**Vidya Bhawan Public School**

**Project report on**

**Student Management System**

**Session: 2022-23**

**Student Details:**

**Name :**Karan Suthar

**Class :**12th

**Roll no :**

**Subject :**Computer Science(083)

**Sub. Teacher :**Mrs. Neha Kataria

# **CERTIFICATE**

This is to certify that **JAYDEEPSHRIMALI**astudentofClassXIISciencehassatisfactorilycompletedtheprojectEntitled “**Grocery Store Management**”, under my **Mrs.NEHA KATARIA Ma’am** guidance and supervision duringthesession2022-2023.

## Iappreciatehiskeeninterestandsincereeffortsincovering all details of the project in a very systematicmanner. Iamvery pleased withhisproject.

**SUBJECTTEACHE PRINCIPAL EXAMINER**

# **ACKNOWLEDGEMENT**

## I would like to express my special thank of gratitude to my Computer Science teacher **Mrs. NEHA KATARIA Ma’am** for her able guidance and support and letting me work on this wonderful project.

I would also like to extend my deep gratitude towards our honourable Principal **Mrs.NEERJA JAIN Ma’am** for providing us all the required facilities and services.

**Karan Suthar**

XII(Science)

# **INDEX**

|  |  |  |
| --- | --- | --- |
| **S NO.** | **TOPICS** | **PAGE NO.** |
| 1. | INTRODUCTION | 1 |
| 2. | STRUCTURE OF PROJECT | 2 |
| 3. | ADMIN PANEL | 3 - 4 |
| 4. | USER PANEL | 5 |
| 5. | SOURCE CODE | 6 - 14 |
| 6. | OUTPUTS | 15 - 19 |
| 7. | TABLE STRUCTURE | 20 -21 |
| 8. | BIBLIOGRAPHY | 22 |

**INTRODUCTION**

The **Student Data Management System** is designed to efficiently handle student records within an educational institution. It streamlines administrative tasks related to student data, including **adding new student profiles**, **updating existing information**, and **removing outdated records**. Through this system, administrators can maintain accurate and up-to-date student details, ensuring smooth operations and effective communication.

**Working:**

* The basics functionality of this is based on mysql which facilates a database and tkinter which provides an gui for better interaction with user .
* This programme provides the an interface for both student and admin where student can see his/her data whereas an admin can manipulate the data of students which includes adding new student profiles, updating existing information, and removing outdated records.

**STRUCTURE OF PROJECT**

1. **Login Window:**
   * Users (both students and administrators) access the system through the login window.
   * The login window prompts users to enter their credentials (username and password).
   * Upon successful authentication, the system grants access to the appropriate interface.
2. **Student Interface:**
   * After logging in, students are directed to their personalized interface.
   * Here, students can view their personal information (name, roll number, contact details, etc.).
3. **Admin Interface:**
   * Administrators (such as teachers, school staff, or system administrators) access the admin panel.
   * Key functionalities in the admin interface include:
     + Adding New Student Profiles:
       - Admins can register new students by inputting their details.
     + Updating Existing Information:
       - Admins can modify student records.
     + Removing Outdated Records:
       - If a student leaves the institution or graduates, admins can archive or delete their records.

**FILE STRUCTURE OF PROJECT**

**SOURCE CODE:**

**Backend.py**

import mysql.connectoras sql  
from CTkMessageboximport CTkMessagebox  
  
  
def insert\_student\_record(roll, name, class\_number, date, gender, phone\_number, guardian\_name, address):  
*# if gender == 0:  
 # gender = "Male"  
 # else:  
 # gender = "Female"*query = "INSERT INTO STUDENT\_RECORDS (roll\_number, name, class, Date\_Of\_Birth, Gender, Phone\_Number, Guardians\_Name, Address)" \  
" VALUES ({}, '{}', {}, '{}', '{}', '{}', '{}', '{}' )".format(str(roll), name,  
str(class\_number), date, gender,  
str(phone\_number), guardian\_name,  
 address)  
try:  
cursor.execute(query)  
mydb.commit()  
CTkMessagebox(message="Data is Inserted successfully.",  
icon="check", option\_1="Ok")  
print("data entry success")  
except Exception as e:  
CTkMessagebox(title="Error", message=("Something went wrong!!!\n {}".format(e)), icon="cancel")  
  
  
  
def update\_student\_record(roll, field, updated\_value):  
query = "update student\_records set {} = \"{}\" where Roll\_Number={}".format(field, updated\_value, roll)  
*#update student\_records set name = where Roll\_Number={}*try:  
cursor.execute(query)  
mydb.commit()  
CTkMessagebox(message="Data is Updated successfully.",  
icon="check", option\_1="Ok")  
print("data update success")  
except Exception as e:  
print(e)  
CTkMessagebox(title="Error", message=("Something went wrong!!!\n {}".format(e)), icon="cancel")  
  
def delete\_student\_record(roll):  
query = "delete from student\_records where roll\_number = {}".format(roll)  
try:  
cursor.execute(query)  
mydb.commit()  
CTkMessagebox(message="Data is Deleted successfully.",  
icon="check", option\_1="Ok")  
print("data delete success")  
except Exception as e:  
CTkMessagebox(title="Error", message="Something went wrong!!!", icon="cancel")  
  
  
def get\_all\_data():  
mydb.commit()  
cursor.execute('SELECT \* FROM STUDENT\_RECORDS')  
data = cursor.fetchall()  
return data  
  
def get\_student\_data(roll):  
mydb.commit()  
cursor.execute('SELECT \* FROM STUDENT\_RECORDS')  
data = cursor.fetchall()  
student\_data= ()  
for iin data:  
if i[0] == int(roll):  
student\_data= i  
break  
  
 return student\_data  
  
  
  
mydb = sql.connect(host="localhost", user="root", passwd="root") *# Establishing SQL connection*cursor = mydb.cursor() *# Creating Cursor object*cursor.execute("CREATE DATABASE IF NOT EXISTS STUDENT\_MANAGEMENT\_SYSTEM")  
cursor.execute("USE STUDENT\_MANAGEMENT\_SYSTEM")  
cursor.execute('''  
CREATE TABLE IF NOT EXISTS STUDENT\_RECORDS (  
Roll\_Number INT PRIMARY KEY,  
 Name VARCHAR(50),  
 Class INT,  
Date\_Of\_Birth DATE,  
 Gender ENUM('Male', 'Female'),  
Phone\_Number BIGINT,  
Guardians\_Name VARCHAR(50),  
 Address VARCHAR(255)  
);  
''')

**Admin\_panel.py**

import customtkinteras c  
from CTkTableimport CTkTable  
from Backend.backendimport \*  
from Center\_windowimport center\_window  
  
  
def create\_admin\_window():  
root = c.CTk()  
root.title("Student Form")  
root.resizable(False, False)  
root.columnconfigure(1, weight=3, uniform="a")  
root.columnconfigure(2, weight=7, uniform="a")  
  
def panel\_selector(value, frame):  
frame.destroy()  
  
frame = c.CTkFrame(root)  
frame.grid(row=1, column=0, sticky="news", padx=10, pady=10)  
  
if value == "Delete": *# create delete window  
 # Roll Number Label and Entry*roll\_number\_label= c.CTkLabel(frame, text="Roll Number to be deleted ")  
roll\_number\_label.grid(row=0, column=0, sticky="we", columnspan=3, padx=10, pady=(10, 0))  
  
roll\_number\_entry= c.CTkEntry(frame)  
roll\_number\_entry.grid(row=1, column=0, sticky="we", columnspan=3, padx=10, pady=(0, 10))  
  
