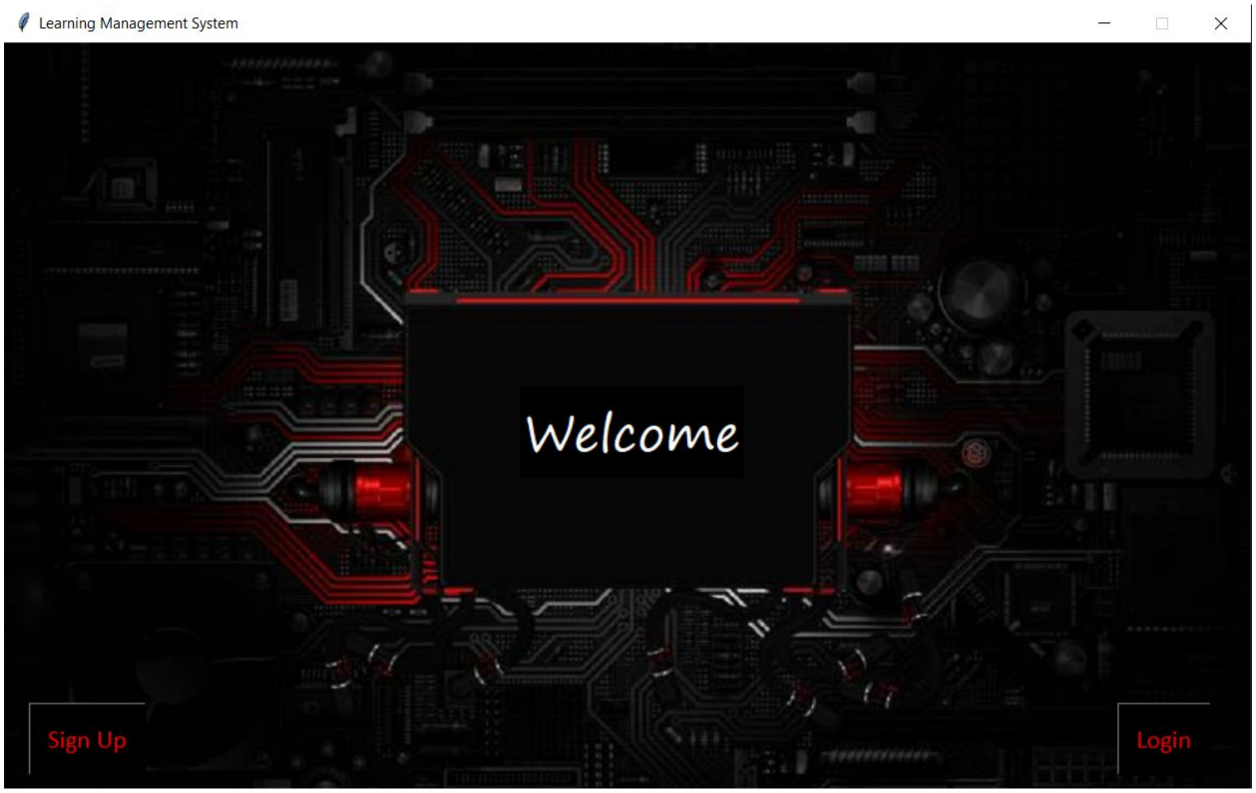


Learning Management System

Computer Science Project 2020-21

Malhaar Arora
Roll no. 18
Class XII - J
Delhi Public School, Sector-45, Gurgaon

Index

Overview

For my School Project 2020-21, I've made a learning management system with Python using Tkinter GUI and MySQL database which enables teachers and students to create and manage accounts, create and submit assignments, attend live classes, access notes and video lessons and keep track of attendance through graphs.

Goal

Due to the Covid-19 pandemic and lockdown, online learning has become a necessity. Learning Management System is designed to make your life easier by managing all the tools you need in one place.

Certificate

This is to certify that Malhaar Arora of class XII - J has prepared this project. This report is a culmination of his efforts and endeavours and has been accepted as the final project report for the subject Computer of class XII.

Ms. Chanchal Chandna

Acknowledgement

I would like to express my sincere gratitude to my Computer Science teacher, Ms. Chanchal Chandna, for her able guidance and support, without which this project would not have come to be.

