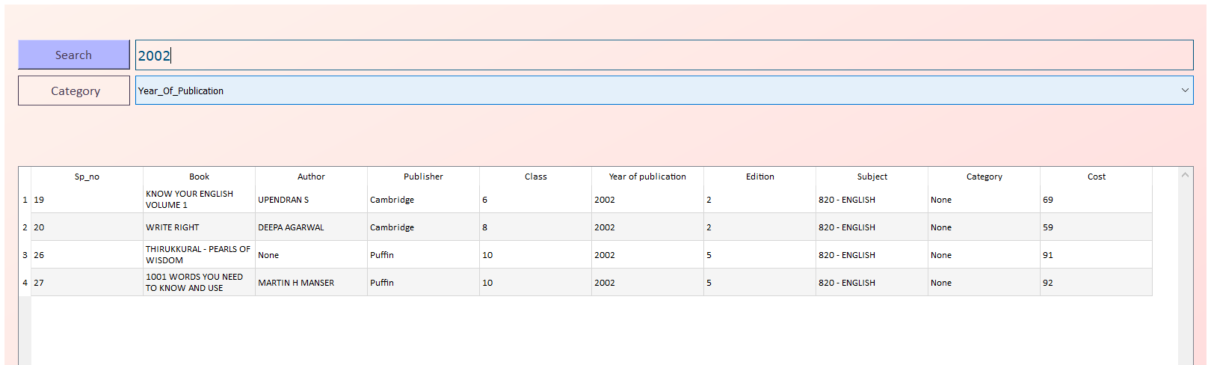

LIBRARY RECORD

	Sp_no	Book_title	Author_name	Publisher	Class	Year_Of_Publication	Edition	Subject	Category	Cost
1	1	SELECTED WRITINGS FOR ...	RABINDRANATH ...	Oxford	6	1999	2	820 - ENGLISH	None	100


Search: 36

Category: Sp_No

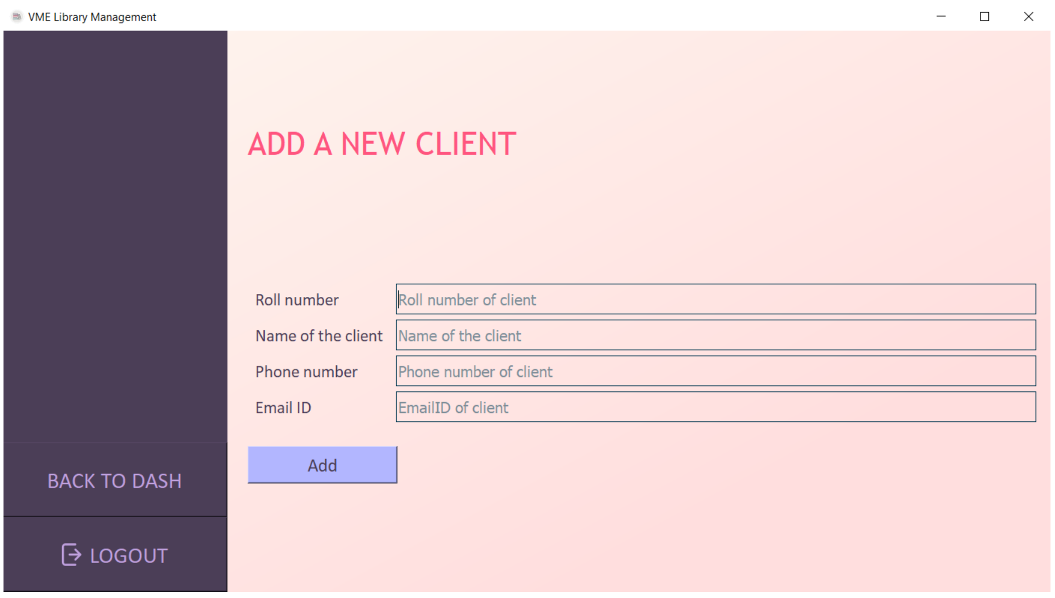
No such record found.

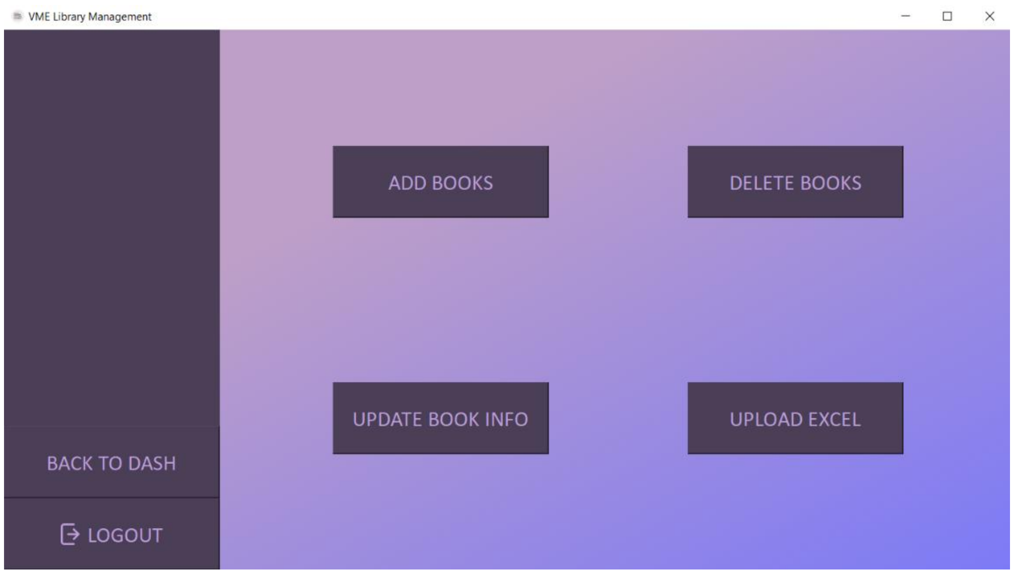

Search: 2002

Category: Year_Of_Publication

	Sp_no	Book	Author	Publisher	Class	Year of publication	Edition	Subject	Category	Cost
1	19	KNOW YOUR ENGLISH VOLUME 1	UPENDRAN S	Cambridge	6	2002	2	820 - ENGLISH	None	69
2	20	WRITE RIGHT	DEEPA AGARWAL	Cambridge	8	2002	2	820 - ENGLISH	None	59
3	26	THIRUKKURAL - PEARLS OF WISDOM	None	Puffin	10	2002	5	820 - ENGLISH	None	91
4	27	1001 WORDS YOU NEED TO KNOW AND USE	MARTIN H MANSER	Puffin	10	2002	5	820 - ENGLISH	None	92

