Software Testing Plan Version 2 for Team “Fish Out of Water”
4/8/2024

Project Sponsor: John Fennell
Faculty Mentor: Saisri Muttineni

Team Members:
Jack Shanley
Corey Moreno
Nicholas Robishaw
Jaron Bauers

Table of Contents

Introduction

In the canyon bordering Northern Arizona and Southern Utah lies Lee's Ferry, renowned in the community of anglers for its rainbow trout fishing along the Colorado River. The Arizona Game and Fish Department is tasked with safeguarding the trout population which is vital to the river's ecosystem and is an important part of the local economy. To do this, they diligently monitor the fishing activities that go on at Lee's Ferry. However, traditional data collecting methods, such as in-person interviews with anglers, are difficult to conduct efficiently for the sake of larger scale data collection. To bolster conservation efforts, the Arizona Game and Fish Department implemented a game camera near the dock, capturing upwards of 1,800 images daily. Our client, John Fennell and his team, have to review these photos manually in order to be able to provide insight into traffic on the river as well as current fishing levels. These photos are extremely useful to John and his team, but the manual review process is highly labor intensive, highlighting a challenge of balancing resource limitations while supporting conservation efforts at Lee's Ferry. Our desktop application aims to ease this burden for John and his team. Utilizing the Python tool, YOLO, we have implemented a program that automates much of the image recognition process which will save John and his team several hours of having to look through pictures to identify various types of watercraft.

Our program utilizes a simple and efficient front end that is integrated with the backend logic of our program with the help of our integration tests. After new functionality is added to our code base and successfully integrated into our GUI, we will add/edit unit tests for our new methods. Finally, run the unit tests to make sure they are all passing as expected. This document will go into more detail about the testing strategy that we have implemented to ensure that our product goes off without a hitch and is a successful solution to our client's problem.

What is Software Testing?

Software testing is a vital part of the software development process and ensures that the deployment of your software goes smoothly. There are many different ways to test your software but we have mainly focused on integration tests and unit tests for this project. Unit tests focus on testing our methods' functionality while our integration tests were steps we put in place to make sure our methods worked against our GUI properly. After adding new functionality to our code base, we utilized these tests to ensure that our code was working accordingly. Using software testing in parallel to our development process helped us immensely in developing our code and for seamless deployment.

Unit Testing

Unit testing is a critically important part of the software development process. Unit testing is a testing technique where individual units and/or components of an application are tested by themselves to make sure that they are working properly. This is done to ensure that the code executes and functions as expected according to the design specifications that have been developed in tandem with the customer. Unit tests can cover functions, classes, or even smaller parts of the code. Unit testing allows developers to catch issues early on in the development process by testing things independently. Ultimately, this leads to higher quality code as well as more maintainable code. This is important in the modern software development landscape as it promotes a more robust and reliable system. It is important to be utilizing methods such as test driven development.

For our program, we decided to implement Pytest for our unit testing. Pytest is a popular testing framework for Python as it simplifies writing and running tests. It has a useful set of

capabilities and it was suitable for the features we were looking for. Because of Pytest's versatility, it is popular in the developer community for both small-scale projects and large-scale projects alike. Pytest also supports a variety of types of Python applications, making the testing framework adaptable to tech that changes. Pytest will allow us to focus more closely on the functionality of our code without having to develop and write complex setups for our testing.

Our program has several key functionalities that need to be tested. Our application operates in three main modes including sorting mode, processing mode, and training mode.

Unit Testing - Sorting Mode

Testing our sorting mode will cover functions that carry out various tasks. These include functions that iterate through a large folder containing several hundred photos that are taken by the game camera at the dock. These functions were often tested during development. Typically, we would isolate functions as we went to confirm proper working order. Testing this functionality is important to ensure that our program can correctly and confidently iterate through the regular influx of photos that will be fed to the program. It is also important to test that it correctly grabs the metadata from those photos for the sake of sorting them correctly by date. This is crucial for the organization of the photos before they enter processing. This will allow for the photos to be presorted and will still appear in chronological order even after processing. Our client highlighted this feature for the sake of usability for himself and his team. Our unit testing for this mode largely centered on functions that were responsible for fetching the metadata that indicated the chronological order of all the pictures in a given folder. Our unit tests for functions, "organize_pic_by_date" and "get_image_time", were critical to test the sorting mode within our program.