*# Instruction Label*instruct = c.CTkLabel(frame,  
text="All the data of the student with the selected roll number will be deleted")  
instruct.grid(row=0, column=3, columnspan=3, sticky="news", padx=10, pady=(10, 0))  
  
*# Delete Button*delete\_button\_button= c.CTkButton(frame, text="Delete",  
command=lambda: delete\_student\_record(roll\_number\_entry.get()))  
delete\_button\_button.grid(row=3, column=0, columnspan=3, sticky="we", padx=10, pady=10)  
  
  
elifvalue == "Update": *# create update window  
 # Roll Number Label and Entry*roll\_number\_label= c.CTkLabel(frame, text="Roll Number")  
roll\_number\_label.grid(row=0, column=0, sticky="w", padx=10, pady=(10, 0))  
  
roll\_number\_entry= c.CTkEntry(frame)  
roll\_number\_entry.grid(row=1, column=0, sticky="we", padx=10, pady=(0, 10))  
  
*# Field to Update Label and Combobox*field\_label= c.CTkLabel(frame, text="Field to Update")  
field\_label.grid(row=0, column=1, sticky="w", padx=10, pady=(10, 0))  
  
fields = ["Name", "Class", "Date\_Of\_Birth", "Gender", "Phone\_Number", "Guardians\_Name", "Address"]  
field\_combobox= c.CTkOptionMenu(frame, values=fields)  
field\_combobox.grid(row=1, column=1, sticky="we", padx=10, pady=(0, 10))  
  
*# New Value Label and Entry*new\_value\_label= c.CTkLabel(frame, text="New Value")  
new\_value\_label.grid(row=0, column=2, sticky="w", padx=10, pady=(10, 0))  
  
new\_value\_entry= c.CTkEntry(frame)  
new\_value\_entry.grid(row=1, column=2, sticky="we", padx=10, pady=(0, 10))  
  
*# Update Button*update\_data\_button= c.CTkButton(frame, text="Update",  
command=lambda: update\_student\_record(roll\_number\_entry.get(),  
field\_combobox.get(),  
new\_value\_entry.get())  
 )  
update\_data\_button.grid(row=2, column=0, columnspan=2, sticky="we", padx=10, pady=10)  
  
else: *# create insert window  
 # Create labels and entry fields*roll\_number\_label= c.CTkLabel(frame, text="Roll Number")  
roll\_number\_label.grid(row=0, column=0, sticky="w", padx=10, pady=(10, 0))  
roll\_number\_entry= c.CTkEntry(frame)  
roll\_number\_entry.grid(row=1, column=0, sticky="we", padx=10, pady=(0, 10))  
  
name\_label= c.CTkLabel(frame, text="Name")  
name\_label.grid(row=0, column=1, sticky="w", padx=10, pady=(10, 0))  
name\_entry= c.CTkEntry(frame)  
name\_entry.grid(row=1, column=1, sticky="we", padx=10, pady=(0, 10))  
  
class\_label= c.CTkLabel(frame, text="Class")  
class\_label.grid(row=0, column=2, sticky="w", padx=10, pady=(10, 0))  
class\_entry= c.CTkEntry(frame)  
class\_entry.grid(row=1, column=2, sticky="we", padx=10, pady=(0, 10))  
  
dob\_label= c.CTkLabel(frame, text="Date Of Birth")  
dob\_label.grid(row=0, column=3, sticky="w", padx=10, pady=(10, 0))  
dob\_entry= c.CTkEntry(frame)  
dob\_entry.grid(row=1, column=3, sticky="we", padx=10, pady=(0, 10))  
  
gender\_label= c.CTkLabel(frame, text="Gender")  
gender\_label.grid(row=0, column=4, sticky="w", padx=10, pady=(10, 0))  
gender\_entry= c.CTkEntry(frame)  
gender\_entry.grid(row=1, column=4, sticky="we", padx=10, pady=(0, 10))  
  
phone\_label= c.CTkLabel(frame, text="Phone Number")  
phone\_label.grid(row=0, column=5, sticky="w", padx=10, pady=(10, 0))  
phone\_entry= c.CTkEntry(frame)  
phone\_entry.grid(row=1, column=5, sticky="we", padx=10, pady=(0, 10))  
  
guardian\_label= c.CTkLabel(frame, text="Guardians Name")  
guardian\_label.grid(row=0, column=6, sticky="w", padx=10, pady=(10, 0))  
guardian\_entry= c.CTkEntry(frame)  
guardian\_entry.grid(row=1, column=6, sticky="we", padx=10, pady=(0, 10))  
  
address\_label= c.CTkLabel(frame, text="Address")  
address\_label.grid(row=0, column=7, sticky="w", padx=10, pady=(10, 0))  
address\_entry= c.CTkEntry(frame)  
address\_entry.grid(row=1, column=7, sticky="we", padx=10, pady=(0, 10))  
  
insert\_data\_button= c.CTkButton(frame, text="insert", command=lambda: insert\_student\_record(  
roll\_number\_entry.get(), name\_entry.get(), class\_entry.get(), dob\_entry.get(), gender\_entry.get(),  
phone\_entry.get(), guardian\_entry.get(), address\_entry.get()))  
  
insert\_data\_button.grid(row=2, column=0, columnspan=2, sticky="we", padx=10, pady=10)  
  
frame\_panel= c.CTkFrame(root)  
frame\_panel.grid(row=1, column=0, sticky="news", padx=10, pady=(0, 10))  
  
seg\_selector= c.CTkSegmentedButton(root, values=["Insert", "Update", "Delete"],  
command=lambda value, frame=frame\_panel: panel\_selector(value, frame))  
seg\_selector.grid(row=0, column=0, sticky="news", padx=10, pady=(10, 0))  
seg\_selector.set("Insert")  
  
panel\_selector("Insert", frame\_panel)  
  
frame\_headings= c.CTkFrame(root)  
frame\_headings.grid(row=2, column=0, sticky="news", padx=10, pady=(10, 0))  
  
labels = \  
 [c.CTkLabel(frame\_headings, text="Roll Number", width=150),  
c.CTkLabel(frame\_headings, text="Name", width=150),  
c.CTkLabel(frame\_headings, text="Class", width=130),  
c.CTkLabel(frame\_headings, text="Date Of Birth", width=150),  
c.CTkLabel(frame\_headings, text="Gender", width=140),  
c.CTkLabel(frame\_headings, text="Phone Number", width=130),  
c.CTkLabel(frame\_headings, text="Guardians Name", width=200),  
c.CTkLabel(frame\_headings, text="Address", width=150)]  
*# Applying characteristics to each label*for label in labels:  
label.grid(row=0, column=labels.index(label), sticky="we")  
  
