APPENDIX 2

Program Code

```
from tkinter import *

import tkinter as tk

from tkinter import ttk

from datetime import datetime

from tkinter import messagebox as msgbox

import mysql.connector

from mysql.connector import Error


class Dashboard:

    def __init__(self, master):

        self.master = master

        self.master.geometry("850x500")

        self.master.title("Dashboard - CAS Management System") # set the title of GUI window

        self.master.resizable(height = 0, width = 1000)

        self.project_status_dict = {}

        self.fetch_project_status()

        self.show_dashboard()


    def show_dashboard(self):

        self.frame1 = tk.Frame(self.master)

        # create a Form label

        Label(self.frame1, text="Dashboard", bg="Orange", font=("Calibri", 15,"bold")).grid(row=0,
column=0, columnspan=10)
```

```
        Label(self.frame1, text="Project List", font=("Calibri", 12,)).grid(row=1, column=0,
columnspan=10)
        self.frame1.pack()


        self.frame2 = tk.Frame(self.master)
        cols = ('ProjectID', 'Project Name', 'Start Date','End
Date','Remarks','Status','StudentID','SName','Grade','Batch')
        self.listBox = ttk.Treeview(self.frame2, columns=cols, show='headings',selectmode
='browse')


        for col in cols:
            self.listBox.column(col, width = 10, anchor ='c')
            self.listBox.heading(col, text=col)
        self.listBox.pack(fill='x',expand=True)


        tempList = self.fetch_project_details()
        for i in tempList:
            self.listBox.insert("", "end", values=i)
        self.frame2.pack(fill='x',expand=True)


        self.frame = tk.Frame(self.master)
        Label(self.frame, text="").grid(row=3, column=0, columnspan=10)
        self.create_button("Manage Projects",Projects,4,1)
        Label(self.frame, text=" <== Click this button to open Manage Projects
Screen").grid(row=4, column=2)
        Label(self.frame, text="").grid(row=5, column=0, columnspan=10)
```

```
        Label(self.frame, text="").grid(row=5, column=0, columnspan=10)
        self.create_button("Manage Students",Students,6,1)
        Label(self.frame, text=" <== Click this button to open Manage Students
Screen").grid(row=6, column=2)

        Label(self.frame, text="").grid(row=7, column=0, columnspan=10)
        self.create_button("Manage Attendance",Attendance,8,1)
        Label(self.frame, text=" <== Click this button to open Manage Attendance
Screen").grid(row=8, column=2)

        Label(self.frame, text="").grid(row=9, column=0, columnspan=10)
        refresh_btn = Button(self.frame, text="Refresh List", width=15,
command=self.refresh_projects_info).grid(row=10, column=1)
        Label(self.frame, text=" <== Click this button to refresh Project List").grid(row=10,
column=2)

        self.frame.pack()

    def refresh_projects_info(self):
        for i in self.listBox.get_children():
            self.listBox.delete(i)
        tempList = self.fetch_project_details()
        for i in tempList:
            self.listBox.insert("", "end", values=i)
```

```
    def create_button(self, text, _class,rowval,columnval):
        "Button that creates a new window"
        tk.Button(
            self.frame, text=text,
            command=lambda: self.new_window(_class)).grid(row=rowval, column=columnval)


    def new_window(self, _class):
        self.win = tk.Toplevel(self.master)
        self.win.transient(self.master)
        self.win.grab_set()
        _class(self.win)


    def fetch_project_status(self):
        try:
            self.project_status_dict.clear()
            connection = mysql.connector.connect(host='localhost', database='casdb', user='root',
password='')
            if connection.is_connected():
                db_Info = connection.get_server_info()
                cursor = connection.cursor(dictionary=True)
                cursor.execute("select * from tbl_projectstatuslookup;")
                records = cursor.fetchall()
                for row in records:
                    self.project_status_dict[row["PStatusID"]] = row["PStatusName"]
        except Error as e:
```

```
            print("Error while connecting to MySQL", e)
        finally:
            if (connection.is_connected()):
                cursor.close()
                connection.close()