Unit Testing - Processing Mode

The next section to begin testing in is processing mode. Processing mode will also have to iterate through all of the photos, thus testing that it can do so correctly is foundational for ensuring that all images are processed. As for the actual processing, the program is responsible for checking the current photo and comparing it to the training data. If the photo contains a boat, kayak, or raft, then a confidence interval will be assigned depending on how sure it is that the object is what it thinks it is. And depending on what the floor confidence interval is set to, that photo will be put into its respective folder (archive, review, or discard). Testing this section of the processing is vital to completing a successful product, as this part of the processing is the crux of the project. Thus we will have to test that the confidence interval gets set correctly and that photos are placed as expected according to their contents. This was important for implementation in the function, “find_boat”. Unit testing for this part of the code was also utilized on our “move_file_to_folder” function. After processing and categorizing all the photos is complete, it is then necessary to test that metadata from each photo is able to be written/exported to an excel file, according to the requirement specifications that were discussed with the client. Testing this functionality ensures that their team can accurately procure and organize data. The excel file, the folders, and their respective photos then need to be outputted to their correct destination in the file system. Testing these functions will confirm that the information gathered is readily available and accessible to our customer’s team for further analysis. Unit testing for this aspect of our program was conducted extensively on our “write_to_excel” function. Additionally, a shortcut will be available within the GUI that will take the user from the graphical interface directly to the location where the data resides in the file system. Testing this feature ensures a simpler and more efficient user experience, which is important to our client.

Unit Testing - Training Mode

Lastly, we are testing our training mode, this is currently a work in progress. Test driven development will be utilized extensively for this section. This mode allows the user to train based off of new data, further aiding in making this product multi capable, adaptable, and modifiable. The user is responsible for creating their own model, however, we will be providing input validation to ensure that the model is correctly configured for program execution.

In order to conduct testing for most of the functions, or “units”, we will be utilizing objects called mocks. In Pytest, mocks are able to mimic the behavior and characteristics of real objects that are tested within the system. Pytest provides a built-in mechanism that allows a developer to create their own mocks, this is possible with the use of the ‘pytest-mock’ plugin. This offers a straightforward way for creating and managing our mock objects for our testing functions. The following describes an example in which we would utilize mocks for our unit testing.

Function to test

```
def splitdatetime(dateTime):
    strList = dateTime.split(' ')
    date = strList[0].split(':')
    date = swap(date)
    dateString = date[0] + '-' + date[1] + '-' + date[2]
    timeString = strList[1]
    return timeString, dateString
```

The function to mock (swap)

```
def swap(date):
    return [date[2], date[1], date[0]]
```

Test function

```
def test_split_date_time(mocker): # Mock the split method of strings
    mocker.patch("builtins.str.split") # Mock the swap function
    mocker.patch("__main.swap", side_effect=lambda date: [date[2], date[1], date[0]])
    # Set up the input and expected output
    dateTime = "2024:03:23 12:30:45"
    expected_date = "23-03-2024"
    expected_time = "12:30:45"
    # Call the function under test
    timeString, dateString = split_date_time(dateTime)
    # Assert the expected output
    assert timeString == expected_time
    assert dateString == expected_date
```

Using mocks, we will be able to test most of our code in a similar fashion. Isolation of each section will ensure that things are working as expected in a variety of test cases. It is important that we do our due diligence so that the client doesn't have issues with the product that could have been prevented in development.

Integration Testing

Integration testing is another area of testing that is important to investigate for the sake of a successful software project. Integration testing is especially relevant for programs that involve APIs or a web implementation. As far as integration testing is concerned, we did not have a significant amount of code that needed to be integrated since we didn't utilize an API and we didn't create a web application. Our program was solely a desktop application. Most of our testing for the GUI was implemented in the form of test-driven development, a lot of the testing was done as we went. Most of our testing was done on the backend logic, so most of our concern was concentrated on ensuring that it was functioning properly. Testing for the front end was ensuring that the interface would respond accordingly based on user inputs. An example of this was testing for the confidence interval alteration bar actually having an effect on the confidence interval in the program. A successful test of this was evident in the outcome of the programs, meaning that there would be more or less photos in the archive folder according to the confidence interval. We also tested to ensure that error handling in the GUI was present to ensure that the program wouldn't be run without proper inputs.