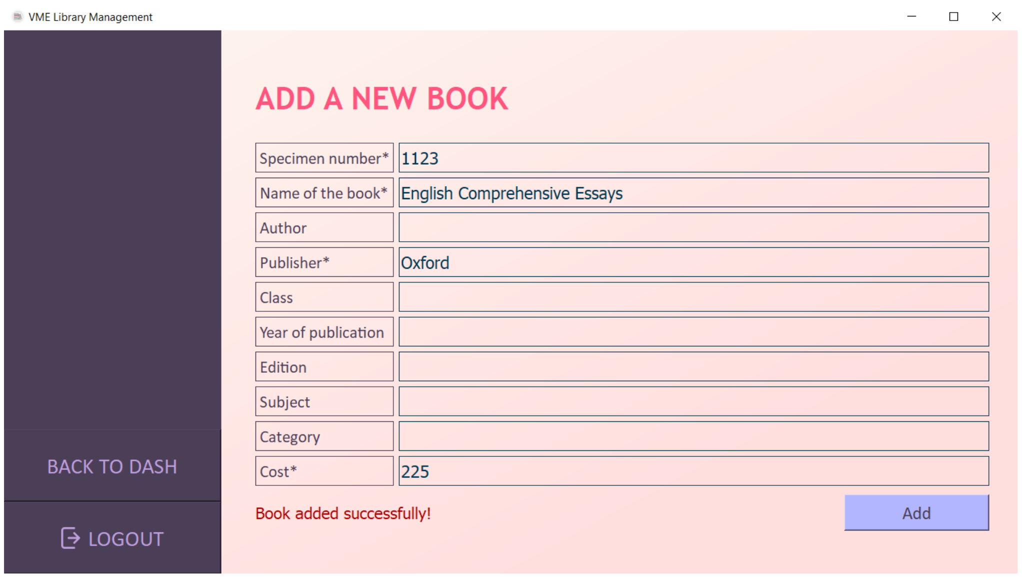
Adding a new user





The update menu

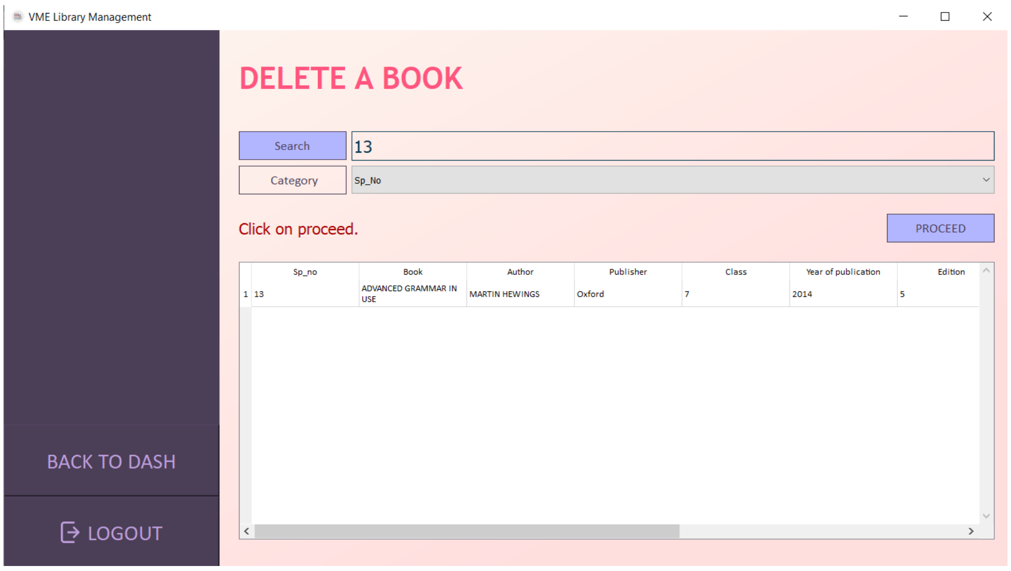
Adding a new book (All fields aren't compulsory)



ADD A NEW BOOK

Specimen number*	1123
Name of the book*	English Comprehensive Essays
Author	
Publisher*	Oxford
Class	
Year of publication	
Edition	
Subject	
Category	
Cost*	225

Book added successfully!



DELETE A BOOK

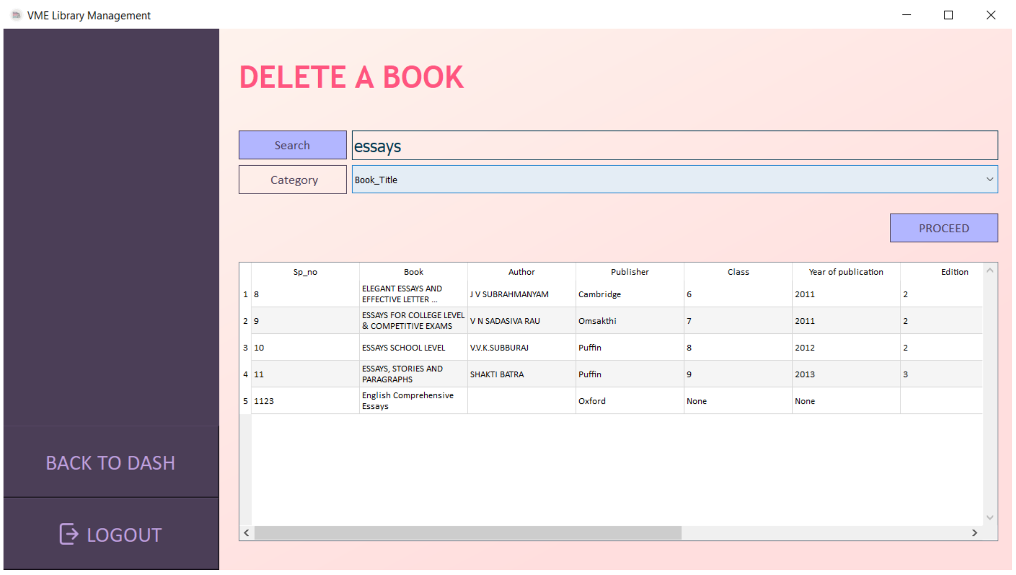
Search: 13

Category: Sp_No

Click on proceed.

	Sp_no	Book	Author	Publisher	Class	Year of publication	Edition
1	13	ADVANCED GRAMMAR IN USE	MARTIN HEWINGS	Oxford	7	2014	5