Python Code

```
import mysql.connector
from tkinter import *
from tkinter import filedialog
from PIL import ImageTk, Image
from functools import partial
import os
import matplotlib.figure
import matplotlib.patches
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import datetime

mydb = mysql.connector.connect(
host = "localhost",
user = "root",
password = "********",
database = "school_portal")
mycursor = mydb.cursor()

class Teacher():

    def __init__(self, name, regno, dob, contactno, subject, pin, email):
        self.name = name
        self.regno = int(regno)
        self.dob = dob
        self.contactno = int(contactno)
        self.subject = subject
        self.pin = int(pin)
        self.emailID = email

class Student():

    def __init__(self, name, regno, dob, contactno, grade, section, pin,
email):
        self.name = name
        self.regno = int(regno)
        self.dob = dob
        self.contactno = int(contactno)
        self.grade = int(grade)
        self.section = section
        self.pin = int(pin)
        self.emailID = email

def start():

    welcomeLabel = Label(root, text = "Welcome", fg = "white", bg = "black",
font=("Segoe Print", 29)).grid(column = 1, row = 0, padx = 280, pady = (275,
180))

    createUserButton = Button(root, text = "Sign Up", padx = 10, pady = 10,
command = create_user, borderwidth = 1, bg = "black", fg = "red", font =
('calibri', 15)).grid(column = 0, row = 1, padx = (20, 20))

    loginButton = Button(root, text = "Login", padx = 10, pady = 10, command =
login, borderwidth = 1, bg = "black", fg = "red", font = ('calibri',
15)).grid(row = 1, column = 2, padx = (20, 20))


def create_user():
    root.wm_state('iconic')
    window1 = Toplevel()
    window1.title("Create User")
    window1.geometry("1000x600")
    window1.configure(bg = "#f5f5dc")
    window1.resizable(False, False)

    createUserLabel = Label(window1, text = "Create User", font=("Comic Sans
MS", 24)).grid(row = 0, column = 1, padx = 30, pady = 40)

    RadioValue = StringVar()
    teacherRadio = Button(window1, text = "Teacher", font = ('Segoe Print',
12), command = create_teacher).grid(row = 1, column = 0, padx = (20, 20))
    studentRadio = Button(window1, text = "Student", font = ('Segoe Print',
12), command = create_student).grid(row = 1, column = 2, padx = (20, 20))

def create_teacher():

    window_teacher = Toplevel()
    window_teacher.title("Create User")
    window_teacher.geometry("1000x600")
    window_teacher.configure(bg = "#f5f5dc")
    window_teacher.resizable(False, False)

    createUserLabel = Label(window_teacher, text = "Create User", font=("Comic
Sans MS", 24)).grid(row = 0, column = 1, padx = 30, pady = 40)

    nameLabel = Label(window_teacher, text = "Name").grid(row = 2, column = 0,
pady = 10)
    nameVar = StringVar()
    nameInput = Entry(window_teacher, textvariable = nameVar).grid(row = 2,
column = 1, padx = (20, 20), pady = 10)
```

```
    regNoLabel = Label(window_teacher, text = "Registration number").grid(row
= 3, column = 0, pady = 10)
    regNoVar = IntVar()
    regNoInput = Entry(window_teacher, textvariable = regNoVar).grid(row = 3,
column = 1, padx = (20, 20), pady = 10)

    dobLabel = Label(window_teacher, text = "Date of birth
(YYYY-MM-DD)").grid(row = 4, column = 0, pady = 10)
    dobVar = StringVar()
    dobInput = Entry(window_teacher, textvariable = dobVar).grid(row = 4,
column = 1, padx = (20, 20), pady = 10)

    contactLabel = Label(window_teacher, text = "Contact number").grid(row =
5, column = 0, pady = 10)
    contactVar = IntVar()
    contactInput = Entry(window_teacher, textvariable = contactVar).grid(row =
5, column = 1, padx = (20, 20), pady = 10)

    subjectLabel = Label(window_teacher, text = "Subject").grid(row = 6,
column = 0)
    subjectVar = StringVar()
    physicsRadio = Radiobutton(window_teacher, text = "Physics", value =
"Physics", font = ('Segoe Print', 12), indicator = 0, background = "light
blue", variable = subjectVar).grid(row = 7, column = 0, padx = (20, 20), pady
= 50)
    mathsRadio = Radiobutton(window_teacher, text = "Maths", value = "Maths",
font = ('Segoe Print', 12), indicator = 0, background = "light blue", variable
= subjectVar).grid(row = 7, column = 1, padx = (20, 20), pady = 50)
    chemistryRadio = Radiobutton(window_teacher, text = "Chemistry", value =
"Chemistry", font = ('Segoe Print', 12), indicator = 0, background = "light
blue", variable = subjectVar).grid(row = 7, column = 2, padx = (30, 100), pady
= 50)
    csRadio = Radiobutton(window_teacher, text = "Computer Science", value =
"CS", font = ('Segoe Print', 12), indicator = 0, background = "light blue",
variable = subjectVar).grid(row = 7, column = 3, padx = (20, 20), pady = 50)

    emailLabel = Label(window_teacher, text = "Enter email address").grid(row
= 8, column = 0, pady = 10)
    emailVar = StringVar()
    emailInput = Entry(window_teacher, textvariable = emailVar).grid(row = 8,
column = 1, pady = 10)

    pinLabel = Label(window_teacher, text = "Enter 4-digit pin").grid(row = 9,
column = 0, padx = (20, 20), pady = 10)
    pinVar = IntVar()
    pinEntry = Entry(window_teacher, textvariable = pinVar).grid(row = 9,
column = 1, padx = (20, 20), pady = 10)

    def submit_createUser():

        try:
            teacherChar = Teacher(nameVar.get().title(), regNoVar.get(),
dobVar.get(), contactVar.get(), subjectVar.get(), pinVar.get(),
emailVar.get())
            sql = "INSERT INTO teachers (Name, Pin, Subject, RegistrationNo,
DOB, ContactNo, EmailID) VALUES (%s, %s, %s, %s, %s, %s, %s)"
            val = (teacherChar.name, teacherChar.pin, teacherChar.subject,
teacherChar.regno, teacherChar.dob, teacherChar.contactno,
teacherChar.emailID)
            mycursor.execute(sql, val)
            mydb.commit()
            accountCreatedLabel = Label(window_teacher, text = "Account
created successfully!").grid(row = 11, column = 0)
```

```
        except Exception as e:
            accoundNotCreatedLabel = Label(window_teacher, text = "Oops! We could not create the account. Please check all the details and try again.").grid(row = 11, column = 0)
            print(e)

    submitButton = Button(window_teacher, text = "Submit", command = submit_createUser).grid(row = 10, column = 0, pady = 30)

def create_student():

    window_student = Toplevel()
    window_student.title("Create User")
    window_student.geometry("1000x600")
    window_student.configure(bg = "#f5f5dc")
    window_student.resizable(False, False)

    createUserLabel = Label(window_student, text = "Create User", font=("Comic Sans MS", 24)).grid(row = 0, column = 1, padx = 30, pady = 40)

    nameLabel = Label(window_student, text = "Name").grid(row = 2, column = 0, pady = 10)
    nameVar = StringVar()
    nameInput = Entry(window_student, textvariable = nameVar).grid(row = 2, column = 1, padx = (20, 20), pady = 10)

    regNoLabel = Label(window_student, text = "Registration number").grid(row = 3, column = 0, pady = 10)
    regNoVar = IntVar()
    regNoInput = Entry(window_student, textvariable = regNoVar).grid(row = 3, column = 1, padx = (20, 20), pady = 10)
    dobLabel = Label(window_student, text = "Date of birth
(YYYY-MM-DD)").grid(row = 4, column = 0, pady = 10)
    dobVar = StringVar()
    dobInput = Entry(window_student, textvariable = dobVar).grid(row = 4,
column = 1, padx = (20, 20), pady = 10)

    contactLabel = Label(window_student, text = "Contact number").grid(row =
5, column = 0, pady = 10)
    contactVar = IntVar()
    contactInput = Entry(window_student, textvariable = contactVar).grid(row =
5, column = 1, padx = (20, 20), pady = 10)

    gradeLabel = Label(window_student, text = "Grade").grid(row = 6, column =
0, pady = 10)
    gradeVar = IntVar()
    gradeInput = Entry(window_student, textvariable = gradeVar).grid(row = 6,
column = 1, pady = 10)

    sectionLabel = Label(window_student, text = "Section").grid(row = 7,
column = 0, pady = 10)
    sectionVar = StringVar()
    sectionInput = Entry(window_student, textvariable = sectionVar).grid(row =
7, column = 1, pady = 10)

    emailLabel = Label(window_student, text = "Enter email address").grid(row
= 8, column = 0, pady = 10)
    emailVar = StringVar()
    emailInput = Entry(window_student, textvariable = emailVar).grid(row = 8,
column = 1, pady = 10)

    pinLabel = Label(window_student, text = "Enter 4-digit pin").grid(row = 9,
column = 0, padx = (20, 20), pady = 10)
    pinVar = IntVar()
    pinEntry = Entry(window_student, textvariable = pinVar).grid(row = 9,
column = 1, padx = (20, 20), pady = 10)

    def submit_createUser():

        try:
            studentChar = Student(nameVar.get().title(), regNoVar.get(),
dobVar.get(), contactVar.get(), gradeVar.get(), sectionVar.get(),
pinVar.get(), emailVar.get())
            sql = "INSERT INTO students (Name, Pin, RegistrationNo, DOB,
ContactNo, Grade, Section, EmailID) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"
            val = (studentChar.name, studentChar.pin, studentChar.regno,
studentChar.dob, studentChar.contactno, studentChar.grade,
studentChar.section, studentChar.emailID)
            mycursor.execute(sql, val)
            mydb.commit()

            sql = f"INSERT INTO attendance VALUES ('{studentChar.name}', 0)"
            mycursor.execute(sql)
            mydb.commit()

            sql = f"ALTER TABLE assignments ADD {studentChar.name}
VARCHAR(100)"
            mycursor.execute(sql)
            mydb.commit()

            accountCreatedLabel = Label(window_student, text = "Account
created successfully!").grid(row = 11, column = 0)

        except Exception as e:
            accoundNotCreatedLabel = Label(window_student, text = "Oops! We could not create the account. Please check all the details and try again.").grid(row = 11, column = 0)
            print(e)

    submitButton = Button(window_student, text = "Submit", command = submit_createUser).grid(row = 10, column = 0, pady = 30)


def login():

    root.wm_state('iconic')
    window2 = Toplevel()
    window2.title("Login")
    window2.geometry("1000x600")
    window2.configure(bg = "#f5f5dc")
    window2.resizable(False, False)

    createUserLabel = Label(window2, text = "Login", font=("Comic Sans MS", 24)).grid(row = 0, column = 1, padx = 30, pady = 20)

    RadioValue = StringVar()
    teacherRadio = Radiobutton(window2, text = "Teacher", value = "Teacher", font = ('Segoe Print', 12), indicator = 0, background = "light blue", variable = RadioValue).grid(row = 1, column = 0, padx = (20, 20), pady = 50)
    studentRadio = Radiobutton(window2, text = "Student", value = "Student", font = ('Segoe Print', 12), indicator = 0, background = "light blue", variable = RadioValue).grid(row = 1, column = 2, padx = (20, 20), pady = 50)

    regNoLabel = Label(window2, text = "Enter your registration number", font = ('Segoe Print', 12)).grid(row = 2, column = 0)
    global regNoVar
    regNoVar = IntVar()
    nameInput = Entry(window2, textvariable = regNoVar).grid(row = 2, column =
1, padx = (20, 20), pady = 60)

    pinLabel = Label(window2, text = "Enter 4-digit pin", font = ('Segoe
Print', 12)).grid(row = 3, column = 0, padx = (20, 20), pady = 60)
    pinVar = IntVar()
    pinEntry = Entry(window2, textvariable = pinVar, show = "*").grid(row = 3,
column = 1, padx = (20, 20), pady = 60)

    def submit_login():

        try:
            mycursor.execute(f"select Pin from {RadioValue.get()}s where
RegistrationNo = '{regNoVar.get()}'")
            actual_pin = mycursor.fetchall()

            if pinVar.get() == actual_pin[0][0]:
                if RadioValue.get() == "Teacher":
                    teacher()
                    window2.destroy()
                elif RadioValue.get() == "Student":
                    student()
                    window2.destroy()
            else:
                wrongPinLabel = Label(window2, text = "Wrong pin
entered").grid(row = 5, column = 0)

        except Exception as e:
            errorLabel = Label(window2, text = "Oops! We could not login.
Please check all the details and try again.").grid(row = 5, column = 0)
            print(e)
    loginButton = Button(window2, text = "Login", command = 
submit_login).grid(row = 4, column = 0)



def teacher():

    window3 = Toplevel()
    window3.title("Home Page")
    window3.geometry("1100x800")
    background_label = Label(window3, image = homescreen)
    background_label.place(x=0, y=0, relwidth=1, relheight=1)
    window3.resizable(False, False)

    sql = f"SELECT Name FROM teachers WHERE RegistrationNo = {regNoVar.get()}"
    mycursor.execute(sql)
    teacher_name = mycursor.fetchall()
    welcomeLabel = Label(window3, text = f"Welcome, {teacher_name[0][0]}!", 
font=("Helvetica", 24, "bold"), fg = "blue").grid(row = 0, column = 0, padx = 
30, pady = (40, 0))

    def live_class():

        liveclassframe = Frame(window3)
        liveclassframe.grid(row = 1, column = 0)

        linkLabel = Label(liveclassframe, text = "Enter live class 
link:").grid(row = 0, column = 0)
        linkValue = StringVar()
        linkInput = Entry(liveclassframe, textvariable = linkValue).grid(row = 
0, column = 1, padx = 5)
```

```
        def submit_link():

            sql = "DELETE FROM LiveClassLink"
            mycursor.execute(sql)
            mydb.commit()

            sql = f"INSERT INTO LiveClassLink VALUES ('{linkValue.get()}')"
            mycursor.execute(sql)
            mydb.commit()
            linkUploadSuccessfulLabel = Label(liveclassframe, text = "Link 
uploaded successfully").grid(row = 1, column = 0)

        def quit():

            liveclassframe.grid_forget()
            liveclassframe.destroy()

        linkSubmitButton = Button(liveclassframe, text = "Submit", command = 
submit_link).grid(row = 2, column = 0, pady = 30)
        quitButton = Button(liveclassframe, text = "Quit", font = ('calibri', 
10, 'bold', 'underline'), foreground = 'red', command = quit).grid(row = 2, 
column = 1, pady = 30, padx = 10)


    def check_attendance():

        window_attendance = Toplevel()
        window_attendance.title("Attendance")
        window_attendance.geometry("1000x600")
        window_attendance.configure(bg = "#f5f5dc")
        window_attendance.resizable(False, False)
        sql = "SELECT * FROM attendance ORDER BY Student ASC"
        mycursor.execute(sql)
        result = mycursor.fetchall()

        for i in range(len(result)):

            studLabel = Label(window_attendance, text = result[i][0]).grid(row
= 0, column = i)

            fig = matplotlib.figure.Figure(figsize=(2,2))
            ax = fig.add_subplot(111)

            school_start = datetime.datetime(2020, 3, 1)
            now = datetime.datetime.now()
            time_difference = now - school_start
            days_passed = time_difference.days
            present = result[i][1]
            absent = days_passed - result[i][1]

            ax.pie([present, absent])
            ax.legend([f"Present: {present}", f"Absent: {absent}"])

            circle=matplotlib.patches.Circle( (0,0), 0.7, color='white')
            ax.add_artist(circle)

            canvas = FigureCanvasTkAgg(fig, master=window_attendance)
            canvas.get_tk_widget().grid(row = 1, column = i)
            canvas.draw()

    def assignment():

        assignmentframe = Frame(window3)
        assignmentframe.grid(row = 2, column = 0)

        chapterLabel = Label(assignmentframe, text = "Chapter name").grid(row
= 0, column = 0, pady = 20)
        chapterValue = StringVar()
        chapterInput = Entry(assignmentframe, textvariable =
chapterValue).grid(row = 0, column = 1, pady = 20, padx = 5)

        topicLabel = Label(assignmentframe, text = "Topic").grid(row = 1,
column = 0, pady = 20)
        topicValue = StringVar()
        topicInput = Entry(assignmentframe, textvariable =
topicValue).grid(row = 1, column = 1, pady = 20, padx = 5)

        lastDateLabel = Label(assignmentframe, text = "Last date of submission
(YYYY-MM-DD)").grid(row = 2, column = 0, pady = 20)
        lastDateValue = StringVar()
        lastDateInput = Entry(assignmentframe, textvariable =
lastDateValue).grid(row = 2, column = 1, padx = 5, pady = 20)

        def choose_file():

            filename = filedialog.askopenfilename(initialdir = "*", title =
"Select a file", filetypes = (("pdf files", "*.pdf"), ("text files",
"*.txt")))

            sql = f"SELECT Subject FROM teachers WHERE RegistrationNo =
{regNoVar.get()}"
            mycursor.execute(sql)
            result = mycursor.fetchall()

            sql = f"INSERT INTO assignments (Subject, Chapter, Topic, Link,
LastDate) VALUES ('{result[0][0]}', '{chapterValue.get()}',
'{topicValue.get()}', '{filename}', '{lastDateValue.get()}')"
            mycursor.execute(sql)
            mydb.commit()

            assignmentUploadSuccessful = Label(assignmentframe, text =
"Assignment uploaded successfully.").grid(row = 4, column = 0)

        def quit():

            assignmentframe.grid_forget()
            assignmentframe.destroy()

        chooseFileButton = Button(assignmentframe, text = "Choose file",
command = choose_file).grid(row = 3, column = 0, pady = 20)
        quitButton = Button(assignmentframe, text = "Quit", font = ('calibri',
10, 'bold', 'underline'), foreground = 'red', command = quit).grid(row = 3,
column = 1, pady = 20, padx = 10)


    def class_notes():

        notesframe = Frame(window3)
        notesframe.grid(row = 2, column = 1)

        chapterLabel = Label(notesframe, text = "Chapter name").grid(row = 0,
column = 0, pady = 20)
        chapterValue = StringVar()
        chapterInput = Entry(notesframe, textvariable = chapterValue).grid(row
= 0, column = 1, pady = 20, padx = 5)

        classNoLabel = Label(notesframe, text = "Class number").grid(row = 1,
column = 0, pady = 20)
        classNoValue = IntVar()
        classNoInput = Entry(notesframe, textvariable = classNoValue).grid(row
= 1, column = 1, pady = 20, padx = 5)

        def choose_file():

            filename = filedialog.askopenfilename(initialdir = "*", title =
"Select a file", filetypes = (("pdf files", "*.pdf"), ("text files",
"*.txt")))

            sql = f"SELECT Subject FROM teachers WHERE RegistrationNo =
{regNoVar.get()}"
            mycursor.execute(sql)
            result = mycursor.fetchall()

            sql = f"INSERT INTO {result[0][0]} (Chapter, ClassNumber, Notes)
VALUES ('{chapterValue.get()}', {classNoValue.get()}, '{filename}')"
            mycursor.execute(sql)
            mydb.commit()

            classNotesUploadSuccessful = Label(notesframe, text = "Notes
uploaded successfully.").grid(row = 3, column = 0)

        def quit():

            notesframe.grid_forget()
            notesframe.destroy()

        chooseFileButton = Button(notesframe, text = "Choose file", command =
choose_file).grid(row = 2, column = 0, pady = 20)
        quitButton = Button(notesframe, text = "Quit", font = ('calibri', 10,
'bold', 'underline'), foreground = 'red', command = quit).grid(row = 2, column
= 1, pady = 20, padx = 10)

    def class_recording():

        recordframe = Frame(window3)
        recordframe.grid(row = 2, column = 2)

        chapterLabel = Label(recordframe, text = "Chapter name").grid(row = 0,
column = 0, pady = 20)
        chapterValue = StringVar()
        chapterInput = Entry(recordframe, textvariable =
chapterValue).grid(row = 0, column = 1, pady = 20, padx = 5)

        classNoLabel = Label(recordframe, text = "Class number").grid(row = 1,
column = 0, pady = 20)
        classNoValue = IntVar()
        classNoInput = Entry(recordframe, textvariable =
classNoValue).grid(row = 1, column = 1, pady = 20, padx = 5)

        def choose_file():

            filename = filedialog.askopenfilename(initialdir = "*", title =
"Select a file", filetypes = [("Mp4 files", "*.mp4")])

            sql = f"SELECT Subject FROM teachers WHERE RegistrationNo =
{regNoVar.get()}"
            mycursor.execute(sql)
            result = mycursor.fetchall()

            sql = f"INSERT INTO {result[0][0]} (Chapter, ClassNumber,
Recording) VALUES ('{chapterValue.get()}', {classNoValue.get()},
'{filename}')"
            mycursor.execute(sql)
            mydb.commit()

            classRecordingUploadSuccessful = Label(recordframe, text = 
"Recording uploaded successfully.").grid(row = 3, column = 0)

        def quit():

            recordframe.grid_forget()
            recordframe.destroy()

        chooseFileButton = Button(recordframe, text = "Choose file", command = 
choose_file).grid(row = 2, column = 0, pady = 20)
        quitButton = Button(recordframe, text = "Quit", font = ('calibri', 10, 
'bold', 'underline'), foreground = 'red', command = quit).grid(row = 2, column 
= 1, pady = 20, padx = 10)


    LiveClassLinkButton = Button(window3, image = liveClassPhoto, command = 
live_class).grid(row = 1, column = 0, padx = 30, pady = 40)
    checkAttendanceButton = Button(window3, image = attendancePhoto, command = 
check_attendance).grid(row = 1, column = 1, padx = 30, pady = 40)
    newAssignmentButton = Button(window3, image = assignmentPhoto, command = 
assignment).grid(row = 2, column = 0, padx = 30, pady = 40)
    classNotesButton = Button(window3, image = classNotesPhoto, command = 
class_notes).grid(row = 2, column = 1, padx = 30, pady = 40)
    classRecordingButton = Button(window3, image = classRecordingPhoto, 
command = class_recording).grid(row = 2, column = 2, padx = 30, pady = 40)


def student():
    window4 = Toplevel()
    window4.title("Home Page")
    window4.geometry("1100x800")
    background_label = Label(window4, image = homescreen)
    background_label.place(x=0, y=0, relwidth=1, relheight=1)
    window4.resizable(False, False)

    sql = f"SELECT Name FROM students WHERE RegistrationNo = {regNoVar.get()}"
    mycursor.execute(sql)
    student_name = mycursor.fetchall()
    welcomeLabel = Label(window4, text = f"Welcome, {student_name[0][0]}!",
font=("Helvetica", 24, "bold"), fg = "blue").grid(row = 0, column = 0, padx =
30, pady = 40)

    def live_class():

        sql = f"SELECT Name FROM students WHERE RegistrationNo =
{regNoVar.get()}"
        mycursor.execute(sql)
        studName = mycursor.fetchall()

        sql = f"UPDATE attendance SET NoOfDaysPresent = NoOfDaysPresent + 1
where Student = '{studName[0][0]}'"
        mycursor.execute(sql)
        mydb.commit()
        print("Attendance marked successfully.")

        print("Joining live class...")
        from selenium import webdriver
        chromedriver = r"C:\Users\Malhaar\Downloads\chromedriver.exe"
        driver = webdriver.Chrome(chromedriver)
        sql = "SELECT Link FROM LiveClassLink"
        mycursor.execute(sql)
        result = mycursor.fetchall()[0][0]
        driver.get(result)
        driver.maximize_window()
        while True:
            pass

    def check_attendance():

        window_attendance = Toplevel()
        window_attendance.title("Attendance")
        window_attendance.geometry("600x600")
        window_attendance.configure(bg = "#f5f5dc")
        window_attendance.resizable(False, False)

        sql = f"SELECT Name FROM students WHERE RegistrationNo =
{regNoVar.get()}"
        mycursor.execute(sql)
        student_name = mycursor.fetchall()

        sql = f"SELECT * FROM attendance WHERE Student =
'{student_name[0][0]}'"
        mycursor.execute(sql)
        result = mycursor.fetchall()

        studLabel = Label(window_attendance, text = result[0][0]).grid(row =
0, column = 0)

        fig = matplotlib.figure.Figure(figsize=(5, 5))
        ax = fig.add_subplot(111)
        school_start = datetime.datetime(2020, 3, 1)
        now = datetime.datetime.now()
        time_difference = now - school_start
        days_passed = time_difference.days
        present = result[0][1]
        absent = days_passed - result[0][1]

        ax.pie([present, absent])
        ax.legend([f"Present: {present}", f"Absent: {absent}"])

        circle=matplotlib.patches.Circle( (0,0), 0.7, color='white')
        ax.add_artist(circle)

        canvas = FigureCanvasTkAgg(fig, master=window_attendance)
        canvas.get_tk_widget().grid(row = 1, column = 0)
        canvas.draw()

    def assignment():

        assignmentWindow = Toplevel()
        assignmentWindow.title("Assignments")
        assignmentWindow.geometry("1200x700")
        background_label = Label(assignmentWindow, image = matrixBackground)
        background_label.place(x=0, y=0, relwidth=1, relheight=1)
        assignmentWindow.resizable(False, False)
        window4.wm_state('iconic')

        def open_this(address):
            os.startfile(address)

        def physics_assignments():
            sql = "SELECT * FROM assignments WHERE Subject = 'Physics' order
by Chapter"
            mycursor.execute(sql)
            result = mycursor.fetchall()

            physicsframe = Frame(assignmentWindow)
            physicsframe.grid(row = 0, column = 0)

            chapter = Label(physicsframe, text = "Chapter", font=("Helvetica",
15, "underline")).grid(row = 0, column = 0, padx = 20, pady = 20)
            topic = Label(physicsframe, text = "Topic", font=("Helvetica", 15,
"underline")).grid(row = 0, column = 1, padx = 20, pady = 20)
            deadline = Label(physicsframe, text = "Due in", font=("Helvetica",
15, "underline")).grid(row = 0, column = 2, padx = 20, pady = 20)

            def choose_file():

                filename = filedialog.askopenfilename(initialdir = "*", title
= "Select a file", filetypes = (("pdf files", "*.pdf"), ("text files",
"*.txt")))

                sql = f"SELECT Name FROM students WHERE RegistrationNo =