We created functions that would do some operation on the user's computer like creating new directories, moving files, sorting images from a file, reading in data from their computer, etc. We would test that the functions would work as expected by feeding in hard-coded values and checking to make sure that the data we entered was processed correctly and giving us the outcome we expected. After ensuring that the function worked properly, we would then hook it up to our GUI where we would feed that function's dynamic variables and ensure that the function was again giving us the same outcome and working properly with the data being

transferred from our GUI to the new function. After successfully integrating this new logic into our GUI we finally would test our entire desktop application to ensure that the program was working properly. Although we didn't have much integration testing, it was crucial that we implemented a smooth connection between our backend logic and frontend interface.

Usability Testing

Usability testing is important for any software design process. Usability testing is a testing technique that is focused on how the software and the user interact with each other. This is crucial to ensure that the software that the user will be operating on is easy to use and understand. Our client and their team are not technology professionals, therefore it is important that our software is designed in a manner that is straightforward so that anybody can pick up the software and use it without any struggles. In order to achieve successful usability testing, it is vital that we include the user (our client) throughout all phases of development so that they see exactly where their product is through each step of the process. This needs to be done in parallel with testing the product ourselves. We accomplished this by having weekly, if not biweekly, meetings with the client in which we showcased a detailed orientation of what our progress was and how our program was looking. This became especially important when showing the user the GUI. Additionally, the product will only be used by the Arizona Game and Fish department at Lee's Ferry, so the program needs to cater to the needs of this group as much as possible. If the program does not fit the usability requirements that our client needs, the program will cause more problems than necessary and will slow down the process of iterating through the pictures.

When it comes to testing the program for usability, we want to make sure that our client is satisfied with the design, functionality, and accuracy of our program. The first step was to

check with our client to see what they thought of the GUI design. After showing our GUI design and demonstrating how the program operates, our client informed us that the GUI was easy to use and understand. However, we did implement a section of our GUI that our client does not want anymore. To ensure usability, we will modify our code to take out the unwanted section and in our next meeting, we will demonstrate our changes and the functionality of the program once again. Once we get another confirmation from our client that the GUI is complete and easy-to-use, that will assure us that the program is usable and meets all expectations. An effective and simple GUI is not to be understated, it ends up having a massive impact on how the user will interact with the product. Even if the logic on the backend works well, a bad GUI will undermine all of that.

The next step in testing usability will be to run the program on different machines to see how fast the program will perform. Our client's machine is an HP EliteDesk VPro 9th generation computer, which has an i5 processor and about 8 gigabytes of RAM. In order to ensure that this program will run on that computer, we need to test this program on a machine that has similar components. Most of NAU's computers have i5 processors and contain anywhere between 8 and 16 gigabytes of RAM. Once we test this program on a computer of this kind, it will allow us to gauge how fast it will take for the program to run on the computer at Lee's Ferry. From there, we can modify and refactor any code that might slow down the image processing process. And although achieving an efficient time is important, the time taken for the program to run will ultimately still have a significant impact on the man hours that are spent by John and his team analyzing photos. A large part of our design for this program was for it to be easily maintainable and modifiable. Although our client has changed their mind on a couple of their requirements. It is a lightweight program that is simple to follow and thus not difficult to add on to in the future

as needs arise. We wanted our client to be familiar with training the model so that when new boats are seen on the water, the program can be trained to recognize them as well. This was taken into consideration when choosing the software that would be used for annotating pictures, since this is also a simple process that the client can implement themselves if necessary. These are the aspects of usability testing that are most relevant and pertinent to our project.

Conclusion

Overall, the purpose of testing this program is to see from a developer's point of view what can be improved and what would most likely break our program. Pytest is a great tool to test if the project is running as it should and if it handles errors as it was programmed to. Using these helpful feedback tests, we as developers can add more error handling and anticipate how our program will handle bad inputs and outputs to limit crashing. This way, the customer doesn't have to worry about reaching back out to us to fix mistakes that should have been addressed in the first place. Throughout the whole project, we as the developers strived to make this project as simple as possible for the customer, and we also know that there will also be a ton of different inputs that the customer will accidentally select. Once testing is completed, we will see where the program is at its weakest with certain input ranges and be able to refine those functionalities to aid the customer in inputting the correct information needed to run correctly. The majority of the functionality in this project is based on assumptions, like many other real-world programs. This is easy to initially develop but difficult to validate the inputs since there are so many variables to keep in mind. Testing is a vital part of this project, and the development team would like to give our customers a really easy and simple project to interact with and learn from.