    def fetch_project_details(self):
        try:
            connection = mysql.connector.connect(host='localhost', database='casdb', user='root',
password='')
            if connection.is_connected():
                cursor = connection.cursor(dictionary=True)
                cursor.execute("select * from tbl_projects;")
                records = cursor.fetchall()
                return_list = []
                for row in records:
                    ProjectID = row["ProjectID"]
                    ProjectName = row["ProjectName"]
                    ProjectStatus = row["PStatusID"]
                    StartDate = row["ProjectStartDate"]
                    EndDate = row["ProjectEndDate"]
                    Remarks = row["Remarks"]
                    stu_cursor = connection.cursor(dictionary=True)
                    stu_cursor.execute("SELECT * FROM tbl_students WHERE StudentID IN (SELECT
StudentID FROM tbl_spmap WHERE ProjectID = '"+str(row["ProjectID"])+"');")
                    stu_records = stu_cursor.fetchall()
```

```
                for stu_row in stu_records:
                    temp_list = [ProjectID,ProjectName,StartDate,EndDate,Remarks,self.project_status_dict[ProjectStatus],stu_row["StudentID"],stu_row["StudentName"],stu_row["Grade"],stu_row["Batch"]]
                    return_list.append(temp_list)
            return return_list

    except Error as e:
        print("Error while connecting to MySQL", e)
    finally:
        if (connection.is_connected()):
            cursor.close()
            connection.close()


class Projects(Dashboard):
    def __init__(self, master):
        self.master = master
        self.master.title("Manage Projects - CAS Management System") # set the title of GUI window
        self.master.geometry("1240x540")
        self.master.resizable(height = 0, width = 0)
        self.project_status_dict = {}
        self.projectid = StringVar()
        self.projectname = StringVar()
        self.status  = StringVar()
```

```
        self.startdate  = StringVar()
        self.enddate  = StringVar()
        self.studentid= StringVar()
        self.remarks = StringVar()
        self.fetch_project_status()
        self.show_projects_window()

    def show_projects_window(self):
        self.frame1 = tk.Frame(self.master)
        # create a Form label
        Label(self.frame1, text="Projects", width=123, height=2, bg="Orange", font=("Calibri",
15,"bold")).grid(row=0, column=0, columnspan=10)
        refresh_btn = Button(self.frame1, text="Refresh List", width=24,
bg='green',fg='white',font=("Calibri", 12,"bold"),
command=self.refresh_projects_info).grid(row=1, sticky="E",column=0, columnspan=10)
        self.frame1.pack()

        self.frame2 = tk.Frame(self.master)
        cols = ('ProjectID', 'Project Name', 'Start Date','End
Date','Remarks','Status','StudentID','SName','Grade','Batch')
        self.listBox = ttk.Treeview(self.frame2, columns=cols, show='headings',selectmode
='browse')

        for col in cols:
            self.listBox.column(col, width = 10, anchor ='c')
            self.listBox.heading(col, text=col)
```

```
        self.listBox.pack(fill='x',expand=True)


        tempList = self.fetch_project_details()
        for i in tempList:
            self.listBox.insert("", "end", values=i)
        self.frame2.pack(fill='x',expand=True)



        self.frame = tk.Frame(self.master)
        # create a Form label
        Label(self.frame, text="").grid(row=3, column=0)
        Label(self.frame, text="Add project details",bg="Orange",height="1", font=("Calibri",
12,"bold")).grid(row=4, column=0,columnspan=10)
        Label(self.frame, text="").grid(row=5, column=0)


        Label(self.frame, text="Project ID", bg="lightgray", font=("Calibri", 13)).grid(row=6,
column=0)
        Label(self.frame, text="Project Name", bg="lightgray", font=("Calibri", 13)).grid(row=6,
column=1)
        Label(self.frame, text="Status", bg="lightgray", font=("Calibri", 13)).grid(row=6, column=2)
        Label(self.frame, text="Start Date", bg="lightgray", font=("Calibri", 13)).grid(row=6,
column=3)
        Label(self.frame, text="End Date", bg="lightgray", font=("Calibri", 13)).grid(row=6,
column=4)
        Label(self.frame, text="Student ID", bg="lightgray", font=("Calibri", 13)).grid(row=6,
