Ministerul Educației și Cercetării al Republicii MoldovaUniversitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică

Tehnici si mecanisme de proiectare software

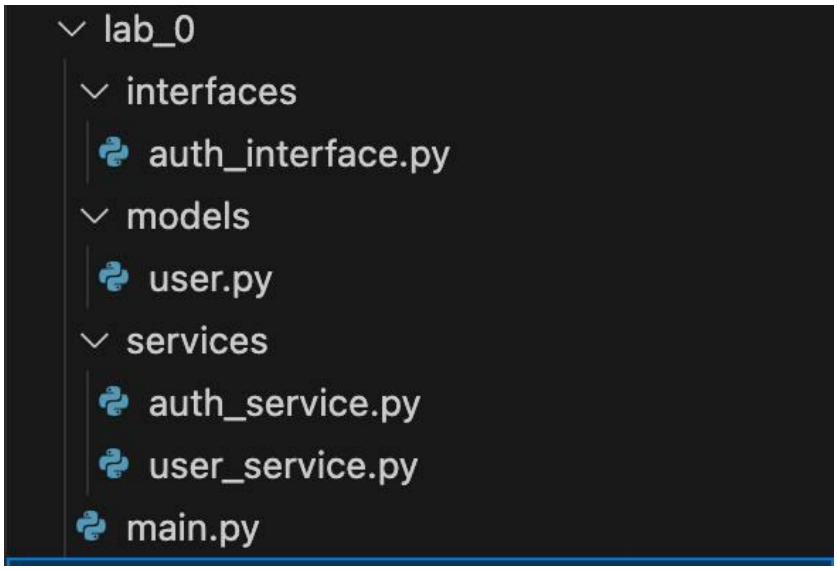
Laborator work 0

Elaborated:
st. gr. FAF-222 Dimbitchi Sergiu

Chişinău - 2023

Scopul lucrări :

Exercitiile practice :



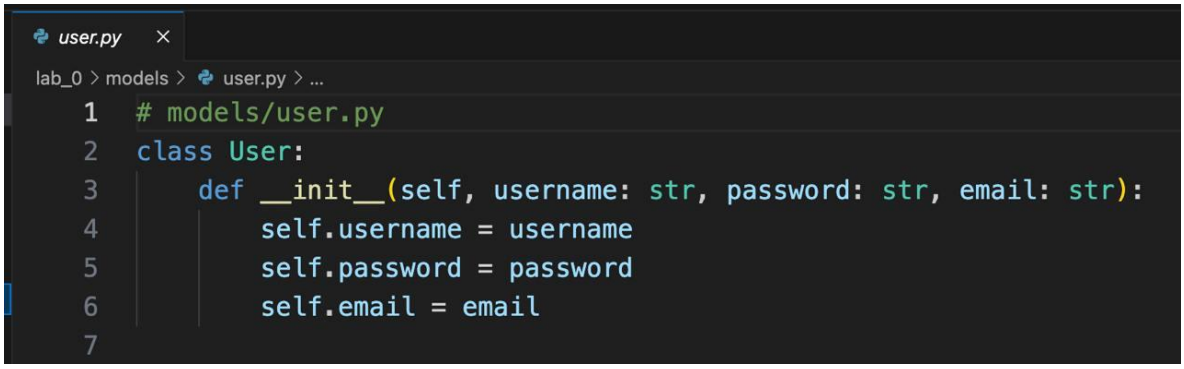
Explicația exercițiului :

