Test Data Manager

Testing DataManager to check if user data is handled as expected:

Blackbox Tests:

1. Create manager object
2. Check initial signed-in state
3. Load user data from csv and chk signed in
4. Get user data when loaded from mocked csv read
5. Get user data when file is empty
6. Register user (new user) when file is empty
7. Register user (overwrite existing user) when file has data (mocked csv read)
8. Check initial has preferences is False
9. Check has preferences after loading preferences from csv(mocked csv read)
10. Check get preferences after loading preferences from csv
    (chk mocked csv read output is formatted correctly)
11. Check get preferences after loading preferences from empty csv
    (load current state of csv in testing env (empty file))
12. Register Preferences processes preferences to correct data format initial (no current preferences)
13. Register Preferences processes preferences to correct data format with existing preferences (overwrite preferences)

Whitebox Tests:

1. Test file does not exist case (to check if file not found error is handled)
2. Test save input is of the wrong type
3. Test unexpected exceptions while file reading
4. Test save user to file
5. Test save preferences to file

Tests Implementation: https://github.com/SGTK06/Job-Application-Tracker/blob/f39cf96de8ea966d38b4f4f4a5d0883a3db42b8e/tests/test_data_manager.py

Test Report:

Test ID	Input	Expected Output	Actual Output	Pass /Fail
BT01	Initialize DataManager()	DataManager object created successfully	created successfully	Pass
BT02	Call is_signed_in()	False (no user data loaded)	False	Pass
BT03	Mock pandas.read_csv to give DataFrame({'user_name':['Peter'], 'user_mail':['parker@marvel.com']})	Signed in = True (user data loaded)	True	Pass
BT04	Mock pandas.read_csv to give DataFrame({'user_name':['Peter'], 'user_mail':['parker@marvel.com']})	{'user_name': 'Peter', 'user_mail': 'parker@marvel.com'} (give data as dict directly no lists)	{'user_name': 'Peter', 'user_mail': 'parker@marvel.com'}	Pass
BT05	Empty csv file	{'user_name': '', 'user_mail': ''}	{'user_name': '', 'user_mail': ''}	Pass
BT06	register_user('abc', 'abc@def.com')	{'user_name': 'abc', 'user_mail': 'abc@def.com'}	{'user_name': 'abc', 'user_mail': 'abc@def.com'}	Pass
BT07	register_user('a1b2', 'a12@bc3.com') then overwrite with ('abc', 'abc@def.com')	{'user_name': 'abc', 'user_mail': 'abc@def.com'}	{'user_name': 'abc', 'user_mail': 'abc@def.com'}	Pass
BT08	Check has_preferences() while empty preferences csv file	False	False	Pass

BT09	Mock pandas.read_csv to give DataFrame({'user_skills':['skill1, skill2, skill3, skill4, skill5'], 'min_salary':[10]})	True (preferences loaded)	True	Pass
BT10	Mock pandas.read_csv to give DataFrame({'user_skills':['skill1, skill2, skill3, skill4, skill5'], 'min_salary':[10]})	{'user_skills': ['skill1','skill2','skill3','skill4','skill5'], 'min_salary': 10}	{'user_skills': ['skill1','skill2','skill3','skill4','skill5'], 'min_salary': 10}	Pass
BT11	Empty preferences CSV	{'user_skills': [], 'min_salary': 0}	{'user_skills': [], 'min_salary': 0}	Pass
BT12	register_preferences ('skill1, skill2, skill3, skill4, skill5', '10') as strings	{'user_skills': ['skill1','skill2','skill3','skill4','skill5'], 'min_salary': 10} of correct data type (list of skills and int)	{'user_skills': ['skill1','skill2','skill3','skill4','skill5'], 'min_salary': 10}	Pass
BT13	register_preferences ('skill1, skill2, skill3, skill4, skill5, skill6', '20')	{'user_skills': ['skill1','skill2','skill3','skill4','skill5','skill6'], 'min_salary': 20}	{'user_skills': ['skill1','skill2','skill3','skill4','skill5','skill6'], 'min_salary': 20}	Pass
WT01a	Mock pandas.read_csv to raise FileNotFoundError	Handle exception without stopping program	No crash	Pass
WT01b	Mock pandas.read_csv to raise pandas.errors.EmptyDataError	Handle exception without stopping program	No crash	Pass
WT02a	register_preferences ('skill1, skill2', 'hundred')	min_salary set to 0	min_salary = 0	Pass

WT 02b	register_preferences (1234, 10)	user_skills converted to ['1234'], convert unexpected datatype to string to avoid processing errors	user_skills = ['1234']	Pass

Mock Documentation:

Test ID	Test Name	What Was Mocked	Why It Was Mocked
BT03	test_signed_in_after_loading_data_bt03	pandas.read_csv	To return fake csv user data without loading actual file
BT04	test_get_user_data_bt04	pandas.read_csv	To provide fake user csv data for get_user_data
BT09	test_loaded_preferences_bt09	pandas.read_csv	To give mock preferences on csv loading
BT10	test_get_preferences_data_bt10	pandas.read_csv	To check preferences processing correctness
WT01a	test_missing_file_error_handled_wt01_a	pandas.read_csv	To raise FileNotFoundError and check error handling
WT01b	test_empty_file_error_handled_wt01_b	pandas.read_csv	To raise EmptyDataError and check error handling