Deleting a book



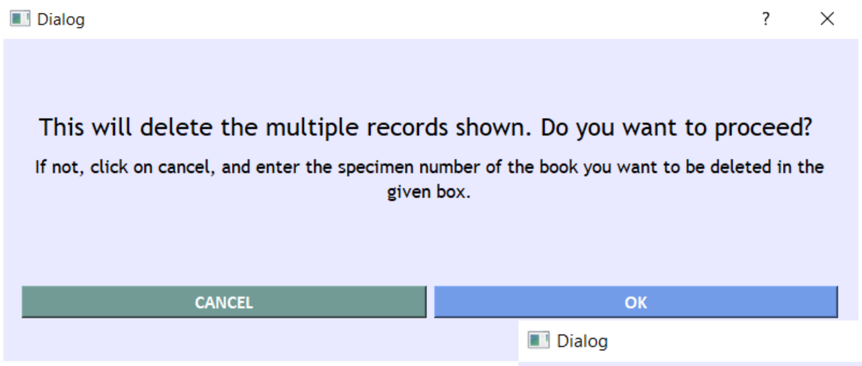
DELETE A BOOK

Search: essays

Category: Book_Title

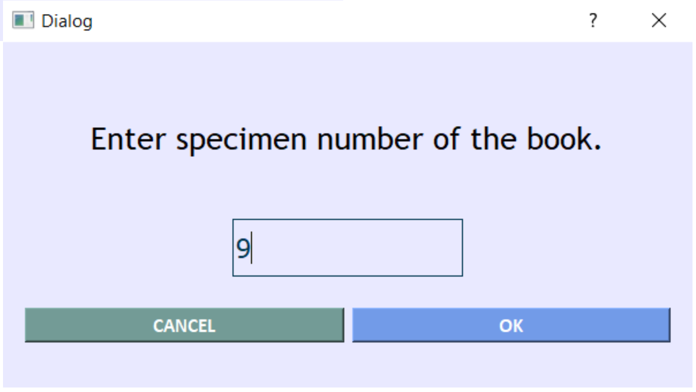
	Sp_no	Book	Author	Publisher	Class	Year of publication	Edition
1	8	ELEGANT ESSAYS AND EFFECTIVE LETTER ...	J V SUBRAHMANYAM	Cambridge	6	2011	2
2	9	ESSAYS FOR COLLEGE LEVEL & COMPETITIVE EXAMS	V N SADASIVA RAU	Omsakthi	7	2011	2
3	10	ESSAYS SCHOOL LEVEL	V.V.K.SUBBURAJ	Puffin	8	2012	2
4	11	ESSAYS, STORIES AND PARAGRAPHS	SHAKTI BATRA	Puffin	9	2013	3
5	1123	English Comprehensive Essays		Oxford	None	None	

When there are multiple records



This will delete the multiple records shown. Do you want to proceed?

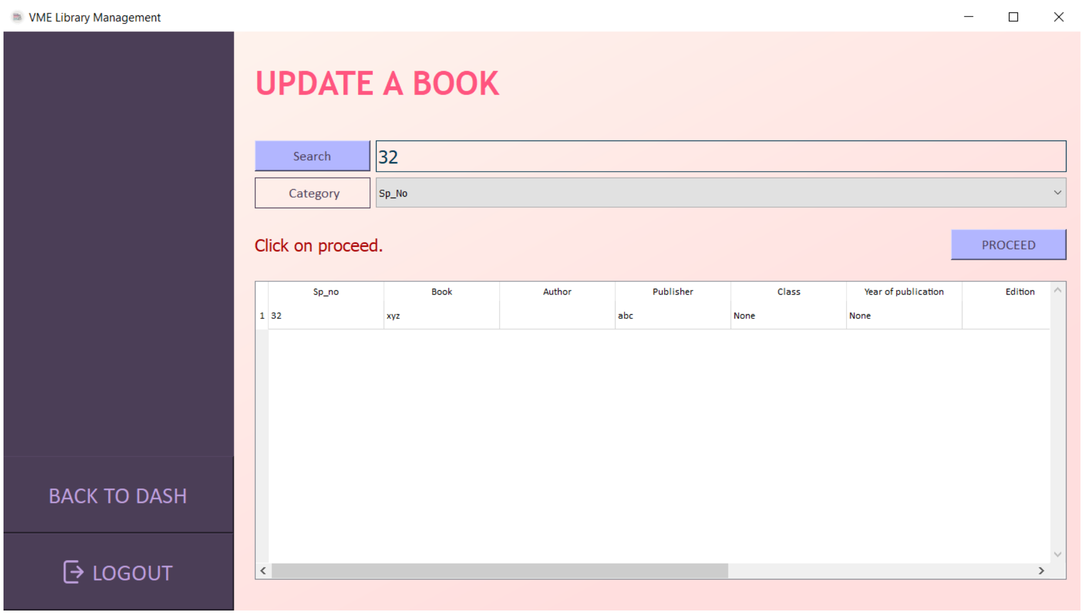
If not, click on cancel, and enter the specimen number of the book you want to be deleted in the given box.



Enter specimen number of the book.

9

A dialog box is shown, whether to delete all the records or just one among them based on its Sp.no



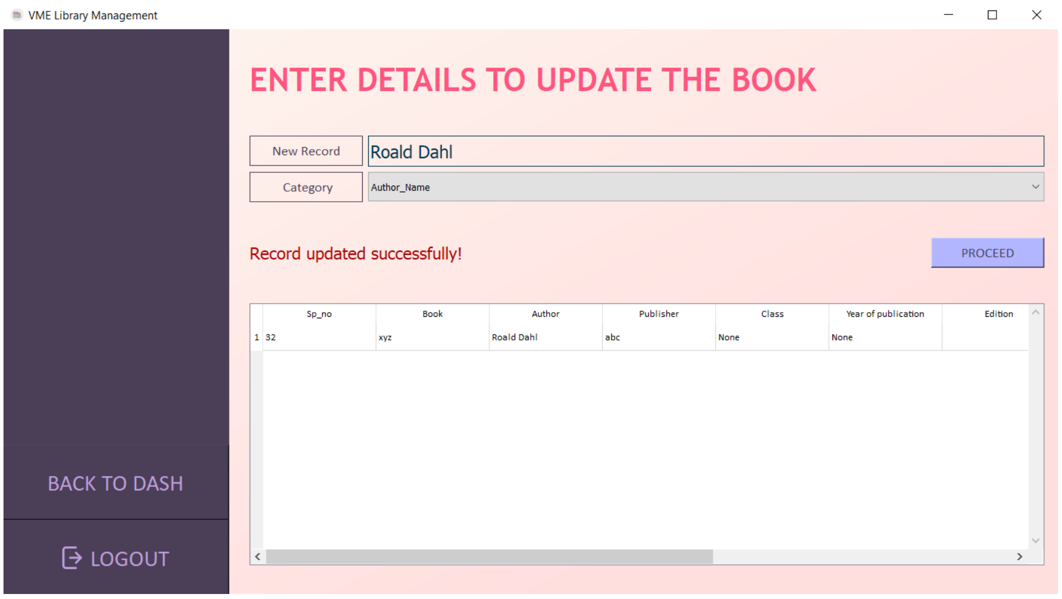
Search: 32

Category: Sp_No

Click on proceed.