auth_interface.py

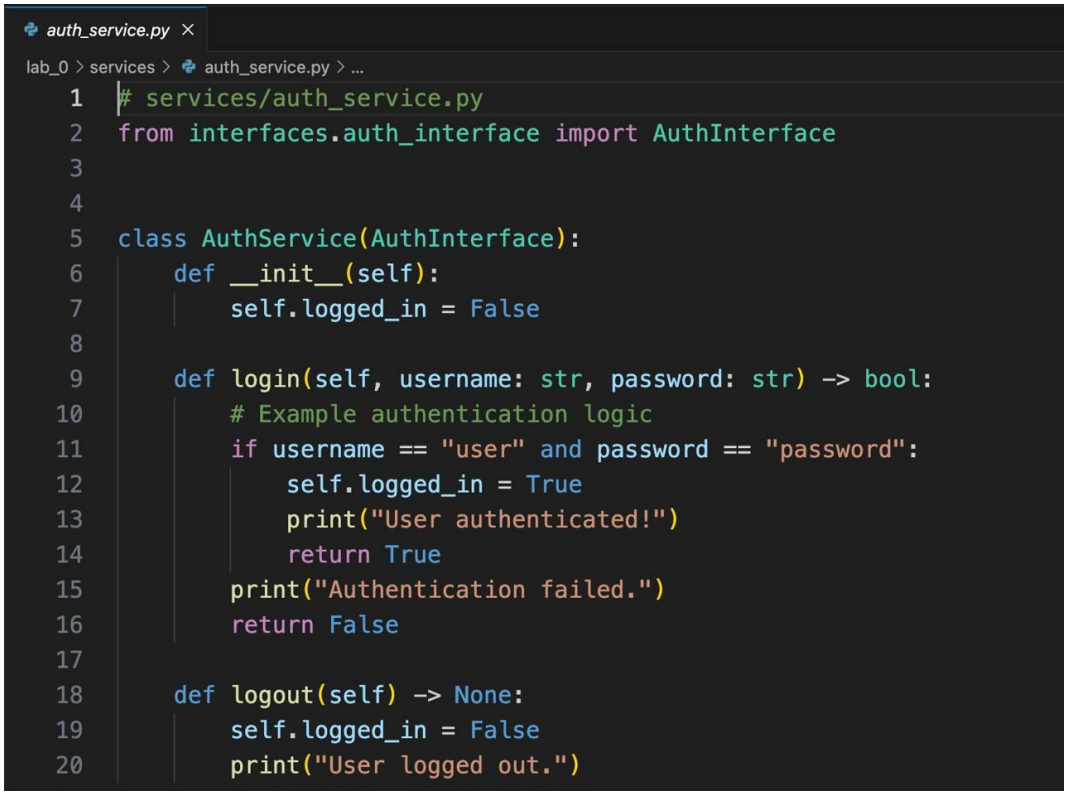
lab_0 > interfaces > auth_interface.py > ...

```python
# interfaces/auth_interface.py
from abc import ABC, abstractmethod


class AuthInterface(ABC):
    @abstractmethod
    def login(self, username: str, password: str) -> bool:
        pass

    @abstractmethod
    def logout(self) -> None:
        pass
```

auth_interfaces.py - defines an abstract base class called `AuthInterface` using Python's `abc` module. It specifies two abstract methods: `login`, which takes a username and password as parameters and returns a boolean indicating success or failure, and `logout`, which does not take any parameters and does not return a value. Classes that inherit from `AuthInterface` must implement these methods to provide concrete authentication functionality.

