Testing

Unit Testing

For our education tools, unit tests need to be done within the infrastructure of our website, or any other media platforms we create. Testing buttons, pages, links, and scroll features to only name a few. In order to do this, small portions of the website will need to be isolated in order to test if these features work properly.

Another part of unit testing that could be important regards the user experience. Since we plan to run this website on Iowa State servers, we need to test whether users outside of the Iowa State network can access this website.

Interface Testing

We will be hosting a website which should get on average between 50 and 100 hits per day. Usually from the same users, almost all of whom will be on the Iowa State University Local Area Network.

A Linux Virtual Machine and host will need to be acquired, likely from the ETG in Coover. From there we can host it. Git-Lab will likely be used to keep track of the codebase.

Main interfaces consist of: Client Web Browser <-> ISU LAN Server Web Server

YouTube Server <-> JavaScript/ HTML embedding API

GitLab Project Code Repository -> ISU LAN Server Webserver

Ebookcentral <-> JavaScript / HTML API

Integration Testing

Integration testing will be done after our website is completed. That is, once all the components are designed and tested individually. This will include the testing of buttons, tabs, links, scroll

features, and embedded educational material (videos and interactions). We will test all these components by simulating the average user. This means putting ourselves in the shoes of someone wanting to learn about the material, and make sure the website features run seamlessly.

The backend features like server connections will also need to be tested for speed and reliability.

System Testing

After the above testing is complete and sufficient, we can look at the big picture. We will need to test website reliability in regards to user load and bandwidth restrictions. To do this, we will need to test a large number of users using the website at once to see if the server can handle it. Our main goal is to make the website as reliable to the user as possible, so we cannot afford any crashes or required server resets.

Regression Testing

In order for us to test regression with our website, we need to make sure that new pages of the website do not break it completely. This will most likely be a simple task, as each new page of the website will cause little to no issue. If we do come across some bugs with adding features, we will need to refactor some backend code to fix the issue.

Acceptance Testing

Our non-functional testing for the website will be done based on the user. We will create a simple Google Form that will give us feedback from users. Each user that helps us test the website will be asked questions based on the experience using our website, as well as if the website was educational. We will also be monitoring the quiz results and other methodology of collecting scores within the educational website to analyze how well the users are retaining the information.

We will also be in contact with our client to make sure that the educational material is important for further learning of our users. It is important that our website is engaging to the user, but also provides accurate and up-to-date information.

As far as functional design testing goes, there will be a part in the form that the user can indicate where they had bugs, if any. These responses during our testing phase will be really helpful in creating the best user experience possible.

Security Testing

- We want to make it so the site isn't too abusable if malicious users want to hurt it or deny its use.
- We will have to review our server hoster's operating system security.
- We will have to review GitLab's security.
- We will have to look at who deserves access to modify this website currently and after completion.

Results

Note: Only successful test results as of currently resulted from network performance and tuning.

Network Performance Test Results:

Testers will also use an iPerf3 setup which is a tool for testing network performance and tuning. We use this setup where the client will run on the UE side with the server on the network side. UDP traffic will be generated. Once we start the iPerf3 server, it will then listen for traffic coming from the UE.

```
---------Signal-----------|-----------------DL-----------------|----------UL-
↪----------
rat  pci  rsrp   pl   cfo | mcs  snr  iter  brate  bler  ta_us | mcs   buff ␣
↪brate  bler
 nr  500    29    0  -16u |  28  n/a   1.1   9.9M    0%    0.0 |  28    48k ␣
↪9.5M    0%
 nr  500    25    0  -18u |  27   70   1.1    13M    0%    0.0 |  28    61k ␣
↪ 13M    0%
 nr  500    28    0  -16u |  27   70   1.1    11M    0%    0.0 |  28   6.7k ␣
↪ 12M    0%
 nr  500    30    0  -14u |  28   70   1.1   9.2M    0%    0.0 |  28    48k ␣
↪9.6M    0%
 nr  500    26    0  -13u |  27   71   1.1    12M    0%    0.0 |  28    30k ␣
↪ 12M    0%
 nr  500    31    0  -17u |  27  n/a   1.1   8.8M    0%    0.0 |  28    43k ␣
↪8.8M    0%
 nr  500    29    0  -14u |  27   70   1.1   9.9M    0%    0.0 |  28    52k ␣
↪ 10M    0%
```

```
 nr  500    27    0 -7.0u |  27   70   1.1    11M    0%    0.0 |  28    47k ␣
↪ 11M    0%
 nr  500    26    0  -14u |  27   71   1.1    11M    0%    0.0 |  28    57k ␣
↪ 12M    0%
 nr  500    27    0  -16u |  27   70   1.1    11M    0%    0.0 |  28    49k ␣
↪ 12M    0%
 nr  500    28    0  -10u |  27   71   1.1    11M    0%    0.0 |  28    41k ␣
↪ 11M    0%
```