

**Software Engineering Project**

**School**

Submitted By:

|  |  |
| --- | --- |
| Name | Group - number |
| Rana Awad Ahmed Mohamed | G1 IS |
| Menna Ibrahim Samy Mohamed Rahmo | G2 IS |
| Hisham Ayman Atia Mohamed(Team Leader) | G2 IS |
| Mohamed Ramadan | G2 Is |
| Abdelrhman Essam Abdelhamid | G2 IS |

Presented to:

* Dr. Marwa
* T.A Riham Fares
* T.A Abdelrahman Essam
* T.A Nada El Sayed
* T.A Nada Hussein
* T.A Dina Ahmed

## TABLE OF CONTENTS

Contents

[TABLE OF CONTENTS 2](#_Toc153804871)

[INTRODUCTION 2](#_Toc153804872)

[Disadvantages: 4](#_Toc153804873)

[REQUIREMENTS OF PROJECT 5](#_Toc153804874)

[FUNCTIONAL REQUIREMENTS: 5](#_Toc153804875)

[NON-FUNCTIONAL REQUIREMENTS: 8](#_Toc153804876)

[PROJECT DIAGRAMS 9](#_Toc153804877)

[USE CASE DIAGRAM: 9](#_Toc153804878)

[USE CASE : 10](#_Toc153804879)

[MISUSE CASE : 11](#_Toc153804880)

[Class Diagram : 12](#_Toc153804881)

[Activity Diagram: 13](#_Toc153804882)

[Sequence Diagram: 15](#_Toc153804883)

[Design 16](#_Toc153804884)

[PROTOTYPE: 16](#_Toc153804885)

[Testing 18](#_Toc153804886)

[TESTCASE 18](#_Toc153804887)

[Implementation 21](#_Toc153804889)

[Library Used : 21](#_Toc153804890)

[PlainText : 23](#_Toc153804891)

[- DatabaseManager 23](#_Toc153804892)

[Code Implementation : 24](#_Toc153804911)

[Database Implementation : 36](#_Toc153804912)

[1. Login Table 36](#_Toc153804913)

[2. Teacher Table 36](#_Toc153804914)

[3. Student Table 37](#_Toc153804915)

[4. New\_User Table 37](#_Toc153804916)

[5. Typee Table 38](#_Toc153804917)

[6. Courses Table 38](#_Toc153804918)

[Code table : 39](#_Toc153804919)

[TOOLS USED IN THIS PROJECT ARE : 42](#_Toc153804920)

# INTRODUCTION

The school website system provides an easy and effective way in terms of communication between students and the school administration, including teachers, without the need to be present in the school, as it is a distinctive way to provide many tasks for students and their parents, as well as teachers, as it allows parents to identify all matters related to the level of their children and also allows access to all reports, attendance rate and exam results for students.Advantages:

1. Flexibility

Among the many benefits of an online learning, you’ll find virtual classrooms are great for people who are advancing their education while working.

2. Reduced Costs

Education can be expensive, but virtual learning can provide a number of ways for students to save.

3. More Free Time

Because your schedule isn’t dictated by classes, you can spend more time doing the things you want.

4. Increased Course Variety

Another reason why online school is better for some is the increased variety of education options.

5.Personalized Education

Students who find their focus suffers from classroom activity may benefit from online classes.

## Disadvantages:

**.** One of the biggest problems with any computerized system is the potential for a system crash. A corrupt hard drive, power issues, and other technical issues can result in the loss of needed data. At the least, teachers are interrupted when they are unable to access the data they need.

**.** There can be fraud and security problems

# REQUIREMENTS OF PROJECT

## FUNCTIONAL REQUIREMENTS:

**Functional requirements define the basic system behaviors, they are what the system must do or must not do and can be thought of in terms of how the system responds to inputs.**

**Functional requirements are features that allow the system to function as it was intended. Put another way, if the functional requirements are not met, the system will not work. Functional requirements are product features and focus on user requirements.**

**The System’s functional requirements are:**

**Student:**

**Login**:

Secure login with a unique student ID and password.

**Register**:

Ability for new students to register with necessary information, including personal details and contact information.

**View Courses**:

Access to a dashboard displaying the courses the student is enrolled in.

**View grades**:

View and track grades for assignments, exams, and overall performance in each course.

**Teacher**:

**Login:**

Secure login with a unique username and password.

**Register:**

Ability for new teachers to register with necessary information, including personal details and qualifications.

**View Courses:**

Access to a dashboard displaying the courses assigned to the teacher.

**Submit Grades:**

Input and update student grades for assignments, exams, and overall course performance.

**View Students:**

Access a list of students enrolled in each course.

**Database System:**

**Check validation:**

Validate input data to prevent errors and ensure data integrity.

**Create account**

**Update Account:**

Allow administrators to update teacher and student account information.

**Delete Account:**

Implement a secure process for administrators to delete teacher and student accounts.

**Admin:**

**Login:**

Secure login with a unique username and password for administrators.

**Delete user:**

Implement a secure process for administrators to delete teacher and student accounts.

**Update user:**

Allow administrators to update teacher and student account information.

**Manage Courses:**

Create, update, and delete courses within the school system.

## NON-FUNCTIONAL REQUIREMENTS:

**While functional requirements define what the system does or must not do, non-functional requirements specify how the system should do it. Non-functional requirements do not affect the basic functionality of the system (hence the name, non-functional requirements). Even if the non-functional requirements are not met, the system will still perform its basic purpose.**

The System’s functional requirements are:

* Performance
* Scalability
* Capacity
* Availability
* Reliability
* Recoverability
* Maintainability
* Serviceability
* Security
* Manageability
* Data Integrity
* Usability

# PROJECT DIAGRAMS

## USE CASE DIAGRAM:

**(USER):**

* Log in: This allows the customer to log into the system.
* Signup (Registration): this allows the customer to make an account to log into the system.
* Search: This allows the customer to search for what game he/she wants.
* View item: view a list of the available products (games) that the customer should choose.

.