*# to delete the data in realtime*def update\_table(previous\_data, table):  
new\_data= get\_all\_data()  
  
if new\_data!= previous\_data: *# i.e. there is a change in data*table.destroy()  
previous\_data= new\_data  
if len(new\_data) >12:  
table = CTkTable(scrollable\_frame, row=len(new\_data), column=8, values=previous\_data)  
table.grid(sticky="nsew", columnspan=8)  
else:  
table = CTkTable(scrollable\_frame, row=12, column=8, values=previous\_data)  
table.grid(sticky="nsew", columnspan=8) *# to not change the number of rows*root.after(5000, update\_table, previous\_data, table) *# pass the reference, not the result of the function call*scrollable\_frame= c.CTkScrollableFrame(root, width=1200, height=400)  
scrollable\_frame.grid(row=3, column=0, sticky="news", padx=10, pady=(5, 10))  
  
initial\_data= get\_all\_data()  
  
table\_widget= CTkTable(scrollable\_frame, row=20, column=8, values=initial\_data)  
table\_widget.grid(sticky="nsew", columnspan=8)  
  
update\_table(initial\_data, table\_widget)  
  
center\_window(root)  
root.mainloop()

**Center\_window.py**

def center\_window(window):  
*# Get the screen resolution*screen\_width= window.winfo\_screenwidth()  
screen\_height= window.winfo\_screenheight()  
  
*# Get the window size*window.update\_idletasks() *# Ensure that the window size is updated*req\_width= window.winfo\_reqwidth()  
req\_height= window.winfo\_reqheight()  
  
*# Calculate the X and Y coordinates to center the window*x = (screen\_width- req\_width) // 2  
y = (screen\_height- req\_height) // 2  
  
*# Set the position of the window*window.geometry(f"+{x}+{y}")

**Login.py**

import tkinteras tk  
import customtkinteras c  
from Splash\_Screenimport show\_splash\_screen  
from Center\_windowimport center\_window  
from Backend.backendimport \*  
from Admin\_panelimport create\_admin\_window  
from Student\_panelimport create\_student\_panel  
  
c.set\_appearance\_mode("System") *# Modes: "System" (standard), "Dark", "Light"*c.set\_default\_color\_theme("dark-blue") *# Themes: "blue" (standard), "green", "dark-blue"*def change\_widgets():  
global user\_login\_choice  
  
*# to delete the current widgets*for widget in widget\_frame.winfo\_children():  
widget.grid\_remove()  
  
if user\_login\_choice.get() == 0:  
username\_entry.grid(row=0, column=1, padx=10, pady=[20, 5])  
username\_label.grid(row=0, column=0, padx=15, pady=[20, 5])  
roll\_no\_entry.grid(row=1, column=1, padx=10, pady=[5, 20])  
roll\_no\_label.grid(row=1, column=0, padx=15, pady=[5, 20])  
else:  
admin\_id\_entry.grid(row=1, column=1, padx=10, pady=[20, 5])  
admin\_id\_label.grid(row=1, column=0, padx=15, pady=[20, 5])  
password\_entry.grid(row=2, column=1, padx=10, pady=[5, 20])  
password\_label.grid(row=2, column=0, padx=15, pady=[5, 20])  
  
*# Call update\_idletasks to force an immediate update*login.update\_idletasks()  
  
  
def authenticate():  
if user\_login\_choice.get() == 0: *# login as a student*roll = roll\_no\_entry.get()  
username = username\_entry.get()  
data = get\_student\_data(roll)  
  
*# data can be either () or the student data*if data == (): *# if data is ()*print("No data found with the corresponding roll number")  
CTkMessagebox(title="Error", message="No data found for the Roll Number.", icon="cancel")  
else:  
login.destroy()  
create\_student\_panel(roll, username)  
  
  
  
elifuser\_login\_choice.get() == 1: *# login as admin*if admin\_id\_entry.get() == "admin" and password\_entry.get() == "root": *# if the id and pass are correct*login.destroy()  
create\_admin\_window()  
else: *# if the password and id is false*CTkMessagebox(title="Login Error",  
message="Oops! Incorrect login details. Please check and try again",  
icon="cancel", option\_1="Retry")  
else: *# don't know how this condition will be archived but still what if :)*CTkMessagebox(title="Login Error",  
message="Oops! Something went wrong ",  
icon="cancel", option\_1="Retry")  
  
  
show\_splash\_screen()  
  
login = c.CTk()  
login.resizable(False, False)  
  
title = c.CTkLabel(login, width=200, text="Login", font=("roboto", 36))  
title.grid(row=0, column=0, padx=200, pady=[20, 10])  
  
login\_frame = c.CTkFrame(login, corner\_radius=20)  
login\_frame.grid(row=1, column=0, padx=200, pady=[5, 20])  
  
user\_login\_choice = tk.IntVar(value=1)  
student = c.CTkRadioButton(login\_frame, text="Student", variable=user\_login\_choice, value=0, command=change\_widgets)  
student.grid(row=1, column=0, padx=10, pady=[20, 10])  
admin = c.CTkRadioButton(login\_frame, text="Admin", variable=user\_login\_choice, value=1, command=change\_widgets)  
admin.grid(row=2, column=0, padx=10, pady=10)  
  
*# Frame*widget\_frame = c.CTkFrame(login\_frame, corner\_radius=10)  
widget\_frame.grid(row=3, column=0, columnspan=2, padx=10, pady=10)  
  
*# login as a student*username\_label = c.CTkLabel(widget\_frame, text=" Username :")  
username\_entry = c.CTkEntry(widget\_frame, corner\_radius=10)  
roll\_no\_label = c.CTkLabel(widget\_frame, text="Roll Number:")  
roll\_no\_entry = c.CTkEntry(widget\_frame, corner\_radius=10)  
  
*# login as Admin*admin\_id\_label = c.CTkLabel(widget\_frame, text=" Admin id :")  
admin\_id\_entry = c.CTkEntry(widget\_frame, corner\_radius=10)  
password\_label = c.CTkLabel(widget\_frame, text=" Password :")  
password\_entry = c.CTkEntry(widget\_frame, show="\*", corner\_radius=10)  
  
*# Login Button*login\_button = c.CTkButton(login\_frame, text="Login", width=250, fg\_color="#138808",  
hover\_color="#0E6606", corner\_radius=50, command=authenticate)  
login\_button.grid(row=4, column=0, padx=10, pady=[50, 10], sticky="ew")  
  
change\_widgets()  
center\_window(login)  
  
login.mainloop()

