Testing Overview Document

for

Ai Invigilation System

Version 1.0

Prepared by Zicheng Guo, Xing Li, Yongxin Zhao

Department of Computing and Software, McMaster University

Supervisor: Dr. Rong Zheng

Table of Contents

1. Overview

This document describes the test plan for the system, and it's organized by the main features of the system. In each section below, the testing plan, justifications and test results will be explained.

2. Features

2.1 Supervisors authentication

Feature Description

This feature is a part of the WI system, and it's embedded in the react app. In the react app, users are required to identify their user type (supervisor or staff member) and authenticate using institution id and code. Once the authentication succeeds, the users will be redirected to the corresponding homepage. The main functionality of the authentication is to classify users and give them different levels of access.

Test plan

To test this feature, we must test both supervisors and staff members authentications. The test would require programmers to select a user type and manually type in pre-set institution id and hashcode. Then, once the log-in button is clicked, the programmer needs to check whether the user is redirected to the correct homepage and whether the correct accesses are given to the user on the homepage.

Justifications

This test mimics the operations that regular users would do when they are using the system. Besides, in the test implementation, pre-set id and code are used to test the logic of the authentication process. In real-world scenarios, the user's id will be stored in the database (which is not in the scope of this project), and the one-time hashcode will be generated right before the exam starts, leading supervisor and regular supervisor will get their access hashcode 15 minutes earlier than the exam via internal communication media(which is not in the scope of the project), we assume all the communication between staff are encrypted and safe. Within the scope of this project, the system shall test whether the user is a supervisor/staff, and the list can represent the database, thus checking whether the input id is in the list can simulate the process of checking whether a user is in the institutional database. Then, the system can generate a hashcode, and the hashcode will be printed in the python terminal to show the generation process. The testing will use the pre-set string, comparing the pre-set string with the user input will simulate the process of authenticating the hashcode.

Implementation

There is no code to test the log-in feature, we use manual testing instead.

For the supervisor log-in, the programmer will click “supervisor” on the main page to see if the system can redirect to the supervisor log-in page. Then, the programmer will test the following scenarios:

1. A random user id and the pre-set code
2. A random user id and a random code
3. An existing institution id and a random code
4. An existing institution id and the pre-set code

Then, check if the system works as expected: the 1-3 scenarios will keep the user on the log-in page and only scenario 4 can redirect the user to the supervisor's homepage.

Test result

Once the programmer clicks the “supervisor” on the main page, the system redirects to the supervisor log-in page. Then, in scenarios 1-3 above, the system stays on the same page and does not allow the users to proceed. In scenario 4, the system directs to the supervisor's homepage.

Therefore, this feature passes the test.

2.2 Suspicious Behavior Detection

Feature description

Once the supervisor starts the exam, the system will detect suspicious behavior of the examinee and flag an examinee if the pre-set criteria are met.

Test plan

To test this feature, we will ask all our group member to record a recording of them self pretending to write an exam, will in the process of writing, he/she will try to behave suspiciously during the process, including but not limited to look left and right, trying talk to the people next to, etc… .

After that, we will manually review the recording and mark the suspicious behavior and compare to the output of our program.

Justifications

Since this feature is hard to test using pre-build data or other popular testing method like unit test, so we need to memik real-world scenarios where individuals may attempt to cheat during an exam. By asking group members to record themselves while pretending to cheat, the testing team can replicate the behavior of real-life cheaters and capture potential cheating patterns that the program may miss. In addition, by ask members to try to do different kind of behaviors, this will cover all flags that included in our software, which cover most of the functioning codes. Moreover, by manually reviewing the result, we can have more understanding of the programs' performance hence we can improve the accuracy. The comparison between the program's output and the manually marked suspicious behavior can help identify the underlying problems in our system.