column=5)
```

```
        Label(self.frame, text="Remarks", bg="lightgray", font=("Calibri", 13)).grid(row=6,
column=6,columnspan=4)


        projectid_entry = Entry(self.frame, font=("Calibri", 12,"bold"),
textvariable=self.projectid).grid(row=7, column=0)
        projectname_entry = Entry(self.frame, font=("Calibri", 12,"bold"),
textvariable=self.projectname).grid(row=7, column=1)


        status_combo = ttk.Combobox(self.frame, width = 27,  textvariable =
self.status,state="readonly")


        status_list = []
        for i in self.project_status_dict.keys():
            status_list.append(str(i) + " - " + self.project_status_dict[i])


        status_combo['values'] = tuple(status_list)
        status_combo.grid(row = 7,column = 2)
        status_combo.current(0)


        startdate_entry = Entry(self.frame, font=("Calibri", 12,"bold"),
textvariable=self.startdate).grid(row=7, column=3)
        enddate_entry = Entry(self.frame, font=("Calibri", 12,"bold"),
textvariable=self.enddate).grid(row=7, column=4)


        studentid_combo = ttk.Combobox(self.frame, width = 27,  textvariable =
self.studentid,state="readonly")
```

```
        studentid_combo['values'] = self.fetch_student_detail()
        studentid_combo.grid(row = 7,column = 5)
        studentid_combo.current(0)
        remarks_entry = Entry(self.frame, font=("Calibri", 12),
textvariable=self.remarks).grid(row=7, column=6,columnspan=4)
        Label(self.frame, text="").grid(row=8, column=0)

        addnew_btn = Button(self.frame, text="Add (+)", font=("Calibri",
12,"bold"),command=self.add_projects_projects_screen).grid(row=9, column=0)

        Label(self.frame, text="").grid(row=10, column=0)
        viewstudents_btn = Button(self.frame, text="View Students Information", font=("Calibri",
12,"bold"), command= lambda: self.new_window(Students)).grid(row=11, column=0)
        Close_btn = Button(self.frame, text="Close", width = 20,
command=self.close_window,font=("Calibri", 12,"bold")).grid(row=11, column=8)
        Label(self.frame, text="").grid(row=12, column=0)

        self.frame.pack()

    def fetch_student_detail(self):
        try:
            connection = mysql.connector.connect(host='localhost', database='casdb', user='root',
password='')
            if connection.is_connected():
                cursor = connection.cursor()
                cursor.execute("SELECT * FROM tbl_students ;")
```

```
            records = cursor.fetchall()
            return_records = ['']
            if cursor.rowcount>0:
                for row in records:
                    return_records.append(str(row[0]) + " - " + row[1])
            else:
                msgbox.showerror('Add Project','Error: Unable to find the given student details')
            return tuple(return_records)

    except Error as e:
        print("Error while connecting to MySQL", e)
    finally:
        if (connection.is_connected()):
            cursor.close()
            connection.close()



def close_window(self):
    self.master.destroy()


def refresh_projects_info(self):
    for i in self.listBox.get_children():
        self.listBox.delete(i)
    tempList = self.fetch_project_details()
    for i in tempList:
```

```
            self.listBox.insert("", "end", values=i)


    def fetch_project_status(self):
        try:
            self.project_status_dict.clear()
            connection = mysql.connector.connect(host='localhost', database='casdb', user='root',
password='')
            if connection.is_connected():
                db_Info = connection.get_server_info()
                cursor = connection.cursor(dictionary=True)
                cursor.execute("select * from tbl_projectstatuslookup;")
                records = cursor.fetchall()
                for row in records:
                    self.project_status_dict[row["PStatusID"]] = row["PStatusName"]
        except Error as e:
            print("Error while connecting to MySQL", e)
        finally:
            if (connection.is_connected()):
                cursor.close()
                connection.close()


    def fetch_project_details(self):
        try:
            connection = mysql.connector.connect(host='localhost', database='casdb', user='root',
password='')
            if connection.is_connected():
```

```