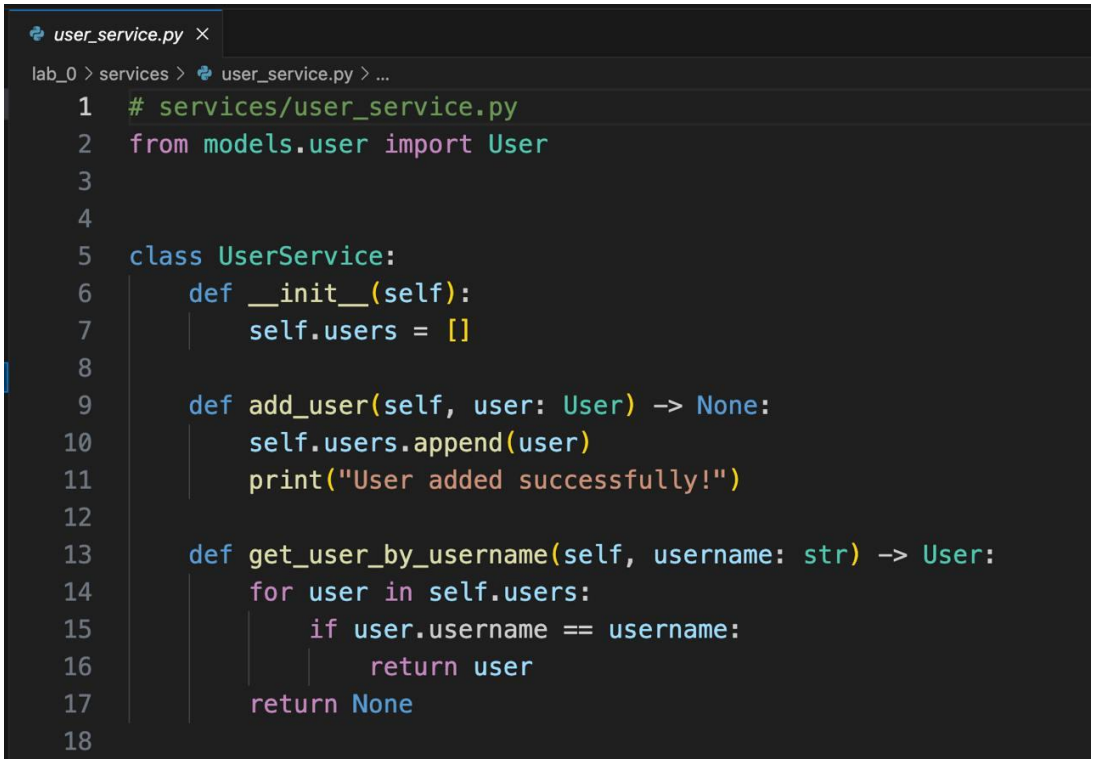


user.py - defines a `User` class that represents a user in an application. The `__init__` method initializes a new `User` object with three attributes: `username`, `password`, and `email`, which are all expected to be of type `str`. When a new instance of the `User` class is created, these attributes are set to the values passed as arguments to the constructor.