{regNoVar.get()}"
                mycursor.execute(sql)
                studname = mycursor.fetchall()

                sql = f"UPDATE assignments SET {studname[0][0]} = '{filename}'
WHERE Chapter = '{result[i][1]}' AND Topic = '{result[i][2]}'"
                mycursor.execute(sql)
                mydb.commit()
                assignmentUploadSuccessful = Label(physicsframe, text =
"Assignment uploaded successfully.").grid(row = i+1, column = 5)

            def quit():

                physicsframe.grid_forget()
                physicsframe.destroy()

            for i in range(len(result)):

                    chapterLabel = Label(physicsframe, text =
f"{result[i][1]}").grid(row = i+1, column = 0, padx = 20, pady = 20)
                    topicButton = Button(physicsframe, text =
f"{result[i][2]}", command = partial(open_this, result[i][3])).grid(row = i+1,
column = 1, padx = 20, pady = 20)

                    today = datetime.date.today()
                    duedate = result[i][4]

                    if today < duedate:
                        daysleft = str(duedate - today)
                        daysleft = daysleft.split(",")
                        daysleftLabel = Label(physicsframe, text =
f"{daysleft[0]}").grid(row = i+1, column = 2, padx = 20, pady = 20)

                    elif today > duedate:
                        daysleftLabel = Label(physicsframe, text = "Deadline
passed").grid(row = i+1, column = 2, padx = 20, pady = 20)

                    elif today == duedate:
                        daysleftLabel = Label(physicsframe, text =
"Today").grid(row = i+1, column = 2, padx = 20, pady = 20)
```

```
                    sql = f"SELECT Name FROM students WHERE RegistrationNo =
{regNoVar.get()}"
                    mycursor.execute(sql)
                    studname = mycursor.fetchall()

                    sql = f"SELECT {studname[0][0]} FROM assignments WHERE
Chapter = '{result[i][1]}' AND Topic = '{result[i][2]}'"
                    mycursor.execute(sql)
                    check = mycursor.fetchall()

                    if check[0][0] == None:
                        chooseFileButton = Button(physicsframe, text =
"Upload", command = choose_file).grid(row = i+1, column = 3,padx = 20, pady =
20)

                    else:
                        alreadyuploadedLabel = Label(physicsframe, text =
"Assignment submitted").grid(row = i+1, column = 3, padx = 20, pady = 20)

            quitButton = Button(physicsframe, text = "Quit", font =
('calibri', 10, 'bold', 'underline'), foreground = 'red', command =
quit).grid(row = len(result)+1, column = 0, pady = 20, padx = 10)

        def maths_assignments():

            sql = "SELECT * FROM assignments WHERE Subject = 'CS' order by
Chapter"
            mycursor.execute(sql)
            result = mycursor.fetchall()

            mathsframe = Frame(assignmentWindow)
            mathsframe.grid(row = 0, column = 1)

            chapter = Label(mathsframe, text = "Chapter", font=("Helvetica",
15, "underline")).grid(row = 0, column = 0, padx = 20, pady = 20)
            topic = Label(mathsframe, text = "Topic", font=("Helvetica", 15,
"underline")).grid(row = 0, column = 1, padx = 20, pady = 20)
            deadline = Label(mathsframe, text = "Due in", font=("Helvetica",
15, "underline")).grid(row = 0, column = 2, padx = 20, pady = 20)

            def choose_file():

                filename = filedialog.askopenfilename(initialdir = "*", title
= "Select a file", filetypes = (("pdf files", "*.pdf"), ("text files",
"*.txt")))

                sql = f"SELECT Name FROM students WHERE RegistrationNo =
{regNoVar.get()}"
                mycursor.execute(sql)
                studname = mycursor.fetchall()

                sql = f"UPDATE assignments SET {studname[0][0]} = '{filename}'
WHERE Chapter = '{result[i][1]}' AND Topic = '{result[i][2]}'"
                mycursor.execute(sql)
                mydb.commit()

                assignmentUploadSuccessful = Label(mathsframe, text =
"Assignment uploaded successfully.").grid(row = i+1, column = 5)

            def quit():

                mathsframe.grid_forget()
                mathsframe.destroy()
            for i in range(len(result)):

                    chapterLabel = Label(mathsframe, text =
f"{result[i][1]}").grid(row = i+1, column = 0, padx = 20, pady = 20)
                    topicButton = Button(mathsframe, text = f"{result[i][2]}",
command = partial(open_this, result[i][3])).grid(row = i+1, column = 1, padx =
20, pady = 20)

                    today = datetime.date.today()
                    duedate = result[i][4]

                    if today < duedate:
                        daysleft = str(duedate - today)
                        daysleft = daysleft.split(",")
                        daysleftLabel = Label(mathsframe, text =
f"{daysleft[0]}").grid(row = i+1, column = 2, padx = 20, pady = 20)

                    elif today > duedate:
                        daysleftLabel = Label(mathsframe, text = "Deadline
passed").grid(row = i+1, column = 2, padx = 20, pady = 20)

                    elif today == duedate:
                        daysleftLabel = Label(mathsframe, text =
"Today").grid(row = i+1, column = 2, padx = 20, pady = 20)

                    sql = f"SELECT Name FROM students WHERE RegistrationNo =
{regNoVar.get()}"
                    mycursor.execute(sql)
                    studname = mycursor.fetchall()

                    sql = f"SELECT {studname[0][0]} FROM assignments WHERE
```

```
Chapter = '{result[i][1]}' AND Topic = '{result[i][2]}'"
                    mycursor.execute(sql)
                    check = mycursor.fetchall()

                    if check[0][0] == None:
                        chooseFileButton = Button(mathsframe, text = "Upload",
command = choose_file).grid(row = i+1, column = 3,padx = 20, pady = 20)

                    else:
                        alreadyuploadedLabel = Label(mathsframe, text =
"Assignment submitted").grid(row = i+1, column = 3, padx = 20, pady = 20)

            quitButton = Button(mathsframe, text = "Quit", font = ('calibri',
10, 'bold', 'underline'), foreground = 'red', command = quit).grid(row =
len(result)+1, column = 0, pady = 20, padx = 10)

        def chemistry_assignments():

            sql = "SELECT * FROM assignments WHERE Subject = 'CS' order by
Chapter"
            mycursor.execute(sql)
            result = mycursor.fetchall()

            chemframe = Frame(assignmentWindow)
            chemframe.grid(row = 1, column = 0)

            chapter = Label(chemframe, text = "Chapter", font=("Helvetica",
15, "underline")).grid(row = 0, column = 0, padx = 20, pady = 20)
            topic = Label(chemframe, text = "Topic", font=("Helvetica", 15,
"underline")).grid(row = 0, column = 1, padx = 20, pady = 20)
            deadline = Label(chemframe, text = "Due in", font=("Helvetica",
15, "underline")).grid(row = 0, column = 2, padx = 20, pady = 20)
```

```
            def choose_file():

                filename = filedialog.askopenfilename(initialdir = "*", title
= "Select a file", filetypes = (("pdf files", "*.pdf"), ("text files",
"*.txt")))

                sql = f"SELECT Name FROM students WHERE RegistrationNo =
{regNoVar.get()}"
                mycursor.execute(sql)
                studname = mycursor.fetchall()

                sql = f"UPDATE assignments SET {studname[0][0]} = '{filename}'
WHERE Chapter = '{result[i][1]}' AND Topic = '{result[i][2]}'"
                mycursor.execute(sql)
                mydb.commit()

                assignmentUploadSuccessful = Label(chemframe, text =
"Assignment uploaded successfully.").grid(row = i+1, column = 5)

            def quit():

                chemframe.grid_forget()
                chemframe.destroy()

            for i in range(len(result)):

                    chapterLabel = Label(chemframe, text =
f"{result[i][1]}").grid(row = i+1, column = 0, padx = 20, pady = 20)
                    topicButton = Button(chemframe, text = f"{result[i][2]}",
command = partial(open_this, result[i][3])).grid(row = i+1, column = 1, padx =
20, pady = 20)
```

```
                    today = datetime.date.today()
                    duedate = result[i][4]

                    if today < duedate:
                        daysleft = str(duedate - today)
                        daysleft = daysleft.split(",")
                        daysleftLabel = Label(chemframe, text =
f"{daysleft[0]}").grid(row = i+1, column = 2, padx = 20, pady = 20)

                    elif today > duedate:
                        daysleftLabel = Label(chemframe, text = "Deadline
passed").grid(row = i+1, column = 2, padx = 20, pady = 20)

                    elif today == duedate:
                        daysleftLabel = Label(chemframe, text =
"Today").grid(row = i+1, column = 2, padx = 20, pady = 20)

                    sql = f"SELECT Name FROM students WHERE RegistrationNo =
{regNoVar.get()}"
                    mycursor.execute(sql)
                    studname = mycursor.fetchall()

                    sql = f"SELECT {studname[0][0]} FROM assignments WHERE
Chapter = '{result[i][1]}' AND Topic = '{result[i][2]}'"
                    mycursor.execute(sql)
                    check = mycursor.fetchall()

                    if check[0][0] == None:
                        chooseFileButton = Button(chemframe, text = "Upload",
command = choose_file).grid(row = i+1, column = 3,padx = 20, pady = 20)
```

```
                    else:
                        alreadyuploadedLabel = Label(chemframe, text =
"Assignment submitted").grid(row = i+1, column = 3, padx = 20, pady = 20)

            quitButton = Button(chemframe, text = "Quit", font = ('calibri',
10, 'bold', 'underline'), foreground = 'red', command = quit).grid(row =
len(result)+1, column = 0, pady = 20, padx = 10)

        def cs_assignments():

            sql = "SELECT * FROM assignments WHERE Subject = 'CS' order by
Chapter"
            mycursor.execute(sql)
            result = mycursor.fetchall()

            csframe = Frame(assignmentWindow)
            csframe.grid(row = 1, column = 1)

            chapter = Label(csframe, text = "Chapter", font=("Helvetica", 15,
"underline")).grid(row = 0, column = 0, padx = 20, pady = 20)
            topic = Label(csframe, text = "Topic", font=("Helvetica", 15,
"underline")).grid(row = 0, column = 1, padx = 20, pady = 20)
            deadline = Label(csframe, text = "Due in", font=("Helvetica", 15,
"underline")).grid(row = 0, column = 2, padx = 20, pady = 20)

            def choose_file():

                filename = filedialog.askopenfilename(initialdir = "*", title
= "Select a file", filetypes = (("pdf files", "*.pdf"), ("text files",
"*.txt")))

                sql = f"SELECT Name FROM students WHERE RegistrationNo =
{regNoVar.get()}"
                mycursor.execute(sql)
                studname = mycursor.fetchall()

                sql = f"UPDATE assignments SET {studname[0][0]} = '{filename}'
WHERE Chapter = '{result[i][1]}' AND Topic = '{result[i][2]}'"
                mycursor.execute(sql)
                mydb.commit()

                assignmentUploadSuccessful = Label(csframe, text = "Assignment
uploaded successfully.").grid(row = i+1, column = 5)

            def quit():

                csframe.grid_forget()
                csframe.destroy()

            for i in range(len(result)):

                    chapterLabel = Label(csframe, text =
f"{result[i][1]}").grid(row = i+1, column = 0, padx = 20, pady = 20)
                    topicButton = Button(csframe, text = f"{result[i][2]}",
command = partial(open_this, result[i][3])).grid(row = i+1, column = 1, padx =
20, pady = 20)

                    today = datetime.date.today()
                    duedate = result[i][4]

                    if today < duedate:
                        daysleft = str(duedate - today)
                        daysleft = daysleft.split(",")
                        daysleftLabel = Label(csframe, text =
f"{daysleft[0]}").grid(row = i+1, column = 2, padx = 20, pady = 20)

                    elif today > duedate:
                        daysleftLabel = Label(csframe, text = "Deadline
passed").grid(row = i+1, column = 2, padx = 20, pady = 20)

                    elif today == duedate:
                        daysleftLabel = Label(csframe, text =
"Today").grid(row = i+1, column = 2, padx = 20, pady = 20)

                    sql = f"SELECT Name FROM students WHERE RegistrationNo =
{regNoVar.get()}"
                    mycursor.execute(sql)
                    studname = mycursor.fetchall()

                    sql = f"SELECT {studname[0][0]} FROM assignments WHERE
Chapter = '{result[i][1]}' AND Topic = '{result[i][2]}'"
                    mycursor.execute(sql)
                    check = mycursor.fetchall()

                    if check[0][0] == None:
                        chooseFileButton = Button(csframe, text = "Upload",
command = choose_file).grid(row = i+1, column = 3,padx = 20, pady = 20)

                    else:
                        alreadyuploadedLabel = Label(csframe, text =
"Assignment submitted").grid(row = i+1, column = 3, padx = 20, pady = 20)

            quitButton = Button(csframe, text = "Quit", font = ('calibri', 10,
'bold', 'underline'), foreground = 'red', command = quit).grid(row =
len(result)+1, column = 0, pady = 20, padx = 10)
```

```
        physicsAssignmentButton = Button(assignmentWindow, image =
physicsphoto, font = ('Segoe Print', 12)).grid(row = 0, column = 0, padx =
(20, 20), pady = 50)
        mathsAssignmentButton = Button(assignmentWindow, image = mathsphoto,
font = ('Segoe Print', 12)).grid(row = 0, column = 1, padx = (20, 20), pady =
50)
        chemistryAssignmentButton = Button(assignmentWindow, image =
chemphoto, font = ('Segoe Print', 12)).grid(row = 1, column = 0, padx = (20,
20), pady = 40)
        csAssignmentButton = Button(assignmentWindow, image = csphoto, font =
('Segoe Print', 12), command = cs_assignments).grid(row = 1, column = 1, padx
= (20, 20), pady = 40)


    def class_notes():

        window_notes = Toplevel()
        window_notes.title("Class Notes")
        window_notes.geometry("1200x700")
        background_label = Label(window_notes, image = matrixBackground)
        background_label.place(x=0, y=0, relwidth=1, relheight=1)
        window_notes.resizable(False, False)
        window4.wm_state('iconic')

        def open_this(address):
            os.startfile(address)

        def physics_notes():

            physicsframe = Frame(window_notes)
            physicsframe.grid(row = 0, column = 0)
            chapter = Label(physicsframe, text = "Chapter", font=("Helvetica",
15, "underline")).grid(row = 0, column = 0, padx = 20, pady = 20)
            classno = Label(physicsframe, text = "Class No.",
font=("Helvetica", 15, "underline")).grid(row = 0, column = 1, padx = 20, pady
= 20)

            sql = "SELECT * FROM physics where Notes is not NULL"
            mycursor.execute(sql)
            result = mycursor.fetchall()

            for i in range(len(result)):
                link = f"Class {result[i][1]}"
                global address
                address = result[i][2]
                chapterLabel = Label(physicsframe, text =
result[i][0]).grid(row = i+1, column = 0)
                noteLinkButton = Button(physicsframe, text = link, command =
partial(open_this, address)).grid(row = i+1, column = 1, pady = 10)

        def maths_notes():

            mathsframe = Frame(window_notes)
            mathsframe.grid(row = 0, column = 1)

            chapter = Label(mathsframe, text = "Chapter", font=("Helvetica",
15, "underline")).grid(row = 0, column = 0, padx = 20, pady = 20)
            classno = Label(mathsframe, text = "Class No.", font=("Helvetica",
15, "underline")).grid(row = 0, column = 1, padx = 20, pady = 20)

            sql = "SELECT * FROM maths where Notes is not NULL"
            mycursor.execute(sql)
            result = mycursor.fetchall()

            for i in range(len(result)):
                link = f"Class {result[i][1]}"
                global address
                address = result[i][2]
                chapterLabel = Label(mathsframe, text = result[i][0]).grid(row
= i+1, column = 0)
                noteLinkButton = Button(mathsframe, text = link, command =
partial(open_this, address)).grid(row = i+1, column = 1, pady = 10)

        def chemistry_notes():

            chemframe = Frame(window_notes)
            chemframe.grid(row = 1, column = 0)

            chapter = Label(chemframe, text = "Chapter", font=("Helvetica",
15, "underline")).grid(row = 0, column = 0, padx = 20, pady = 20)
            classno = Label(chemframe, text = "Class No.", font=("Helvetica",
15, "underline")).grid(row = 0, column = 1, padx = 20, pady = 20)

            sql = "SELECT * FROM chemistry where Notes is not NULL"
            mycursor.execute(sql)
            result = mycursor.fetchall()

            for i in range(len(result)):
                link = f"Class {result[i][1]}"
                global address
                address = result[i][2]
                chapterLabel = Label(chemframe, text = result[i][0]).grid(row
= i+1, column = 0)
                noteLinkButton = Button(chemframe, text = link, command =
partial(open_this, address)).grid(row = i+1, column = 1, pady = 10)

        def cs_notes():

            csframe = Frame(window_notes)
            csframe.grid(row = 1, column = 1, padx = (20, 20), pady = 40)

            chapter = Label(csframe, text = "Chapter", font=("Helvetica", 15,
"underline")).grid(row = 0, column = 0, padx = 20, pady = 20)
            classno = Label(csframe, text = "Class No.", font=("Helvetica",
15, "underline")).grid(row = 0, column = 1, padx = 20, pady = 20)

            sql = "SELECT * FROM cs where Notes is not NULL"
            mycursor.execute(sql)
            result = mycursor.fetchall()

            for i in range(len(result)):
                link = f"Class {result[i][1]}"
                global address
                address = result[i][2]
                chapterLabel = Label(csframe, text = result[i][0]).grid(row =
i+1, column = 0)
                noteLinkButton = Button(csframe, text = link, command =
partial(open_this, address)).grid(row = i+1, column = 1, pady = 10)

        physicsnotesButton = Button(window_notes, image = physicsphoto, font =
('Segoe Print', 12), command = physics_notes).grid(row = 0, column = 0, padx =
(20, 20), pady = 50)
        mathsnotesButton = Button(window_notes, image = mathsphoto, font =
('Segoe Print', 12), command = maths_notes).grid(row = 0, column = 1, padx =
(20, 20), pady = 50)
        chemistrynotesButton = Button(window_notes, image = chemphoto, font =
('Segoe Print', 12), command = chemistry_notes).grid(row = 1, column = 0, padx
= (20, 20), pady = 40)
        csnotesButton = Button(window_notes, image = csphoto, font = ('Segoe
Print', 12), command = cs_notes).grid(row = 1, column = 1, padx = (20, 20),
pady = 40)

    def class_recording():

        window_recording = Toplevel()
        window_recording.title("Class Recording")
        window_recording.geometry("1200x700")
        background_label = Label(window_recording, image = matrixBackground)
        background_label.place(x=0, y=0, relwidth=1, relheight=1)
        window_recording.resizable(False, False)
        window4.wm_state('iconic')


        def open_this(address):
            os.startfile(address)

        def physics_recording():

            physicsframe = Frame(window_recording)
            physicsframe.grid(row = 0, column = 0)

            chapter = Label(physicsframe, text = "Chapter", font=("Helvetica",
15, "underline")).grid(row = 0, column = 0, padx = 20, pady = 20)
            classno = Label(physicsframe, text = "Class No.",
font=("Helvetica", 15, "underline")).grid(row = 0, column = 1, padx = 20, pady
= 20)

            sql = "SELECT Chapter, ClassNumber, Recording FROM physics where
```

```
Recording is not NULL"
            mycursor.execute(sql)
            result = mycursor.fetchall()

            for i in range(len(result)):
                link = f"Class {result[i][1]}"
                global address
                address = result[i][2]
                chapterLabel = Label(physicsframe, text =
result[i][0]).grid(row = i+1, column = 0)
                noteLinkButton = Button(physicsframe, text = link, command =
partial(open_this, address)).grid(row = i+1, column = 1, pady = 10)

        def maths_recording():

            mathsframe = Frame(window_recording)
            mathsframe.grid(row = 0, column = 1)

            chapter = Label(mathsframe, text = "Chapter", font=("Helvetica",
15, "underline")).grid(row = 0, column = 0, padx = 20, pady = 20)
            classno = Label(mathsframe, text = "Class No.", font=("Helvetica",
15, "underline")).grid(row = 0, column = 1, padx = 20, pady = 20)

            sql = "SELECT Chapter, ClassNumber, Recording FROM maths where
Recording is not NULL"
            mycursor.execute(sql)
            result = mycursor.fetchall()

            for i in range(len(result)):
                link = f"Class {result[i][1]}"
                global address
                address = result[i][2]
                chapterLabel = Label(mathsframe, text = result[i][0]).grid(row
= i+1, column = 0)
                noteLinkButton = Button(mathsframe, text = link, command =
partial(open_this, address)).grid(row = i+1, column = 1, pady = 10)

        def chemistry_recording():

            chemframe = Frame(window_recording)
            chemframe.grid(row = 1, column = 0)

            chapter = Label(chemframe, text = "Chapter", font=("Helvetica",
15, "underline")).grid(row = 0, column = 0, padx = 20, pady = 20)
            classno = Label(chemframe, text = "Class No.", font=("Helvetica",
15, "underline")).grid(row = 0, column = 1, padx = 20, pady = 20)

            sql = "SELECT Chapter, ClassNumber, Recording FROM chemistry where
Recording is not NULL"
            mycursor.execute(sql)
            result = mycursor.fetchall()

            for i in range(len(result)):
                link = f"Class {result[i][1]}"
                global address
                address = result[i][2]
                chapterLabel = Label(chemframe, text = result[i][0]).grid(row
= i+1, column = 0)
                noteLinkButton = Button(chemframe, text = link, command =
partial(open_this, address)).grid(row = i+1, column = 1, pady = 10)

        def cs_recording():

            csframe = Frame(window_recording)
            csframe.grid(row = 1, column = 1, padx = (20, 20), pady = 40)

            chapter = Label(csframe, text = "Chapter", font=("Helvetica", 15,
"underline")).grid(row = 0, column = 0, padx = 20, pady = 20)
            classno = Label(csframe, text = "Class No.", font=("Helvetica",
15, "underline")).grid(row = 0, column = 1, padx = 20, pady = 20)

            sql = "SELECT Chapter, ClassNumber, Recording FROM cs where
Recording is not NULL"
            mycursor.execute(sql)
            result = mycursor.fetchall()

            for i in range(len(result)):
                link = f"Class {result[i][1]}"
                global address
                address = result[i][2]
                chapterLabel = Label(csframe, text = result[i][0]).grid(row =
i+1, column = 0)
                noteLinkButton = Button(csframe, text = link, command =
partial(open_this, address)).grid(row = i+1, column = 1, pady = 10)

        physicsrecButton = Button(window_recording, image = physicsphoto, font
= ('Segoe Print', 12), command = physics_recording).grid(row = 0, column = 0,
padx = (20, 20), pady = 50)
        mathsrecButton = Button(window_recording, image = mathsphoto, font =
('Segoe Print', 12), command = maths_recording).grid(row = 0, column = 1, padx
= (20, 20), pady = 50)
        chemistryrecButton = Button(window_recording, image = chemphoto, font
= ('Segoe Print', 12), command = chemistry_recording).grid(row = 1, column =
0, padx = (20, 20), pady = 40)
        csrecButton = Button(window_recording, image = csphoto, font = ('Segoe
Print', 12), command = cs_recording).grid(row = 1, column = 1, padx = (20,
20), pady = 40)


    LiveClassLinkButton = Button(window4, image = liveClassPhoto, command =
live_class).grid(row = 1, column = 0, padx = 30, pady = 40)
    checkAttendanceButton = Button(window4, image = attendancePhoto, command =
check_attendance).grid(row = 1, column = 1, padx = 30, pady = 40)
    newAssignmentButton = Button(window4, image = assignmentPhoto, command =
assignment).grid(row = 2, column = 0, padx = 30, pady = 40)
    classNotesButton = Button(window4, image = classNotesPhoto, command =
class_notes).grid(row = 2, column = 1, padx = 30, pady = 40)
    classRecordingButton = Button(window4, image = classRecordingPhoto,
command = class_recording).grid(row = 2, column = 2, padx = 30, pady = 40)


root = Tk()
root.title("Learning Management System")
root.geometry("1000x600")
welcomePhoto = PhotoImage(file = r"Images\welcome.png")
background_label = Label(root, image = welcomePhoto)
background_label.place(x=0, y=0, relwidth=1, relheight=1)
root.attributes("-fullscreen", False)
root.resizable(False, False)

#----------------------------IMAGES----------------------------------
#You need to initialise them here instead of inside the functions, otherwise
they take too much time to load

homescreen = PhotoImage(file = r"Images\home.png")
liveClassPhoto = PhotoImage(file = r"Images\liveclass.png")
classNotesPhoto = PhotoImage(file = r"Images\classnotes.png")
classRecordingPhoto = PhotoImage(file = r"Images\recording.png")
```