            cursor = connection.cursor(dictionary=True)
            cursor.execute("select * from tbl_projects;")
            records = cursor.fetchall()
            return_list = []
            for row in records:
                ProjectID = row["ProjectID"]
                ProjectName = row["ProjectName"]
                ProjectStatus = row["PStatusID"]
                StartDate = row["ProjectStartDate"]
                EndDate = row["ProjectEndDate"]
                Remarks = row["Remarks"]
                stu_cursor = connection.cursor(dictionary=True)
                stu_cursor.execute("SELECT * FROM tbl_students WHERE StudentID IN (SELECT
StudentID FROM tbl_spmap WHERE ProjectID = '"+str(row["ProjectID"])+"');")
                stu_records = stu_cursor.fetchall()
                for stu_row in stu_records:
                    print(stu_row)
                    temp_list =
[ProjectID,ProjectName,StartDate,EndDate,Remarks,self.project_status_dict[ProjectStatus],stu
_row["StudentID"],stu_row["StudentName"],stu_row["Grade"],stu_row["Batch"]]
                    return_list.append(temp_list)
            return return_list

        except Error as e:
            print("Error while connecting to MySQL", e)
        finally:
```

```
            if (connection.is_connected()):
                cursor.close()
                connection.close()


    def add_projects_projects_screen(self):
        projectid_info = self.projectid.get()
        projectname_info = self.projectname.get()
        status_info = self.status.get()
        startdate_info = self.startdate.get()
        enddate_info = self.enddate.get()
        student_info = self.studentid.get()
        remarks_info = self.remarks.get()


        try:
            connection = mysql.connector.connect(host='localhost', database='casdb', user='root',
password='')
            if connection.is_connected():
                db_Info = connection.get_server_info()
                cursor = connection.cursor()
                cursor.execute("INSERT INTO tbl_projects(ProjectID,ProjectName,
PStatusID,ProjectStartDate,ProjectEndDate,Remarks)
VALUES('"+projectid_info+"','"+projectname_info+"','"+status_info+"','"+startdate_info+"','"+e
nddate_info+"','"+remarks_info+"');")
                if cursor.rowcount>0:
                    msgbox.showinfo('Add New Project','Success: Student details are added
successfully')
```

```
            else:
                msgbox.showerror('Add New Project','Error: Unable to add the given details. Please verify and proceed')

            cursor.execute("INSERT INTO tbl_spmap(StudentID, ProjectID) VALUES('"+student_info+"','"+projectid_info+"');")
            connection.commit()
        except Error as e:
            connection.rollback()
            print("Error while connecting to MySQL", e)
        finally:
            if (connection.is_connected()):
                cursor.close()
                connection.close()

    def fetch_projects_projects_screen():
        global project_status_dict
        try:
            connection = mysql.connector.connect(host='localhost', database='casdb', user='root', password='')
            if connection.is_connected():
                db_Info = connection.get_server_info()
                cursor = connection.cursor(dictionary=True)
                cursor.execute("select * from tbl_projects;")
                records = cursor.fetchall()
                for row in records:
```

```
                print("Project Id = ", row["ProjectID"], )
                print("Project Name = ", row["ProjectName"])
                print("Project Status  = ", project_status_dict[row["PStatusID"]])
                print("Project Start Date  = ", row["ProjectStartDate"])
                print("Project End Date  = ", row["ProjectEndDate"])
                print("Remarks  = ", row["Remarks"])
        except Error as e:
            print("Error while connecting to MySQL", e)
        finally:
            if (connection.is_connected()):
                cursor.close()
                connection.close()


class Students(Dashboard):
    def __init__(self, master):
        self.master = master
        self.master.title("Manage Students - CAS Management System") # set the title of GUI
window
        self.master.geometry("800x475")