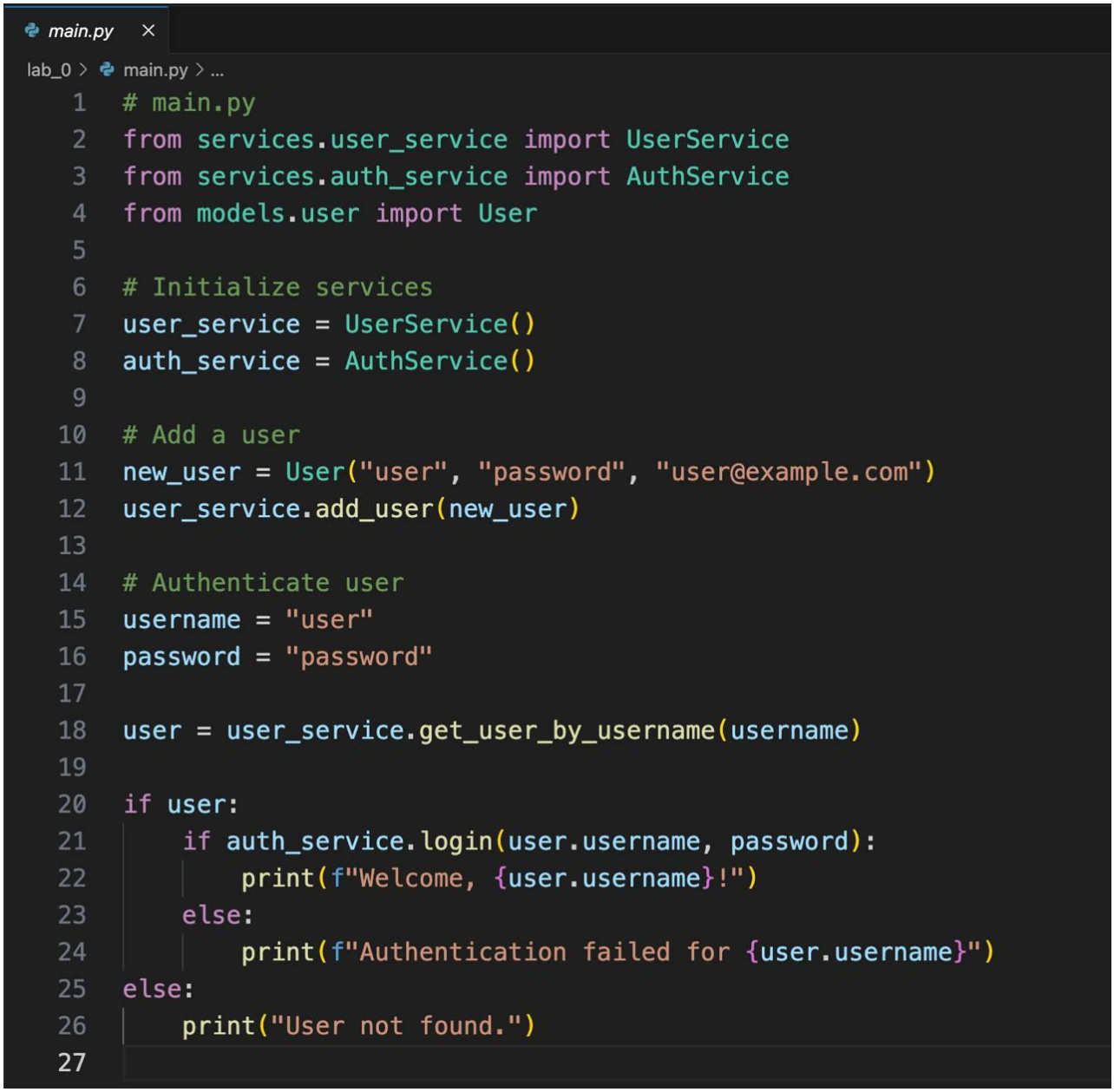
auth-service.py - defines a class `AuthService` that inherits from the `AuthInterface` abstract base class. In the `__init__` method, it initializes an instance variable `logged_in` to `False`, indicating that the user is not currently logged in.

- The `login` method takes a username and password, checks them against the values "user" and "password", and, if they match, sets `logged_in` to `True`, prints a success message, and returns `True`. If authentication fails, it prints a failure message and returns `False`.
- The `logout` method sets `logged_in` to `False` and prints a message indicating the user has logged out.



user_service.py - defines a class `UserService` that manages a list of `User` objects. In the `__init__` method, it initializes an empty list called `users` to store the user instances.

- The `add_user` method takes a `User` object as a parameter, appends it to the `users` list, and prints a success message.
- The `get_user_by_username` method searches through the `users` list for a user with a matching username; if found, it returns that `User` object. If no user is found with the given username, it returns `None`.



main.py :

Service Initialization: It starts by importing the necessary classes and initializing instances of `UserService` and `AuthService`.
User Creation and Addition: A new `User` object (`new_user`) is created with a username, password, and email. This user is then added to the `UserService` using the `add_user` method.

User Authentication:

- The code defines the `username` and `password` variables for authentication.
- It attempts to retrieve the user from `UserService` by calling `get_user_by_username(username)`. If the user is found, it proceeds to authenticate the user using `auth_service.login()`.
- If authentication is successful, it prints a welcome message. If authentication fails, it prints an error message indicating that authentication failed for the user.
- If the user is not found, it prints "User not found."