**Report on Lab-4 Assignment: Software Application Development using Python**

By: Sharanya Chakraborty

22CS10088

This program uses classes in Python to set up an academic unit with a hierarchy as follows:  
- Person

-Teacher

-Student

-UG Student

-PG Student

It also uses the Tkinter module to set up a basic GUI to perform CRUD operations as well as some other niche functions.

1. Classes:  
   - For the Academic Unit:

* Person
* Teacher (extends Person)
* Student (extends Person)
* UG Student (extends Student)
* PG Student (extends Student)

All of these are as per the specifications in the assignment.

* For the GUI Interfaces:
* UserRegistrationGUI : GUI for the User Registration Page.
* SignInGUI : GUI for the Sign In Page.
* TeacherProfileEditGUI : GUI for the teachers for adding and updating their details, as well as deregistering.
* UGStudentProfileEditGUI : GUI for the teachers for adding and updating their details, as well as deregistering.
* PGStudentProfileEditGUI : GUI for the PG Students for adding and updating their details, as well as deregistering.
* WelcomeGUI : GUI for the welcome screen.
* MainApplication : Triggered in the main function, basically tells which GUI should we begin with in the beginning. Also triggers the call of the function load\_data\_from\_file, which will be discussed in Functions section below.

All these GUI class extend the built in ‘Frame’ class in the Tkinter module.

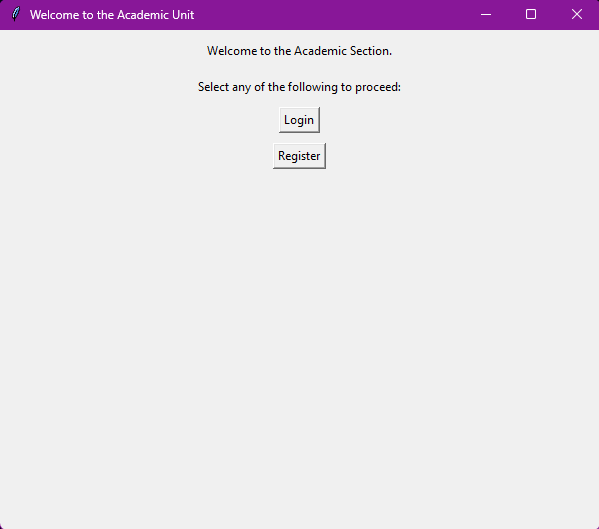
1. Functions:
   * Utility Functions:

* unique\_email(email): This checks if an email ID (to be used as the User ID) is already in use.
* email\_validator(email): Checks if the email entered by the user is of a valid format (has only one @ symbol, and the period (.) comes after the @ symbol)
* password\_validator(password): This follows the specifications in the assignments. It returns various error codes to display suitable error messages like:
  + - 501: Password not in between 8-12 characters in length
    - 502: Password does not have an upper case, lower case or digit in it
    - 503: Password does not have a special character from the set “!@#$%&\*”
    - 504: Password has blank spaces
  + Backend Functions:
    - * + load\_data\_from\_file(): It opens a file called “academic.txt” (if it exists, else the list remains empty) and splits the chunks of data in the files which are separated by a new line character (essentially ‘/n/n’). Within each chunk, it splits the key value pairs which are one after another (essentially ‘/n’). To take care of Boolean values, it checks for values equal to “True” or “False” and explicitly changes them to their Boolean forms. Rest all fields are parsed as strings. After parsing, these are then filtered by the user types to create objects of those types, which then populate the object list.
        + parse\_attributes(attributes): For every key value pair, it splits it using ‘:’ then creates a dictionary with these key value pairs.
        + save\_data\_to\_file(): It iterates through the members list, checks what instance every object is, and then determines the user type. It then stores all fields (except is\_authenticated and valid\_attempts) for every object in the file.