	Sp_no	Book	Author	Publisher	Class	Year of publication	Edition
1	32	xyz		abc	None	None	

Updating a book

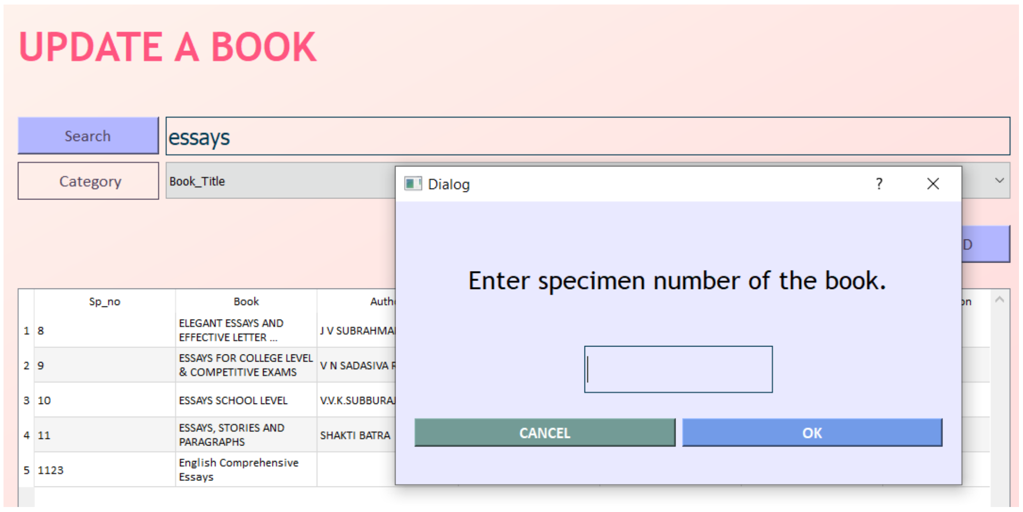


New Record: Roald Dahl

Category: Author_Name

Record updated successfully!

	Sp_no	Book	Author	Publisher	Class	Year of publication	Edition
1	32	xyz	Roald Dahl	abc	None	None	



Search: essays

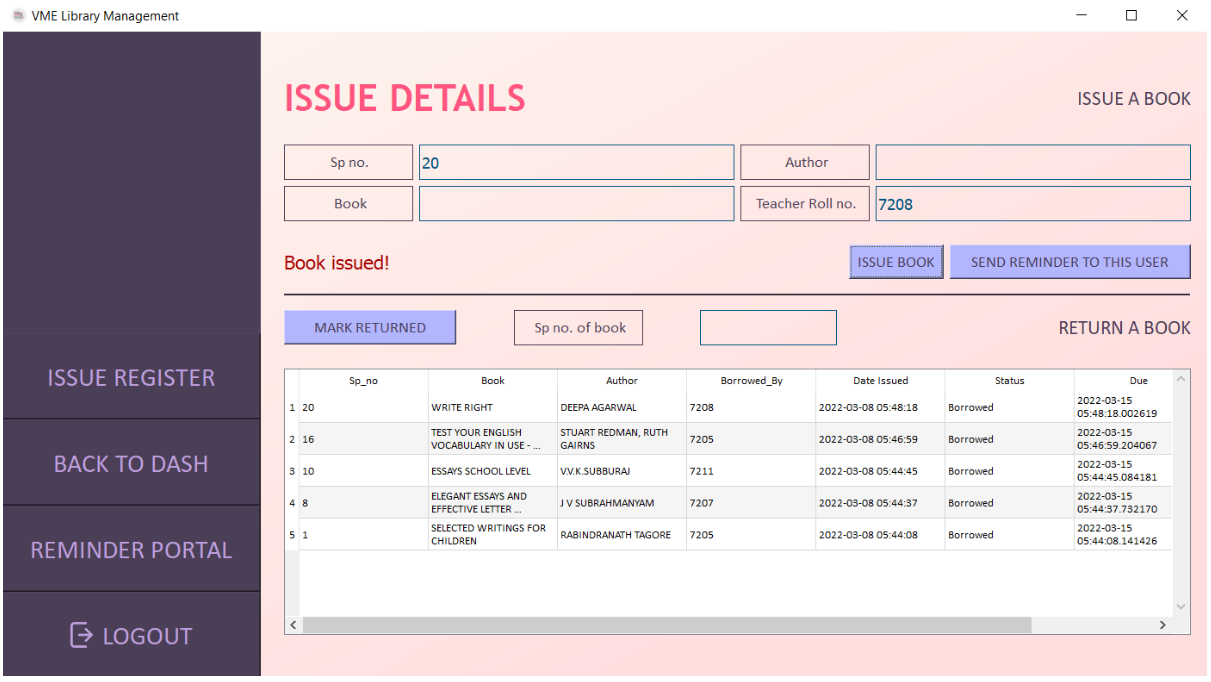
Category: Book_Title

Enter specimen number of the book.

	Sp_no	Book	Auth
1	8	ELEGANT ESSAYS AND EFFECTIVE LETTER ...	J V SUBRAHMA
2	9	ESSAYS FOR COLLEGE LEVEL & COMPETITIVE EXAMS	V N SADASIVA R
3	10	ESSAYS SCHOOL LEVEL	V.V.K.SUBBURA
4	11	ESSAYS, STORIES AND PARAGRAPHS	SHAKTI BATRA
5	1123	English Comprehensive Essays	

Entering specimen no. when there are similar records

Issue Details, through which books can be issued & marked returned


VME Library Management

ISSUE DETAILS

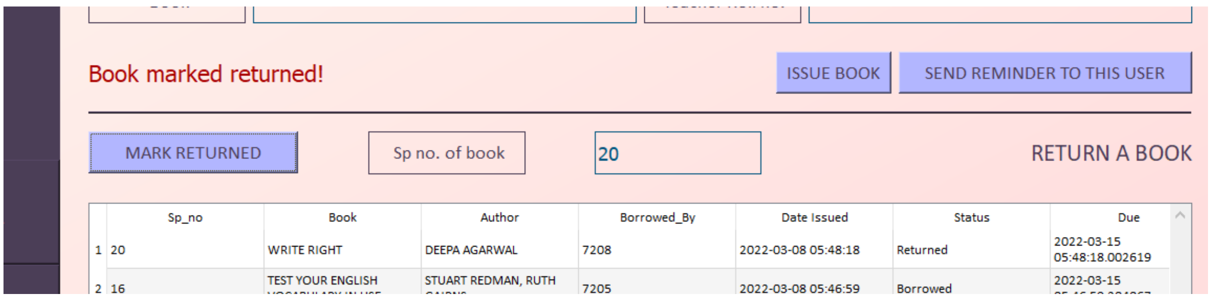
ISSUE A BOOK

Sp no.	20	Author	
Book		Teacher Roll no.	7208

Book issued!