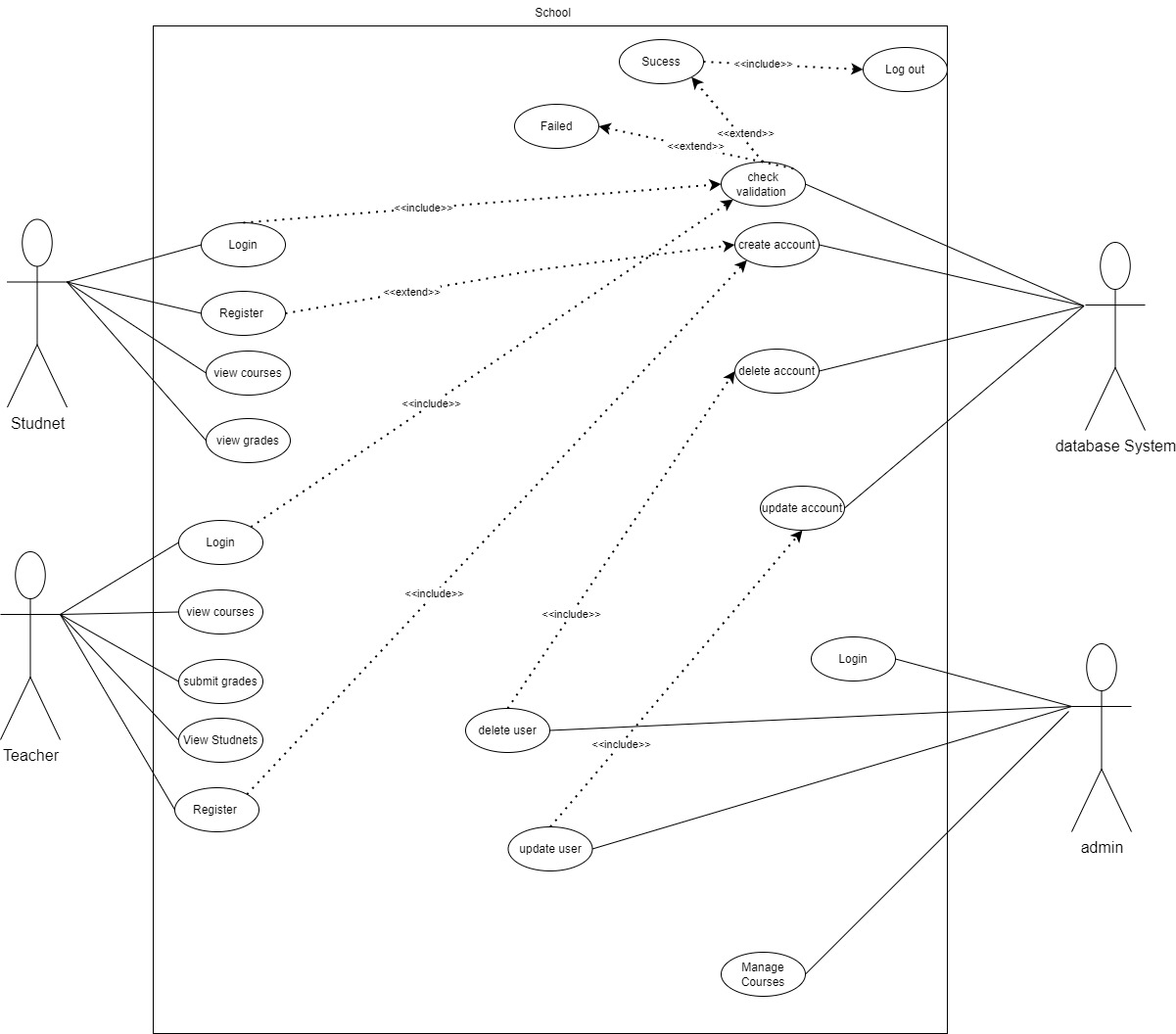
**(ADMIN):**

* Mange Category: which allows admin (Manger of Store) to make a lot of transactions like (add, update, and delete) categories.
* Log in: This allows the admin to log into the system.

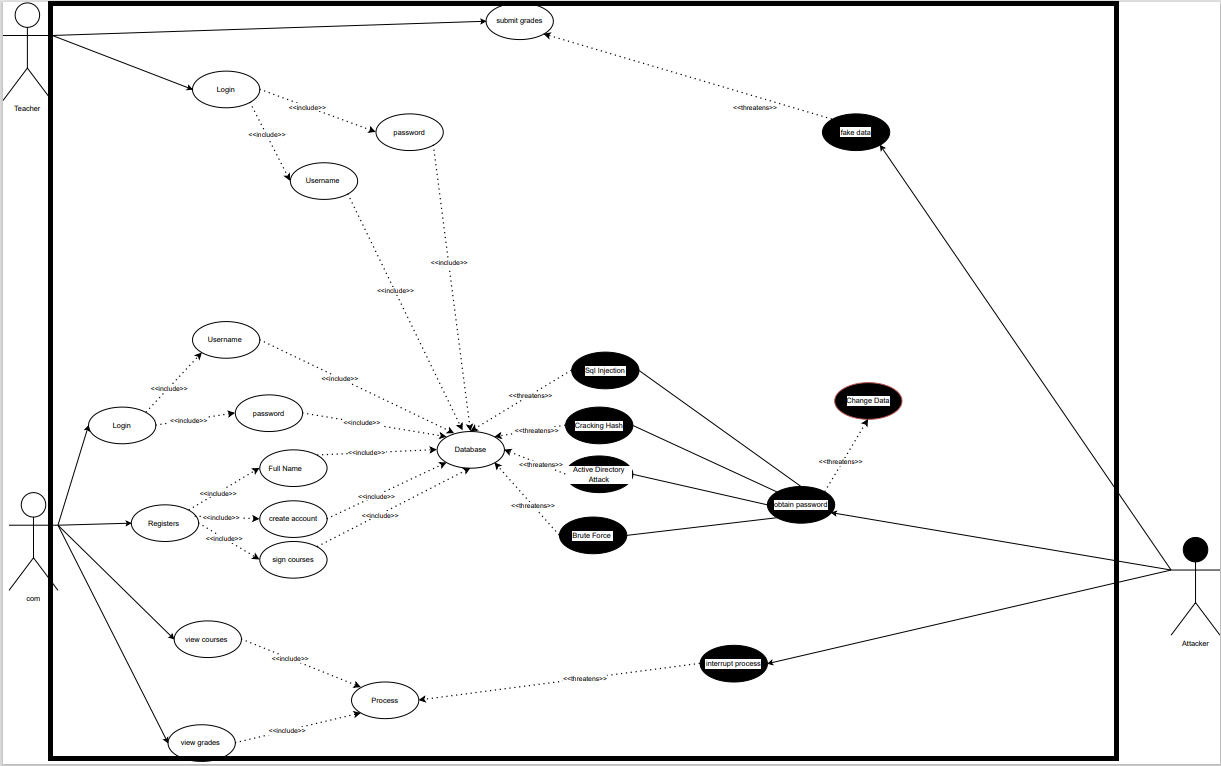
**(DATABASE):**

* Search product: which stores all these products in the database.
* Load data.
* Save data.

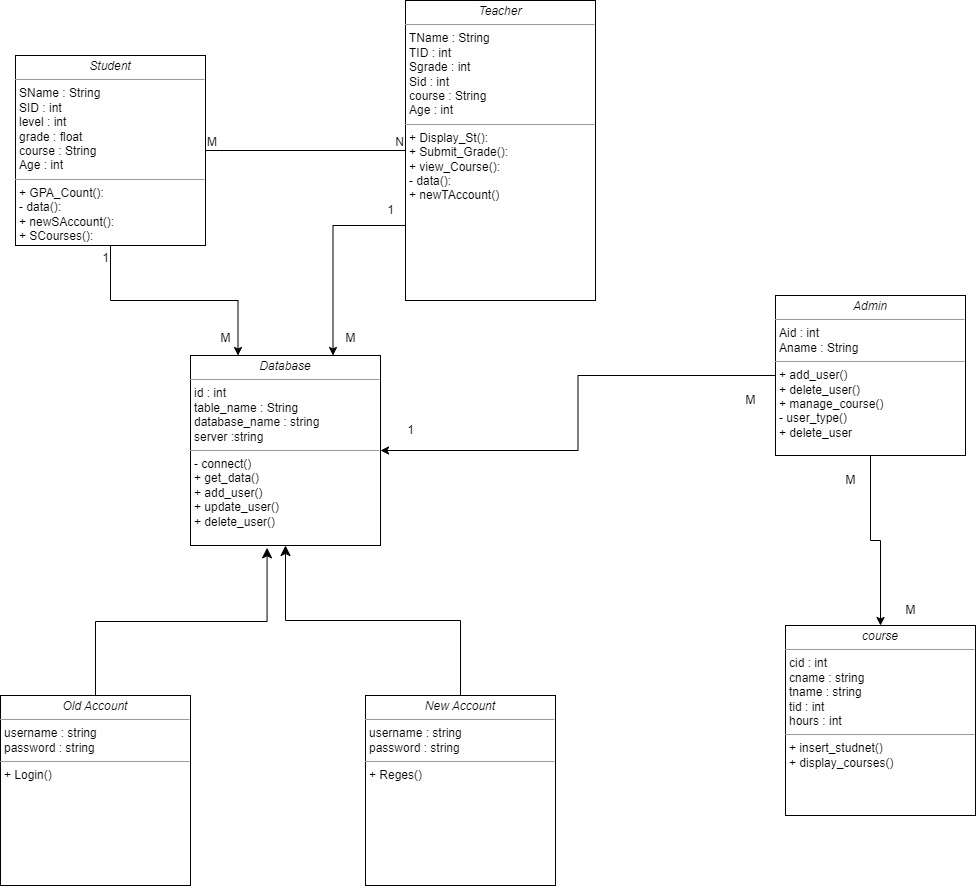
## USE CASE :