        self.master.resizable(height = 0, width = 0)
        self.show_students_window()


    def show_students_window(self):
         self.frame1 = tk.Frame(self.master)
         # create a Form label
```

```
        Label(self.frame1, text="Manage Students", width = "80", bg="Orange", height="2",
font=("Calibri", 15,"bold")).grid(row=0, column=0, columnspan=4)

        refresh_btn = Button(self.frame1, text="Refresh List", width=15,
command=self.refresh_students_info).grid(row=0, column=3)

        self.frame1.pack()


        self.frame2 = tk.Frame(self.master)

        cols = ('StudentID','Student Name','Batch','Grade')

        self.listBox = ttk.Treeview(self.frame2, columns=cols, show='headings',selectmode
='browse')

        for col in cols:

          self.listBox.column(col, width = 40, anchor ='c')

          self.listBox.heading(col, text=col)


        self.listBox.pack(fill='x',expand=True)

        tempList = self.fetch_student_details()

        for i in tempList:

           self.listBox.insert("", "end", values=i)


#       self.listBox.grid(row=1, column=0, columnspan=4)

        self.frame2.pack(fill='x',expand=True)


        self.frame = tk.Frame(self.master)



        Label(self.frame, text="").grid(row=2, column=0)
```

```
        Label(self.frame, text="Add New Student",bg="Orange",height="1", font=("Calibri",
12,"bold")).grid(row=3, column=0,columnspan=4)

        Label(self.frame, text="").grid(row=4, column=0)

        Label(self.frame, text="Student ID", bg="lightgray", font=("Calibri", 13)).grid(row=5,
column=0)

        Label(self.frame, text="Student Name", bg="lightgray", font=("Calibri", 13)).grid(row=5,
column=1)

        Label(self.frame, text="Batch", bg="lightgray", font=("Calibri", 13)).grid(row=5, column=2)

        Label(self.frame, text="Grade", bg="lightgray", font=("Calibri", 13)).grid(row=5, column=3)


        global studentid_verify

        global studentname_verify

        global batch_verify

        global grade_verify


        studentid_verify = StringVar()

        studentname_verify =StringVar()

        batch_verify  = StringVar()

        grade_verify  = StringVar()

        studentID_entry = Entry(self.frame, font=("Calibri", 12,"bold"),
textvariable=studentid_verify).grid(row=6, column=0)

        studentname_entry = Entry(self.frame, font=("Calibri", 12,"bold"),
textvariable=studentname_verify).grid(row=6, column=1)

        batch_entry = Entry(self.frame, font=("Calibri", 12,"bold"),
textvariable=batch_verify).grid(row=6, column=2)

        grade_entry = Entry(self.frame, font=("Calibri", 12,"bold"),
textvariable=grade_verify).grid(row=6, column=3)
```

```
        Label(self.frame, text="").grid(row=7, column=0)
        addnew_btn = Button(self.frame, text="Add (+)", width=15,
command=self.add_students_info).grid(row=8, column=0)
        Close_btn = Button(self.frame, text="Close",  font=("Calibri", 12,"bold"), width=15,
command=self.close_window).grid(row=8, column=3,pady=10)
        Label(self.frame, text="").grid(row=9, column=0)
        self.frame.pack()

    def close_window(self):
        self.master.destroy()

    def fetch_student_details(self):
        try:
            connection = mysql.connector.connect(host='localhost', database='casdb', user='root',
password='')
            if connection.is_connected():
                cursor = connection.cursor(dictionary=True)
                cursor.execute("SELECT * FROM tbl_students;")
                records= cursor.fetchall()
                return_list = []
                for row in records:
                    temp_list = [row["StudentID"],row["StudentName"],row["Grade"],row["Batch"]]
                    return_list.append(temp_list)
                return return_list
        except Error as e:
            print("Error while connecting to MySQL", e)
```

```
        finally:
            if (connection.is_connected()):
                cursor.close()
                connection.close()


    def refresh_students_info(self):
        for i in self.listBox.get_children():
            self.listBox.delete(i)
        tempList = self.fetch_student_details()
        for i in tempList:
            self.listBox.insert("", "end", values=i)


    def add_students_info(self):
        studentid_info = studentid_verify.get()