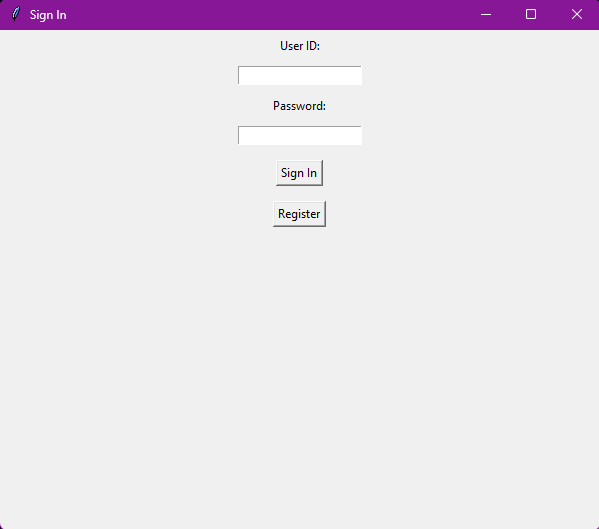
Some assumptions:

I have assumed that every time a user successfully logs in, the user’s “valid attempts remaining” is reset to 3, since this resembles a login portal more.

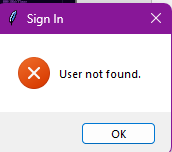
Also, for deregistration, I am completely removing the account from the list (and therefore the backend file).

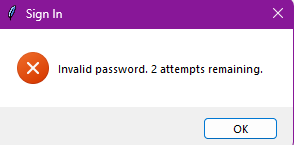
Screenshots of various GUIs under various scenarios:  
Welcome Screen:  


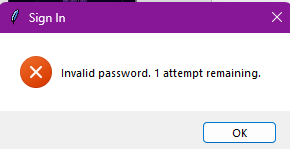
Login Page:

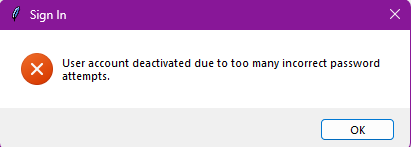


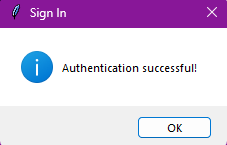
Login page message when no user is found in the list of members:

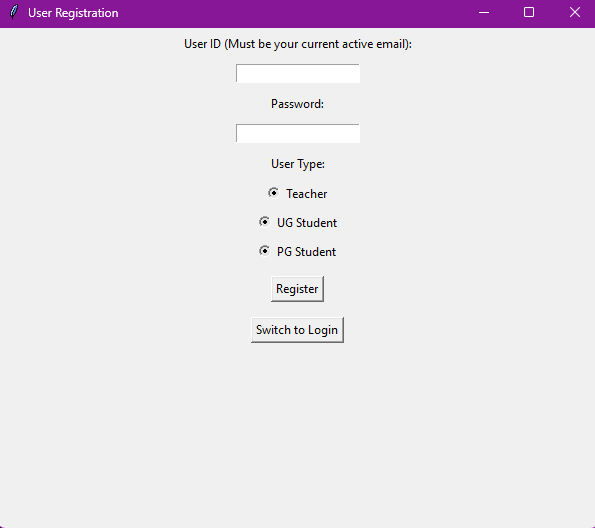


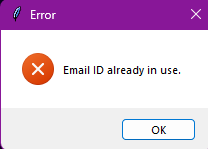
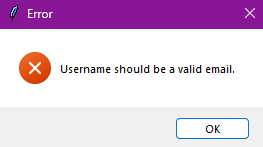
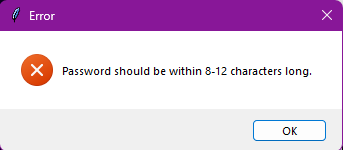
Login page messages for incorrect password:  


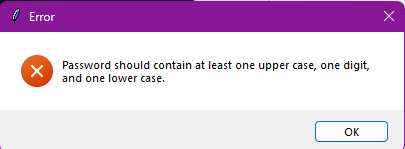


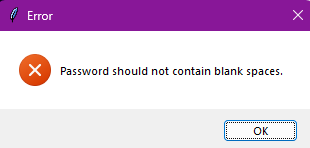
Login page message for deactivation for too many incorrect attempts:  
  