****

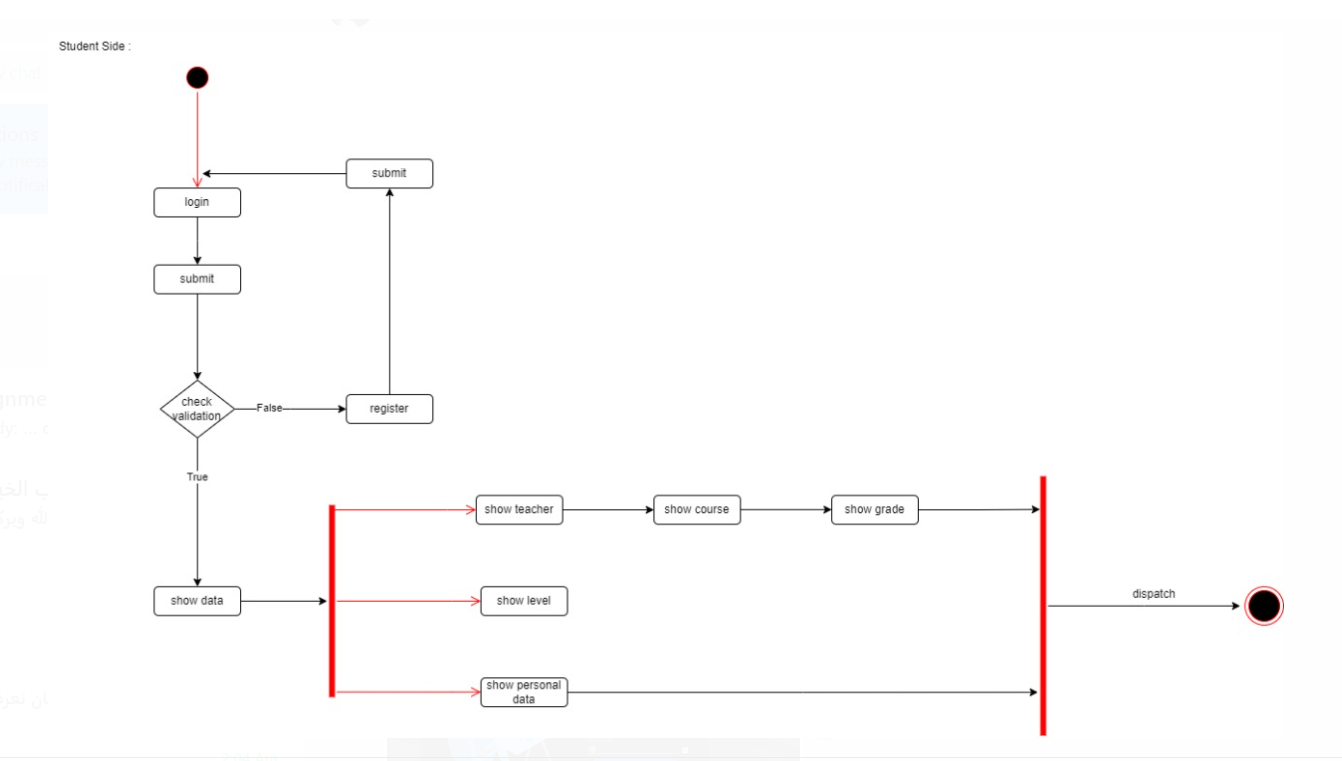
## MISUSE CASE :