**Splash\_screen.py**

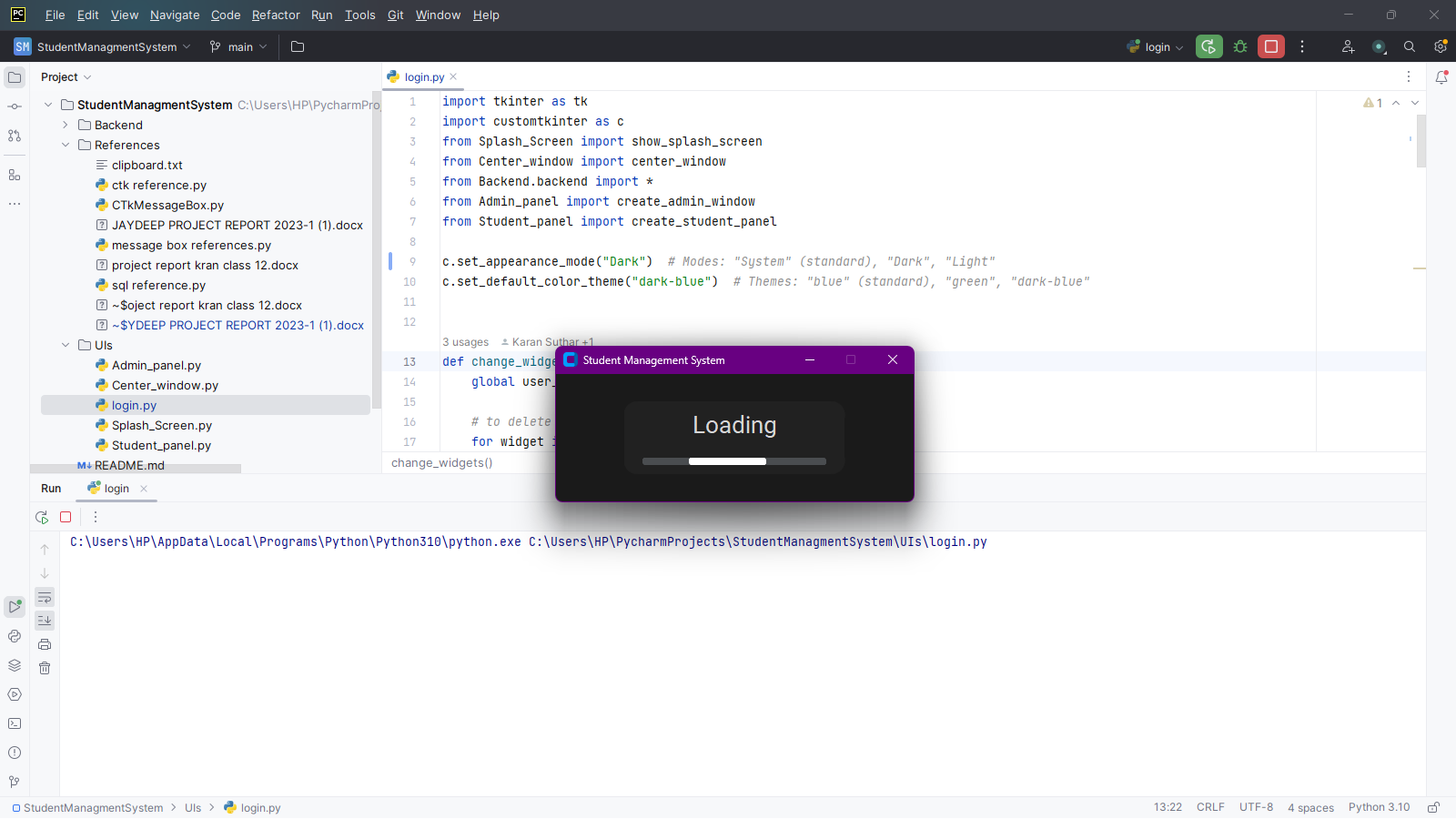
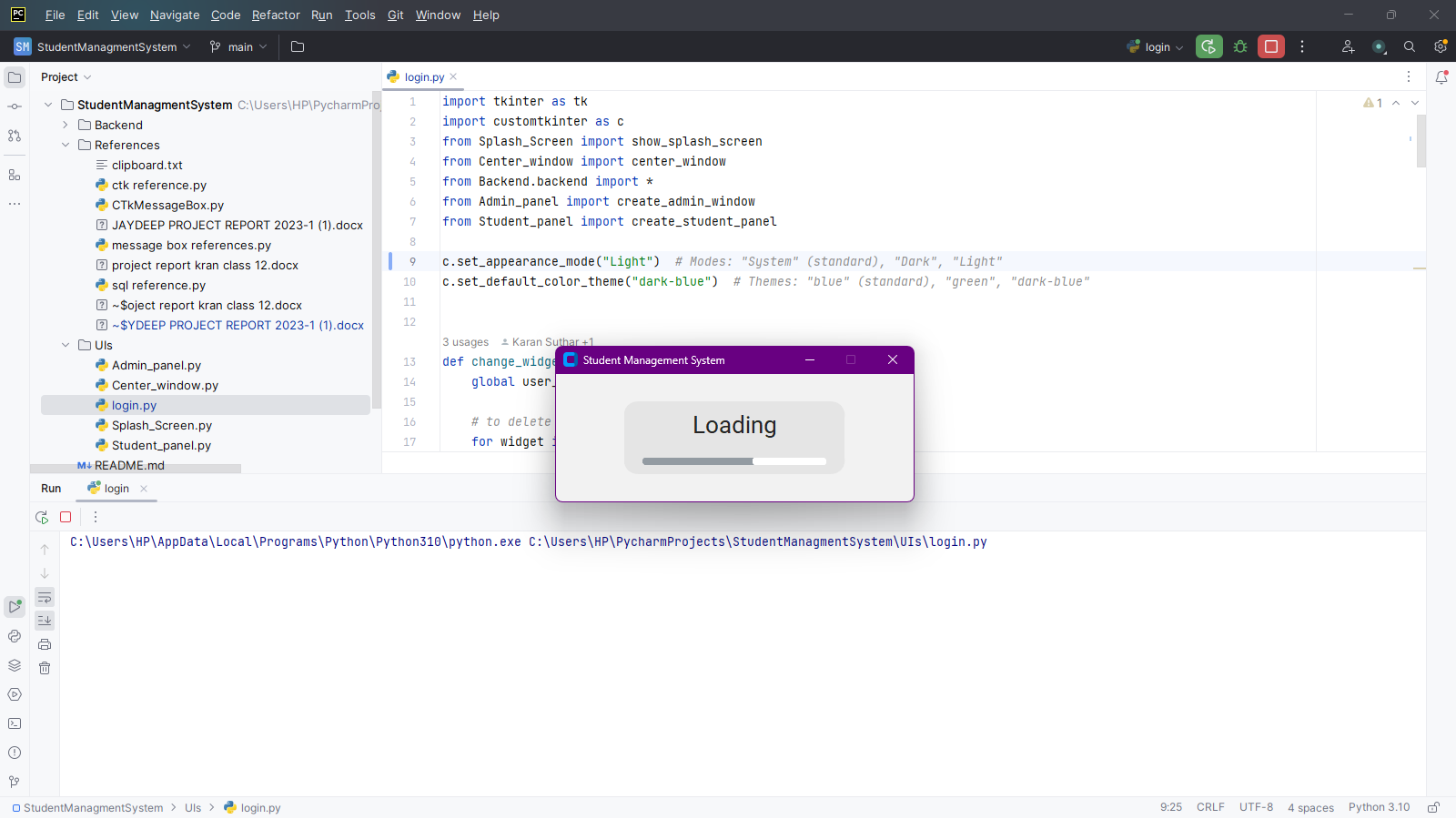
import customtkinteras c  
from Center\_windowimport center\_window  
  
c.set\_appearance\_mode("System") *# Modes: "System" (standard), "Dark", "Light"*c.set\_default\_color\_theme("dark-blue") *# Themes: "blue" (standard), "green", "dark-blue"*def show\_splash\_screen():  
splash\_root= c.CTk()  
splash\_root.resizable(False, False)  
splash\_root.title("Student Management System")  
  
frame = c.CTkFrame(splash\_root, corner\_radius=15)  
frame.grid(row=0, column=0, padx=75, pady=30)  
  
title = c.CTkLabel(frame, text="Loading", font=(("roboto", 26)))  
title.grid(row=0, column=0, padx=75, pady=10)  
  