	Sp_no	Book	Author	Borrowed_By	Date Issued	Status	Due
1	20	WRITE RIGHT	DEEPA AGARWAL	7208	2022-03-08 05:48:18	Borrowed	2022-03-15 05:48:18.002619
2	16	TEST YOUR ENGLISH VOCABULARY IN USE - ...	STUART REDMAN, RUTH GAIRNS	7205	2022-03-08 05:46:59	Borrowed	2022-03-15 05:46:59.204067
3	10	ESSAYS SCHOOL LEVEL	V.V.K.SUBBURAJ	7211	2022-03-08 05:44:45	Borrowed	2022-03-15 05:44:45.084181
4	8	ELEGANT ESSAYS AND EFFECTIVE LETTER ...	J V SUBRAHMANYAM	7207	2022-03-08 05:44:37	Borrowed	2022-03-15 05:44:37.732170
5	1	SELECTED WRITINGS FOR CHILDREN	RABINDRANATH TAGORE	7205	2022-03-08 05:44:08	Borrowed	2022-03-15 05:44:08.141426

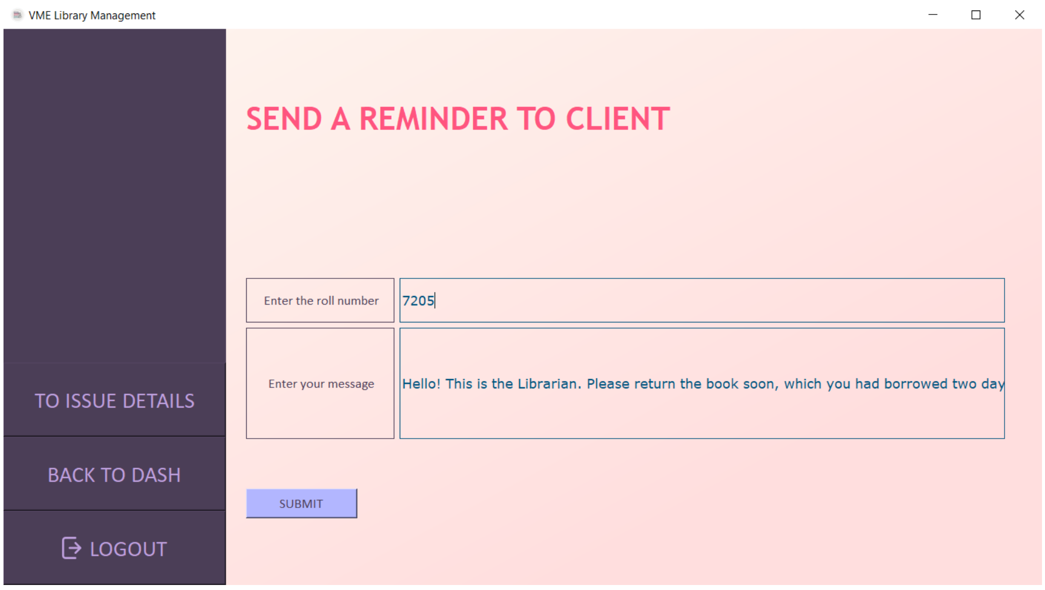
Book marked returned


Book marked returned!

Sp no. of book: 20

	Sp_no	Book	Author	Borrowed_By	Date Issued	Status	Due
1	20	WRITE RIGHT	DEEPA AGARWAL	7208	2022-03-08 05:48:18	Returned	2022-03-15 05:48:18.002619
2	16	TEST YOUR ENGLISH	STUART REDMAN, RUTH	7205	2022-03-08 05:46:59	Borrowed	2022-03-15

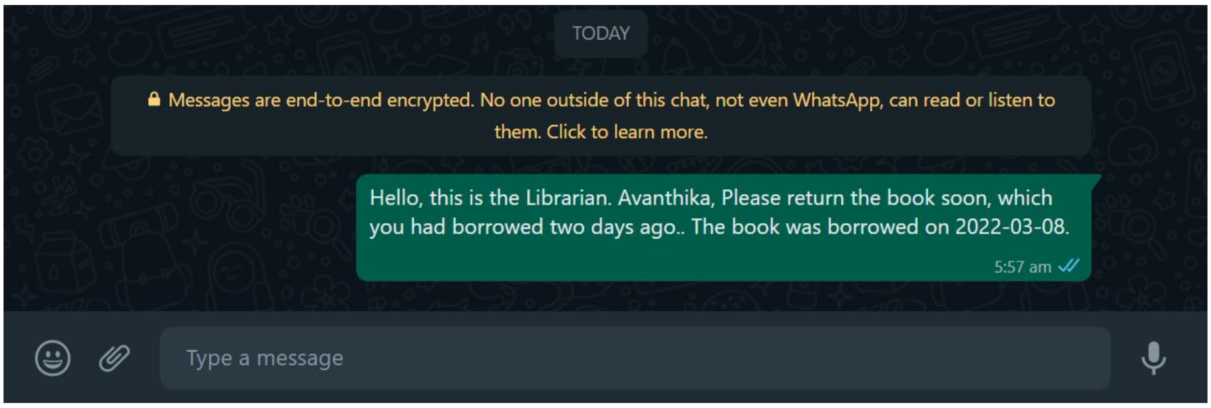
The reminder portal, to send reminder to users


VME Library Management

SEND A REMINDER TO CLIENT

Enter the roll number	7205
Enter your message	Hello! This is the Librarian. Please return the book soon, which you had borrowed two day

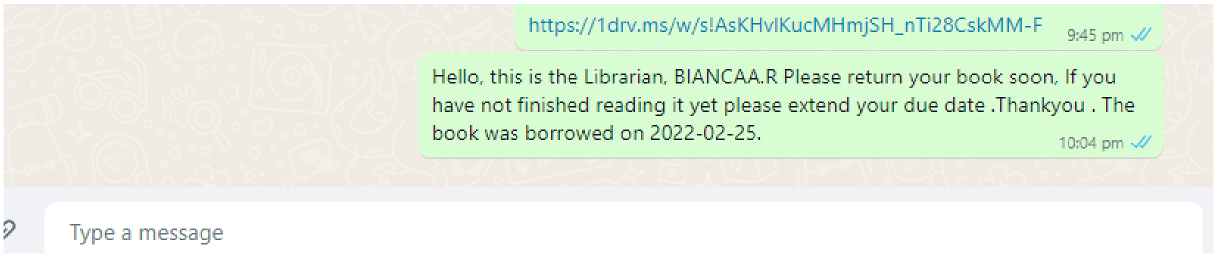
Message sent through whatsapp to the user



🔒 Messages are end-to-end encrypted. No one outside of this chat, not even WhatsApp, can read or listen to them. Click to learn more.