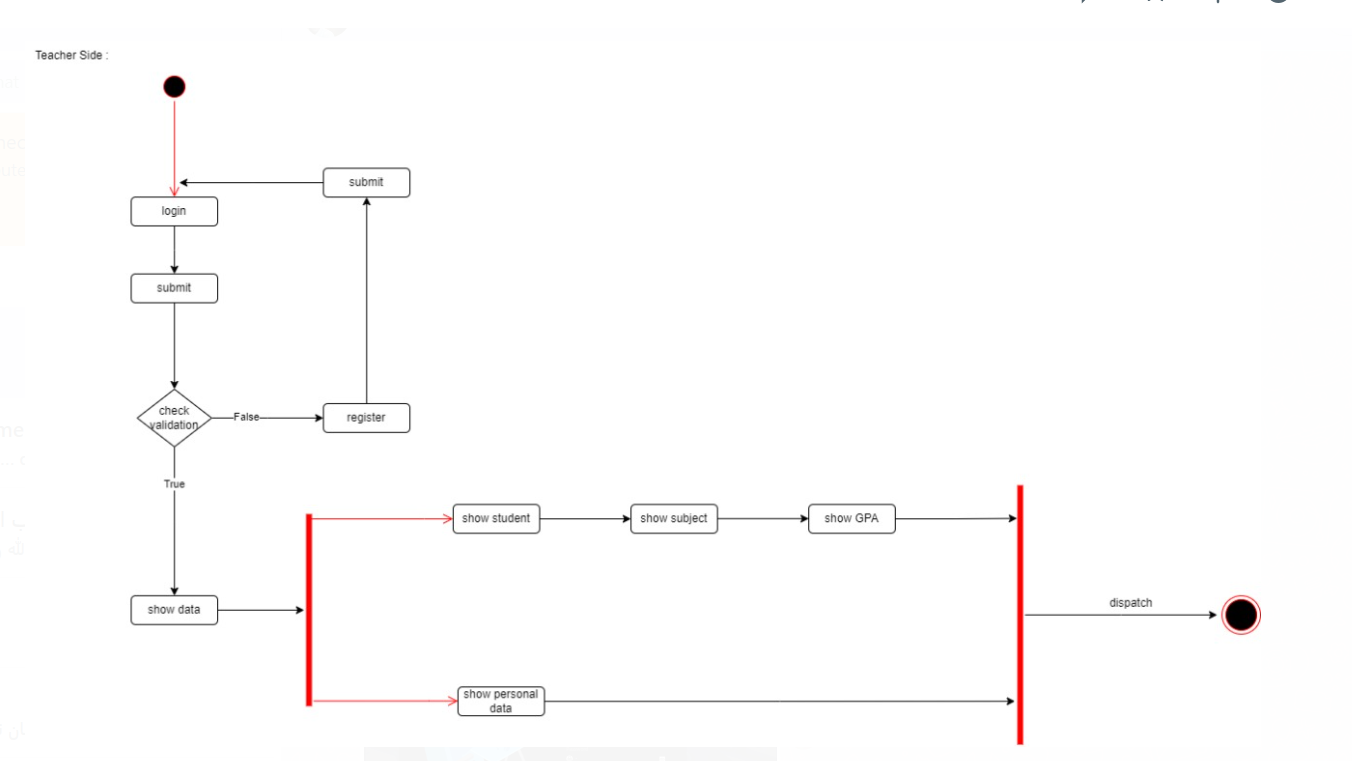


## Class Diagram :

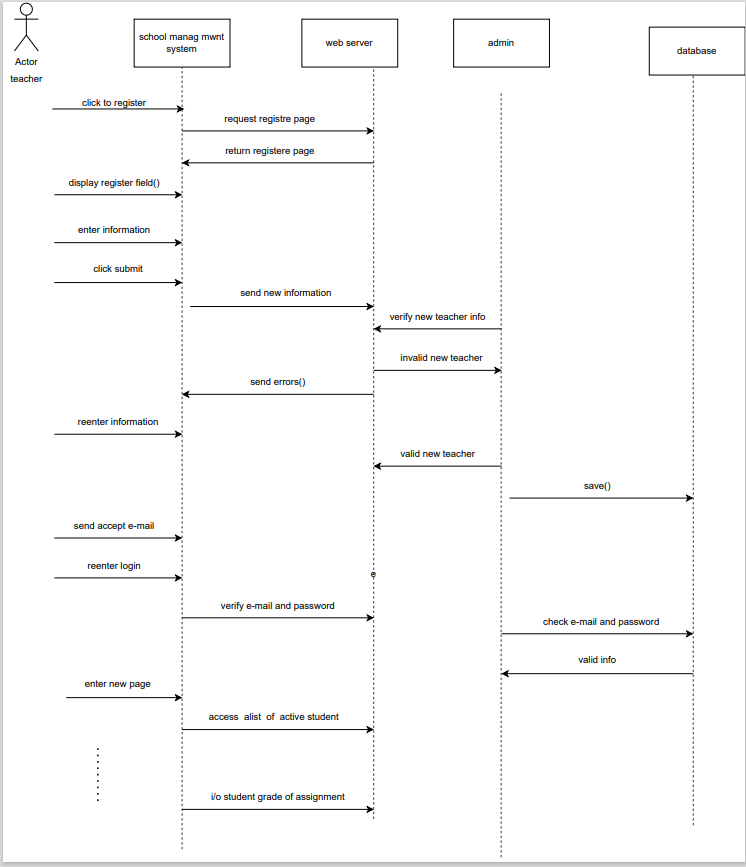


## Activity Diagram:

****

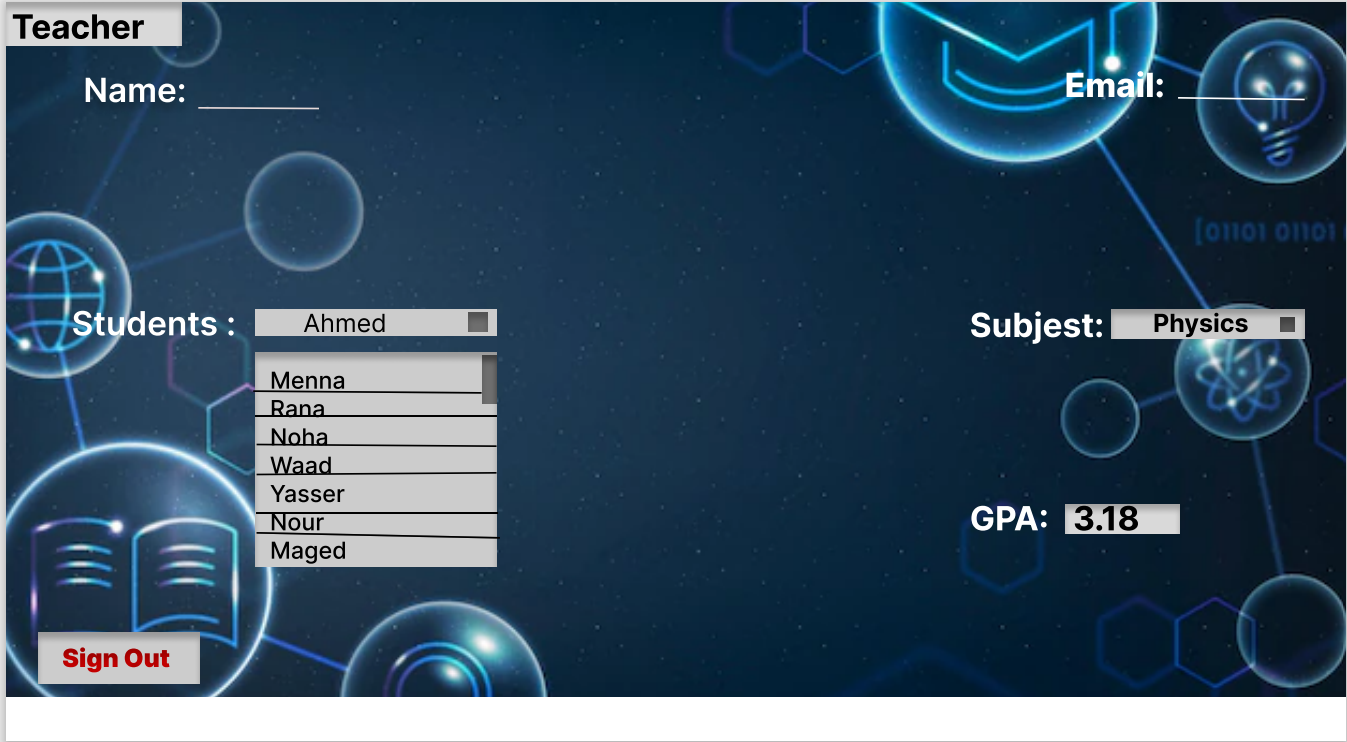
****

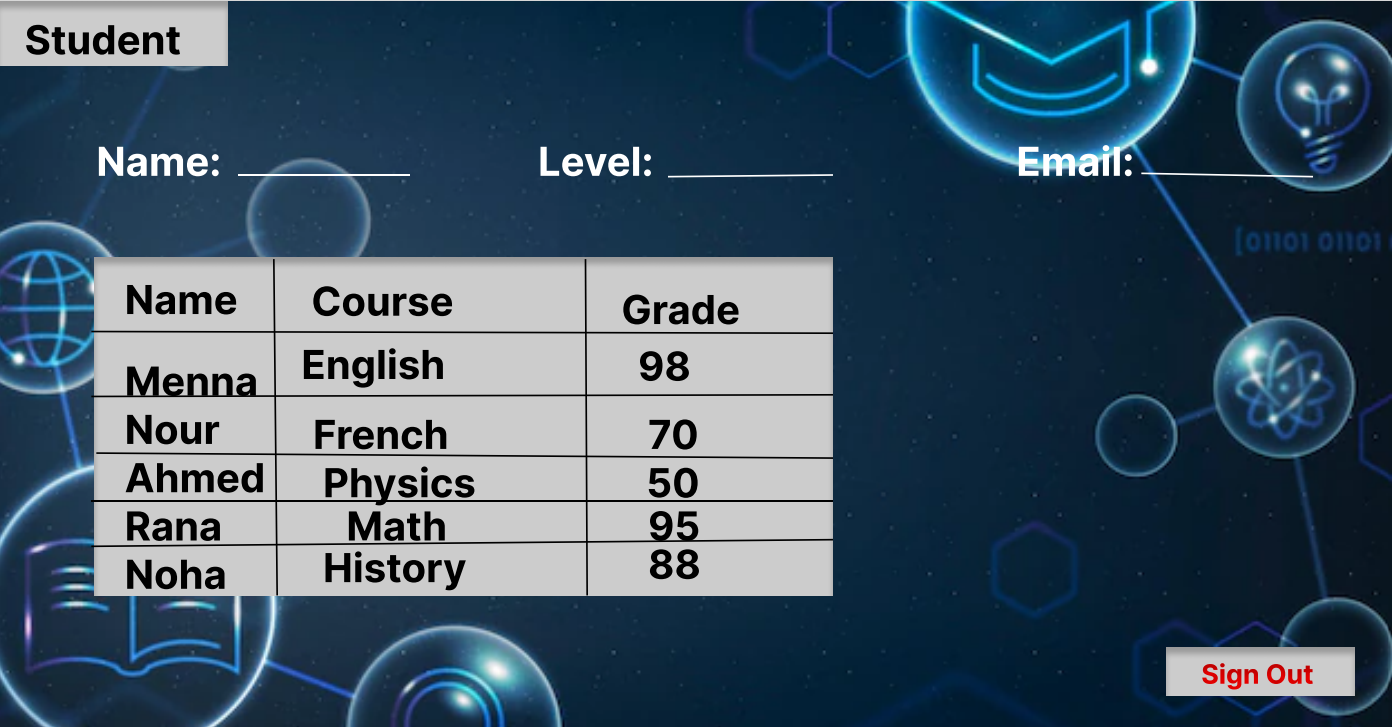
## Sequence Diagram:



