Project Info

This project is a simple demo app created for interview purposes.

Test Objective

Objective of the test is to find out if the app meets the Acceptance Criteria or not.
And the acceptance Criteria are as follows:

As a UI user I can:

1: Register through web portal

2: Review my own user information from the main view

As an API Consumer I can:

1: Review users registered in system

2: If authenticated I can get personal information of users

3: If authenticated I can update personal information of users

Approach

UI test was done simply with Robotframework where for API testing, two different approaches were taken. First approach was a simple api endpoints test using the pytest framework. It implements proper use of fixtures also.
Second approach was writing python functions for api requests which were later tested with unittest framework.

Test Summary

First, the test was done as a UI user. For that, the test was performed using robotframework.

Both the acceptance criteria for UI users were met meaning the robot was successful in registering the user and logging them in to review the User Information.

Additional test was done by providing the same username to register again and also providing the wrong password to login. Both the tests failed like they should because proper testing is not only passing when right information is provided but also failing when wrong information is passed.

Log reports of UI tests can be found in the "*ui_test_reports*" folder.

As for the API Consumer, all the acceptance criteria did not meet. It was not possible to create new user nor was it possible to update personal information even after proper authentication was passed. It throws a severe "500 server error" when trying to update the personal information. It is a severe bug.

Total of 13 different tests were run for the API endpoints out of which only 9 passed. All 3 criterias were tested providing, token, wrong token and no token at all.

Bugs found from API test:

1) Consumer can view users even without token
2) Consumers can view users even with wrong token
3) Consumers can not update update user information even when authenticated
4) Consumer trying to update user information with wrong data format gets 500 server error

As from the bug itself we can see that there is the biggest security breach with the app as anyone can access usernames.

Console logged "*AttributeError: 'dict' object has no attribute 'iteritems'*" While trying to PUT request to update user information, this means the the version of python instructed in the app and the version of python the app was written for doesn't match as for python3, dict.iteritems() was removed.

Log reports of UI tests can be found in the "*api_test_reports*" folder.

Exploratory Testing

As it is true that exploratory testing discovers overall more defects than scripted testing. Specially for the UI, it discovers even more.

So from my exploratory testing, I found that once the user is logged in, user can review User Information but when, "Demo app" is clicked, User Information disappears and there is no way to get it back. User needs to log out and then log back in again to see the same information. This is a bug and need to be fixed.

Improvements

The first thing to focus is on securing the data as data privacy is the biggest issue in the current scenario. Second thing is to consider the usability of the app in the future also when the newer version of the libraries that the app is dependent on arrives. It is also important to make it stable while updating the app so that it is easier to maintain in future and do not invalidate automated tests.

It is important to fix all the available bugs.

How do you review code?
Code review is important for improving coding skills, learning better and cleaner ways to write codes, to learn different approaches, to understand logic and performance of code and to solve problems. I prefer peer code review and before doing so, I review the code in brief and try to understand myself, the logic, approaches, style and performance. Once that is done, during the peer review I like to hear briefly from them about the code, approaches and so on. And finally I would give my feedback. Code review is about improvement and not competition so it is very important in a team while doing projects.

How do you make sure that the product is testable?
If the product meets characteristics like observability, simplicity and stability then it is considered as testable products. Also if testable requirements principle is followed, then it becomes easier to test.
I will simply follow the mentioned characteristics and as for the coding part, I will make sure to use KISS principle.

How do you enforce coding standards?
Different companies have their own specific standards required for the software and I enforce those standards simply by following rules. As for individuals, I try to make code readability by using proper naming conventions, segmenting the code, using proper comments and documentations. I try to force myself to write clean codes.

How do you plan what kind of approach you take for test automation - what libraries to use, how does it work in a couple of years, how to make it easy to maintain, etc? What are the main points to consider?

I think the question answers itself. As per planning, the first question that should be asked is what is the goal? Once that is answered then only I would move to the next step which can either be looking for the framework first or identifying the tools and technology. I would rather go with selecting the framework first and then identifying if it needs additional tools(libraries). In doing so, the main point to consider is to research how stable those tools are, if they are open-sourced or licensed versions, what is the budget if I am about to choose licensed versions. In order to find the right tool doing personal research, asking colleagues or friends and exploring the most downloadable tools are some options. So all in all I would

- List out project requirements (To find the goal)
- Define the budget for test automation
- Consider tech stack (finding frameworks and tools)
- Analyze and compare the found tech stacks
- Verify the choice
- Make final decision