*# Create and place the custom progress bar widget*progressbar\_1 = c.CTkProgressBar(frame, progress\_color="#ffffff")  
progressbar\_1.grid(row=1, column=0, padx=20, pady=(10, 10), sticky="ew")  
progressbar\_1.configure(mode="indeterminate")  
progressbar\_1.start()  
  
*# to kill the activity after 3 seconds*splash\_root.after(2000, lambda: splash\_root.destroy()) *# Use lambda to delay the execution  
  
 # Centering the window*center\_window(splash\_root)  
  
*# Start the Tkinter event loop for the splash screen*splash\_root.mainloop()

**Student\_panel.py**

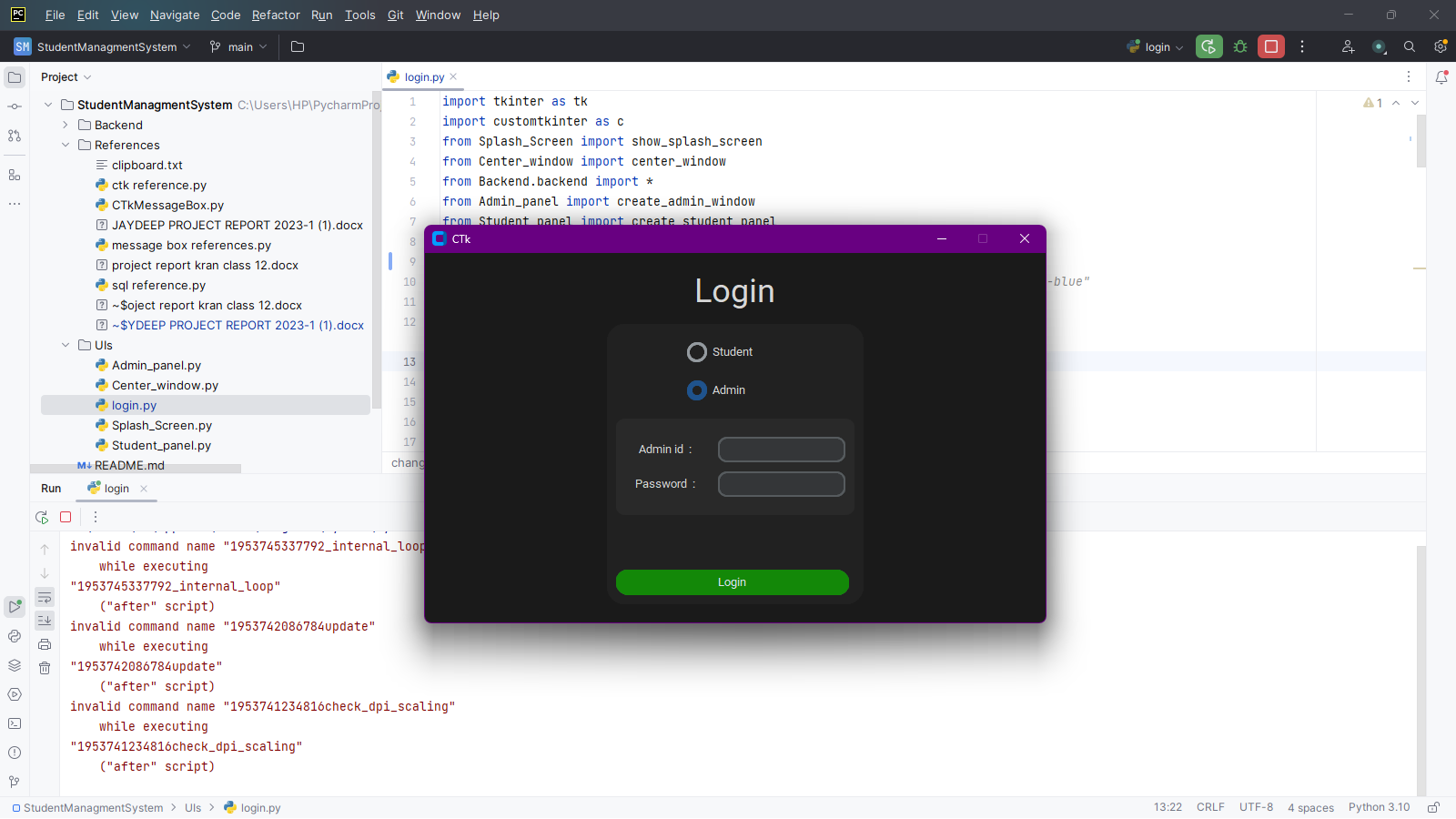
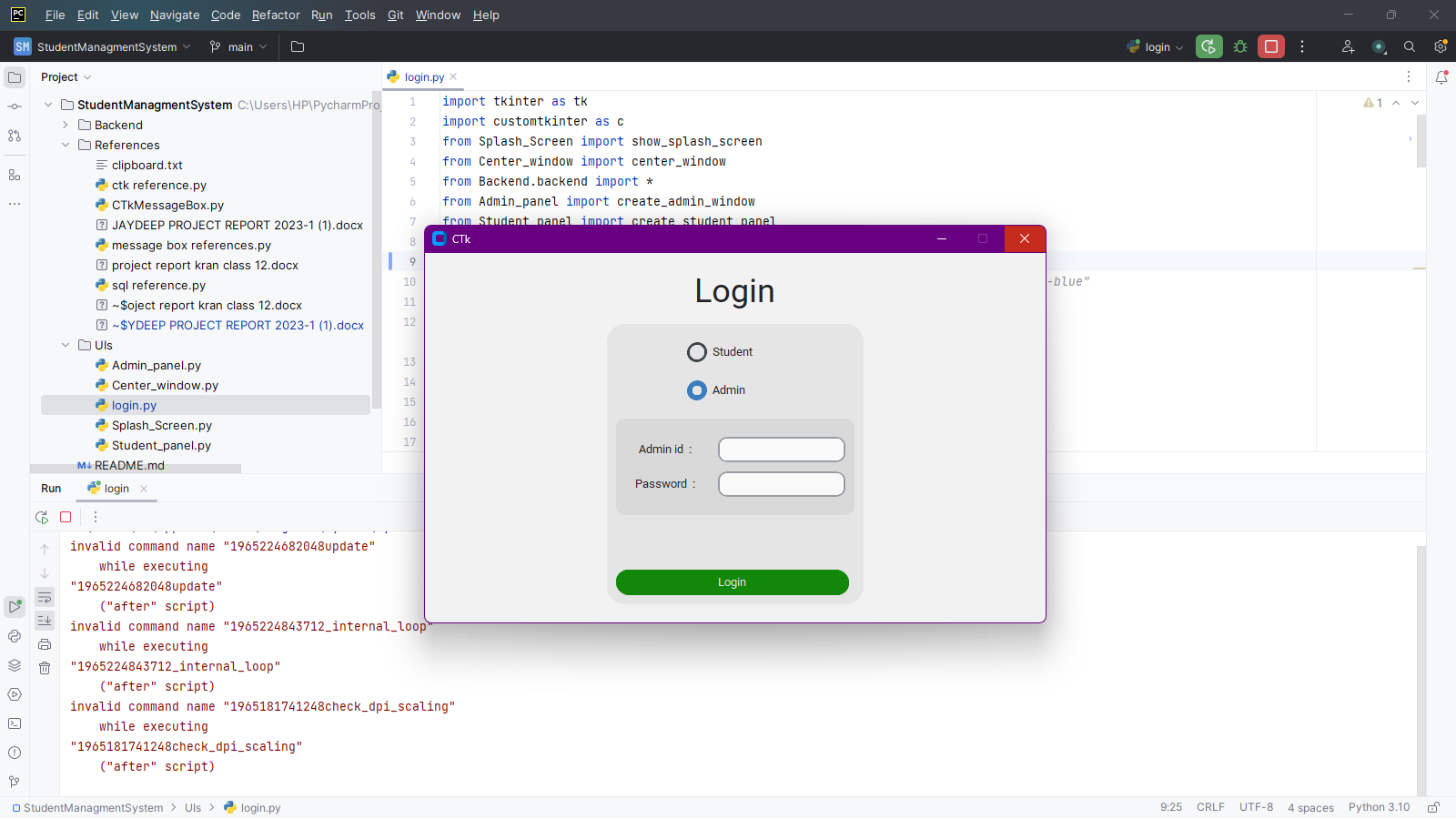
*# Import necessary modules*import customtkinteras c  
from Center\_windowimport center\_window  
from Backend.backendimport get\_student\_data*# Import your backend function here*def create\_student\_panel(roll\_number, username):  
root = c.CTk()  
root.title("Student Panel")  
root.resizable(False, False)  
  