# Design

## PROTOTYPE:

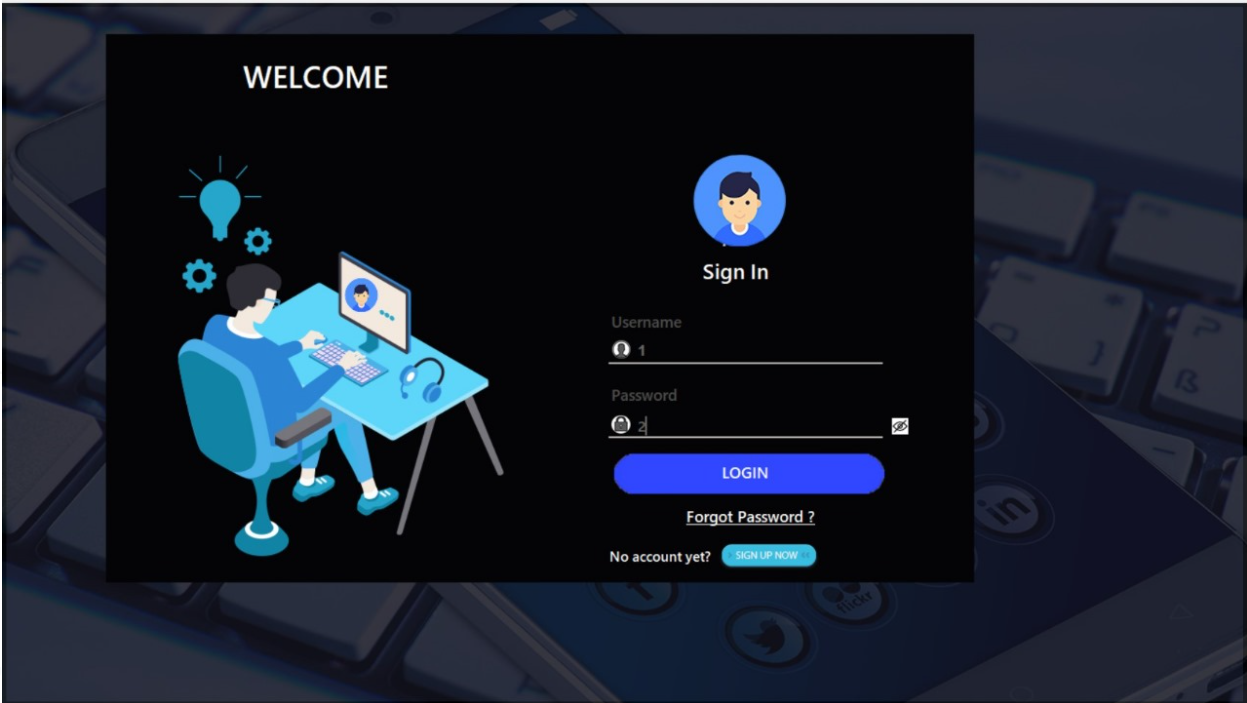


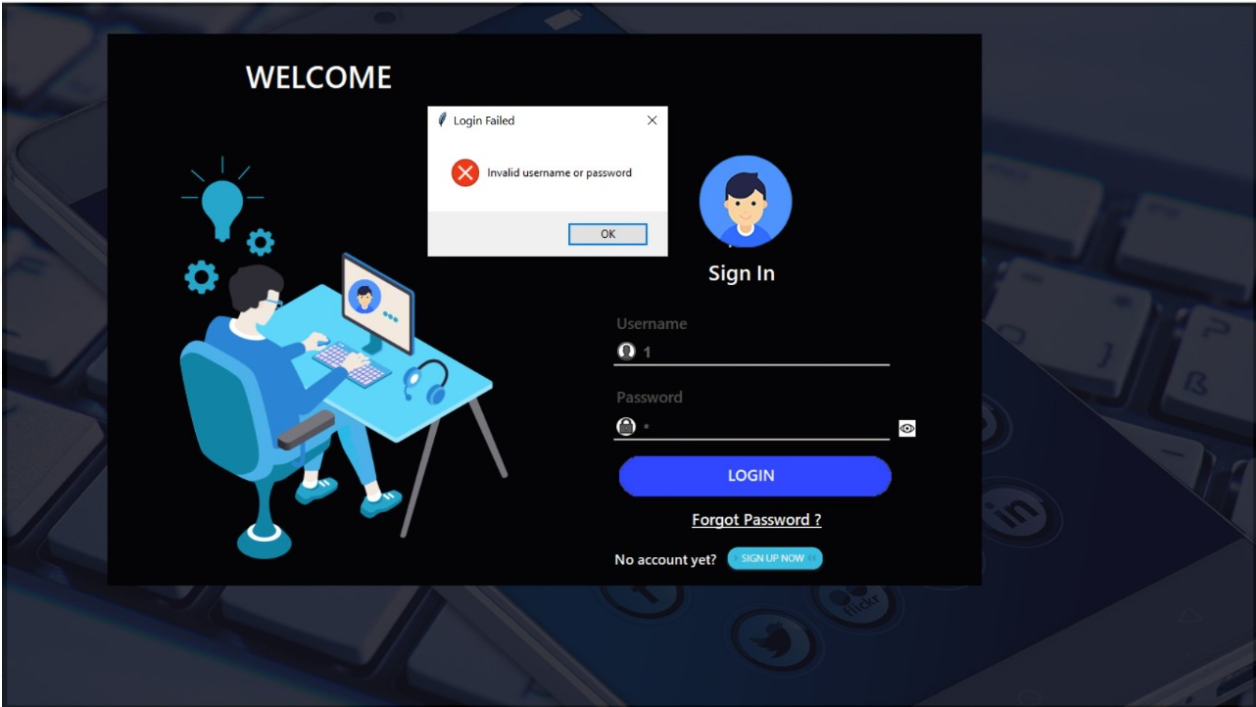


# Testing

## TESTCASE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GPA\_1 | **Verify that a user can successfully login with valid credentials.** | | **Choose student**  **Choose subject** | **Choose Ahmed**  **Chose english** | **The GPA will appear 3.18** |
| GPA\_2 | **Verify that a user can successfully login with valid credentials.** | **Choose student**  **Choose subject** | | **Choose Rana**  **Chose math** | **The GPA will appear 3.5** |





# 

# Implementation

## Library Used :

Tkinter Library:

Tkinter is the standard GUI toolkit for Python. It provides a set of tools for building graphical user interfaces.

It allows the creation of windows, buttons, menus, and other GUI elements to interact with the user.

The code uses Tkinter for creating GUI elements such as windows, labels, buttons, and more.

PIL (Python Imaging Library) Library:

PIL, now known as Pillow, is a library for opening, manipulating, and saving many different image file formats.

It provides image processing capabilities such as resizing, cropping, and applying various filters to images.

In the code, PIL is used to work with images, load image files, and display images within the GUI using Tkinter.

Pyodbc:

Pyodbc is a Python module that provides access to ODBC (Open Database Connectivity) databases.

