Software Test Plan

GeoLearn

3/27/2020



Mentor: Fabio Santos

Sponser: Dr. Chris Doughty

Team members: Samuel Prasse (lead), Kaitlyn Grubb,

Tyler Pehringer, Joshus Tenakhongva

**Table of Contents**

1. *Introduction…………………………………………………………………….pg3*
2. *Unit Testing……………….…………………………………………………....pg4*
3. *Integration Testing……………………....…………………………………....pg8*
4. *Usability Testing …………………..……..…………………………………...pg9*
5. *Conclusion……………………………………………………………………..pg11*

**1.0 Introduction**

Enviroslides is an application that automatically generates Google Slides presentations based on local environmental information. The application should be able to pull information about different topics on the4 environment; such as biodiversity, climate change and land-use change. The presentations must have some form of interactivity such as a game called Dobble, which is a form of a matching game. These presentations should then be able to be sent to the users Google Drive account so that they are able to edit them if they wish.

 This leads us into the software testing itself. Software testing is great to ensure that we can get rid of any major bugs in an application. We can use software testing to ensure that our users have the best experience possible. We can use unit testing to test each individual module of the application itself to ensure that they are all running as smoothly as possible on their own. We then move onto integration testing. Integration testing is making sure that after all of the unit testing is done, we then check to see how each module interacts with each other. This will allow us to find even bigger bugs in the application. We will also be making sure that the overall application is easily usable for all users.

As a plan, we will be doing unit testing on being able to pull data from the databases our client has provided us. We will check to see if information can be formatted onto slides formatted by our client. We will test each important module for accuracy if information, such as making sure the animal information is correct based on the database. We will then test how each of the modules work with each other by making sure information is sent to a user's google drive with the right information. We will also be making sure the application can be easily navigated by the end users by testing it out in classrooms.

**2.0 Unit Testing**

Our software has four main components: data scripts, individual slides, website, server. Each will be tested separately so that when all these components are working together we can assure that the software is working as expected. In this section the specifics of testing all of these separate components will be discussed.

***2.1 Testing Data Scripts***

Go over methods in scripts and how to test separately

To test our many different types of data scripts, we will have to perform several different types of tests. These scripts include the Google Maps API, the Python scripts to obtain a time lapse video, and the biodiversity database scripts. Now that we have defined the scripts we will be testing, we will now examine each test case.

*2.1.1 Maps Location*

This test case takes a look at our Google Maps API. This map that is provided from this API is very robust and well developed meaning that there is very little that we can do to test it for failure. If the tester were to scroll out until the world was shown multiple times on the map and then selected a latitude and longitude, this may cause an issue with the form input since the API might not pull the correct location.

*2.1.2 Bad Latitude and Longitude input*

In this case, the tester would provide a set of bad coordinates to our submission form. We would have to test the edge cases of “good longitude and bad latitude” and “bad longitude and good latitude” as well as “both bad longitude and latitude” to fully understand what our program would do with this data. Depending on the results of this test we may need to input failsafes into the code that only allow the user to input valid latitudes and longitudes as determined by a regular expression or some other verification method. Submission with no input would result in no video being added to the slideshow presentation every time this function was called.

*2.1.3 Video Conversion*

The video conversion function of our project would need to be tested with a video file that is located in the GeoLearn Google Drive. From here, the video should be downloaded automatically by id and name and then converted using the ffmpeg tool. Once the video file has been output as a gif on the local machine, it is then uploaded back to the Drive as a GIF, deleted from the local machine, and included in the presentation using a find and replace function in the Google Apps script code.

*2.1.4 Database Scripts*

Our database scripts are run using valid longitude and latitude data from the earlier Maps and Python scripts. Once the biodiversity script has the correct location, it will then search the database of shapefiles for animals that are within that region. We have ensured that appropriate animal data and pictures are gathered by searching the internet using their scientific names. This guarantees us that we will find an appropriate image of the animal and provide current and accurate data to the user.

***2.2 Testing Individual Slides***

In order to test the slide creation a folder dummy data will be put into the Google Drive. This data will cover all edge cases, such as not enough information to fill the slide completely, to make sure the slide generation script can handle this. This process will need to be completed for each slide type and difficulty variant. Let's look closer at the test cases.

*2.2.1 Picture Size*

One picture will have a big resolution. This makes sure that the picture will not overfill the slide. Another picture will be a vertical picture replacing a horizontal picture. This makes sure that no stretching will happen when pictures are inserted. The quality of pictures will be tested in the data scripts section.

*2.2.2 Picture Doesn’t Exist*

There will be a missing picture to cover when there is not enough data to fill a slide. This makes sure slide generation will still work with just one missing element.

*2.2.3 Type Checking*

The test will insert every type of data used in the slide show to make sure that all kinds of data such as video, picture, text, table, and graphs are all supported and inserted correctly.

