Jester

Testing Report

Students

Paul Treanor - 16347251

Colin Gorman - 17354073

Supervisor

Dr. Paul Clarke

Date completed

02/05/2021

Table of contents

1. Testing Overview **4**

2. Testing strategy **4**

2.1 Server side tests 4

2.2 Front end tests 4

2.3 Continuous integration 4

3. Testing Results **5**

3.1 Server side tests 5

3.1.1 Dataset preprocessing tests 5

3.1.2 Confidence function tests 6

3.1.3 Classifier pipeline tests 7

3.1.4 Server tests 8

3.1.5 Classifier profiling tests 9

3.2 Front end tests 10

3.2.1 Camera button tests 10

3.2.2 Page load tests 11

3.2.3 Manual front end tests 11

3.3 User testing 12

3.3.1 App functionality questions 12

3.3.2 User experience questions 12

3.3.3 User suggestions 12

4. Environments **13**

4.1 Test environments 13

1. Testing Overview

Testing was carried out on all components of Jester in order to ensure that the system was of a high quality and behaved as expected.

This document outlines the testing strategy that was followed, as well as the tests run and the results of the test.

2. Testing strategy

The system's tests can be divided into server side tests which test the classifier and API, and front end tests which test the web app.

2.1 Server side tests

As the classifier and API are both written in Python, the pytest module was used to write unit and integration tests for server side code.

A test driven development approach was carried out when developing the Jester classifier and API. Unit tests were written before the code, and the code was not considered complete until all tests passed. Ad hoc tests were carried out during development to allow for easier debugging.

Tests that profile the performance of the classification algorithms were also carried out to compare and optimise the algorithms.

2.2 Front end tests

Due to the web app's reliance on input from the device's camera and heavy integration between front end JavaScript code and the HTML document object model, unit and integration tests were difficult to write. Instead a range of manual tests, automated end to end tests, and user tests were carried out.

1. Ad hoc tests were carried out during development using console.log() and the Vue.js DevTools browser extension.
2. Automated end to end tests were carried out with Nightwatch.js to ensure that the system’s user interface was functional.
3. Manual tests were carried out to ensure that the gesture recognition requirements were met.
4. User tests were carried out to get feedback on the app’s user experience.

2.3 Continuous integration

Automated tests were integrated into a GitLab CI pipeline and Git Hooks which ran before code commits. For full details of this see section 5.10 of the technical manual document.

3. Testing Results

3.1 Server side tests

3.1.1 Dataset preprocessing tests

Description:

These tests validate that the keypoint standardisation algorithms used in the dataset preprocessing stage, and in the final classifier work as expected.

Location:

/src/dataset/data-preprocessing/data-transformations/data_transformer_test.py

Tests:

Test	Purpose	Status
test_translation_point_0	Assert the translate() method sets specific keypoints of a test gesture object to the expected coordinates.	Pass
test_translation_point_12		Pass
test_translation_point_15		Pass
test_enlargement_point_0	Assert the enlargement() method sets specific keypoints of a test gesture object to the expected coordinates.	Pass
test_enlargement_point_12		Pass
test_enlargement_point_15		Pass
test_rotation_point_0	Assert the rotate() method sets specific keypoints of a test gesture object to the expected coordinates.	Pass
test_rotation_point_12		Pass
test_rotation_point_15		Pass

3.1.2 Confidence function tests

Description:

These tests validate that the methods that make up the confidence value algorithms that were tested work as expected.

Location:

/src/server-side/classifier/conf_function_test.py

Tests:

Test	Purpose	Status
test_conf_kth_nn	Assert that the kth-nn confidence algorithm that was implemented returns the expected result.	Pass
test_conf_k_nn	Assert that the k-nn confidence algorithm (different to the knn classifier algorithm) that was implemented returns the expected result.	Pass
test_conf_ldofs	Assert that the local distance outlier factor algorithm that was implemented returns the expected result.	Pass
test_ReachableDistance	Assert that the ReachableDistance method of the local outlier factor algorithm returns the expected result.	Pass
test_local_r_density	Assert that the local_r_density method of the local outlier factor algorithm returns the expected result.	Pass

3.1.3 Classifier pipeline tests

Description:

These tests ensure that the components of the classification pipeline integrate correctly to produce correct classifications.

Location:

/src/server-side/classifier/get_class_test.py

Tests:

Test	Purpose	Status
test_peace_classification	Assert that gesture images are correctly classified.	Pass
test_palm_classification		Pass
test_thumbs_up_classification		Pass
test_alt_peace_classification		Pass
test_alt_palm_classification		Pass
test_alt_thumbs_up_classification		Pass
test_ood_classification	Assert that out-of-distribution gesture is classified as OOD.	Pass
test_alt_ood_classification		Pass
test_fuzzy_classification	Assert that a fuzzy image of a person is classified as OOD.	Pass
test_no_hand_classification	Assert that an image that does not show a person's hands is classified as OOD.	Pass

3.1.4 Server tests

Description:

These tests validate that the classifier API works as expected.