It allows Python programs to connect to a wide range of database management systems using ODBC drivers.

The code uses Pyodbc to establish a connection to a SQL Server database and execute SQL queries.

Random:

The random module provides functions for generating random numbers, selecting random elements from a list, and shuffling sequences.

It is commonly used for tasks such as generating random passwords, simulating random events, or shuffling data.

In the code, the random module may be used for tasks that require randomization or selection of random elements.

## PlainText :

# - DatabaseManager

# - \_\_init\_\_(self)

# - execute\_query(self, query, params=None)

# - close\_connection(self)

# - SignupPage

# - \_\_init\_\_(self, parent\_window)

# - LoginPage

# - \_\_init\_\_(self, window)

# - open\_teacher\_window(self, username)

# - open\_student\_window(self, username)

# - get\_teacher\_info(self, username)

# - set\_background\_image(self, window)

# - show(self)

# - hide(self)

# - validate\_login(self)

# - open\_signup\_window(self)

# - open\_main\_window(self)

# - page(self)

# - close\_connection(self)

# Code Implementation :

from tkinter import \*  
from PIL import ImageTk, Image  
import pyodbc  
from tkinter import messagebox  
from tkinter import Tk, Canvas, Toplevel, Label, Listbox, Scrollbar, Button, END, Frame  
import random  
import tkinter as tk  
class DatabaseManager:  
 def \_\_init\_\_(self):  
 self.connection = pyodbc.connect(  
 'DRIVER={SQL Server};'  
 'SERVER=SAMSEPI0L;'  
 'DATABASE=School\_Soft;'  
 'Trusted\_Connection=yes;'  
 )  
  
 self.cursor = self.connection.cursor()  
  
 def execute\_query(self, query, params=None):  
 try:  
 if params:  
 self.cursor.execute(query, params)  
 else:  
 self.cursor.execute(query)  
 result = self.cursor.fetchone()  
 return result  
 except Exception as e:  
 print(f"Error executing query: {e}")  
 return None  
  
 def close\_connection(self):  
 self.cursor.close()  
 self.connection.close()  
class SignupPage:  
 def \_\_init\_\_(self, parent\_window):  
 self.signup\_window = Toplevel(parent\_window)  
 self.signup\_window.geometry('500x300')  
 self.signup\_window.title('Sign Up')  
  
class LoginPage:  
 def \_\_init\_\_(self, window):  
 self.window = window  
 self.window.geometry('1166x718')  
 self.window.resizable(0, 0)  
 self.window.state('zoomed')  
 self.window.title('Login Page')  
  
 self.bg\_frame = Image.open('images\\background1.png')  
 photo = ImageTk.PhotoImage(self.bg\_frame)  
 self.bg\_panel = Label(self.window, image=photo)  
 self.bg\_panel.image = photo  
 self.bg\_panel.pack(fill='both', expand='yes')  
  
 self.lgn\_frame = Frame(self.window, bg='#040405', width=950, height=600)  
 self.lgn\_frame.place(x=200, y=70)  
  
 self.txt = "WELCOME"  
 self.heading = Label(self.lgn\_frame, text=self.txt, font=('yu gothic ui', 25, "bold"), bg="#040405",  
 fg='white', bd=5, relief=FLAT)  
 self.heading.place(x=80, y=30, width=300, height=30)  
  
 self.side\_image = Image.open('images\\vector.png')  
 photo = ImageTk.PhotoImage(self.side\_image)  
 self.side\_image\_label = Label(self.lgn\_frame, image=photo, bg='#040405')  
 self.side\_image\_label.image = photo  
 self.side\_image\_label.place(x=5, y=100)  
  
 self.sign\_in\_image = Image.open('images\\hyy.png')  
 photo = ImageTk.PhotoImage(self.sign\_in\_image)  
 self.sign\_in\_image\_label = Label(self.lgn\_frame, image=photo, bg='#040405')  
 self.sign\_in\_image\_label.image = photo  
 self.sign\_in\_image\_label.place(x=620, y=130)  
 self.signup\_img = ImageTk.PhotoImage(file='images\\register.png')  
 self.signup\_button\_label = Button(self.lgn\_frame, image=self.signup\_img, bg='#98a65d', cursor="hand2",  
 borderwidth=0, background="#040405", activebackground="#040405",  
 command=self.open\_signup\_window)  
 self.signup\_button\_label.place(x=670, y=555, width=111, height=35)  
 self.sign\_in\_label = Label(self.lgn\_frame, text="Sign In", bg="#040405", fg="white",  
 font=("yu gothic ui", 17, "bold"))  
 self.sign\_in\_label.place(x=650, y=240)  
  
 self.username\_label = Label(self.lgn\_frame, text="Username", bg="#040405", fg="#4f4e4d",  
 font=("yu gothic ui", 13, "bold"))  
 self.username\_label.place(x=550, y=300)  
  
 self.username\_entry = Entry(self.lgn\_frame, highlightthickness=0, relief=FLAT, bg="#040405", fg="#6b6a69",  
 font=("yu gothic ui ", 12, "bold"), insertbackground='#6b6a69')  
 self.username\_entry.place(x=580, y=335, width=270)  
  
 self.username\_line = Canvas(self.lgn\_frame, width=300, height=2.0, bg="#bdb9b1", highlightthickness=0)  
 self.username\_line.place(x=550, y=359)  
  
 self.username\_icon = Image.open('images\\username\_icon.png')  
 photo = ImageTk.PhotoImage(self.username\_icon)  
 self.username\_icon\_label = Label(self.lgn\_frame, image=photo, bg='#040405')  
 self.username\_icon\_label.image = photo  
 self.username\_icon\_label.place(x=550, y=332)  
  
 self.lgn\_button = Image.open('images\\btn1.png')  
 photo = ImageTk.PhotoImage(self.lgn\_button)  
 self.lgn\_button\_label = Label(self.lgn\_frame, image=photo, bg='#040405')  
 self.lgn\_button\_label.image = photo  
 self.lgn\_button\_label.place(x=550, y=450)  
  
 self.login = Button(self.lgn\_button\_label, text='LOGIN', font=("yu gothic ui", 13, "bold"), width=25, bd=0,  
 bg='#3047ff', cursor='hand2', activebackground='#3047ff', fg='white',  
 command=self.validate\_login)  
 self.login.place(x=20, y=10)  
  
 self.forgot\_button = Button(self.lgn\_frame, text="Forgot Password ?",  
 font=("yu gothic ui", 13, "bold underline"), fg="white", relief=FLAT,  
 activebackground="#040405", borderwidth=0, background="#040405", cursor="hand2")  
 self.forgot\_button.place(x=630, y=510)  
  
 self.sign\_label = Label(self.lgn\_frame, text='No account yet?', font=("yu gothic ui", 11, "bold"),  
 relief=FLAT, borderwidth=0, background="#040405", fg='white')  
 self.sign\_label.place(x=550, y=560)  
  
 self.signup\_img = ImageTk.PhotoImage(file='images\\register.png')  
 self.signup\_button\_label = Button(self.lgn\_frame, image=self.signup\_img, bg='#98a65d', cursor="hand2",  
 borderwidth=0, background="#040405", activebackground="#040405")  
 self.signup\_button\_label.place(x=670, y=555, width=111, height=35)  
  
 self.password\_label = Label(self.lgn\_frame, text="Password", bg="#040405", fg="#4f4e4d",  
 font=("yu gothic ui", 13, "bold"))  
 self.password\_label.place(x=550, y=380)  
  
 self.password\_entry = Entry(self.lgn\_frame, highlightthickness=0, relief=FLAT, bg="#040405", fg="#6b6a69",  
 font=("yu gothic ui", 12, "bold"), show="\*", insertbackground='#6b6a69')  
 self.password\_entry.place(x=580, y=416, width=244)  
  
