

CA400 Testing Manual

Project Title: LiveFocus

Students: Cormac Duggan and Sean Hammond

Student Numbers: 17100348, 17374356

Supervisor: Graham Healy

Completion Date: 7/5/2021



System Testing

Due to LiveFocus's usability being heavily reliant on its user interface, it was necessary to undertake robust formal testing in the form of system testing. While individual components could be validated via unit tests and integration tests, ensuring that the software as a whole was entirely usable and accessible was required.

The methods of testing used, and their findings, are as follows:

Usability and Accessibility Testing

Due to LiveFocus being an application that is designed to be used by a wide variety of individuals, it is extremely important to ensure that the software can be used by anyone with the right equipment. In order to ensure this, a user testing survey was conducted. During this survey, users were instructed to complete a number of tasks using the software, and the amount of time taken to complete these tasks was recorded. Users were given a brief demonstration of using the software before being asked to use it themselves.

It was decided that user testing would be considered a success if each of the tasks given could be completed in under 3 minutes. Any task that took longer than this would demonstrate that the application would not be intuitive to use, and more information should be provided. The tasks given were as follows:

1. Using the instructions provided by the application, pair the Muse device to the PC.
2. Start a new recording using the video file located in the folder named "Video" in the PC's home directory, recording frontal EOG, rear EOG, and accelerometer data. The file should be named userTest, followed by your provided subject ID. The group name should be "user testing".
3. Turn on the Muse device and connect it to the application using the "connect" button. Wipe with a dry piece of cloth or towel your forehead and the back of your ears to remove any dead skin that may interfere with the Muse device readings, then place the device on your head. Ensure full contact between the connection points of the device and the skin on your forehead, and between the rubber of the device and the back of your ears. Begin the recording, and start the video when ready. When the video is finished, click "End Recording", and then disconnect the device. (Note: For this task, timing was stopped after the video began to play)
4. Navigate to the Sessions page, and select the session with the name you gave in task 2, as well as the session entitled "exampleSession.fid". Click "View Selected Files". In the drop down menu on the top of the page, change to "Rear EOG".

5. Determine where there is a noticeable change in the graph, and note the start and end time of the change in seconds using the graph's horizontal time axis. Enter these times into the "View Clip" fields, then click "View Clip".
6. Navigate to the "Online" page, and provide the username "admin" and the password "password". Once you have logged in, using the "Upload File" button, select your recorded session.
7. In the search field, enter "databasetest" and press "Search". Click download on the file that appears.

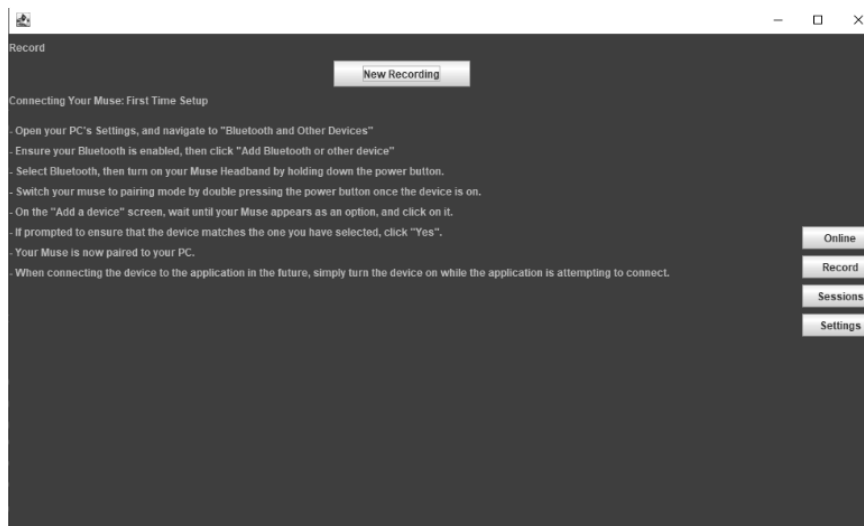
Users were then provided with an anonymous survey to fill out, ensuring that they felt as though the software was usable and intuitive, as well as accessible for them. All users reported that they felt the software was accessible and usable, with no recommendations for changes to be made. All users were able to complete each task within the given time.

Due to limited access to individuals with colour blindness and other impairments that may hinder a user's ability to use the software, accessibility testing was undertaken to ensure that the software was navigable with only the use of a keyboard. While not usable through the Windows Narrator, users with Java Access Bridge installed and a screen reader that supports Java will be able to screen read the text of the GUI. Finally, accessibility for colour blind individuals has been ensured by running screenshots of the application through a colour blindness filter, some examples of which can be seen below. It should be noted that in some cases, the ability of a colour blind user to determine at a glance which line in a graph correlates to which item in the legend may be hindered. This issue can be circumvented to some degree by disabling and re-enabling lines on the graph to determine which line represents which file.

Colour Blindness Testing Examples

Documented below are some examples of the UI with various colour blindness filters attached. Note that testing was done with all forms of colour blindness with all UI panels, below are 3 distinct key examples.