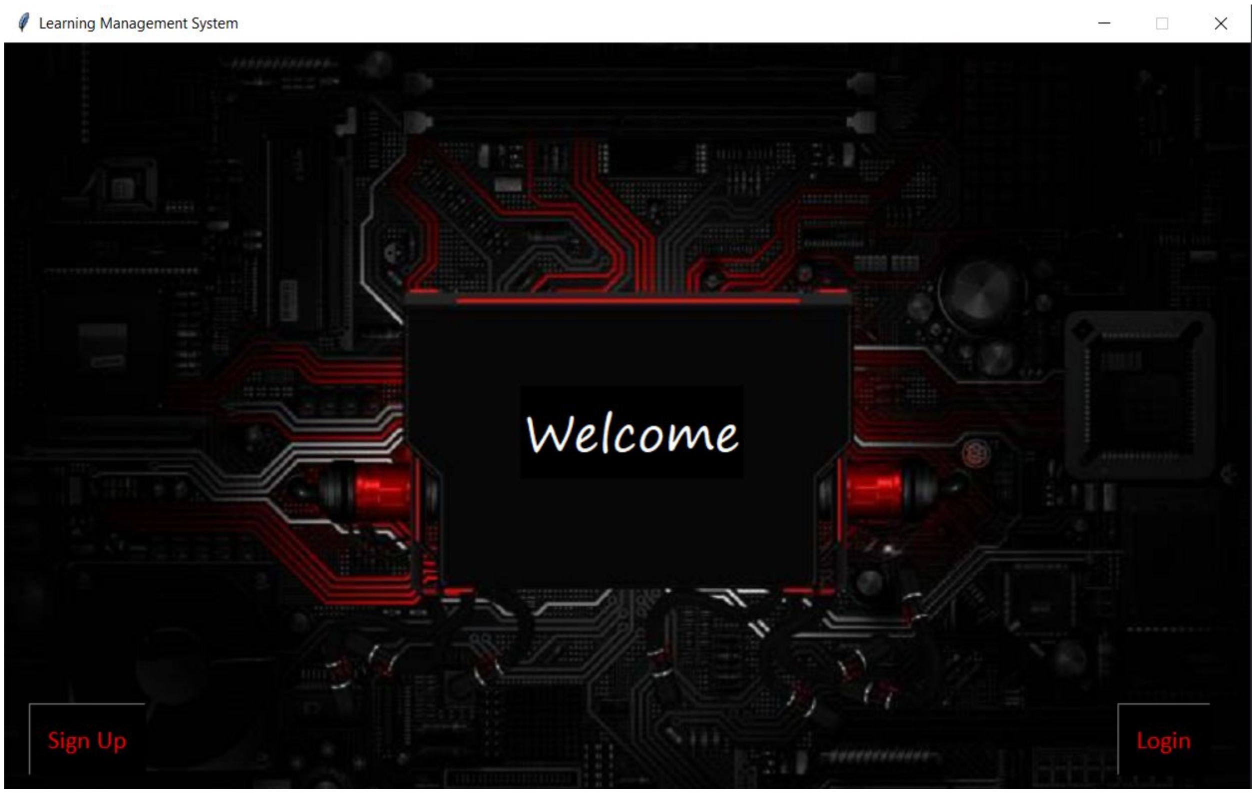
```
assignmentPhoto = PhotoImage(file = r"Images\assignment.png")
attendancePhoto = PhotoImage(file = r"Images\attendance.png")

physicsphoto = PhotoImage(file = r"Images\physics.png")
chemphoto = PhotoImage(file = r"Images\chemistry.png")
mathsphoto = PhotoImage(file = r"Images\maths.png")
csphoto = PhotoImage(file = r"Images\cs.png")
matrixBackground = PhotoImage(file = r"Images\matrix_background.png")

start()

root.mainloop()
```