Hello, this is the Librarian. Avanthika, Please return the book soon, which you had borrowed two days ago.. The book was borrowed on 2022-03-08.
5:57 am

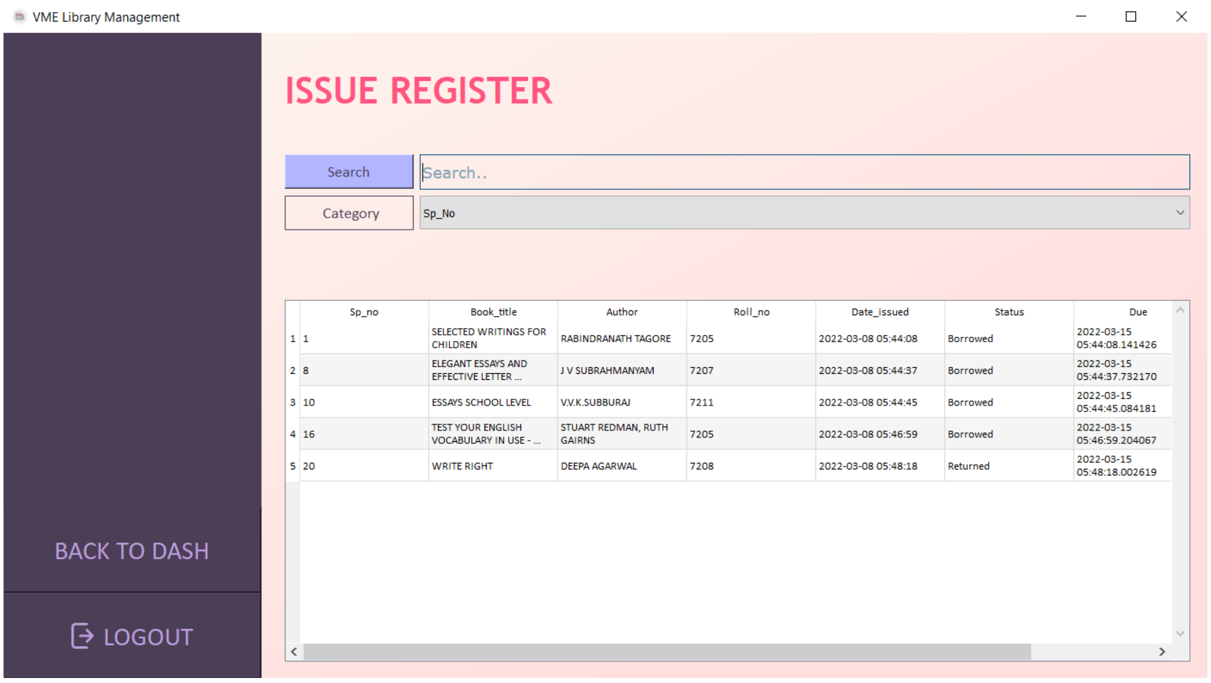
(Offers you the liberty to customize your own reminder messages)



https://1drv.ms/w/s!AsKHvlKucMHmjSH_nTi28CskMM-F
9:45 pm

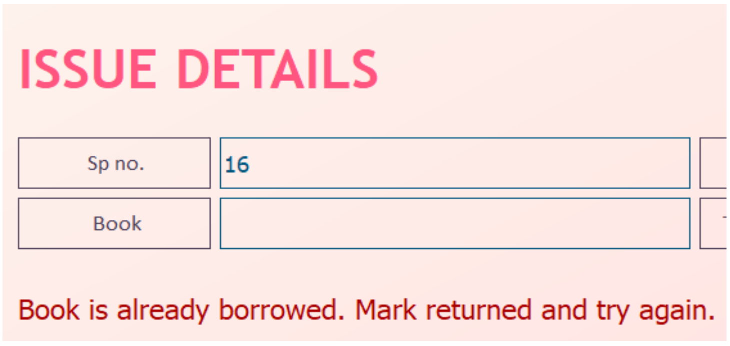
Hello, this is the Librarian, BIANCAA.R Please return your book soon, If you have not finished reading it yet please extend your due date .Thankyou . The book was borrowed on 2022-02-25.
10:04 pm

The issue register, shows all details of issued and borrowed books, can search through them too



	Sp_no	Book_title	Author	Roll_no	Date_issued	Status	Due
1	1	SELECTED WRITINGS FOR CHILDREN	RABINDRANATH TAGORE	7205	2022-03-08 05:44:08	Borrowed	2022-03-15 05:44:08.141426
2	8	ELEGANT ESSAYS AND EFFECTIVE LETTER ...	J V SUBRAHMANYAM	7207	2022-03-08 05:44:37	Borrowed	2022-03-15 05:44:37.732170
3	10	ESSAYS SCHOOL LEVEL	V.V.K.SUBBURAJ	7211	2022-03-08 05:44:45	Borrowed	2022-03-15 05:44:45.084181
4	16	TEST YOUR ENGLISH VOCABULARY IN USE - ...	STUART REDMAN, RUTH GAIRNS	7205	2022-03-08 05:46:59	Borrowed	2022-03-15 05:46:59.204067
5	20	WRITE RIGHT	DEEPA AGARWAL	7208	2022-03-08 05:48:18	Returned	2022-03-15 05:48:18.002619

When a book is already borrowed

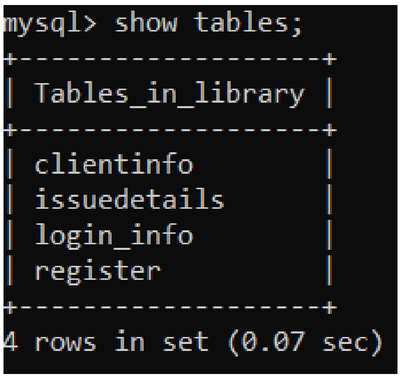


ISSUE DETAILS

Sp no.	16
Book	

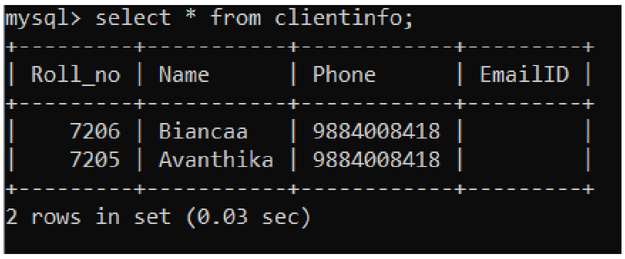
Book is already borrowed. Mark returned and try again.

All tables in the db (background working)



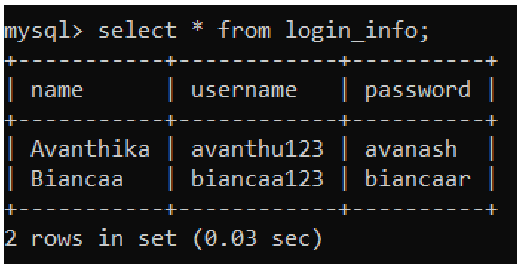
```
mysql> show tables;
+-------------------+
| Tables_in_library |
+-------------------+
| clientinfo        |
| issuedetails      |
| login_info        |
| register          |
+-------------------+
4 rows in set (0.07 sec)
```

Client info table



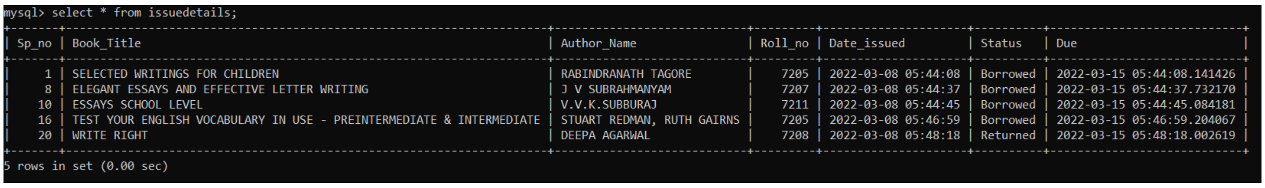
```
mysql> select * from clientinfo;
+---------+-----------+------------+---------+
| Roll_no | Name      | Phone      | EmailID |
+---------+-----------+------------+---------+
|    7206 | Biancaa   | 9884008418 |         |
|    7205 | Avanthika | 9884008418 |         |
+---------+-----------+------------+---------+
2 rows in set (0.03 sec)
```

