Virtual Labs: Circular Dichroism Spectroscopy Lab

Jalees Jahanzaib Mayank Goyal Naren Akash R J P Meena Raja Sree

International Institute of Information Technology Hyderabad

Software Testing Analysis

Acceptance Testing

User acceptance testing is performed by the client to verify the software system developed by our team of developers. After different rounds of unit testing, integration testing and system testing, we cross-verified with our project SRS (Software Requirements Specification) document the objectives and the scope of the software.

We created specific test cases and then applied them in front of our client (over a video conference call) to show them the outputs of the integrated software. We rechecked the functionalities with the requirements specified in the SRS document as well.

We used black box testing method for carrying out acceptance testing. We took several positive as well as negative test scenarios to verify if the software complies with the desired outputs.

- Functional Testing: We went through random experiments and performed several operations to check if the functionalities of different objects are working properly.
- Non-Functional Testing: We randomly checked different modules to check if the module has good user experience as well as has good performance.
- Regression Testing: We run different test cases again to check if different modules work well after all other maintenance done in our codebase.

For functional testing and regression testing, we used **Selenium**, an open-source web automation tool to verify that we get the desired outputs.

System Testing

In system testing, we validate the complete, fully integrated software product. We evaluate the end to end specifications of our product. We also test the user's experience with the application.

In our project, we have 10 different experiments under our CDS lab. We go through each of the ten experiments, start from the introductory page and follow the sequence of the experiment to thoroughly check if each step of the experiment follows the experiment manual and gives us the desired output.

- Regression Testing: We select different combinations of test cases used during integration testing and re-run them to check if they succeed after complete integration of different modules. We used **Selenium**, an open-source tool for browser-based regression testing. It automates the complete web-based laboratory sequence.
- Usability Testing: We check how user-friendly the software system is. We look for user's ease to use the application and flexibility in handling the controls.

For example, in experiment 3, we start from the introductory page and check if all the molecular visualizations are rendered. Then we select the wrong options in the quiz to check if we get failure status. Then, we proceed on by entering the correct options. Then, we choose different samples in the experiment setup and check if the corresponding plots are rendered properly. This process is automated by using a tool called Selenium which helps us in writing automated script for the same. We also check if the colours used for different objects are appealing enough for the users by getting feedback from fellow developers and check if the page navigation and buttons are placed properly.

Integration Testing

Here, in integration testing, we test combinations of different units of a system to assess if they work properly together. By testing the units in groups, we can easily identify if any fault is present in the way how different units interact together.

We have used sandwich testing (aka. hybrid testing). In this method, the top modules are tested with lower modules and at the same time lower modules are integrated with the top modules and tested.

For example, in experiment 8, we have an introductory page where the major steps in the experiment are mentioned. (Layer I)

In the first step, we see the visualizations of the protein molecules at different temperatures. In the next page, we get to know about the agent used in the experiment and its corresponding visualization. (Layer II)

From the page which contains the information of the agent, we are redirected to the animated experiment where we click on the respective objects mentioned in the instruction bar to perform its function. (Layer III).

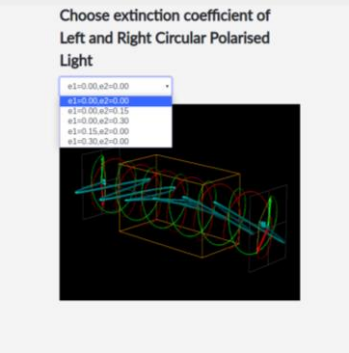
Now, we obtain different test cases by:

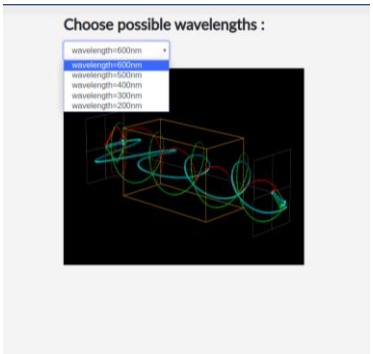
- Testing each layer separately
- Testing after combining layer I and II and II and III. For example, from page 2, the test script clicks on the desired object as per the manual and verifies the corresponding change.
- Testing after combining layers I, II and III.

The main advantage of sandwich integration testing is that it simulates the actual sequence of a user of our project. A user can go forward as well as backward in our experiment. So, this kind of hybrid testing allows us to rigorously test for faults.

Unit Testing

The aim of unit testing is to test each part of the software by separating it. The, we check if each of the units – the smallest testable piece – is fulfilling its functionalities or not. This helps us to test each module separately. Here, we primarily verify the flow of inputs and outputs through the application.





In our software product, we used **PyUnit** tool for white box testing. For example, in experiment 2, we iterate through the options in the drop-down menu of each one of the four different experiment. Then, we check if the corresponding video/image is rendered. Thus, we have used the branch coverage technique.