**Unit Testing of the login function**

The objective of the login function is to validate user credentials and return whether a login attempt was successful. If the user does not exist, the function provides an option to register the new user.

The (username, password) function takes as inputs a username (string) and a password (string). Thus, out of {int, float, string, list}, the first two and the last are invalid types. We create one test case for each of these invalid input types and assert that the function returns an appropriate error message or raises a TypeError.

We now turn to valid inputs. The input domain can be divided into the following equivalence classes (EC):

1. **EC1**: The username exists in the user database, and the password matches.
2. **EC2**: The username exists in the user database, but the password does not match.
3. **EC3**: The username does not exist in the user database.
4. **EC4**: The username and/or password are empty strings.

The coverage criteria we will use are: For each equivalence class, we need to test at least one input.

This leads to the following test requirements:

* **R1**: If input is in EC1, login function should return True and indicate successful login.
* **R2**: If input is in EC2, login function should return False and indicate a failed login attempt.
* **R3**: If input is in EC3, login function should inform the user that the username does not exist and provide an option to register.
* **R4**: If input is in EC4, login function should return False and prompt the user to enter valid credentials.

These lead to the corresponding test cases:

* **TC1**: Input = ("testuser", " Valid123!") (EC1)
* **TC2**: Input = ("testuser", "wrongpass") (EC2)
* **TC3**: Input = ("nouser", "anypassword") (EC3)
* **TC4**: Input = ("", "") (EC4)

In order to execute the tests, the fixtures mock\_user\_data was written to simulate a user database.