Login Info Table



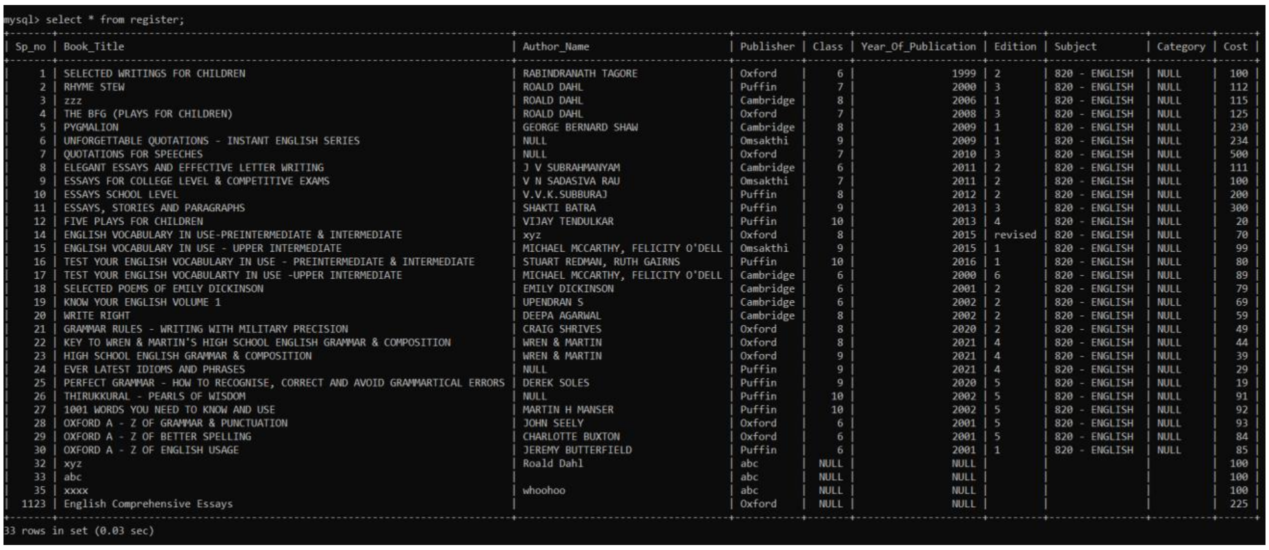
```
mysql> select * from login_info;
+-----------+-------------+----------+
| name      | username    | password |
+-----------+-------------+----------+
| Avanthika | avanthu123  | avanash  |
| Biancaa   | biancaa123  | biancaar |
+-----------+-------------+----------+
2 rows in set (0.03 sec)
```

Issue table



```
mysql> select * from issuedetails;
+-------+----------------------------------------------------------------------+----------------------------+---------+---------------------+----------+----------------------------+
| Sp_no | Book_Title                                                           | Author_Name                | Roll_no | Date_issued         | Status   | Due                        |
+-------+----------------------------------------------------------------------+----------------------------+---------+---------------------+----------+----------------------------+
|     1 | SELECTED WRITINGS FOR CHILDREN                                       | RABINDRANATH TAGORE        |    7205 | 2022-03-08 05:44:08 | Borrowed | 2022-03-15 05:44:08.141426 |
|     8 | ELEGANT ESSAYS AND EFFECTIVE LETTER WRITING                          | J V SUBRAHMANYAM           |    7207 | 2022-03-08 05:44:37 | Borrowed | 2022-03-15 05:44:37.732170 |
|    10 | ESSAYS SCHOOL LEVEL                                                  | V.V.K.SUBBURAJ             |    7211 | 2022-03-08 05:44:45 | Borrowed | 2022-03-15 05:44:45.084181 |
|    16 | TEST YOUR ENGLISH VOCABULARY IN USE - PREINTERMEDIATE & INTERMEDIATE | STUART REDMAN, RUTH GAIRNS |    7205 | 2022-03-08 05:46:59 | Borrowed | 2022-03-15 05:46:59.204067 |
|    20 | WRITE RIGHT                                                          | DEEPA AGARWAL              |    7208 | 2022-03-08 05:48:18 | Returned | 2022-03-15 05:48:18.002619 |
+-------+----------------------------------------------------------------------+----------------------------+---------+---------------------+----------+----------------------------+
5 rows in set (0.00 sec)
```