MySQL Code

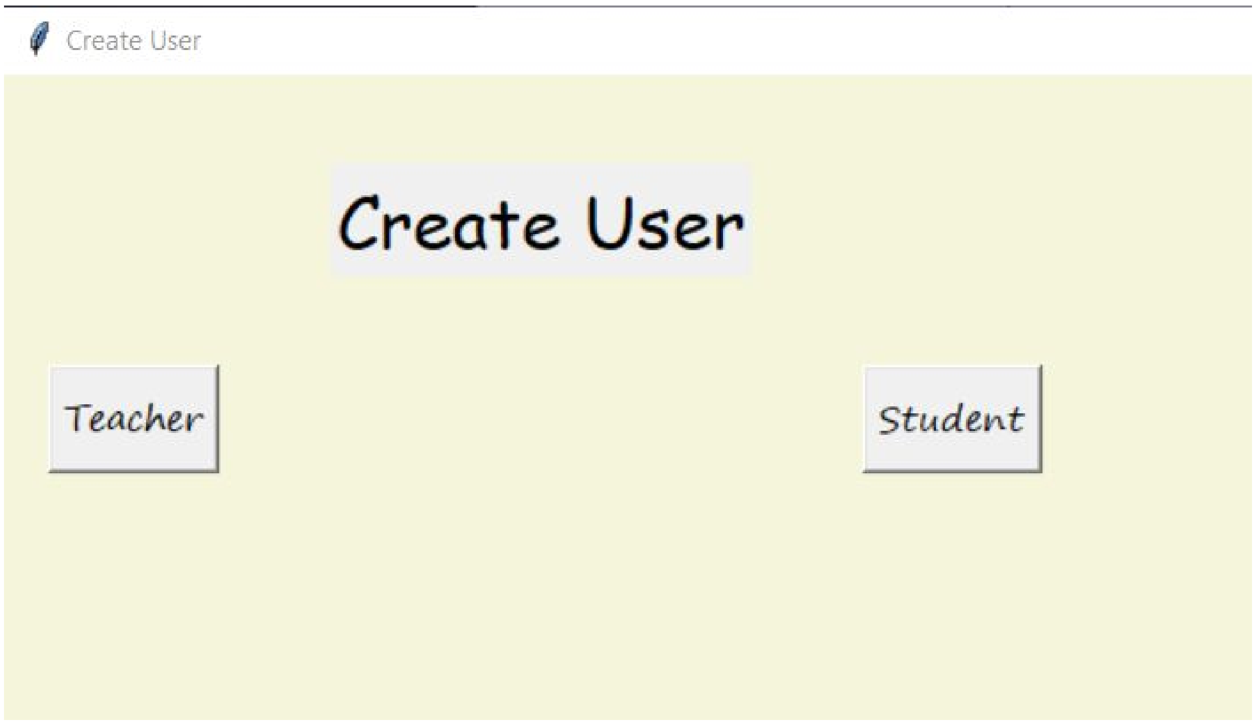
```
mysql> create database school_portal;
Query OK, 1 row affected (0.10 sec)

mysql> create table assignments(
    -> Subject varchar(20),
    -> Chapter varchar(20),
    -> Topic varchar(40),
    -> Link varchar(100),
    -> LastDate date);
Query OK, 0 rows affected (0.17 sec)

mysql> create table attendance(
    -> Student varchar(20),
    -> NoOfDaysPresent int(11)
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> create table chemistry(
    -> Chapter varchar(20),
    -> ClassNumber int(11),
    -> Notes varchar(70),
    -> Recording varchar(70)
    -> );
Query OK, 0 rows affected (0.14 sec)

mysql> create table cs(
    -> Chapter varchar(20),
    -> ClassNumber int(11),
    -> Notes varchar(70),
    -> Recording varchar(70)
    -> );
Query OK, 0 rows affected (0.14 sec)

mysql> create table maths(
    -> Chapter varchar(20),
    -> ClassNumber int(11),
    -> Notes varchar(70),
    -> Recording varchar(70)
    -> );
Query OK, 0 rows affected (0.14 sec)

mysql> create table physics(
    -> Chapter varchar(20),
    -> ClassNumber int(11),
    -> Notes varchar(70),
    -> Recording varchar(70)
    -> );
Query OK, 0 rows affected (0.14 sec)

mysql> create table liveclasslink(
    -> Link varchar(60)
    -> );
Query OK, 0 rows affected (0.11 sec)

mysql> create table students(
    -> Name varchar(20),
    -> Pin int(4),
    -> RegistrationNo int(7) PRIMARY KEY,
    -> DOB date,
    -> ContactNo bigint(10),
    -> Grade int(2),
    -> Section char(1),
    -> emailID varchar(40)
    -> );
Query OK, 0 rows affected (0.14 sec)

mysql> create table teachers(
    -> Name varchar(20),
    -> Pin int(4),
    -> Subject varchar(10),
    -> RegistrationNo int(7) PRIMARY KEY,
    -> DOB date,
    -> ContactNo bigint(10),
    -> emailID varchar(40)
    -> );
Query OK, 0 rows affected (0.21 sec)
```

Output

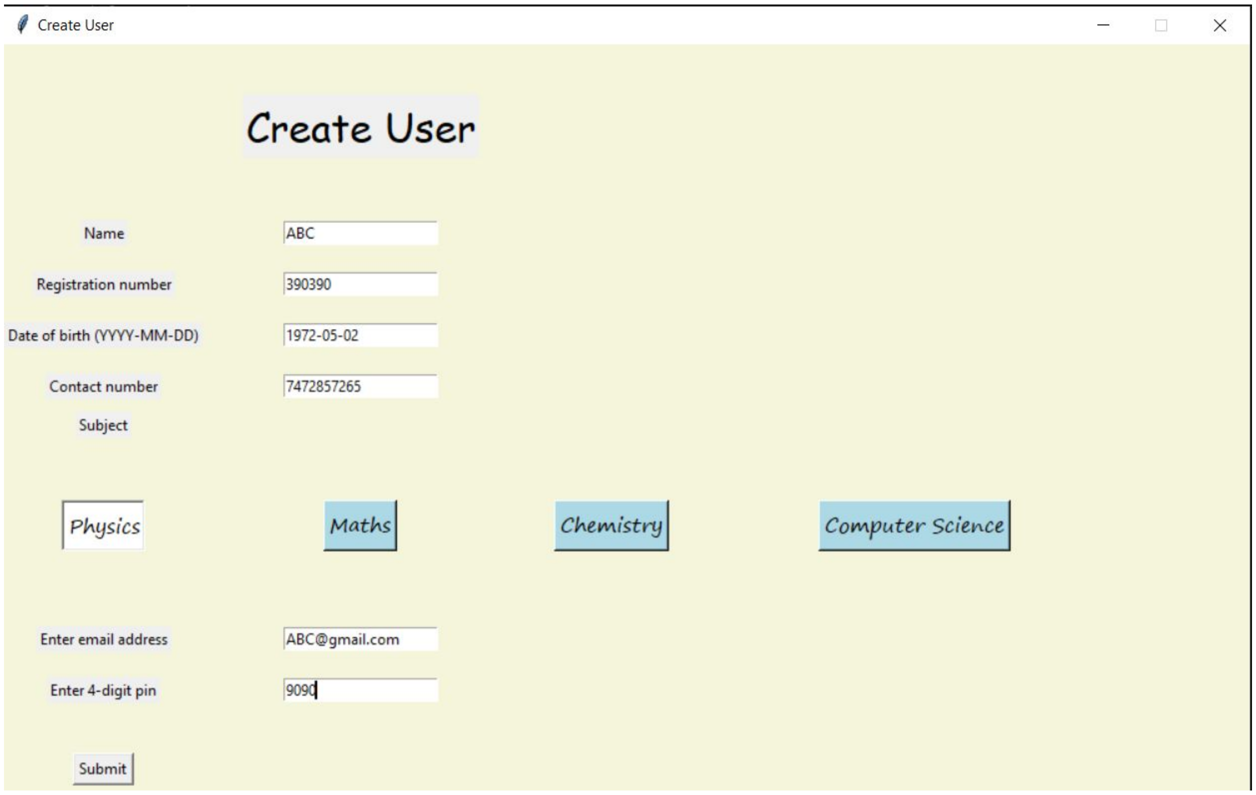
1. WELCOME SCREEN



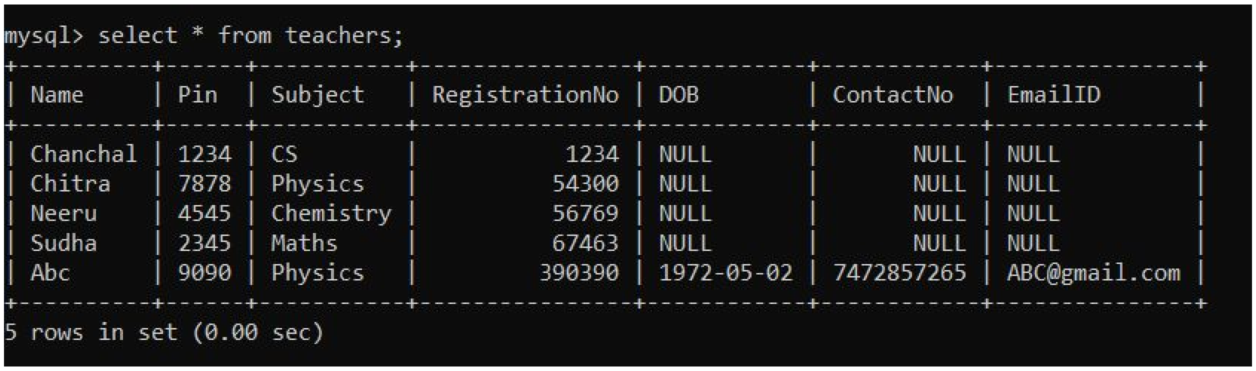
2. SIGN UP



- **TEACHER**

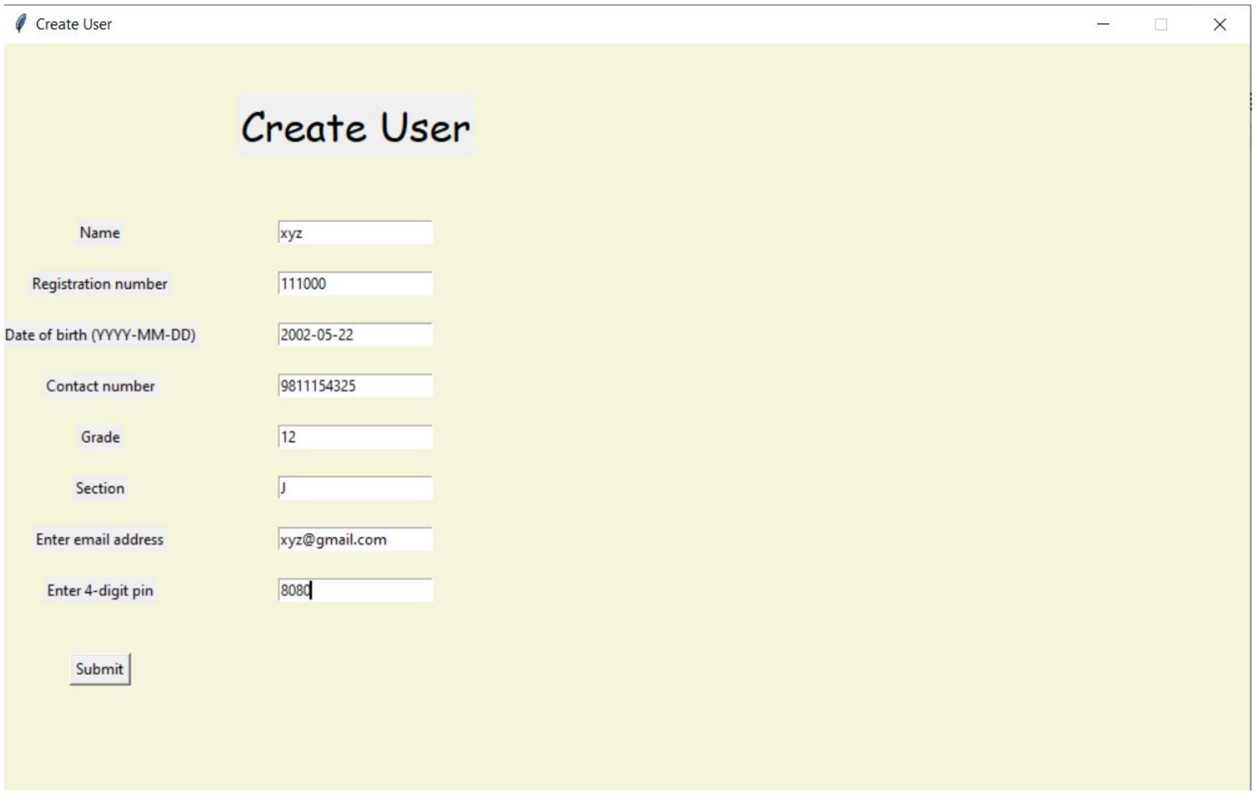


Name	ABC
Registration number	390390
Date of birth (YYYY-MM-DD)	1972-05-02
Contact number	7472857265
Subject	
Enter email address	ABC@gmail.com
Enter 4-digit pin	9090



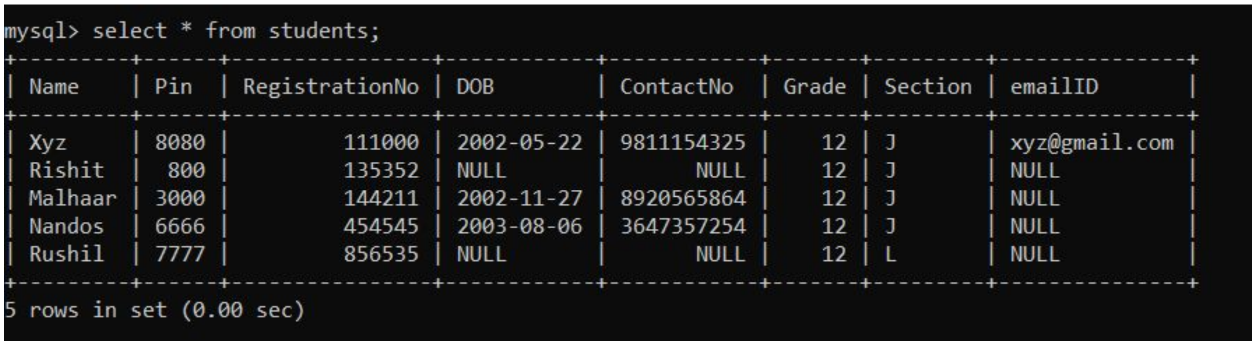
```
mysql> select * from teachers;
+----------+------+-----------+----------------+------------+------------+---------------+
| Name     | Pin  | Subject   | RegistrationNo | DOB        | ContactNo  | EmailID       |
+----------+------+-----------+----------------+------------+------------+---------------+
| Chanchal | 1234 | CS        |           1234 | NULL       |       NULL | NULL          |
| Chitra   | 7878 | Physics   |          54300 | NULL       |       NULL | NULL          |
| Neeru    | 4545 | Chemistry |          56769 | NULL       |       NULL | NULL          |
| Sudha    | 2345 | Maths     |          67463 | NULL       |       NULL | NULL          |
| Abc      | 9090 | Physics   |         390390 | 1972-05-02 | 7472857265 | ABC@gmail.com |
+----------+------+-----------+----------------+------------+------------+---------------+
5 rows in set (0.00 sec)
```

