1.) Internal Testing:

I created manual test cases that test to see the connectivity of the database. This is done by creating a function that will take a snapshot of the database and it will return a boolean statement that is either true or false depending if the database is connected. When I know that the database is connected, I then create a function to see if I can add data to the database, I check this by randomly calling the function with dummy data and seeing if the data gets put into the database.

2.) Unit Test:

There was a problem in which webgazer would not open when you click on the button for a specific test. We tested this also with hard code testing. We first tested to see if the buttons were redirecting to the correct point, which availed to no use. We then check to see if all the functions of web gazers where being used properly. This resulted in a https error. Without this, we can not be able to open webgazer. We then scratched the idea of putting the app in a machine (making it strictly local). The problem was still there, so we tested this by seeing if the camera is being prompted and being activated when the test button was clicked.

3.) Application Test:

Trying to test all the function in our application, I found a bug in the eye movement test. There were times when the video will not fit to the screen and it will just apply to the bottom of the screen and would all be white space. This was a problem with the aspect ratio of the different computers. We tried to fix this by making the video into a specific size for all computers, but this also posed a problem. We tried to make it so it goes straight to full screen, but this will make it so webgazer will not be able to be open while the video is in full screen. Order to fix this we had to make the video as low aspect ratio to modern computer sizes (basically making as small as the smallest screen from our computers). This is just a temporary solution to this problem.

4.) Stress Testing

When our web app is tested with limited connection, webgazer fails to completely load. If the internet is complete shut down while using the web app, data fails to load and user can not login. If the cache is removed then user data is no longer persistent. The application does not work online. That backend uses firebase and mysql, since firebase is limited on the creation of users then if a person where to create a bot of millions of accounts then the service would no longer be free. Mysql uses microsoft azure as the hosting service, this means as the traffic and user data increases the less able the virtualbox can manage such amount of data and traffic, which may lead the machine to break down. There are no service workers which means this web application is not optimized to be a functional web application. In conclusion an internet

connection is essential for the application to function which may be slowed due to traffic or creation of new accounts.