*# Function to display student information*def display\_student\_info():  
student\_data= get\_student\_data(roll\_number)  
  
if student\_data:  
*# Clear previous data*for widget in labels\_frame.winfo\_children():  
widget.destroy()  
  
for widget in values\_frame.winfo\_children():  
widget.destroy()  
  
*# Display student information in the frames*labels = ["Roll Number", "Name", "Class", "Date Of Birth", "Gender", "Phone Number",  
"Guardian Name", "Address"]  
for index, label\_textin enumerate(labels):  
label = c.CTkLabel(labels\_frame, text=label\_text)  
label.grid(row=index, column=0, sticky="w", padx=10, pady=(10, 0))  
  
value = c.CTkLabel(values\_frame, text=str(student\_data[index]))  
value.grid(row=index, column=1, sticky="w", padx=10, pady=(10, 0))  
  
else:  
*# Display a message if no data is found for the provided roll number*message\_label= c.CTkLabel(labels\_frame, text="No data found for the given roll number.")  
message\_label.grid(row=0, column=0, sticky="w", padx=10, pady=(10, 0))  
  
*# Create UI elements*username\_label= c.CTkLabel(root, text=f"Username: {username}")  
username\_label.grid(row=0, column=0, columnspan=2, sticky="w", padx=10, pady=(20, 0))  
  
labels\_frame= c.CTkFrame(root)  
labels\_frame.grid(row=1, column=0, sticky="w", padx=10, pady=(10, 10))  
  
values\_frame= c.CTkFrame(root)  
values\_frame.grid(row=1, column=1, sticky="w", padx=10, pady=(10, 10))  
  
roll\_number\_label= c.CTkLabel(labels\_frame, text="Roll Number:")  
roll\_number\_label.grid(row=0, column=0, sticky="w", padx=10, pady=(10, 0))  
  
roll\_number\_value= c.CTkLabel(values\_frame, text=str(roll\_number))  
roll\_number\_value.grid(row=0, column=0, sticky="w", padx=10, pady=(10, 0))  
  
*# Call the function to display student information*display\_student\_info()  
  
center\_window(root)  
root.mainloop()