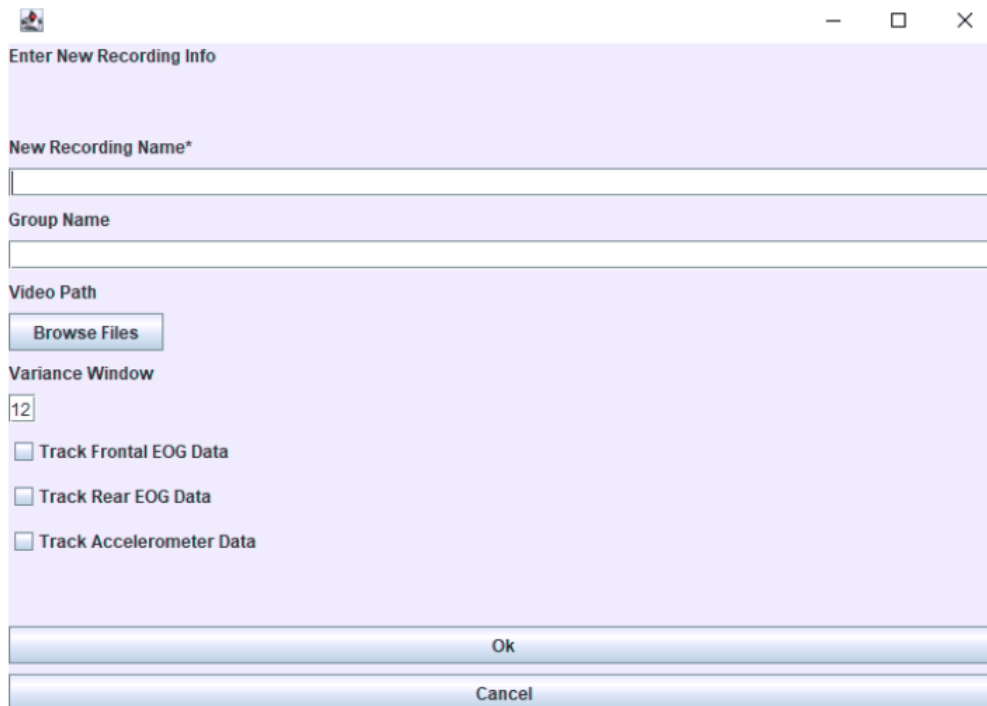
Monochromacy/Achromatopsia



Green-Blind/Deuteranopia



Blue-Blind/Tritanopia



Enter New Recording Info

New Recording Name*

Group Name

Video Path

Browse Files

Variance Window

12

☐ Track Frontal EOG Data

☐ Track Rear EOG Data

☐ Track Accelerometer Data

Ok

Cancel

Functional Completeness Testing

To ensure the application's functional completeness, a number of test cases were devised to ensure that the software did not break or malfunction given an unexpected edge case scenario. These test cases were designed to validate the code without the need for knowledge of the code itself, and were all performed by actually using the application. During each test, the tester went through a full "end to end" use of the software, taking the following steps in sequence:

1. Begin a new recording and connect the Muse to the application.
2. Record a session using the given sample video, then disconnect the Muse.
3. View the recorded session alongside the sample session. View all recorded data types and enable and disable viewing the average.
4. View a clip from the session's video.
5. Log into the online panel using the credentials provided.
6. Upload the recorded session.
7. Enter the search term "functest" and download the file that appears.
8. View the downloaded session against the recorded session.

In order to ensure the integrity of testing, during each test session these actions were taken in the same order with the same parameters with the exception of the test case that was being tested. The test cases devised were as follows:

- Do not provide a name or group name when creating a new recording.
 - Result: No defects found. File named ".fd" is created, this is deemed acceptable and non-anomalous behaviour.
- Close the recording window mid recording.
 - Result: defects found. Data is not written to a file, as expected as the "finish recording" button is not pressed.
- Start a session with no data types selected.
 - Result: No defects found. A file is created with the given info with no readings.
- View a file with no readings in session view.
 - Result: No defects found. Graph shows continuous line at 0.
- Have files without the suffix ".fd" in the sessions directory.
 - Result: No defects found. Incompatible files are detected and not listed.
- Have files invalid files with the suffix ".fd" in the sessions directory.
 - Result: No defects found. Invalid files are detected and not listed.
- Open multiple sessions of varying lengths.
 - Result: No defects found. As intended, graphing is only displayed until the shortest graph ends.
- Delete a video after selecting it for recording.

- Result: No defects found. Video panel remains blank as if no video was selected, data capture and graphing remains functional.
- Provide a number of seconds for the end timestamp of video clipping longer than the actual video length.
 - Result: No defects found. Video panel opens, but is blank, as expected.
- Provide a number of seconds for the start timestamp of video clipping longer than the actual video length.
 - Result: No defects found. Video panel opens, but is blank, as expected.
- Provide a start timestamp greater than the end timestamp for video clipping.
 - Result: No defects found. Video panel opens, but is blank, as expected.
- Be not connected to the internet while trying to log in to the online file storage system.
 - Result: No defects found. Login attempt times out and returns an error.

Additional Testing

In addition to system testing, individual components and their functionality was tested using unit tests and interactions with the GUI were verified using integration testing. Void functions were tested by ensuring dependency injection was used during development, where functions' main dependencies were extracted into separate classes that were then injected into the code. This allowed for classes with untestable dependencies and functionality to be tested safely without risk.

Informal ad hoc testing was also undertaken at every stage of development to ensure that all features were fully operational and that there were no defects in the code that were missed by the more formal software tests. When a feature was complete, it was standard practice during development to run through all features to ensure the application's integrity was maintained.