*2.2.4 Slide Permissions*

In order to test that the slide show can correctly transfer ownership, the team will insert a Google account that is not the GeoLearn account and ensure they have access to edit, save, and delete the presentation.

*2.2.5 Incorrect Information*

When wrong information is given to the slide generation it should produce an error and explain to the user what went wrong. A non-existing or non-Google email should report the slide could not be transfered. Entering a point in the middle of the ocean for latitude and longitude should return no animal data. If there is no animal data this should be reported to the user and cancel the slide process.

***2.3 Testing Website***

The website can be broken down into pages. For each page we need to ensure that no errors are thrown and all buttons on the page have the correct functionality; however, we do not have to consider that these functions are correct because that will be covered by section 4.1, testing data scripts. In addition to these more specialized tests per page there will be a script that connects to each page in the server and makes sure all pages can be accessed without error. For each page we will have a checklist to hand check buttons and user input. We chose to hand check because we have only a few user inputs in the website and an automated test would be excessive. Let's consider the pages in more detail.

*2.3.1 Toolbar*

Each page has a toolbar at the top. Every page needs to be able to redirect to other pages from this toolbar. In order to test this from each page listed below we will navigate to every page listed in the toolbar from left to right. This ensures that all pages are linked up correctly.

*2.3.2 Home Page*

The home page only holds a welcome message and links to other pages. In order to test this we will navigate to all links to other pages on this page to ensure that all embedded links are correct.

*2.3.3 About Page*

The about page displays information about the team and the project. In order to test this page we will navigate to the linked web pages (personal sites of the team members and project web page).

*2.3.4 FAQ Page*

This page just contains text. There is nothing specific to test.

*2.3.5 Slides Topic Page*

This page has three links, one that redirects to a slideshow from per topic. In order to test this page we will make sure that every topic redirects to it’s respected slideshow form

*2.3.6 Slides Form Pages*

This group of pages contains the bulk of our website testing. This test should be repeated for every slideshow topic (currently there are only three). For each page the form gathering needs to be checked. In order to check the values are getting correctly pulled our team plans to log all form fields to the console and check that everything matches accordingly.

***2.4 Testing Server***

Testing our web server is a critical aspect of this project. To do this, we will be ensuring that the correct packages are installed, the server has the correct amount of memory and storage, and provide a log of all data when necessary.

*2.4.1 Server is running*

To test if our server is operational, we can do one of two things: the first is send a ping command to the server and check if we received the packets back from the Amazon Web Server. If we have received all packets, then we know the server is operational and connected to the internet. The second being logging on to the Amazon Web Service console and checking to see if the specific EC2 instance in which we are hosting is live and operational.

*2.4.2 Installed Packages*

To test if we have the necessary packages installed on the web machine, one will be able to use our list of packages on the Github and manually check the versions of each package running on the machine. Or, one could run a Python script we have developed specifically for development on the web server to ensure that we have the correct versions and types of packages installed for deployment.

*2.4.3 Memory and Storage*

 Testing the amount of memory and storage is completed when the web application is running on the server. If there is a lack of either of these critical components, the web page will inform us with a redirect to a page stating what we are lacking. Another way to perform a live check is to use the command Htop. This command will show the user all activity and connections on the site as well as the memory the application is using in total.

**3.0 Integration Testing**

Integration testing is testing that ensures that each individual module is able to interact with each other. The purpose of doing this is to make sure that we can find any bigger bugs. This could be because the modules are not compatible with each other in their current state or they do run, but the results that come from the interaction do not come out as expected. By the end of these tests, we should be able to make sure that the product can go into a proper beta phase for users.

 There are a few goals we have in mind when it comes to integration tests.

* Each individual code snippet responds to each other.
* Each code snippet works together to produce the expected result.
* Each code snippet has all of their major bugs worked out before the beta build.

***3.1 Front end to Scripts***

 With this test, we need to make sure that we can put in information on the website and have that information manipulated in the scripts. To do this, we will be entering in basic user information on the website to create the slides themselves. This will be user email, school name, latitude and longitude and topic of the presentation. Once they push the submit button, they can then see if the scripts properly grabbed the information by checking the information on the slides and then we must check the information that is based inside the databases we retrieved the info from. This will be information from the Google Earth Engine and the Animaps databases and if the information maps up to each other, then we know that the scripts and the front end work coherently to each other.

*3.1.1* *Map API to Scripts*

 We need to make sure that the map API on the website is able to place the chosen latitude and longitude onto the scripts. This will need to read where the user places the marker on the Google Map on the website and records the lat and long of the marker. And that is how we will test its accuracy.

