**How I made University helper (Python project)**

I first started with the login interface for which I used customtkinter for a nice modern GUI look. I then customized the interface to my likings before implementing the login functionality. I then changed around some fonts and sizes until it suited well with the theme I was going for. After that, I implemented a sign-up label at the near bottom for those who want to register and added a hovering effect whenever users hovered over the text using their mouse. The default login and password are “admin” with password being “123” in order to test the login. However, I found that sometimes I kept putting space after admin and it wouldn’t register it as the strings were not equal which I found annoying, hence I decided to remove any spaces from the users input so as long as the username was the same, it would work regardless of if there were any empty spaces between them.

Then I needed to setup the signup feature for new users that don’t have an account, for that I created a new pop-up window whenever the “create a new account” link was clicked. For closing the pop up and making a new one appear, I kept getting weird errors from tkinter when using the .destroy() so I had to use a different method to close windows and make them reappear using .deiconify() and .withdraw() which worked perfectly with no errors as well as using the terminal to exit the application to save up on memory which I had to create a separate function for to implement for each window I created.

However, after reading through the main.py file, I realized how unorganized it looked and the code was already 100 lines and didn’t have a structure. I then proceeded to put the login and signup page in separate files and used classes which were imported to the main file for neater and more organized code. I then setup the .main py file and imported the different classes needed and passed in the login page to the signup class and the sign-up page to the login class. This allowed me to go back and forth from signup to login through the use of a button.

After that I proceeded to setup my signup page and used the grid method to place them next to the entry. I also added icons for them to give it a cleaner. I also wanted the entry bar to light up blue whenever the user hovered near it, so they knew which input box they were on, to do this I made 2 functions called on\_enter and on\_leave but I was unable to pass the CTKButton in order to change the border colour in the function. I tried searching but only results for the normal tkinter came up and I tried using event.widget.config and similar things but that only worked with tkinter not customtkinter hence I created lambda functions that would allow me to pass both the event and the button itself to the functions which worked perfectly rather than creating a separate function for every entry box which would be inefficient and make the code much longer than needed.

Following that I added signup logic to the signup page by making it so that the full name must include 2 words. The way I identified that was to see if the string from the input box had a space between them had, if not it would send an error telling them to give the full name. For the email checker, I used an email pattern which I found through the internet and allowed me to have some validity to check if the email users enter are valid. For the password, I made it so the user has to create a password longer than 6 characters in which it must contain an uppercase and a number. After all tests were passed and the signup button was clicked, the user would get redirected to the main application.

I then had to have a database for the login and signup system that way if the user signs up and then closes the application and uses it later on, they are able to login instead of signing up again. For this I had 2 options, either SQLite or Pandas through using .csv files and reading from them. Although either would do the job as the data was going to be locally, I decided to go with SQLite as it will be more suitable if the dataset grows and is much more efficient taking up less space and resources, and if plan on making it public having more data added will be very easy etc rather than pandas which is less secure although it is much easier to implement. I then proceeded to learn SQLite through resources across the internet but mainly YouTube tutorials and documentations done the job.

I continued to then implement the database through having a separate file called database\_setup.py which added the table with specific columns of id, username, email, and password as well as a default login which was admin with the password of 123 for testing purposes. I then did a check on login page to see if the data the user has entered exists in the database, if not it will give an error saying “Incorrect login” as well as implementing it on the signup page where if the username doesn’t already exist in the database, they are able to create a new account which gets added to the database allowing them to login with it the next time they open the application.

I then proceeded to add the Remember me checkbox functionality so that users only need to enter their login details once and the next time they open the application, it will automatically log them in. This was done through adding a new table to the database called local\_credentials. The login class would then check if the remember me checkbox is on or off and would then add the current username and password to that database and every time the application would check if the credentials existed and would automatically redirect them to the app\_page which was implemented in the main.py. However, the code for the database when updating the values whenever the user presses remember me and login or deleting the local credentials if the user logs out through the main app could all be implemented in the database\_setup hence I decided to convert it into a class and add functions that other scripts could call making it more organised and easier to debug.

After doing that I commented all the code and implemented an existing function to have 2 purposes on the function “does\_user\_exist” for the database class so that the password parameter is optional. This was done so that I could use the same function to check if the username already exists or not when creating an account which was used by the signup class. As well as adding a new function to add custom entries into the database for when users create an account rather than having it in the signup class making the code more organised as well as adding other small functions into the database that the signup and login class can use.

I then proceeded to clean up the code, fix some minor bugs and finally started implementing features on the main page. Initially I added the pomodoro timer with its own frame and adjusted it to be on the top right-hand side of the window. I then implemented the buttons and text for it as well as the ability to customize the work time and break time. There was a small problem where users could spam the start/reset button which would make the timer go down much quicker, so I implemented a cooldown and disabled the reset button until the timer got stopped, causing less fewer bugs. At the near end, I also decided to add a “total time worked” which shows how much time they spent working.

Following that, I added an email sender in the application where users can click the button to get a pop up which allows them to send them an email to their lecturer/teacher if they have a problem or can’t attend the next day etc rather than distracting themselves and going to other apps such as Gmail in order to do the same thing. This minimises the distractions and allows them to stay more concentrated to the Pomodoro timer and effectively makes them be more productive, thus the name University Helper. The email sender uses a Gmail STMP method meaning currently only Gmail users can send mail but perhaps later I’ll implement more mail methods that incorporate other mail accounts such as outlook etc.

Next up was adding a way to see national rail delays in the UK as I travel through train when going to university so this would be rather helpful. I then proceeded to scrap the national rail website for the important dates and formatted it in a readable manner. This allowed me to be able to see exactly on which days which transportations would be affected by strikes etc which was very useful for someone who uses the train daily in order to go to university.

Most importantly, was the To-do-list which allows students to stay organised and have a way to track the things they need to do. This To-do-list has the ability to be removed as it slowly fades away once you hit the checkbox as well as being able to add new tasks with a scrollable frame means they can add as many tasks as they want. I then used the same method to implement my next feature which was deadline notifier as a way of tracking the important dates for students such as when the next assignment for a certain module is due or if there’s an upcoming exam etc. This had the same features as the to-do list with being able to add a new deadline and remove it by clicking the checkbox, but it also allowed students to add the given time in a valid format as well as showing how many days are left until the next deadline which was calculated by comparing the given date and the current date module.

Lastly, I readjusted the layout and fixed up my code comments and make it easier to read as well as fix some minor bugs. This program is meant for students that are in university as it includes all the important stuff that they would need in a productivity application from the pomodoro timer which tracks your hours to a to-do-list with an email sender, deadline notifier to keep students up to date with what deadlines need to be completed first and a way of seeing which trains will be delayed in the upcoming days.

Resources used:

For the GUI interface I used customtkinter: <https://github.com/TomSchimansky/CustomTkinter>

For the custom messageboxes I used CTkMessagebox: <https://github.com/Akascape/CTkMessagebox>

For database management I used SQLite: <https://www.sqlite.org/index.html>