 self.password\_line = Canvas(self.lgn\_frame, width=300, height=2.0, bg="#bdb9b1", highlightthickness=0)  
 self.password\_line.place(x=550, y=440)  
  
 self.password\_icon = Image.open('images\\password\_icon.png')  
 photo = ImageTk.PhotoImage(self.password\_icon)  
 self.password\_icon\_label = Label(self.lgn\_frame, image=photo, bg='#040405')  
 self.password\_icon\_label.image = photo  
 self.password\_icon\_label.place(x=550, y=414)  
  
 self.show\_image = ImageTk.PhotoImage(file='images\\show.png')  
 self.hide\_image = ImageTk.PhotoImage(file='images\\hide.png')  
  
 self.show\_button = Button(self.lgn\_frame, image=self.show\_image, command=self.show, relief=FLAT,  
 activebackground="white", borderwidth=0, background="white", cursor="hand2")  
 self.show\_button.place(x=860, y=420)  
  
 def show(self):  
 self.hide\_button = Button(self.lgn\_frame, image=self.hide\_image, command=self.hide, relief=FLAT,  
 activebackground="white", borderwidth=0, background="white", cursor="hand2")  
 self.hide\_button.place(x=860, y=420)  
 self.password\_entry.config(show='')  
  
 def hide(self):  
 self.show\_button = Button(self.lgn\_frame, image=self.show\_image, command=self.show, relief=FLAT,  
 activebackground="white", borderwidth=0, background="white", cursor="hand2")  
 self.show\_button.place(x=860, y=420)  
 self.password\_entry.config(show='\*')  
  
 def validate\_login(self):  
 entered\_username = self.username\_entry.get()  
 entered\_password = self.password\_entry.get()  
  
 # Query the Login table to check if the credentials are valid  
 login\_query = "SELECT \* FROM Login WHERE username = ? AND password = ?"  
 login\_result = self.db\_manager.execute\_query(login\_query, (entered\_username, entered\_password))  
  
 if login\_result:  
 # Credentials are valid, now retrieve the user type from the typee table  
 type\_query = "SELECT type FROM typee WHERE username = ?"  
 type\_result = self.db\_manager.execute\_query(type\_query, (entered\_username,))  
  
 if type\_result:  
 user\_type = type\_result[0]  
  
 # Open a new window based on the user type  
 if user\_type == 'teacher':  
 self.open\_teacher\_window(entered\_username)  
 elif user\_type == 'student':  
 self.open\_student\_window(entered\_username)  
 else:  
 # Handle other user types as needed  
 print(f"Unknown user type: {user\_type}")  
 else:  
 print("User type not found for the given username")  
 else:  
 messagebox.showerror("Login Failed", "Invalid username or password")  
  
 def open\_teacher\_window(self, username):  
 # Create a new top-level window for the teacher  
 teacher\_window = Toplevel(self.window)  
 teacher\_window.title("Teacher Window")  
 teacher\_window.geometry('1166x718')  
  
 # Set the same background image as the login page  
 self.set\_background\_image(teacher\_window)  
  
 # Display the details from the login page  
 welcome\_label = Label(teacher\_window, text=f"Welcome, {username} (Teacher)!")  
 welcome\_label.pack()  
  
 # Create a heading label similar to the login page  
 heading\_label = Label(teacher\_window, text="WELCOME", font=('yu gothic ui', 25, "bold"), bg="#040405",  
 fg='white', bd=5, relief=FLAT)  
 heading\_label.place(x=(teacher\_window.winfo\_width() - 300) // 2, y=30, width=300, height=30)  
  
 # Calculate the size and position of the black box  
 box\_width = teacher\_window.winfo\_width() \* 0.6  
 box\_height = teacher\_window.winfo\_height() \* 0.6  
 box\_x = (teacher\_window.winfo\_width() - box\_width) // 2  
 box\_y = (teacher\_window.winfo\_height() - box\_height) // 2  
  
 # Create a new frame inside the teacher window  
 teacher\_frame = Frame(teacher\_window, bg='#040405', width=box\_width, height=box\_height)  
 teacher\_frame.place(x=box\_x, y=box\_y)  
  
 # Add elements inside the frame (modify as needed)  
 name\_and\_email\_label = Label(teacher\_frame, text="Name: Hesham Aymen Email: heshamaymen8@gmail.com",  
 fg='white', bg='#040405')  
 name\_and\_email\_label.pack(pady=10)  
  
 # Add space between name and email  
 space\_label = Label(teacher\_frame, text="", bg='#040405')  
 space\_label.pack()  
  
 # Create a listbox of students  
 student\_listbox = Listbox(teacher\_frame, selectmode=tk.SINGLE, height=5)  
 students = ["Student1", "Student2", "Student3", "Student4", "Student5"]  
 for student in students:  
 student\_listbox.insert(tk.END, student)  
 student\_listbox.pack(pady=10)  
  
 # Variable to store selected student  
 selected\_student = StringVar()  
  
 def display\_student\_gpa():  
 selected\_index = student\_listbox.curselection()  
 if selected\_index:  
 selected\_student.set(student\_listbox.get(selected\_index))  
 gpa\_label.config(text=f"GPA: {random.uniform(0, 4):.2f}")  
  
 # Button to display GPA  
 display\_gpa\_button = Button(teacher\_frame, text="Display GPA", command=display\_student\_gpa, bg='#040405', fg='white')  
 display\_gpa\_button.pack()  
  
 # Label to display selected student's GPA  
 gpa\_label = Label(teacher\_frame, text="", fg='white', bg='#040405')  
 gpa\_label.pack(pady=10)  
  
 # Start the main loop for the teacher window  
 teacher\_window.mainloop()  
  
 def open\_student\_window(self, username):  
 # Check the user type  
 type\_query = "SELECT type FROM typee WHERE username = ?"  
 type\_result = self.db\_manager.execute\_query(type\_query, (username,))  
  
 if type\_result and type\_result[0] == 'student':  
 # Hardcoded teacher information for demonstration purposes  
 teacher\_name = "John Doe"  
  
 # Create a new top-level window for the student  
 student\_window = Toplevel(self.window)  
 student\_window.title("Student Window")  
 student\_window.geometry('500x300') # Adjust the window size as needed  
  
 # Set the same background image as the login page  
 self.set\_background\_image(student\_window)  
  
 # Create a black box frame similar to the login page  
 box\_width = student\_window.winfo\_width() \* 0.6  
 box\_height = student\_window.winfo\_height() \* 0.6  
 box\_x = (student\_window.winfo\_width() - box\_width) // 2  
 box\_y = (student\_window.winfo\_height() - box\_height) // 2  
  
 student\_frame = Frame(student\_window, bg='#040405', width=box\_width, height=box\_height)  
 student\_frame.place(x=box\_x, y=box\_y)  
  
 # Display the teacher's name with a similar label style  
 teacher\_name\_label = Label(student\_frame, text=f"Teacher Name : {teacher\_name}",  
 font=('yu gothic ui', 15, "bold"), bg="#040405", fg='white')  
 teacher\_name\_label.pack(pady=10)  
  
 # Static student data for demonstration purposes  
 student\_data = [  
 {"name": "c++", "grade": "A"},  
 {"name": "Math", "grade": "B"},  
 {"name": "Software Engineering", "grade": "C"},  
 {"name": "multimedia", "grade": "A"},  
 {"name": "CCNA", "grade": "B"},  
 ]  
  
 # Display student data with a similar label style  
 for student\_info in student\_data:  
 student\_label = Label(student\_frame, text=f"{student\_info['name']}: Grade {student\_info['grade']}",  
 font=('yu gothic ui', 12), bg='#040405', fg='white')  
 student\_label.pack()  
  