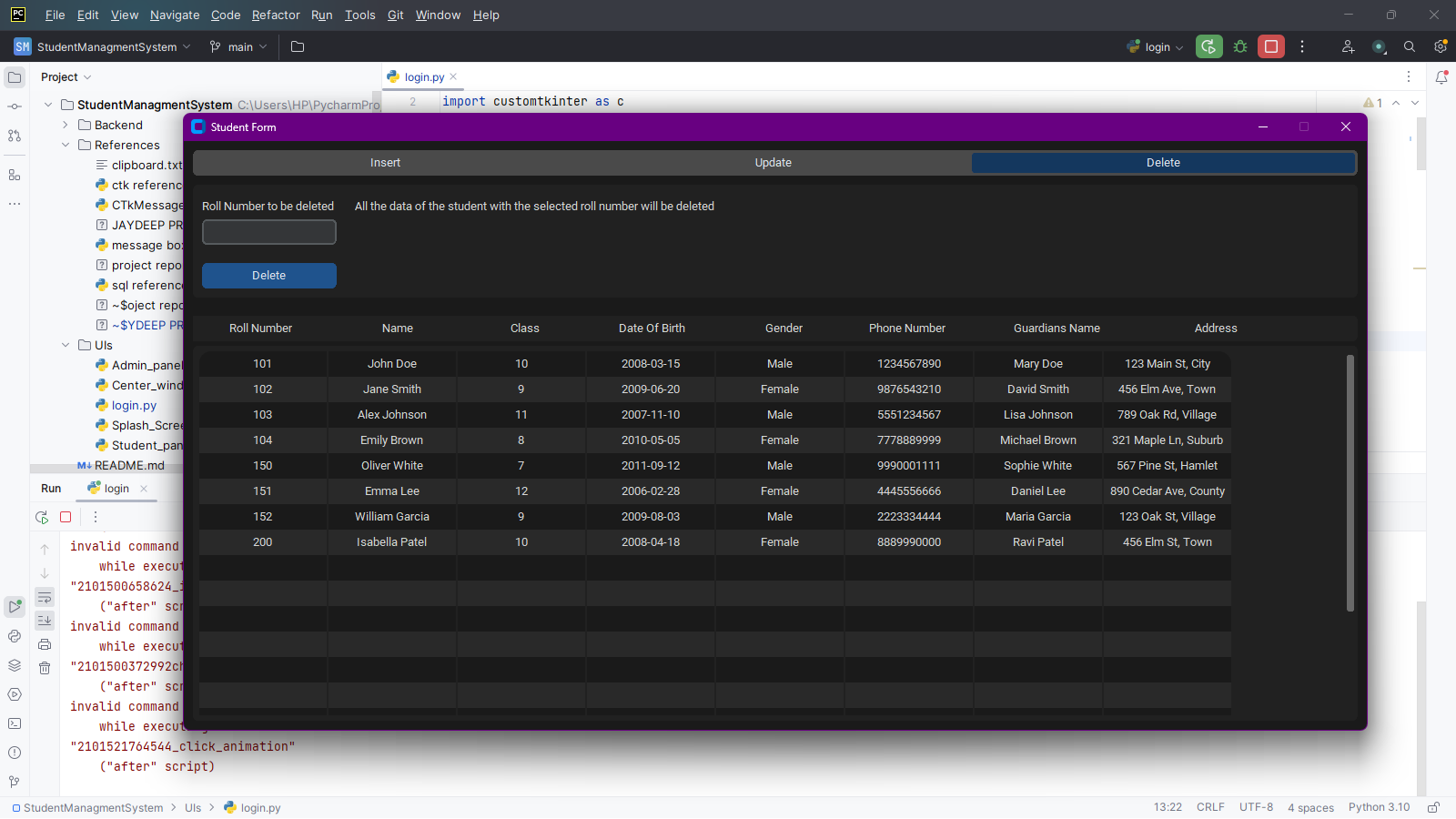
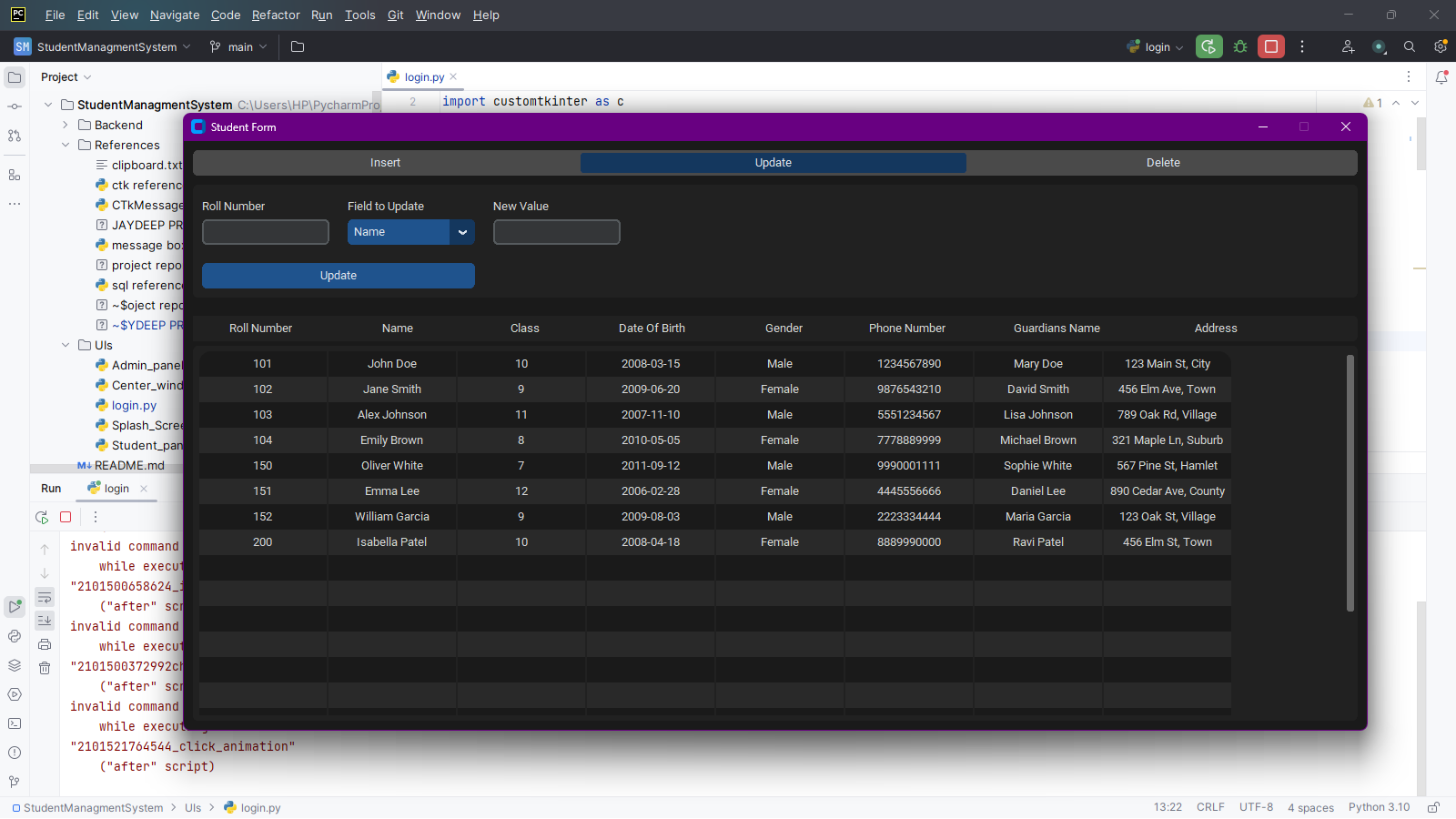
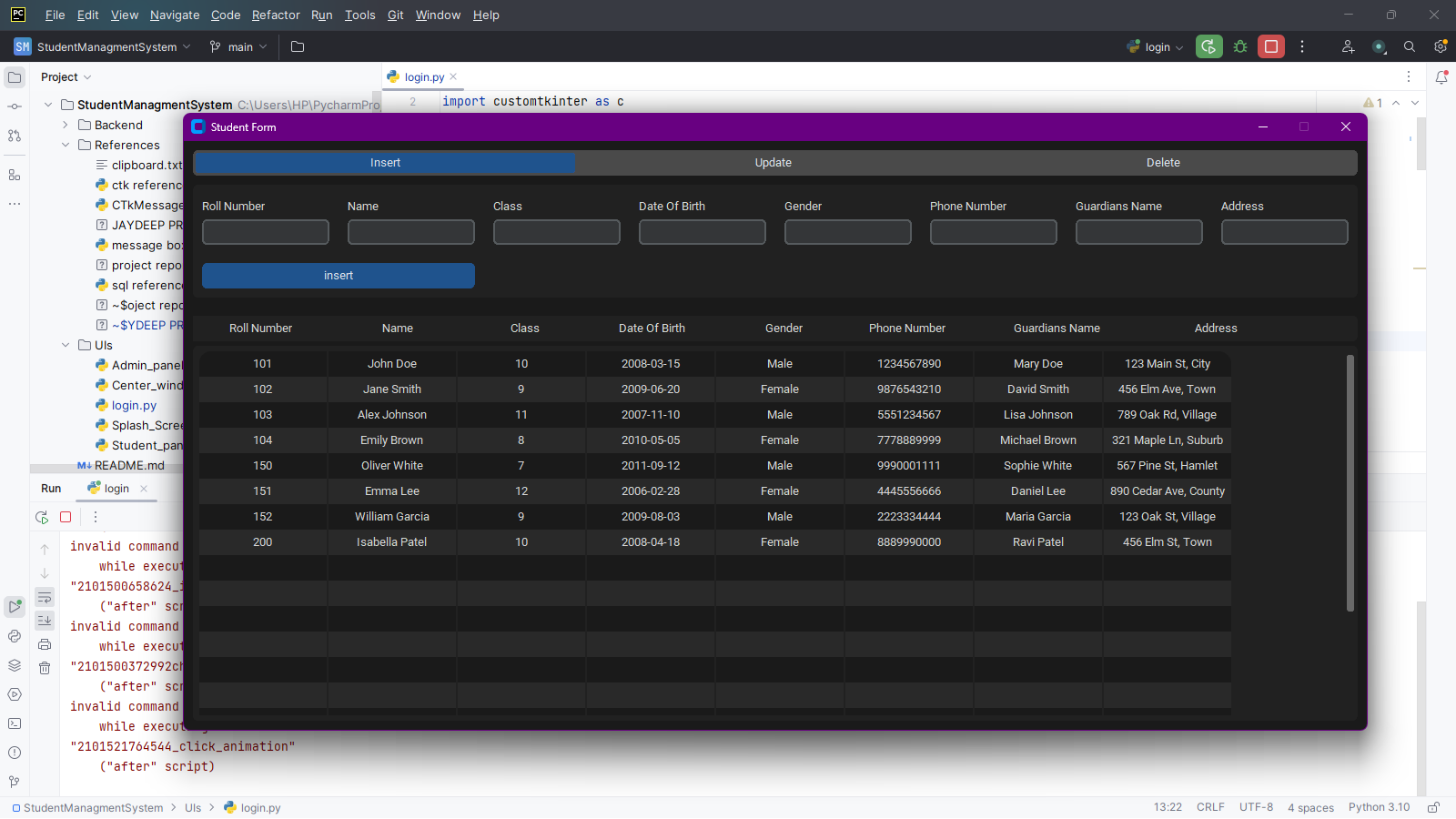
**OUTPUTS**

