auth.py

Imports

```
import tkinter as tk
import customtkinter
from authActivity.backend.UserAccount import UserAccount
```

- `tkinter` → Python's built-in GUI library.
- `customtkinter` → A modern version of Tkinter with better-looking widgets.
- `UserAccount` →class for managing login and registration (username/password).

Auth Class (GUI Frame)

```
class Auth(tk.Frame):
    def __init__(self, master):
        super().__init__(master, bg="#f5f5f5")
        self.user = UserAccount()
        self.pack(fill="both", expand=True)
```

- `Auth` is a **frame** (like a section of a window).
- `master` → parent widget (your main window).
- `bg="#f5f5f5"` → background color of the frame.
- `self.user` → instance of your `UserAccount` to handle login/register.
- `self.pack(fill="both", expand=True)` → make the frame fill the whole window.

Notification Label

```
self.notif_label = customtkinter.CTkLabel(
    self,
    text="",
    fg_color='transparent',
    text_color="#000000",
    font=("Arial", 11),
    padx=20,
```

```
        pady=12,
        corner_radius=5
    )
    self.notif_label.place(relx=1.0, rely=1.0, anchor="se", y=-10, x=-10)
```

- Creates a **label to show messages** (like “Login successful”).
- `fg_color='transparent'` → starts invisible.
- `place(relx=1.0, rely=1.0, anchor="se", x=-10, y=-10)` → puts it **at bottom-right corner**, slightly inside.
- `corner_radius` → rounded corners.

`relx` and `rely` in `place()`

- These are **relative positions** of a widget inside its parent (like a frame or window).
- They are **numbers from 0.0 to 1.0**.

Parameter	What it means	0.0	0.5	1.0
`relx`	Horizontal position (left → right)	left edge	center	right edge
`rely`	Vertical position (top → bottom)	top edge	center	bottom edge

---

Current Frame

```
    self.current_frame = None
    self.show_login()
```

- Keeps track of the frame currently shown (login or register).
- Initially shows the **login screen**.

Notification Functions

```
def show_notif(self, message):
    if message:
        self.notif_label.configure(text=message, fg_color="#ffffff")
```

```
            self.notif_label.lift()
            self.after(3000, self.clear_notif)
        else:
            self.clear_notif()

    def clear_notif(self):
        self.notif_label.configure(text="", fg_color='transparent')
```

- `show_notif(message)` → displays a message for **3 seconds**.
- `clear_notif()` → hides the notification.

`configure`
`self.notif_label.configure(text=message, fg_color="#ffffff")`

- **Purpose:** Change the **properties of a widget** after it has been created.
- Here, it updates:
  - `text` → the message to display
  - `fg_color` → the background color of the label

`lift`
`self.notif_label.lift()`

- **Purpose:** Bring the widget **to the front** above other widgets.
- Ensures the notification **doesn't get hidden behind other frames or widgets**.

`after`
`self.after(3000, self.clear_notif)`

- **Purpose:** Schedule a **function to run after a delay** (in milliseconds).
- `3000` → 3000 milliseconds = **3 seconds**
- `self.clear_notif` → function that will be called after 3 seconds

Clearing Frames

```
    def clear_frame(self):
```

```
    if self.current_frame is not None:
        self.current_frame.destroy()
        self.current_frame = None
```

- Deletes the old frame before showing a new one (login/register).

Show Login Screen

```
def show_login(self):
    self.clear_frame()
    mainFrame = tk.Frame(self, bg="#ffffff", width=420, padx=40, pady=30, height=400)
    mainFrame.place(relx=0.5, rely=0.5, anchor="center")
```

- Creates a **login frame** in the center of the window.
- `mainFrame.pack_propagate(False)` → prevents the frame from resizing to fit widgets.

Show Register Screen

```
def show_register(self):
    self.clear_frame()
    mainFrame = tk.Frame(self, bg="#ffffff", width=420, padx=40, pady=30, height=400)
    mainFrame.place(relx=0.5, rely=0.5, anchor="center")
```

- Same as login, but for **registration**.
- Input fields for **username** and **password**, **Register button**, and **Login link**.

Switch Functions

```
def switch_to_register(self, event=None):
    self.show_register()

def switch_to_login(self, event=None):
    self.show_login()
```

- Called when user clicks **Sign up** or **Login** link.
- Simply shows the corresponding screen.

Handle Login/Register

```
def handle_login(self):
    username = self.username_entry.get()
    password = self.password_entry.get()
    message = self.user.login(username, password)
    self.show_notif(message)

def handle_register(self):
    username = self.username_entry.get()
    password = self.password_entry.get()
    message = self.user.register(username, password)
    self.show_notif(message)
```

- Reads the **username** and **password**.
- Calls the `UserAccount` class to **login** or **register**.
- Shows the **result message** using the notification label.