Part 1. Team github repo
https://github.com/gzhami/research_lab

Part 2. User stories and Acceptance testing (20 points max)

12 points max for there should be acceptance tests for each user story (the user stories given in this report, not necessarily the same as the revised proposal). There should be at least one test with valid input and at least one test with invalid input. If a user story could not possibly involve invalid inputs, that should be explained. Take off 2 points for each user story without a valid test case and two points for each user story without an invalid test case or explanation, up to 12 points off (i.e., 0 for this portion of the 20 points total). This is to account for 3 to 5 user stories, e.g., if there are 5 user stories and none of them have any acceptance test cases, this part would be 0 out of 12, not -8 out of 12.

8 points max for the discussion of the acceptance testing process and bugs found/fixed. 0 out of 8 if this discussion is missing, prorate partial credit among the user stories covered or not covered by this discussion.

The user story we finished right now is

Users: Investment Professionals

<Strategy Upload>: As an investment professional, I want to easily upload all trading strategies so that I can get results and compare them smoothly. My conditions of satisfaction are <

Common case:

1. Investment professionals should log in before they can upload their strategies.
2. Investment professionals upload the valid trading strategies(.py) file and then they have options to upload or not.
3. After uploading, they can choose to use which strategy to run backtesting.
4. After backtesting is done, then can view the result and save to compare with others.

Special cases:

1. Investment professionals upload an invalid strategy, which can't be compiled, a warning message will prompt up to let them know.
2. ~~Investment professionals upload a valid strategy but forget to save it, there's a window prompted up to ask if save or not.~~
3. ~~Investment professionals forget to save results, there's automatic windows to ask if to save or not, all the results will be backed up for one week if not saved. >~~
4. Investment professionals do not have an account

The acceptance test process:

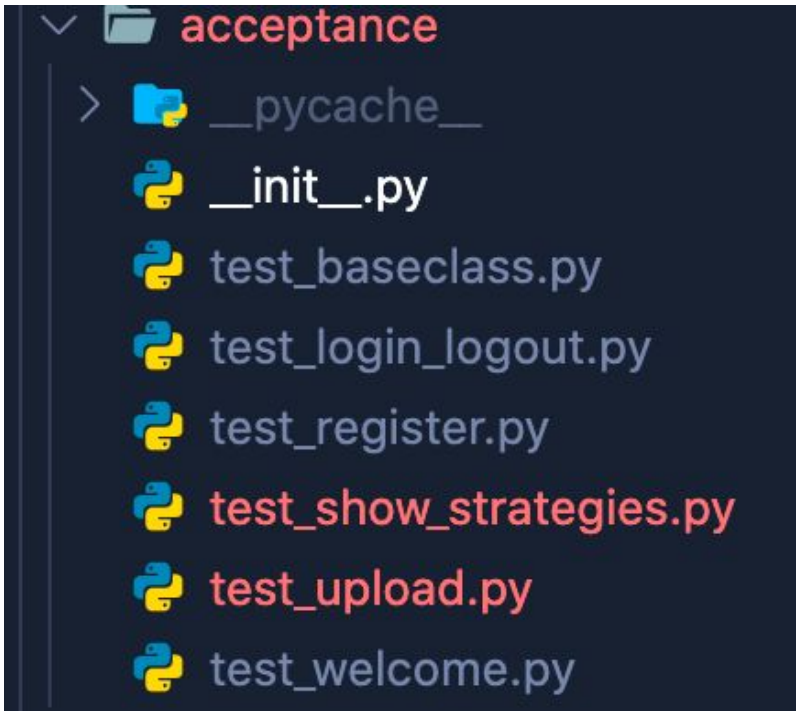
Common case:

1. (Valid, Finished) Investment professionals go to /login endpoint, the input box field should be available to them. After sending a post request with valid email and password, redirect to /home
    a. (Invalid, finished) Investment professionals go to /home endpoint before they log in, redirect them to /login endpoint
2. (Valid, Finished) upload a valid .py file using the /upload endpoint, test the response code (200) and it is successful
3. (Valid, Finished) there should be an "all strategies" button in /upload endpoint that enables users to go to see all strategies they upload. There is no invalid test because it is not testable
4. (Valid, Finished) there should be an "view results" button in /strategies endpoint that enables users to go to see results of their strategies.There is no invalid test because it is not testable
5. (Valid, Finished) new investment professionals go to /register endpoint, the input box field should be available to them. After sending a post request with valid username, email, password and password confirmation, user registration successfully.