        studentname_info = studentname_verify.get()
        batch_info = batch_verify.get()
        grade_info = grade_verify.get()
        try:
            connection = mysql.connector.connect(host='localhost', database='casdb', user='root',
password='')
            if connection.is_connected():
                db_Info = connection.get_server_info()
                cursor = connection.cursor()
                cursor.execute("INSERT INTO tbl_students
VALUES('"+studentid_info+"','"+studentname_info+"','"+batch_info+"','"+grade_info+"');")
                connection.commit()
```

```
            if cursor.rowcount>0:
                msgbox.showinfo('Add New Student','Success: Student details are added
successfully')
            else:
                msgbox.showerror('Add New Student','Error: Unable to add the given details. Please
verify and proceed')
        except Error as e:
            msgbox.showerror('Add New Student','Error: Unable to add the given details. Please
verify and proceed')
        finally:
            if (connection.is_connected()):
                cursor.close()
                connection.close()
            studentid_verify.set("")
            studentname_verify.set("")
            batch_verify.set("")
            grade_verify.set("")


class Attendance(Dashboard):
    def __init__(self, master):
        self.master = master
        self.master.title("Window n.2")
        self.master.geometry("335x250")
        self.master.title("Attendance Entry - CAS Management System") # set the title of GUI
window
        self.master.resizable(height = 0, width = 0)
```

```
        self.studentid = StringVar()
        self.attendance = StringVar()
        self.timestampStr= StringVar()
        self.show_attendance_window()

    def show_attendance_window(self):
        self.frame = tk.Frame(self.master)
        dateTimeObj = datetime.now()
        self.timestampStr = dateTimeObj.strftime("%a %d-%b-%Y  %r")

        # create a Form label
        Label(self.frame, text="Students Attendance Entry", bg="Orange", font=("Calibri",
15,"bold")).grid(row=0, column=0,columnspan=2)
        Label(self.frame, text=self.timestampStr, font=("Calibri", 11)).grid(row=1,
column=0,columnspan=2)

        Label(self.frame, text="").grid(row=2, column=0)
        Label(self.frame, text="Student ID : ").grid(row=3, column=0)

        studentid_combo = ttk.Combobox(self.frame, width = 27,  textvariable = self.studentid
,state="readonly")
        studentid_combo['values'] = self.fetch_student_detail()
        studentid_combo.grid(row = 3,column = 1)
        studentid_combo.current(0)

        Label(self.frame, text="").grid(row=5, column=0)
```

```
        Label(self.frame, text="Attendance : ").grid(row=6, column=0)

        attendance_combo = ttk.Combobox(self.frame, width = 27,  textvariable =
self.attendance,state="readonly")

        attendance_combo['values'] = (' ','Present','Absent')

        attendance_combo.grid(row = 6,column = 1)

        attendance_combo.current(0)


        Label(self.frame, text="").grid(row=7, column=0)

        Save_btn = Button(self.frame, text="Save Attendance",  font=("Calibri", 12,"bold"),
width=15, command=self.save_attendance).grid(row=8, column=0,columnspan=2,padx=5,
pady=5,sticky=W+E+N+S)

        Close_btn = Button(self.frame, text="Close",  font=("Calibri", 12,"bold"), width=15,
command=self.close_window).grid(row=9, column=0,columnspan=2,pady=10)


        self.frame.pack()


    def close_window(self):

        self.master.destroy()


    def fetch_student_detail(self):

        try:

            connection = mysql.connector.connect(host='localhost', database='casdb', user='root',
password='')

            if connection.is_connected():

                cursor = connection.cursor()

                cursor.execute("SELECT * FROM tbl_students ;")
```

```
            records = cursor.fetchall()
            return_records = ['']
            if cursor.rowcount>0:
                for row in records:
                    return_records.append(str(row[0]) + " - " + row[1])
            else:
                msgbox.showerror('Add Project','Error: Unable to find the given student details')
            return tuple(return_records)

        except Error as e:
            print("Error while connecting to MySQL", e)