 # Add more elements to the student window as needed  
  
 # Start the main loop for the student window  
 student\_window.mainloop()  
 else:  
 messagebox.showerror("Error", "Invalid user type or not a student.")  
  
 def get\_teacher\_info(self, username):  
 # Query the database to get teacher information based on the username  
 query = """  
 SELECT Teacher.fname  
 FROM Teacher  
 INNER JOIN Login ON Teacher.id = Login.id  
 WHERE Login.username = ?  
 """  
 result = self.db\_manager.execute\_query(query, (username,))  
  
 return result.fetchone() if result else None  
  
  
 def set\_background\_image(self, window):  
 # Set the same background image as the login page  
 background\_image = Image.open('images\\background1.png')  
 photo = ImageTk.PhotoImage(background\_image)  
 bg\_panel = Label(window, image=photo)  
 bg\_panel.image = photo  
 bg\_panel.pack(fill='both', expand='yes')  
 def open\_signup\_window(self):  
 signup\_page = SignupPage(self.window)  
 def open\_main\_window(self):  
 # Create a new window with the same background and appearance  
 main\_window = Tk()  
 main\_window.geometry('1166x718')  
 main\_window.resizable(0, 0)  
 main\_window.state('zoomed')  
 main\_window.title('Main Window')  
  
 # Copy the appearance and functionality from the login window  
 # For simplicity, you can create a new instance of the LoginPage class  
 main\_page = LoginPage(main\_window)  
  
 # Close the login window  
 self.window.destroy()  
  
 # Start the main loop for the new window  
 main\_window.mainloop()  
  
 def page(self):  
 self.db\_manager = DatabaseManager()  
  
 def close\_connection(self):  
 self.db\_manager.close\_connection()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 window = Tk()  
 login\_page = LoginPage(window)  
 login\_page.page()  
 window.mainloop()

## Database Implementation :

Overview:

The **School Soft** database is designed to manage information related to users, teachers, students, courses, logins, and user types. This document provides an overview of the tables in the database, their structures, and sample queries.

Tables:

### 1. Login Table

* **Description:** Stores login credentials for users.

| **Column** | **Data Type** | **Description** |
| --- | --- | --- |
| id | INT (Primary Key) | Unique identifier |
| username | NVARCHAR(50) | User's username |
| password | NVARCHAR(50) | User's password |

### 2. Teacher Table

* **Description:** Contains information about teachers.

| **Column** | **Data Type** | **Description** |
| --- | --- | --- |
| tid# | INT (Primary Key) | Teacher ID |
| fname | VARCHAR(11) | First name of the teacher |
| lname | VARCHAR(11) | Last name of the teacher |
| id | INT | Unique identifier |

### 3. Student Table

* **Description:** Stores information about students.

| **Column** | **Data Type** | **Description** |
| --- | --- | --- |
| stid# | INT (Primary Key) | Student ID |
| GPA | INT | Student's GPA |
| fname | VARCHAR(11) | First name of the student |
| lname | VARCHAR(11) | Last name of the student |
| tid# | INT | Teacher ID (Foreign Key) |
| id | INT | Unique identifier |

### 4. New\_User Table

* **Description:** Records information about new users.

| **Column** | **Data Type** | **Description** |
| --- | --- | --- |
| nid | INT (Primary Key) | New user ID |
| fname | VARCHAR(11) | First name of the user |
| lname | VARCHAR(11) | Last name of the user |
| username | NVARCHAR(50) | User's username |
| password | NVARCHAR(50) | User's password |

### 5. Typee Table

* **Description:** Specifies the type (teacher or student) of a user.

| **Column** | **Data Type** | **Description** |
| --- | --- | --- |
| tid | INT (Primary Key) | Type ID |
| username | NVARCHAR(50) | User's username |
| type | VARCHAR(50) | User type (teacher/student) |

### 6. Courses Table

* **Description:** Stores information about courses.

| **Column** | **Data Type** | **Description** |
| --- | --- | --- |
| cid# | INT (Primary Key) | Course ID |
| stid# | INT | Student ID (Foreign Key) |
| tid# | INT | Teacher ID (Foreign Key) |
| cname | VARCHAR(20) | Course name |
| grade | INT | Course grade |

## Code table :

CREATE DATABASE School\_Soft;

GO

-- Use the database

USE School\_Soft;

GO

-- Create the Login table

CREATE TABLE Login (

id INT IDENTITY(1,1) PRIMARY KEY,

username NVARCHAR(50),

password NVARCHAR(50)

);

create table Teacher

(

tid# int not null ,

fname varchar(11) not null ,

lname varchar(11) not null ,

id int not null ,

primary key (tid#)

);

create table Studnet

(

stid# int identity(1 ,1 ) primary key ,

GPA int not null ,

fname varchar(11) not null ,

lname varchar(11) not null ,

tid# int not null ,

id int not null ,

foreign key(tid#) references Teacher(tid#)

);

create table new\_user

(

nid int identity(1 ,1 ) not null ,

fname varchar(11) not null ,

lname varchar(11) not null ,

username NVARCHAR(50),

password NVARCHAR(50),

primary key(nid),

);

create table typee

(

tid INT IDENTITY(1,1) not null ,

username NVARCHAR(50) not null ,

type VARCHAR(50) CHECK (type IN ('teacher', 'student'))

primary key(tid , username ) ,

);

drop table courses

create table courses(

cid# int identity(1,1) not null ,

stid# int not null ,

tid# int not null ,

cname varchar(20) not null ,

grade int not null ,

primary key(cid#)

);

insert into new\_user (fname , lname , username , password )

values ('Hesham' , 'Aymen' , 'heshamaym' , 'p@ssword')

select \* from new\_user

insert into typee (username , type)

values ('1' , 'teacher')

select \* from typee

insert into Login (username , type)

values ('1' , 'teacher')

INSERT INTO Login (username, password) VALUES

('1', '2'),

('user2', 'pass2');

-- Insert data into the Teacher table

INSERT INTO Teacher (tid#, fname, lname, id) VALUES

(, 'John', 'Doe', 2),

(2, 'Jane', 'Smith', 102);

-- Insert data into the Studnet table (assuming this is a typo, corrected from 'Studnet' to 'Student')

INSERT INTO Studnet (GPA, fname, lname, tid#, id) VALUES

(3.5, 'Alice', 'Johnson', 1, 5),

(4.0, 'Bob', 'Williams', 2, 202);

-- Insert data into the new\_user table

INSERT INTO new\_user (fname, lname, username, password) VALUES

('New', 'User1', 'newuser1', 'newpass1'),

('New', 'User2', 'newuser2', 'newpass2');

select \* from new\_user

-- Insert data into the typee table

INSERT INTO typee (username, type) VALUES

('1', 'teacher'),

('user2', 'student');

select \* from Login

-- Insert data into the courses table

INSERT INTO courses (stid#, tid#, cname, grade) VALUES

(1, 1, 'Mathematics', 3),

(2, 2, 'Science', 4);

select \* from Studnet

select \* from Login

select \* from Teacher

select \* from typee

SELECT Teacher.fname

FROM Teacher

INNER JOIN Login ON Teacher.id = Login.id

WHERE Login.id = 2

SELECT Teacher.fname

FROM Teacher

INNER JOIN Studnet ON Teacher.tid# = Studnet.tid#

INNER JOIN new\_user ON Teacher.id = new\_user.nid

WHERE new\_user.username = 'user1'

# TOOLS USED IN THIS PROJECT ARE :

* Draw.io Application for Designing.
* pycharm for Implementation.
* SQL Server for Database.
* Word for Documentation.
* Figma for prototype.
* Teams for meeting.