- **STUDENT**



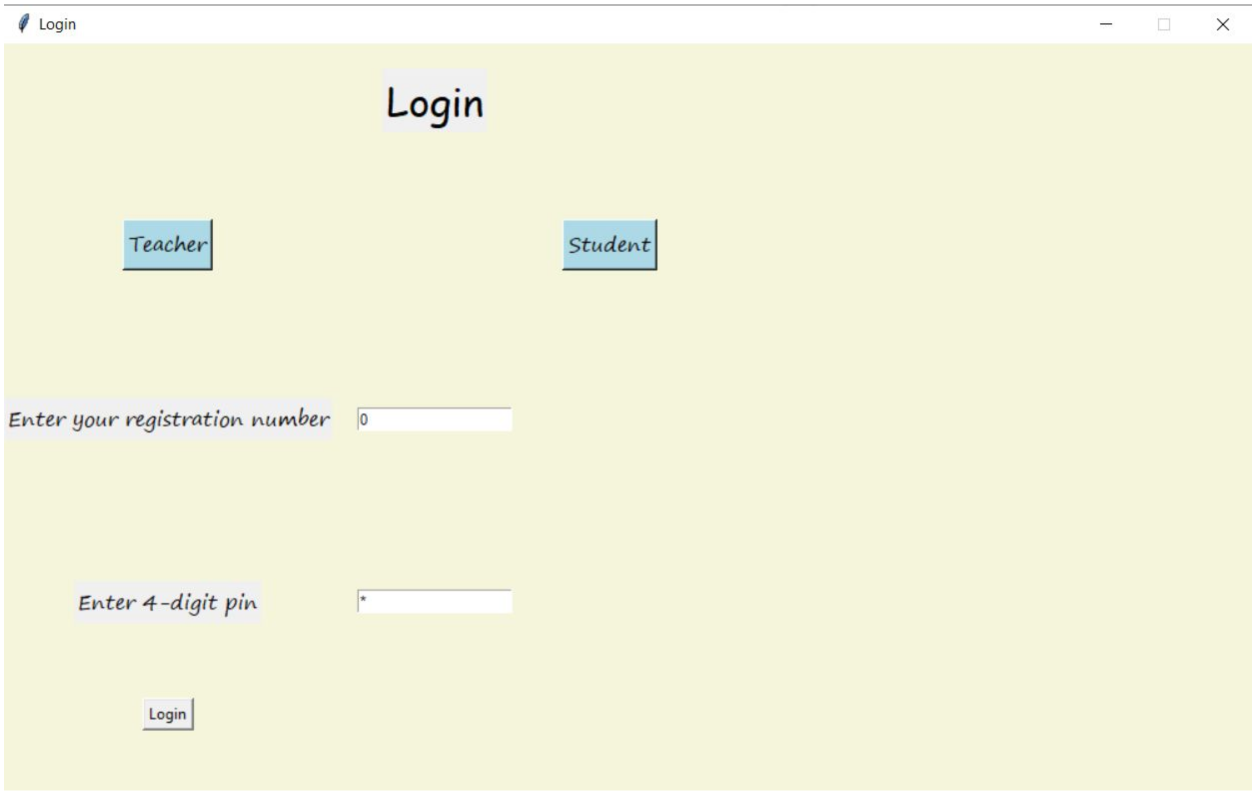
Create User

Name	xyz
Registration number	111000
Date of birth (YYYY-MM-DD)	2002-05-22
Contact number	9811154325
Grade	12
Section	J
Enter email address	xyz@gmail.com
Enter 4-digit pin	8080



```
mysql> select * from students;
+---------+------+----------------+------------+------------+-------+---------+---------------+
| Name    | Pin  | RegistrationNo | DOB        | ContactNo  | Grade | Section | emailID       |
+---------+------+----------------+------------+------------+-------+---------+---------------+
| Xyz     | 8080 |         111000 | 2002-05-22 | 9811154325 |    12 | J       | xyz@gmail.com |
| Rishit  |  800 |         135352 | NULL       |       NULL |    12 | J       | NULL          |
| Malhaar | 3000 |         144211 | 2002-11-27 | 8920565864 |    12 | J       | NULL          |
| Nandos  | 6666 |         454545 | 2003-08-06 | 3647357254 |    12 | J       | NULL          |
| Rushil  | 7777 |         856535 | NULL       |       NULL |    12 | L       | NULL          |
+---------+------+----------------+------------+------------+-------+---------+---------------+
5 rows in set (0.00 sec)
```

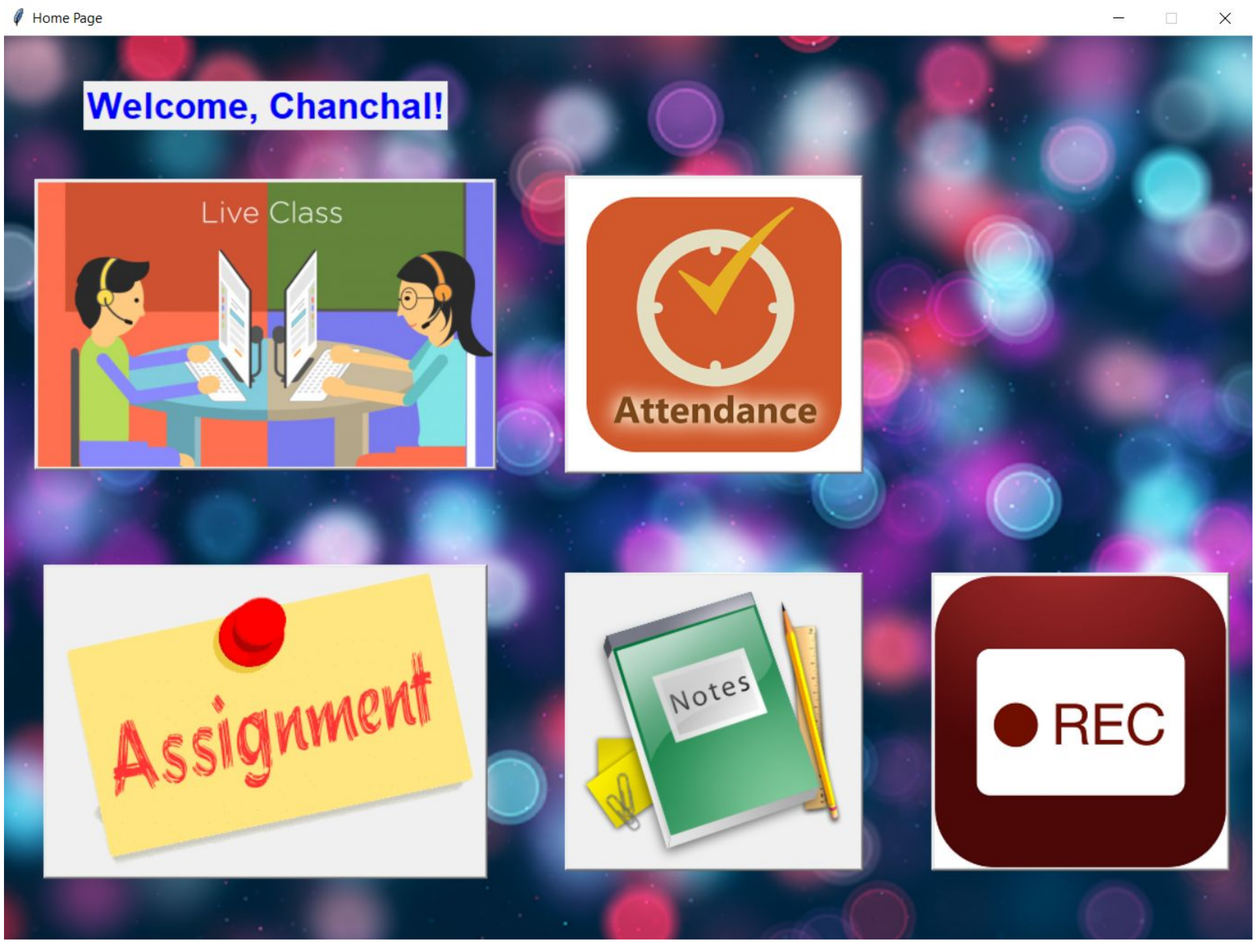
3. LOGIN



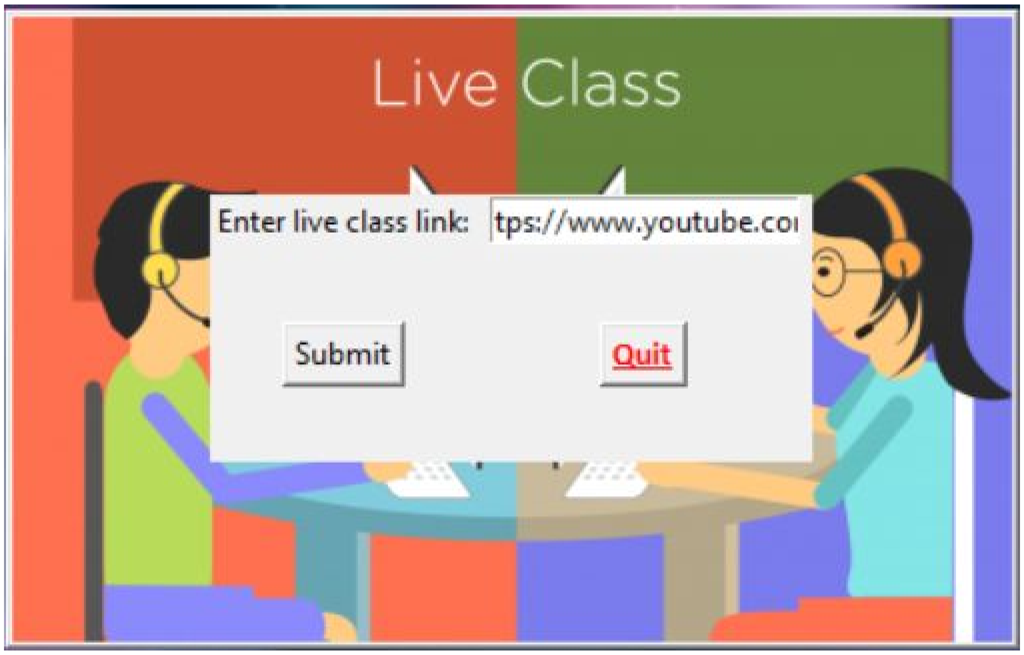
Enter your registration number 0

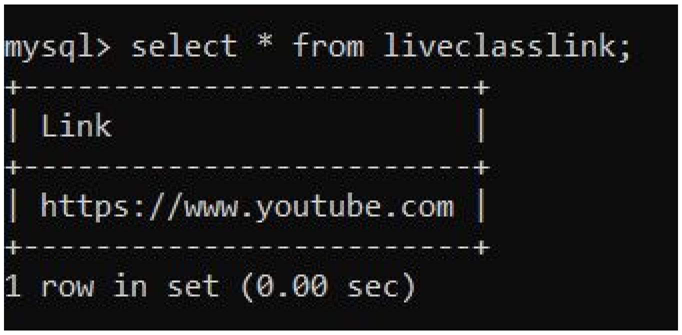
Enter 4-digit pin *

- **TEACHER**


Home Page
Welcome, Chanchal!
Live Class
Attendance
Assignment
Notes
REC

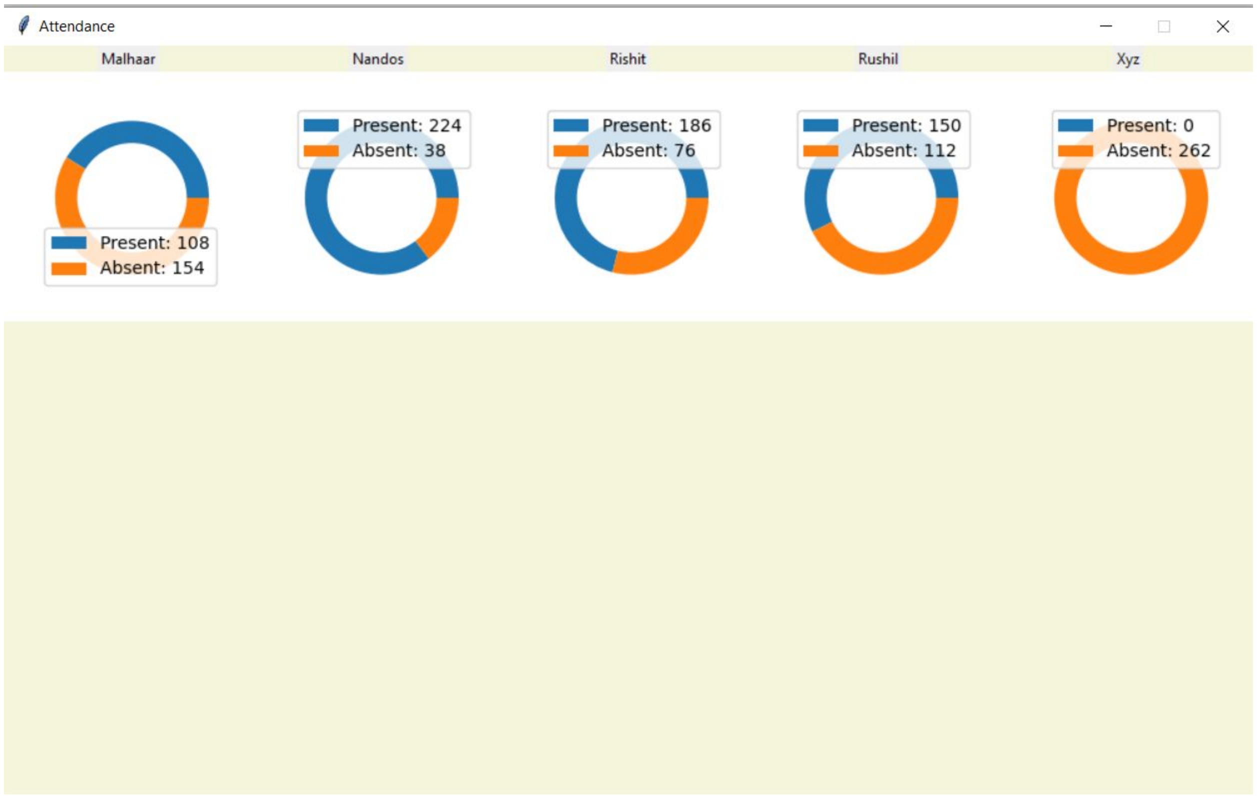
A. LIVE CLASS

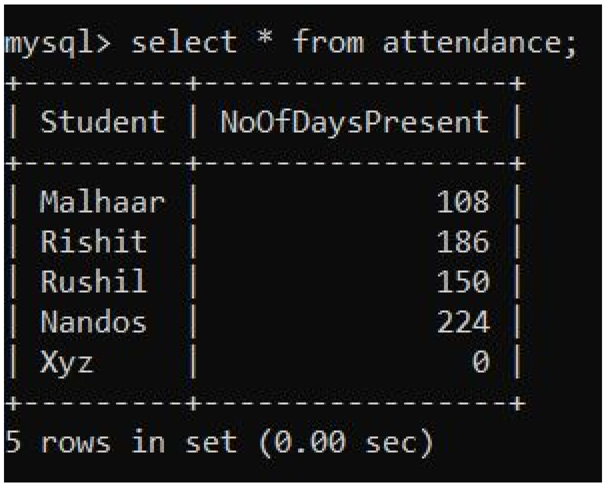




```
mysql> select * from liveclasslink;
+-------------------------+
| Link                    |
+-------------------------+
| https://www.youtube.com |
+-------------------------+
1 row in set (0.00 sec)
```