        finally:
            if (connection.is_connected()):
                cursor.close()
                connection.close()

    def save_attendance(self):
        if self.studentid.get() != "":
            if self.attendance.get() !="":
                self.add_attendance(self.timestampStr,self.studentid.get().split(' -
')[0],self.attendance.get())
            else:
                msgbox.showerror('Add Attendance','Please select the attendance status')
        else:
            msgbox.showerror('Add Attendance','Please select the student detail')
```

```
    def add_attendance(self,attendance_date,studentid,attendance):
        attendance_dict = {"Present":"1","Absent":"0"}
        try:
            connection = mysql.connector.connect(host='localhost', database='casdb', user='root', password='')
            if connection.is_connected():
                db_Info = connection.get_server_info()
                cursor = connection.cursor()
                cursor.execute("INSERT INTO tbl_attendance(DateOfClass,StudentID,AttendanceStatus) VALUES('"+attendance_date+"','"+studentid+"','"+attendance_dict[attendance]+"');")
                connection.commit()
                if cursor.rowcount>0:
                    msgbox.showinfo('Add Attendance','Success: Attendance details are added successfully')
                    self.studentid.set('')
                    self.attendance.set('')

                    # To close the Attendance Window once the attendance is entered
                    #self.master.destroy()
                else:
                    msgbox.showerror('Add Attendance','Error: Unable to add the given details. Please verify and proceed')
        except Error as e:
            print("Error while connecting to MySQL", e)
```

```
        finally:
            if (connection.is_connected()):
                cursor.close()
                connection.close()



login_flag = False
def login_verify():
    global login_flag
    username_info = username_verify.get()
    password_info = password_verify.get()
    try:
        connection = mysql.connector.connect(host='localhost', database='casdb', user='root',
password='')
        if connection.is_connected():
            db_Info = connection.get_server_info()
            cursor = connection.cursor()
            cursor.execute("select Login_ID from tbl_login where Login_ID='"+username_info+"' and
Password='"+password_info+"';")
            record = cursor.fetchone()
            if cursor.rowcount>0:
                main_screen.destroy()
                login_flag = True
            else:
                msgbox.showerror('Login Failed','Error: Invalid Username or Password')
    except Error as e:
```

```
        print("Error while connecting to MySQL", e)
    finally:
        if (connection.is_connected()):
            cursor.close()
            connection.close()


def main_account_screen():
    global main_screen
    main_screen = Tk()   # create a GUI window
    main_screen.geometry("450x250") # set the configuration of GUI window
    main_screen.title("CAS Management System") # set the title of GUI window
    main_screen.resizable(height = 0, width = 0)
    global username_verify
    global password_verify
    username_verify = StringVar()
    password_verify = StringVar()


    Label(text="Login", bg="Orange", width="450", height="2", font=("Calibri", 15,"bold")).pack()


    # Set username entry
    Label(text="Enter the User Name", width="450", font=("Calibri", 12,"bold")).pack()
    global username_entry
    username_entry = Entry(main_screen, font=("Calibri", 15,"bold"),
textvariable=username_verify)
    username_entry.pack()
```

```
    Label(text="").pack()


    # Set password entry
    Label(text="Enter the password", width="450", font=("Calibri", 12,"bold")).pack()
    global password_entry
    password_entry = Entry(main_screen, font=("Calibri", 15,"bold"),
textvariable=password_verify,show= '*')
    password_entry.pack()
    Label(text="").pack()


    # create Login Button
    Button(text="Login", height="1", width="30",font=("Calibri",
12,"bold"),command=login_verify).pack()
    Label(text="").pack()


    main_screen.mainloop() # start the GUI


main_account_screen()
if login_flag == True:
    root = tk.Tk()
    app = Dashboard(root)
    root.mainloop()
```