Special case:

1. (Invalid input of Common case 1, finished) upload an invalid file which is not a valid .py file, and there will be messages for the user to re-upload.
1. (Invalid input of Common case 1, finished) upload an empty file, there will be messages for the user to re-upload.
2. Under development, and it is a frontend feature, not testable by backend test
3. Under development, and it is a frontend feature, not testable by backend test

4. (Valid, Finished) there is a “Register” button which redirects to /register endpoint in the /login endpoint. There is no invalid test because it is not testable



<Backtesting>: As an investment professional, I want to easily compare all the backtesting results from my trading strategies so that I can analyze my strategies.

Common case:

1. Investment professionals should be able to compare multiple backtesting results(the result will include a cumulative return graph to illustrate the profit and a table of statistical results) after running all strategies.
2. Investment professionals should be able to continue to select other results to compare after that.

Special case:

1. Investment professionals can’t see the plot strategies that have not been backtested.
2. Investment professionals can’t select nothing and get plots.

The acceptance test process:

Common case:

1. Investment professionals can select which backtest result to visualize and compare. There is a "View" button on the selection page.
3. After IP checking which backtesting results and clicking the view button, a new window will pop up to show the results.
4. IP can continue to choose other results to visualize.

Special case:

1. The selection page will not include any strategies that have not been backtested yet.
2. If IP didn't select any result and click the "View" button, a warning will show and nothing will happen.

Part 3. Automated build and unit testing (10 points max)
4 points max for the configuration of the build/package manager. 2 of these points for does it look like it will indeed automate build or packaging of the team's application or service? 2 of these points for does it look like it will indeed invoke automated testing?

6 points max for the unit test cases. Does each method, other than constructors, etc., have at least one automated test case? Take off 1 point for each major method that does not have at least one test case, up to 6. For example, if there are 7 methods without test cases, this part is still graded 0, not -1. If the project has less than 6 major methods, ask the instructor to look at it.

build/package manager: conda + pip. The way we specify the dependencies is to **env.yml**
Automatic testing: pytest locally, pylint locally+**git action**

We also test building or package environments with env.yml in **git action.** Env.yml is also used in locally build a conda environment with **conda create -f env.yml** command

We build unit tests to test each package and helper functions in our codes. Unit tests are in **/tests/unit** folder. We test each testable helper function and utility functions in our codebase. Please take a look.

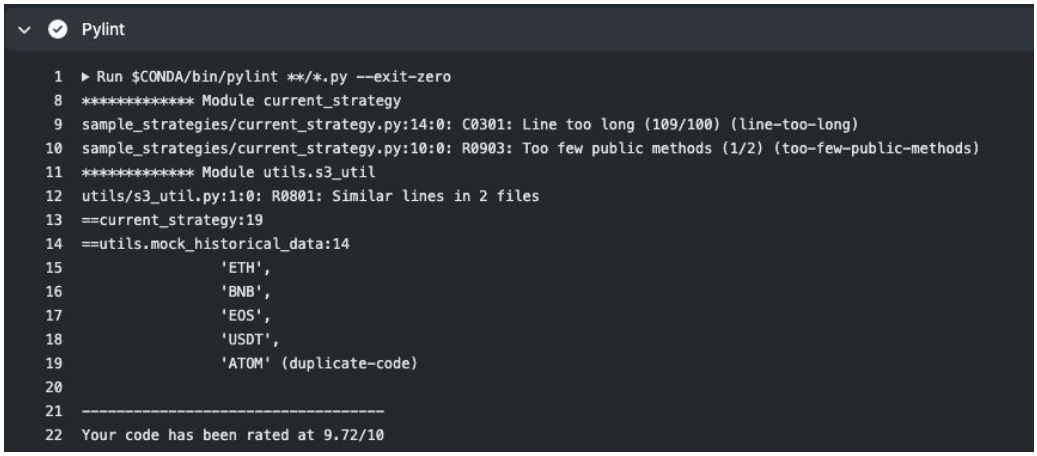
Part 4. Style checker and bug finder (10 points max)

4 points max for the style checker. Is there at least one clean (no errors) style checking report in the repo that appears to be from the latest version of the code (before the deadline for this assignment)? It seems inconceivable that it would not be possible to fix all style errors, but if this comes up, ask the instructor to look at it.

We use PyLint to check style and find bugs. In addition, we display PyLint status on README.md as shown below.

README.md

Also, our Git Action outputs our style check report:



6 points max for the bug finder. Is there at least one clean (no errors) bug finder report in the repo that appears to be from the latest version of the code (before the deadline for this assignment)? Or, if the latest bug finder report still reports errors, is there a plausible written explanation about the difficulty finding/fixing these errors and/or a plausible explanation as to why the reported errors are false positives?

Our current Pylint scores 9.72/10. The “duplicate code” error comes from our “utils.mock_historical_data” method where we are manually including some mock data for demo purposes.

Similarly, in the sample strategy folder, the current_strategy.py only has a few methods inside the Strategy class. Because it is a very simple strategy, we have very few methods. Therefore, we would classify these two error reports as false positives.