Login page on successful login

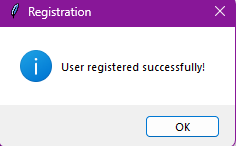


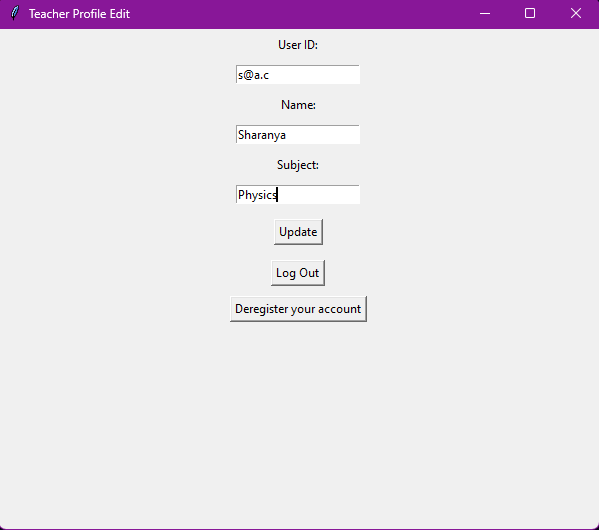
Registration Page:  


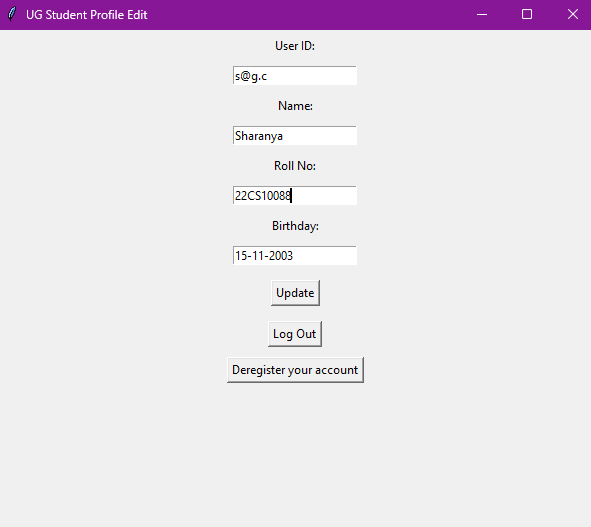
On same email that already exists:  
  
On improper email format:  
  
Invalid password messages:  


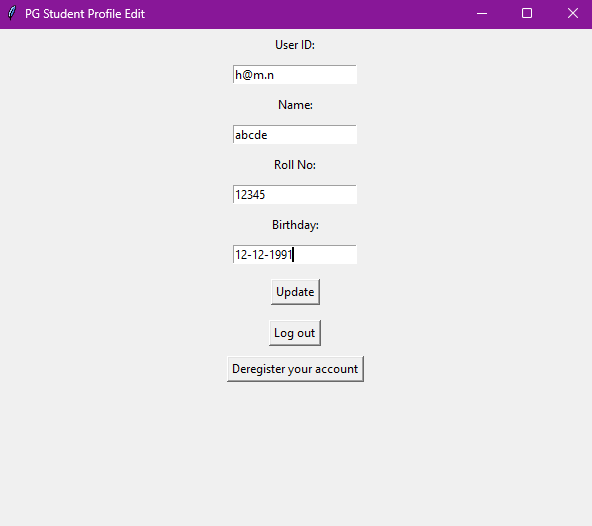


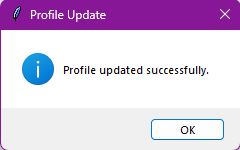


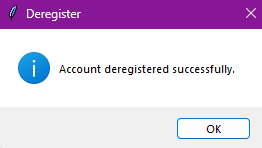
Successful registration:  
  
Teacher Profile Page:



UG Student Page:  


PG Student Page:  


Successful profile update:  


Deregistration successful:  


Some basic padding was added to the elements in the y direction to create space between them.

I have also used a single list to store objects of all types, and this list is the one which is used to update the backend file.