**Splash\_screen.py**

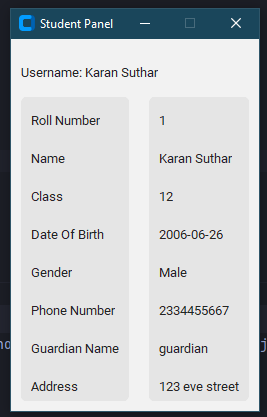
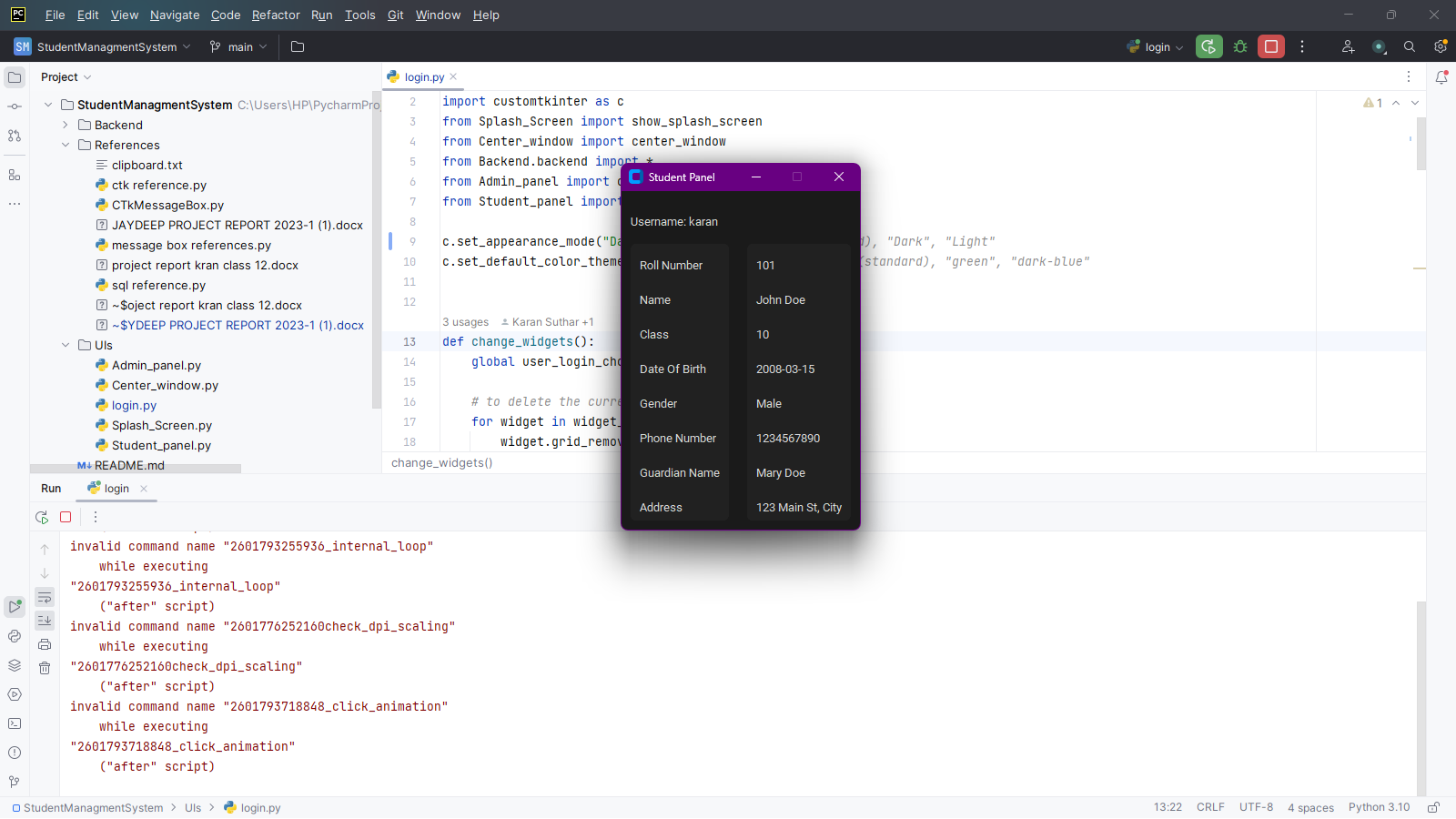
****

**Login.py**

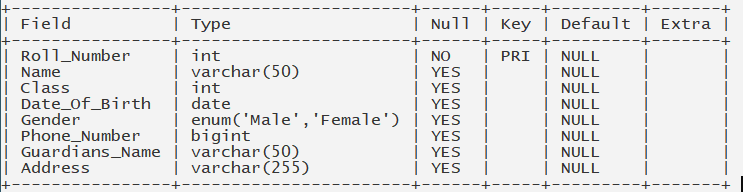
****

**Admin\_panel.py**

**Student\_panel.py**

****

**TABLE STRUCTURE**

****

This table structure represents a database table with the following fields:

1. **Roll\_Number**: This is an integer field and is the primary key of the table. It cannot be NULL.
2. **Name**: This is a variable character field with a maximum length of 50 characters. It can be NULL.
3. **Class**: This is an integer field. It can be NULL.
4. **Date\_Of\_Birth**: This is a date field. It can be NULL.
5. **Gender**: This is an enumerated field which can take the values ‘Male’ or ‘Female’. It can be NULL.
6. **Phone\_Number**: This is a big integer field. It can be NULL.
7. **Guardians\_Name**: This is a variable character field with a maximum length of 50 characters. It can be NULL.
8. **Address**: This is a variable character field with a maximum length of 255 characters. It can be NULL.

**BIBLIOGRAPHY**

1. [www.greeksforgreeks.org](http://www.greeksforgreeks.org/)
2. [github.com/TomSchimansky/ CustomTkinter](https://github.com/TomSchimansky/CustomTkinter)
3. www.w3schools.com/MySQL/default.asp

2.) Help from textbook i.e. computer science class XII by Preeti Arora.

3.) Help from Subject Teacher.