***3.2 Scripts to Slides***

 With this test, we need to make sure the information is saved to a person’s Google Drive account. This will be achieved by first making sure that this information is sent to our own Google Drive account before we send it to the users account. We need to make sure that the slide formats are placed inside the GeoLearn Google Drive account, once there, the scripts should be able to place information correctly. In the scripts, there should be a way to send the newly filled out slides to the users Google account so that they will be able to edit the slides if they wish to do so.

*3.2.1 Slides based on topic*

 We need to use the topics button on the website to run scripts to ensure that we are accurately pulling information. This entails that we check the information once they show up on the slides themselves to ensure the most accurate information possible.

*3.2.2 User info on slides*

 We need to ensure that the info the user inputs onto the websites is also properly displayed onto the slides themselves. We will do this by entering in test info and check the slides once they are generated to make sure that the info is placed correctly.

**4.0 Usability Testing**

Usability testing is testing your product with real people and observing their behaviors and reactions to the interface. The purpose of doing this is to make sure that the actual experience of using the product is effective, understandable, and as bug free as can be. Since virtually everyone on the development team will have some knowledge of the product, their usage of the product will differ from the target audience who will likely use the product with little knowledge beforehand on what exactly to do.

 Usability testing typically has a few goals in mind. Testers want to check if user can:

* easily understand the user interface (UI) of the product
* navigate the UI to accomplish their goals,
* go through the UI without encountering bugs or other technical issues, and
* generally have a pleasant experience.
* Notice any issues looked over by the development team

With this information, we can do a few things.

* Ensure that our prototype works as planned
* Check that users can effectively follow the steps to complete their goals
* Show *Why* other issues may occur

 There are many ways to test usability, but there are a few characteristics of usability testing that we can look at to guide our testing regime. Our test can be moderated or unmoderated. The test can be conducted remotely or in person. We can also explore different test methods, depending on exactly what kind of results we want to find through explorative, assessing, or comparative tests.

 Moderated testing is when you have someone there to guide the user and answer questions, and unmoderated is leaving the user on their own. Tests done with a moderator typically yield more detailed and useful results, but they require more resources to conduct and organize. Unmoderated testing is typically done through surveys and follow-up questions, greatly reducing cost and allowing for a higher quantity of answers. Typically, moderated testing is used to find the reason behind behavior, and unmoderated is used to observe and find behavior patterns.

 Remote testing is done either online or through the phone, and in-person testing is just that, in-person. Similar to moderated testing, in-person testing yields more detailed results, but remote testing allows for a large pool of testers to be used. It’s a conflict between wanting qualitative and quantitative results.

 We also have a few testing methods that we can apply. Explorative tests are taking ideas from testers early on and implementing them with developer insight, then testing them on their own. Assessment testing is used to test a user's satisfaction after they have used the product. Comparative testing is giving testers options, then having them select which one they prefer.

 It is also important to take into account details specific to our project.

* Our target audience is teachers and young students
* There aren’t a lot of automated slideshow generators available, so exactly how our product works might need some explanation
* Since our product is meant to make the process of making education material easier and faster, a poorly designed UI would not incentivize customers to use our product over handcrafting slideshows.

 Since our target audience is people who may not be very tech savvy, we want to ensure that our UI is as simple, effective, and well communicated as possible.

Due to the budget constraints of our project, we will likely opt for a very short round of moderated testing followed by using unmoderated testing through surveys and questionnaires on the website. We will want to begin with some moderated testing to ensure that we begin on as strong a foot as we can. So far, our UI is very simple, requiring only a few clicks and decisions, so it shouldn’t be too costly to run these moderated tests. From there, we will mostly want to refine our UI since we will have the main structure of our website design laid out.

Like the moderation of our testing, we will begin in-person, then use remote surveys and questionnaires for most of our testing. We will ask questions primarily about how well we were able to convey how to use the website, if there were any confusion directions, and the ease of each step the user will have to go through to create a slideshow.

Our surveys will be assessment tests that ask the user about how satisfied they were using our product. With these kinds of tests, we should be able to obtain a large amount of data that either we or future developers on this project will be able to use to refine and build the UI.

**5.0 Conclusion**

 With these tests, we hope to achieve our goal of making sure that we can deliver comprehensive environmental presentations that are automatically generated for students of all ages. With basic unit testing, we believe that we will be able to ensure that each individual module works with minimal bugs, to our abilities; for example, making sure that pulling data from the databases with our scripts runs properly. With Integration testing, we believe that we will be able to make sure that each module interacts to each other in the most appropriate way possible. This includes making sure that the website and the scripts interact with each other properly to gather the right user info to place onto the slides. Lastly, we are hoping to have in person testing with our target users to ensure that they get the best experience possible. In the end, we think that we are able to deliver a product suitable for our users needs with minimal bugs possible, ensuring to enhance the students experience learning about environmental information.