Register table



```
mysql> select * from register;
+-------+---------------------------------------------------------------------------+-----------------------------------+-----------+-------+---------------------+---------+------------------+----------+------+
| Sp_no | Book_Title                                                                | Author_Name                       | Publisher | Class | Year_Of_Publication | Edition | Subject          | Category | Cost |
+-------+---------------------------------------------------------------------------+-----------------------------------+-----------+-------+---------------------+---------+------------------+----------+------+
|     1 | SELECTED WRITINGS FOR CHILDREN                                            | RABINDRANATH TAGORE               | Oxford    |     6 |                1999 | 2       | 820 - ENGLISH    | NULL     |  100 |
|     2 | RHYME STEW                                                                | ROALD DAHL                        | Puffin    |     7 |                2000 | 3       | 820 - ENGLISH    | NULL     |  112 |
|     3 | zzz                                                                       | ROALD DAHL                        | Cambridge |     8 |                2006 | 1       | 820 - ENGLISH    | NULL     |  115 |
|     4 | THE BFG (PLAYS FOR CHILDREN)                                              | ROALD DAHL                        | Oxford    |     7 |                2008 | 3       | 820 - ENGLISH    | NULL     |  125 |
|     5 | PYGMALION                                                                 | GEORGE BERNARD SHAW               | Cambridge |     8 |                2009 | 1       | 820 - ENGLISH    | NULL     |  230 |
|     6 | UNFORGETTABLE QUOTATIONS - INSTANT ENGLISH SERIES                         | NULL                              | Omsakthi  |     9 |                2009 | 1       | 820 - ENGLISH    | NULL     |  234 |
|     7 | QUOTATIONS FOR SPEECHES                                                   | NULL                              | Oxford    |     7 |                2010 | 3       | 820 - ENGLISH    | NULL     |  500 |
|     8 | ELEGANT ESSAYS AND EFFECTIVE LETTER WRITING                               | J V SUBRAHMANYAM                  | Cambridge |     6 |                2011 | 2       | 820 - ENGLISH    | NULL     |  111 |
|     9 | ESSAYS FOR COLLEGE LEVEL & COMPETITIVE EXAMS                              | V N SADASIVA RAU                  | Omsakthi  |     7 |                2011 | 2       | 820 - ENGLISH    | NULL     |  100 |
|    10 | ESSAYS SCHOOL LEVEL                                                       | V.V.K.SUBBURAJ                    | Puffin    |     8 |                2012 | 2       | 820 - ENGLISH    | NULL     |  200 |
|    11 | ESSAYS, STORIES AND PARAGRAPHS                                            | SHAKTI BATRA                      | Puffin    |     9 |                2013 | 3       | 820 - ENGLISH    | NULL     |  300 |
|    12 | FIVE PLAYS FOR CHILDREN                                                   | VIJAY TENDULKAR                   | Puffin    |    10 |                2013 | 4       | 820 - ENGLISH    | NULL     |   20 |
|    14 | ENGLISH VOCABULARY IN USE-PREINTERMEDIATE & INTERMEDIATE                  | xyz                               | Oxford    |     8 |                2015 | revised | 820 - ENGLISH    | NULL     |   70 |
|    15 | ENGLISH VOCABULARY IN USE - UPPER INTERMEDIATE                            | MICHAEL MCCARTHY, FELICITY O'DELL | Omsakthi  |     9 |                2015 | 1       | 820 - ENGLISH    | NULL     |   99 |
|    16 | TEST YOUR ENGLISH VOCABULARY IN USE - PREINTERMEDIATE & INTERMEDIATE      | STUART REDMAN, RUTH GAIRNS        | Puffin    |    10 |                2016 | 1       | 820 - ENGLISH    | NULL     |   80 |
|    17 | TEST YOUR ENGLISH VOCABULARTY IN USE -UPPER INTERMEDIATE                  | MICHAEL MCCARTHY, FELICITY O'DELL | Cambridge |     6 |                2000 | 6       | 820 - ENGLISH    | NULL     |   89 |
|    18 | SELECTED POEMS OF EMILY DICKINSON                                         | EMILY DICKINSON                   | Cambridge |     6 |                2001 | 2       | 820 - ENGLISH    | NULL     |   79 |
|    19 | KNOW YOUR ENGLISH VOLUME 1                                                | UPENDRAN S                        | Cambridge |     6 |                2002 | 2       | 820 - ENGLISH    | NULL     |   69 |
|    20 | WRITE RIGHT                                                               | DEEPA AGARWAL                     | Cambridge |     8 |                2002 | 2       | 820 - ENGLISH    | NULL     |   59 |
|    21 | GRAMMAR RULES - WRITING WITH MILITARY PRECISION                           | CRAIG SHRIVES                     | Oxford    |     8 |                2020 | 2       | 820 - ENGLISH    | NULL     |   49 |
|    22 | KEY TO WREN & MARTIN'S HIGH SCHOOL ENGLISH GRAMMAR & COMPOSITION          | WREN & MARTIN                     | Oxford    |     8 |                2021 | 4       | 820 - ENGLISH    | NULL     |   44 |
|    23 | HIGH SCHOOL ENGLISH GRAMMAR & COMPOSITION                                 | WREN & MARTIN                     | Oxford    |     9 |                2021 | 4       | 820 - ENGLISH    | NULL     |   39 |
|    24 | EVER LATEST IDIOMS AND PHRASES                                            | NULL                              | Puffin    |     9 |                2021 | 4       | 820 - ENGLISH    | NULL     |   29 |
|    25 | PERFECT GRAMMAR - HOW TO RECOGNISE, CORRECT AND AVOID GRAMMARTICAL ERRORS | DEREK SOLES                       | Puffin    |     9 |                2020 | 5       | 820 - ENGLISH    | NULL     |   19 |
|    26 | THIRUKKURAL - PEARLS OF WISDOM                                            | NULL                              | Puffin    |    10 |                2002 | 5       | 820 - ENGLISH    | NULL     |   91 |
|    27 | 1001 WORDS YOU NEED TO KNOW AND USE                                       | MARTIN H MANSER                   | Puffin    |    10 |                2002 | 5       | 820 - ENGLISH    | NULL     |   92 |
|    28 | OXFORD A - Z OF GRAMMAR & PUNCTUATION                                     | JOHN SEELY                        | Oxford    |     6 |                2001 | 5       | 820 - ENGLISH    | NULL     |   93 |
|    29 | OXFORD A - Z OF BETTER SPELLING                                           | CHARLOTTE BUXTON                  | Oxford    |     6 |                2001 | 5       | 820 - ENGLISH    | NULL     |   84 |
|    30 | OXFORD A - Z OF ENGLISH USAGE                                             | JEREMY BUTTERFIELD                | Puffin    |     6 |                2001 | 1       | 820 - ENGLISH    | NULL     |   85 |
|    32 | xyz                                                                       | Roald Dahl                        | abc       |  NULL |                NULL |         |                  |          |  100 |
|    33 | abc                                                                       |                                   | abc       |  NULL |                NULL |         |                  |          |  100 |
|    35 | xxxx                                                                      | whoohoo                           | abc       |  NULL |                NULL |         |                  |          |  100 |
|  1123 | English Comprehensive Essays                                              |                                   | Oxford    |  NULL |                NULL |         |                  |          |  225 |
+-------+---------------------------------------------------------------------------+-----------------------------------+-----------+-------+---------------------+---------+------------------+----------+------+
33 rows in set (0.03 sec)
```

CONCLUSION

In brief, the above program has been designed to make the whole process of keeping track of records in the library easier. With everything digitalised, this has become much more efficient and much more easier. One can perform all operations like issuing books, marking them returned, sending reminders to users on when to return the book, adding, deleting and updating new books in the register and so on.

FUTURE ENHANCEMENTS

The future enhancements for our project is very high. The following developments can be made in our program :

- Sending messages needs network connectivity, so could be changed to SMS type instead of through whatsapp.
- Automatic sending of messages when the books are due(it’s to be sent manualy currently).
- Uploading excel sheets for adding new books instead of adding them one by one.
- Packaging the code as an application to use it across platforms.

BIBLIOGRAPHY

- Python basics, using python libraries and SQL connectivity - Sumita Arora Class 11, Sumita Arora Class 12, NCERT classes 11 and 12 (Com
puter
Science)
- Installing MySQL connector, setup
https://dev.mysql.com/doc/connector-python/en/connector-python-install
ation.html
- Installing PyQt5 tools and designer, setup and basics
https://realpython.com/qt-designer-python/
- PyQt5 tutorial, used as reference
https://www.tutorialspoint.com/pyqt5/index.htm
- Converting .ui file to .py
https://www.tutorialexample.com/convert-qt-desiger-ui-file-to-python-scri
pt-file-py-pyqt-tutorial/
- How to use the pywhatkit module
https://pypi.org/project/pywhatkit/
- https://www.geeksforgeeks.org/introduction-to-pywhatkit-module/
- Basics of pyqt5
https://www.techwithtim.net/tutorials/pyqt5-tutorial/basic-gui-application/