Location:

/src/server-side/*server_test/server_tests.py*

Tests:

Test	Purpose	Status
test_connection	Assert that the API is connecting the network.	Pass
test_ood_image	Assert that the API returns “OOD” when an out-of-distribution image is posted.	Pass
test_gesture_image	Assert that the API returns the correct gesture classification when an in-distribution image is posted.	Pass
test_400_error	Assert that the API returns a 400 error when an invalid file type is posted.	Pass

3.1.5 Classifier profiling tests

Description:

These tests record the time it takes to run the system's classification algorithms. The time was measured using Python's perf_counter() method. The classifier_profiling.py main method runs each of these tests 5 times and prints the average time taken.

The following code snippet shows how the timer begins, code is executed, and the timer stopped. The difference between the start and stop time is the execution time.

```
def knn_profile(gesture_data, k):
    start = time.perf_counter()              # Start timer
    classifier = knn(k)
    nearest_neighbours = classifier.predict(gesture_data)   # Find nearest neighbours
    stop = time.perf_counter()               # Stop timer
    execution_time = stop-start
    return execution_time
```

The profiling tests were carried out on a machine with an AMD Ryzen 3 2200G CPU.

Location:

/src/server-side/classifier/classifier_profiling.py

Tests:

Method	Purpose	Average time (seconds)
knn_profile	Profile K-nearest neighbours	0.0836
Local_distance_outlier_factor_profile	Profile Local distance outlier factor	0.0003
Local_outlier_factor_profile	Profile local outlier factor	2.1820

3.2 Front end tests

Automated front end testing was carried out using nightwatch.js, an end to end testing framework for browser apps.

3.2.1 Camera button tests

Description:

The following tests ensure that the manual camera and video functions implemented on the app (which trigger the same methods as the corresponding gestures do) work correctly with the device's camera and the app's gallery tab.

Location:

/src/web-app/tests/cameraButtonTests.js

Tests:

Test	Purpose	Status
Take photo button	Asserts pressing the app's take photo button will place a photo in the app's gallery	Pass
Take video button	Asserts that pressing the recording button, waiting, and pressing the stop button will place a video recording in the app's gallery.	Pass

3.2.2 Page load tests

Description:

The following tests ensure that the camera, gallery, and info views load correctly within the app.

Location:

/src/web-app/tests/pageLoadTests.js

Tests:

Test	Purpose	Status
Camera tab load tests	Asserts that items on the camera tab load when the app URL is accessed in the browser.	Pass
Camera preview load test	Asserts that camera preview video element is present on startup of the app.	Pass
Gallery page load test	Asserts that gallery tab elements load when the gallery button is pressed.	Pass
Info page load test	Asserts that info tab elements load when the info button is pressed.	Pass

3.2.3 Manual front end tests

Description:

These manual tests were carried out to assert that hand gestures can be used to control the device's camera.

Tests:

Test	Status
Assert that a photo is taken in response to the peace sign hand gesture.	Pass
Assert that a video is taken in response to the thumbs up gesture followed by the palm gesture.	Pass
Assert that photos and videos can be downloaded to the device with the download button.	Pass

3.3 User testing

User testing was limited to members of the development team's households due to the Covid-19 movement restrictions, but valuable feedback was still gathered.

3.3.1 App functionality questions

Question	Success rate
Were you able to use the app to take a photo?	3/3
Were you able to use the app to start video recording?	3/3
Were you able to use the app to stop video recording?	3/3

3.3.2 User experience questions

Question	Responses
Was the info screen helpful?	● Not very helpful: 1/3 ● Somewhat helpful: 1/3 ● Quite helpful: 1/3
Did you have any issues figuring out how to use the app?	● Yes: 3/3

3.3.3 User suggestions

Responses:

1. "I liked the design but it would be better if it was faster"
2. "it was designed well but there was no information telling me to include my elbow in the photo"
3. "It was hard to tell if a photo was being taken because the app was slow."

Takeaways:

1. The app should be more responsive.
2. A more informative in-app help screen is needed.

4. Environments

The system’s functionality has been tested on Android and Windows devices. The automated tests have primarily been carried out on Windows machines locally, with some tests being carried out on Linux machines through the GitLab CI pipeline.

4.1 Test environments

- Android 10.0
- Windows 10 64-bit
- Python 3.9
- Vue.js 2.6
- Node.js 14.16.0
- Nightwatch.js 1.6.3
- Vue DevTools 6.0