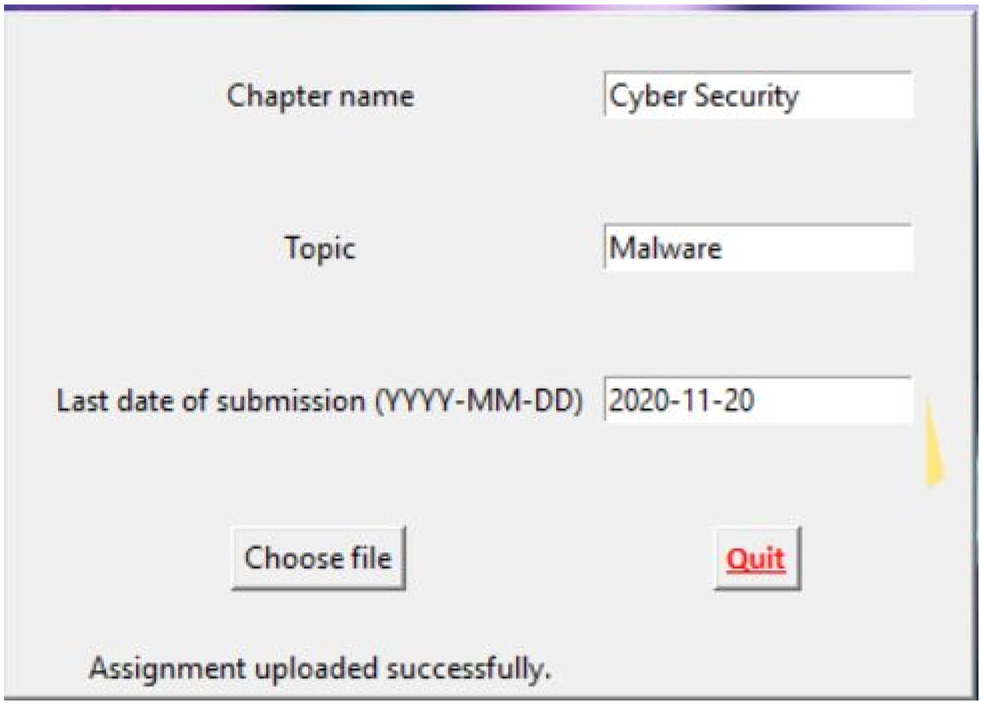
B. ATTENDANCE



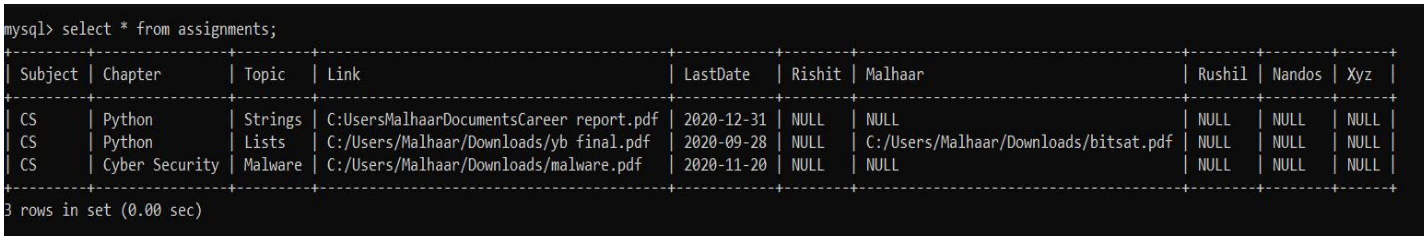


```
mysql> select * from attendance;
+---------+-----------------+
| Student | NoOfDaysPresent |
+---------+-----------------+
| Malhaar |             108 |
| Rishit  |             186 |
| Rushil  |             150 |
| Nandos  |             224 |
| Xyz     |               0 |
+---------+-----------------+
5 rows in set (0.00 sec)
```

C. ASSIGNMENT

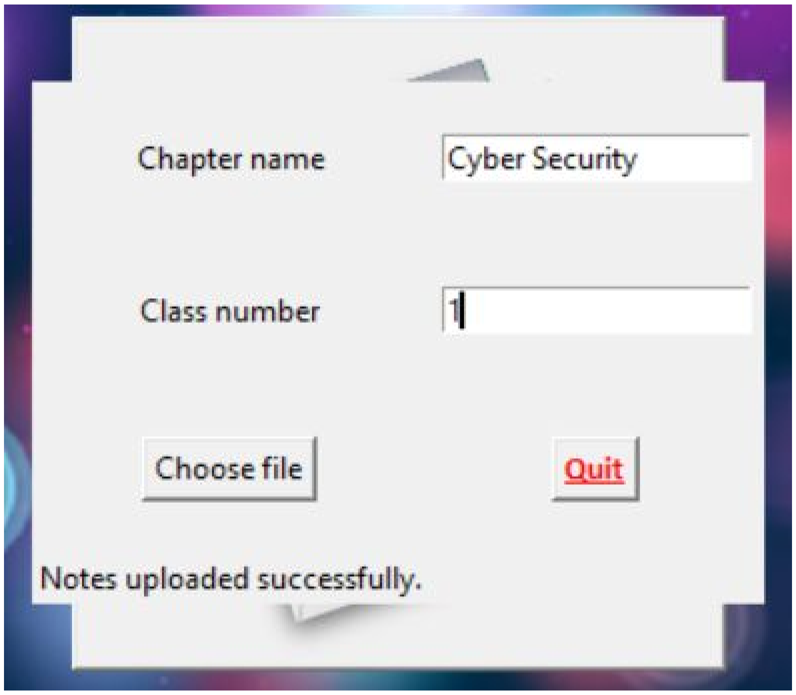


Assignment uploaded successfully.



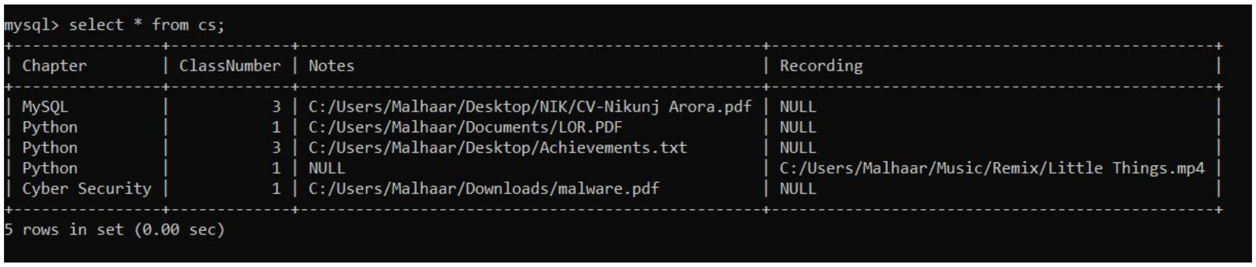
```
mysql> select * from assignments;
+---------+---------------+---------+-----------------------------------------+------------+--------+-----------------------------------------+--------+--------+------+
| Subject | Chapter       | Topic   | Link                                    | LastDate   | Rishit | Malhaar                                 | Rushil | Nandos | Xyz  |
+---------+---------------+---------+-----------------------------------------+------------+--------+-----------------------------------------+--------+--------+------+
| CS      | Python        | Strings | C:UsersMalhaarDocumentsCareer report.pdf | 2020-12-31 | NULL   | NULL                                    | NULL   | NULL   | NULL |
| CS      | Python        | Lists   | C:/Users/Malhaar/Downloads/yb final.pdf | 2020-09-28 | NULL   | C:/Users/Malhaar/Downloads/bitsat.pdf   | NULL   | NULL   | NULL |
| CS      | Cyber Security | Malware | C:/Users/Malhaar/Downloads/malware.pdf  | 2020-11-20 | NULL   | NULL                                    | NULL   | NULL   | NULL |
+---------+---------------+---------+-----------------------------------------+------------+--------+-----------------------------------------+--------+--------+------+
3 rows in set (0.00 sec)
```

D. NOTES


Chapter name Cyber Security

Class number 1

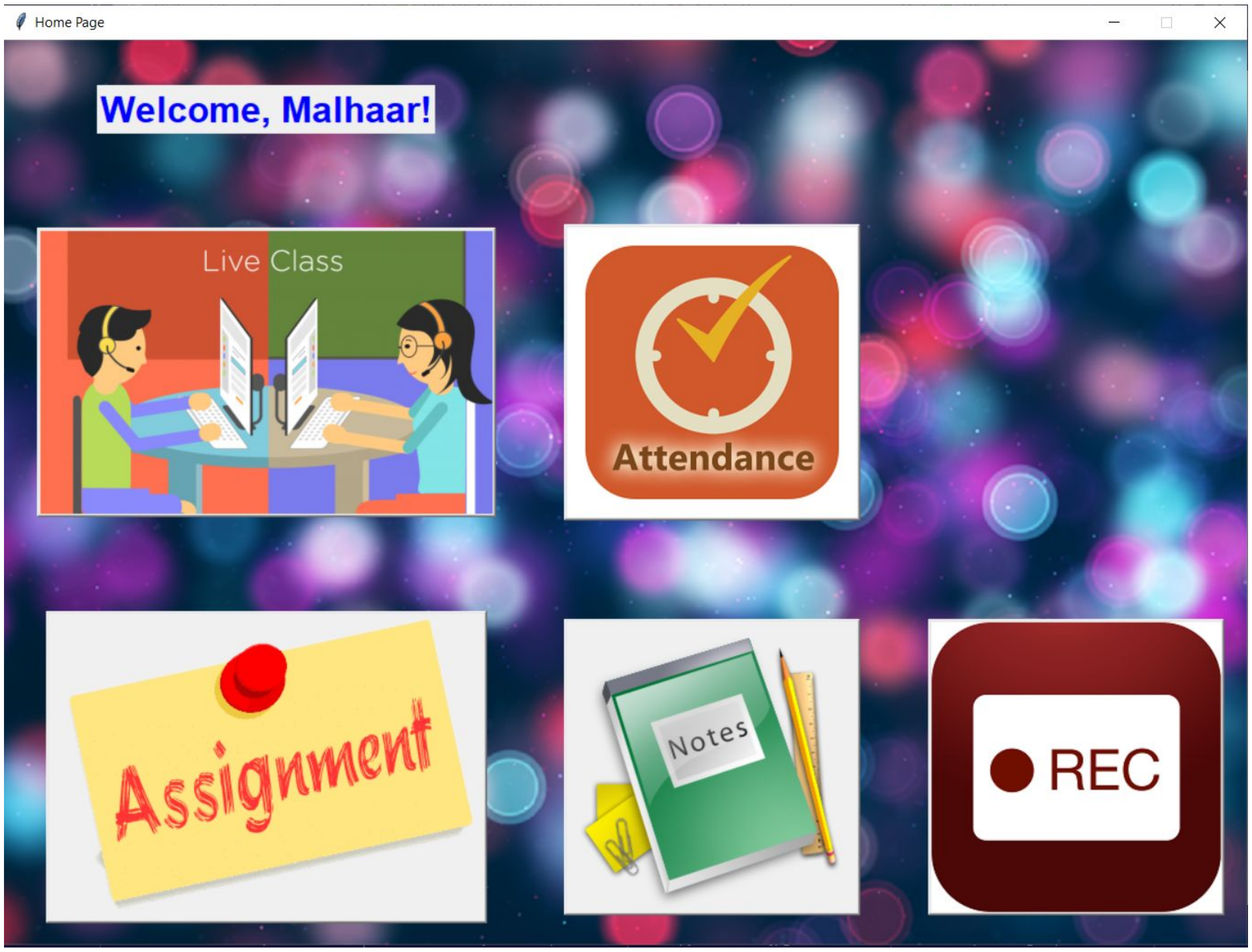
Notes uploaded successfully.


```
mysql> select * from cs;
+---------------+-------------+--------------------------------------------+------------------------------------------------+
| Chapter       | ClassNumber | Notes                                      | Recording                                      |
+---------------+-------------+--------------------------------------------+------------------------------------------------+
| MySQL         |           3 | C:/Users/Malhaar/Desktop/NIK/CV-Nikunj Arora.pdf | NULL                                     |
| Python        |           1 | C:/Users/Malhaar/Documents/LOR.PDF         | NULL                                           |
| Python        |           3 | C:/Users/Malhaar/Desktop/Achievements.txt  | NULL                                           |
| Python        |           1 | NULL                                       | C:/Users/Malhaar/Music/Remix/Little Things.mp4 |
| Cyber Security |          1 | C:/Users/Malhaar/Downloads/malware.pdf     | NULL                                           |
+---------------+-------------+--------------------------------------------+------------------------------------------------+
5 rows in set (0.00 sec)
```

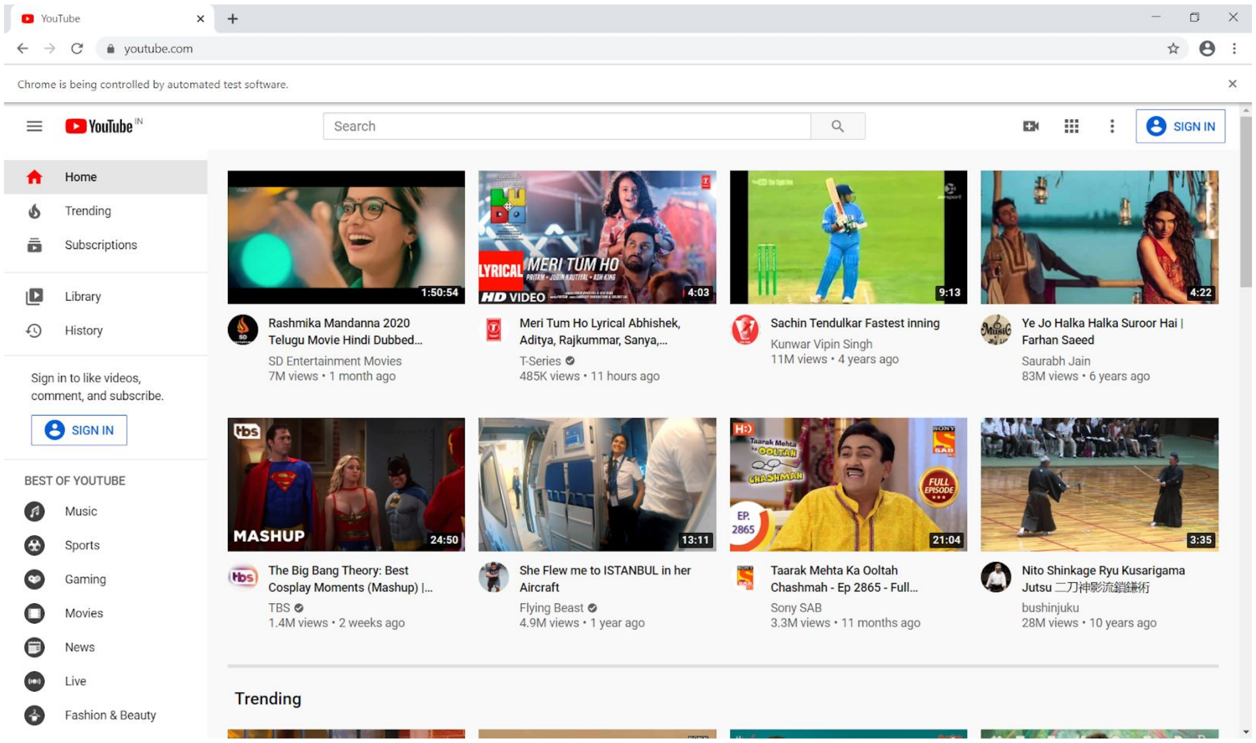
E. RECORDING

Same as notes

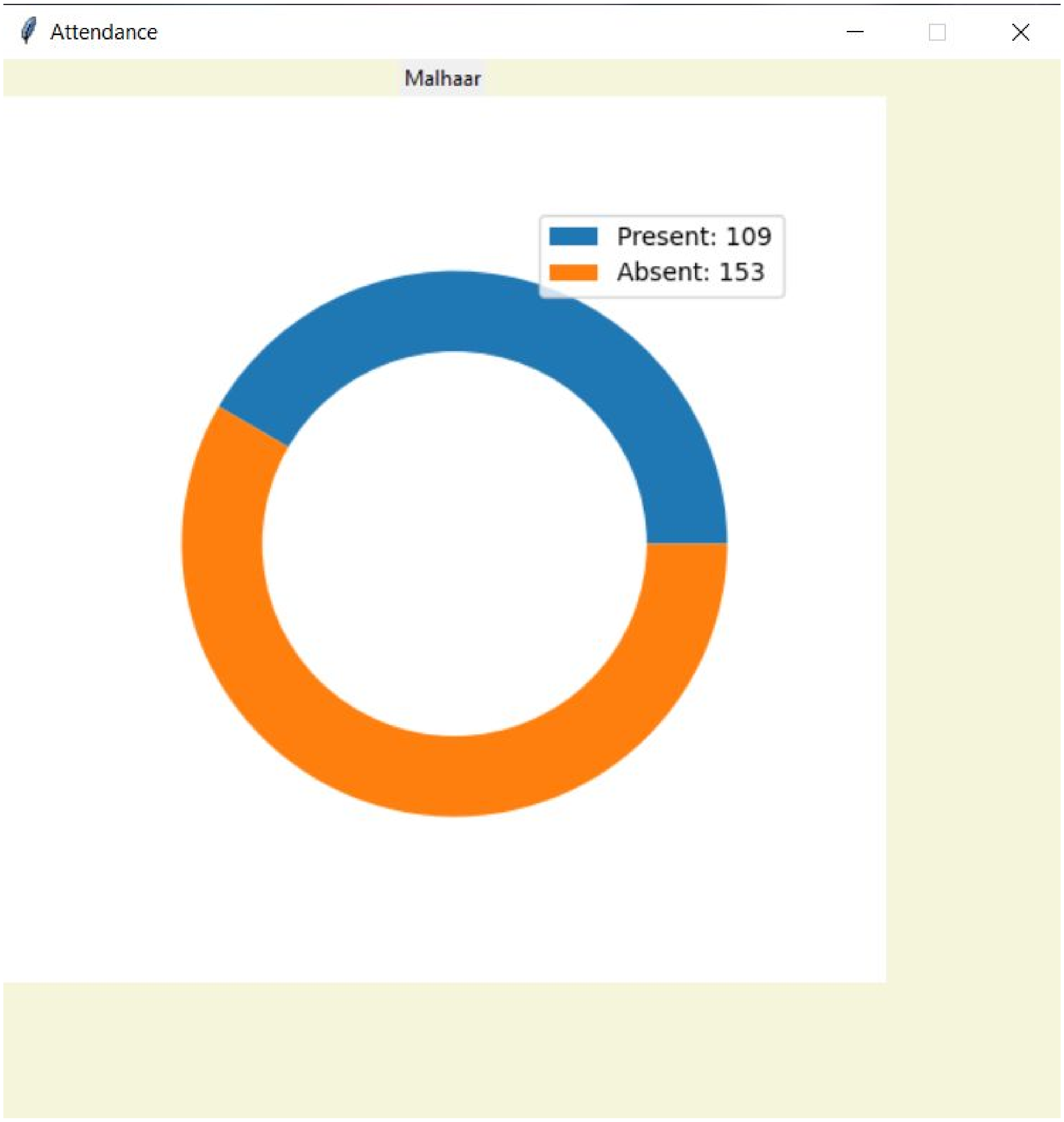
- **STUDENT**


Home Page
Welcome, Malhaar!
Live Class
Attendance
Assignment
Notes
REC

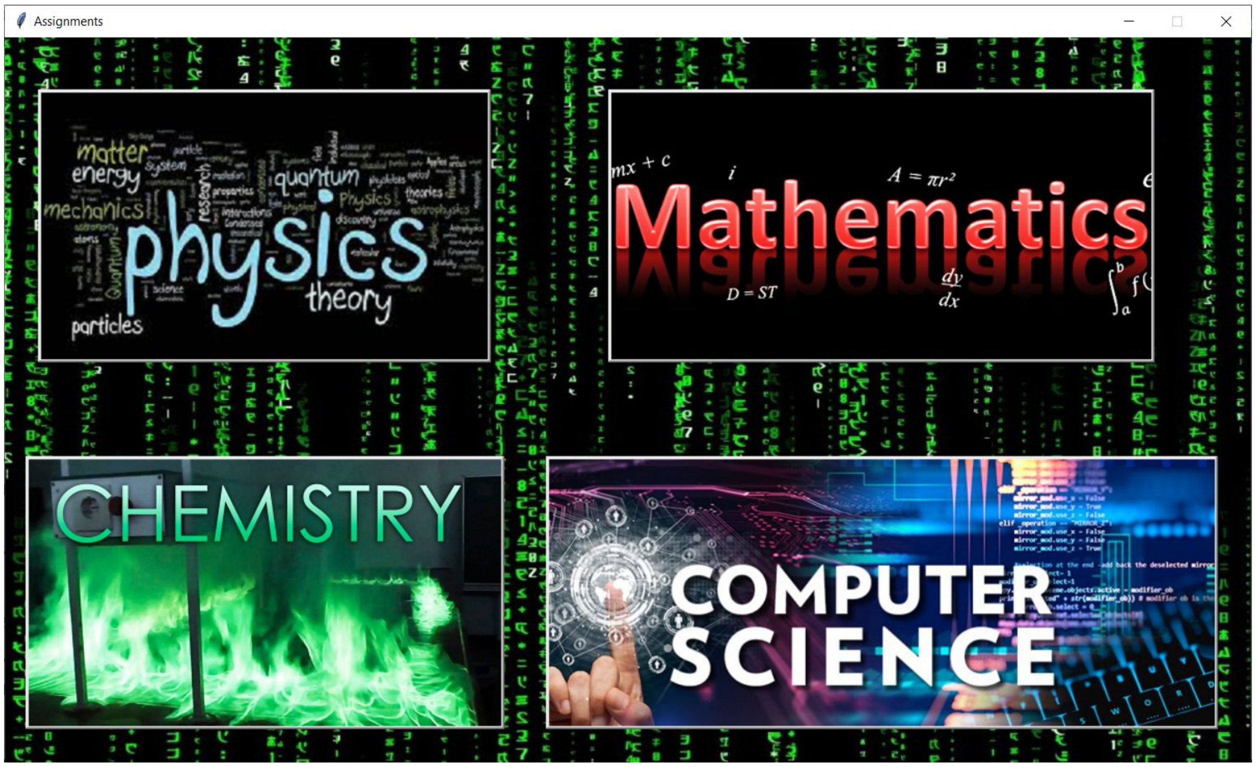
A. LIVE CLASS

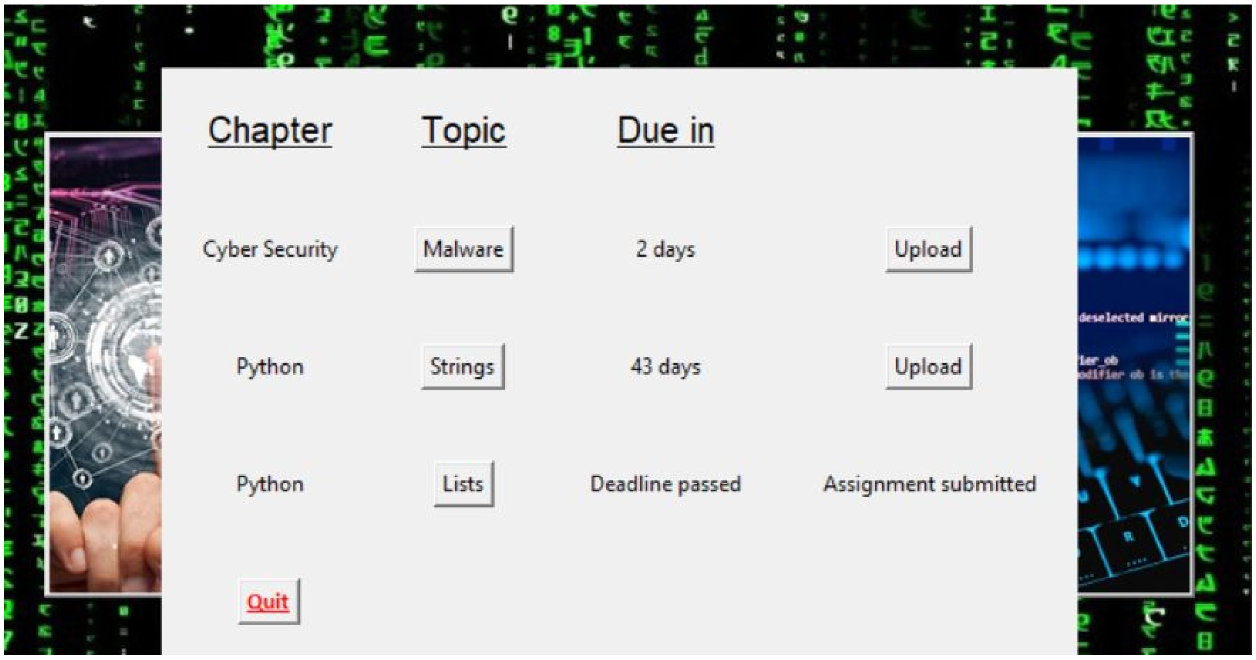

YouTube
youtube.com
Chrome is being controlled by automated test software.
Search
SIGN IN
Home
Trending
Subscriptions
Library
History
Sign in to like videos, comment, and subscribe.
SIGN IN
BEST OF YOUTUBE
Music
Sports
Gaming
Movies
News
Live
Fashion & Beauty
Rashmika Mandanna 2020 Telugu Movie Hindi Dubbed...
SD Entertainment Movies
7M views • 1 month ago
Meri Tum Ho Lyrical Abhishek, Aditya, Rajkummar, Sanya,...
T-Series
485K views • 11 hours ago
Sachin Tendulkar Fastest inning
Kunwar Vipin Singh
11M views • 4 years ago
Ye Jo Halka Halka Suroor Hai | Farhan Saeed
Saurabh Jain
83M views • 6 years ago
The Big Bang Theory: Best Cosplay Moments (Mashup) |...
TBS
1.4M views • 2 weeks ago
She Flew me to ISTANBUL in her Aircraft
Flying Beast
4.9M views • 1 year ago
Taarak Mehta Ka Ooltah Chashmah - Ep 2865 - Full...
Sony SAB
3.3M views • 11 months ago
Nito Shinkage Ryu Kusarigama Jutsu 二刀神影流鎖鎌術
bushinjuku
28M views • 10 years ago
Trending

B. ATTENDANCE



C. ASSIGNMENT

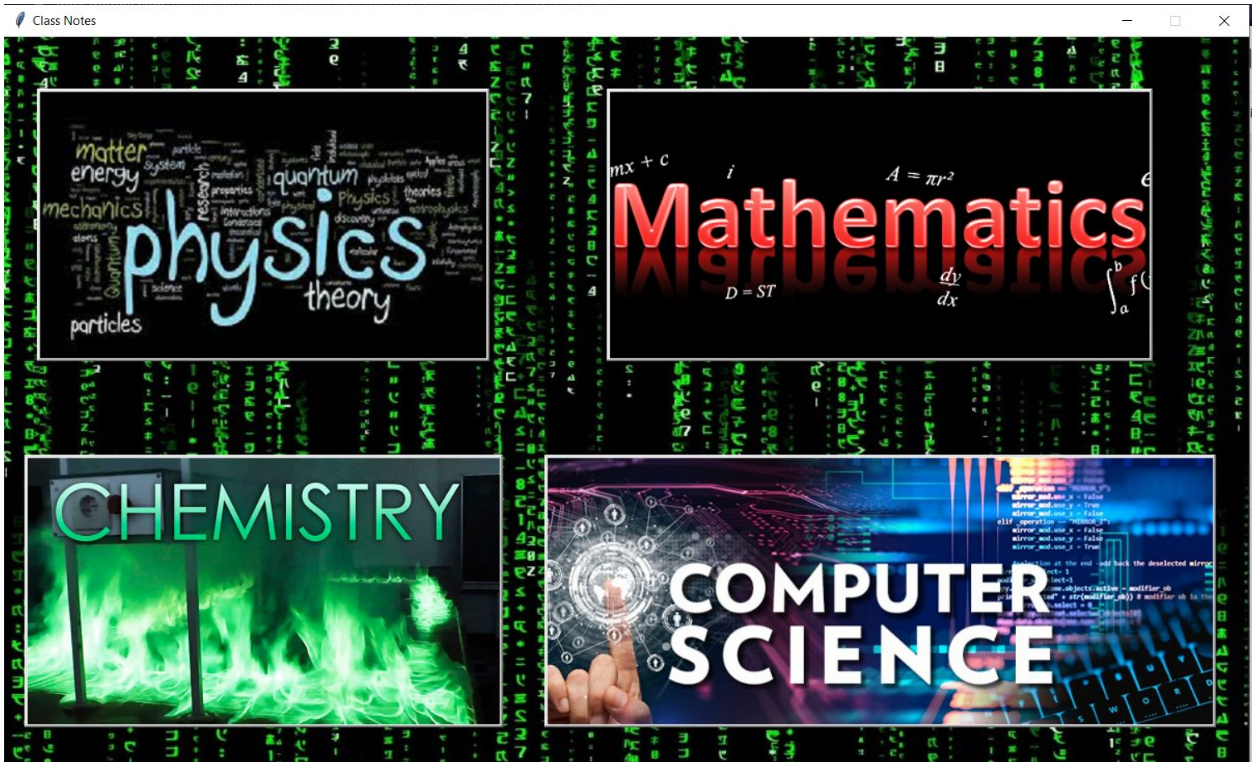


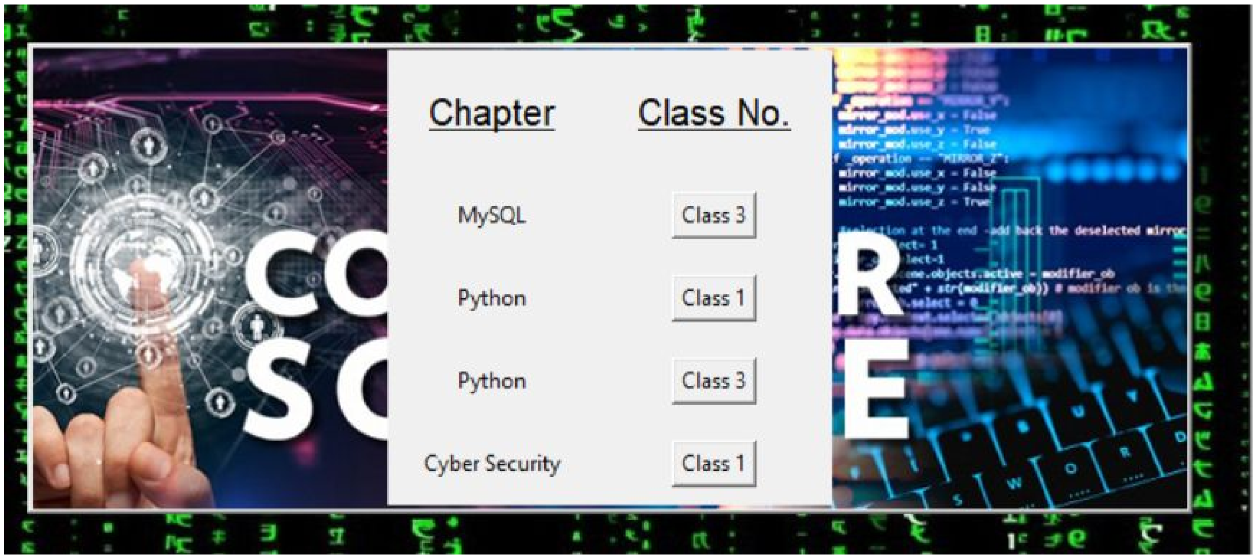


Chapter	Topic	Due in	
Cyber Security	Malware	2 days	Upload
Python	Strings	43 days	Upload
Python	Lists	Deadline passed	Assignment submitted

Quit

D. NOTES





Chapter	Class No.
MySQL	Class 3
Python	Class 1
Python	Class 3
Cyber Security	Class 1

E. RECORDING

Same as notes

THANK YOU