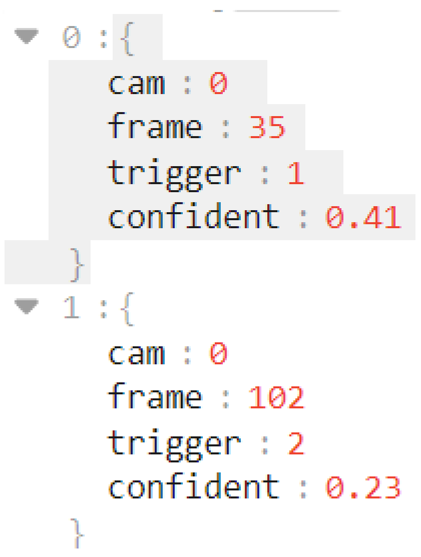
Implementation

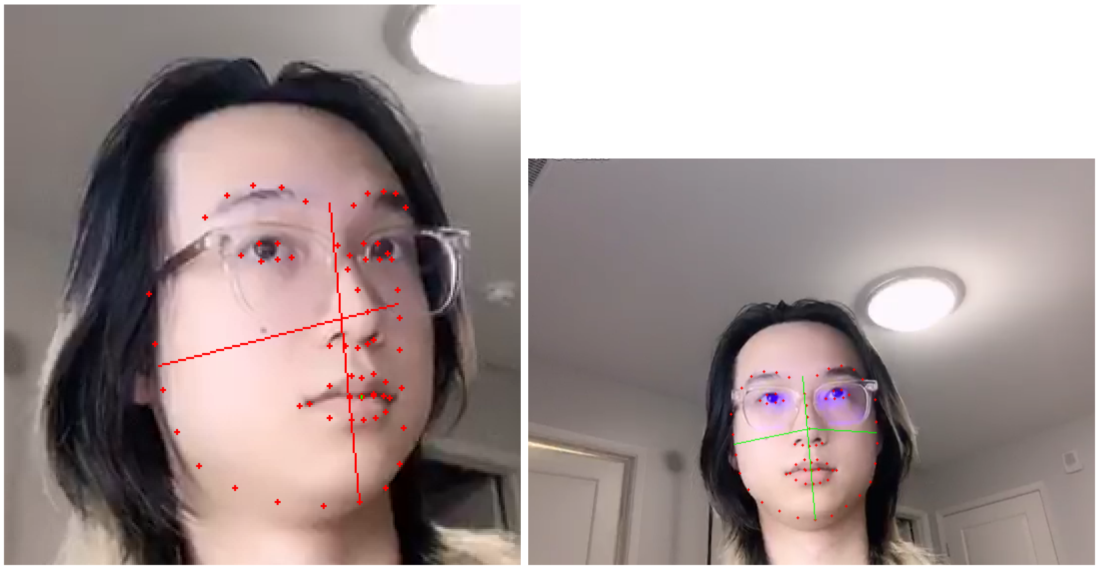
The only implementation need for the proposed testing method is an different entry point of BE where instead of getting streamed, the camera feed is stream by a file. And here is a proposal of procedures:

1. Advise all group members of the purpose and objective of the testing.
2. Setup the environment as above:
    a. Put an camera in front of a normal desk, or use the laptop's camera, and start recording.
    b. Get some random paper and a pen, then start writing anything on that paper.
    c. Try behave in a way that a student may cheating, which include but not limited to:
        i. Look left/right to see other's paper.
        ii. Look front to see the student in front of you.
        iii. Look directly below as you have a smart phone/cheat seet under the table.
        iv. Whisper to the people next to you.
3. Save the recording and run the software to generate an result.
4. Compare the result with the actual video to verify if all behavior is detected.

Test result

Below is screenshots of example output, the json file marked which frame is been detected, and we use #0 as example: screen shot on the left shown below, at the given frame generate by software, it detect the student turning his face to the right(1:face yaw trigger) with confident of 0.41 on camera 0. While compare to the right one where student is look where straight without any suspecious behavior.


```
▼ 0 : {
    cam : 0
    frame : 35
    trigger : 1
    confident : 0.41
  }
▼ 1 : {
    cam : 0
    frame : 102
    trigger : 2
    confident : 0.23
  }
```



2.3 Exam Management

Feature description

This feature is a combination of WI system and BE system, and the users will control the BE system via the WI system (react app will be the interface connecting users and the back-end models).

The system shall allow the leading supervisors to input exam information, set up timers, change the rules and add any specifications about the exam. The system shall also allow supervisors to customize pre-set triggers that the system will detect suspicious behaviors depend on. The suspicious behavior detection is based on these pre-set criteria.

Test plan

This feature consists of several components, and these components will be tested individually first, once they pass the tests, the overall test will be conducted to the feature as a whole.

For the create exam component in the WI system, programmers need to manually input some exam information via the WI interface in the browser, and check whether the information is passed to the back-end correctly and a corresponding exam object is created successfully. At the same time, programmers also need to test whether a new web page is opened in the browser with all the exam information displaying on that webpage as the feature is described in the requirement.

For the set triggers component in both WI system and BE system, programmers need to test whether 3 pre-set triggers can be controlled by the users via the WI system. The programmer need to test whether the triggers can be applied correctly in the following scenarios:

1. Each of the triggers is applied alone
2. Any two triggers are applied (including 3 scenarios)
3. All triggers are applied

In each scenario, the programmer needs to test whether the corresponding trigger status is passed to the back-end, which means the users can control triggers via checkboxes in the WI system.

Justifications

The mechanism of testing this feature is to test each individual component to make sure each one works well, then testing the entire feature as a whole. Since this feature includes many steps to complete, each individual testing can ensure the corresponding step can be done without errors. If errors are detected in an individual testing, programmers can easily locate that error and debug. Otherwise, programmers wouldn't know which component the error is from. Finally, programmers should mimic how a regular user will use the system and test if the system operates as expected. This is because we have to make sure the system's target user group can use the system to achieve their goals, and ensure the final product is error-free.

For the create exam component, there are two data sending procedures: from the current web page to the information display page; from the current webpage to the back-end and create an exam object thus testing should cover both procedures. In the implementation below, printing exam object can test the first data sending procedure above; manually test whether the pop-up window displays the correct input information.

For the set triggers component, it contains two steps: users select / cancel triggers on the webpage (system sends status to the back-end); system active / deactivate triggers (system controls the trigger objects). Therefore, we design tests for each step individually to ensure the data transmission in each step works well. For the first step, programmers need to test whether the status of triggers is stored and sent to the back-end, thus we test it by printing the status as a list. For the second step, programmers need to test whether triggers can control the back-end detection models.

Implementation

For the create exam component in the WI system, programmers need to modify the back-end code temporarily to print the created exam object (override the to string function) and information. Once input the exam info, programmers need to check whether the printed information is correct (which means that the correct exam object is created successfully) and the pop-up window also contains the exam info.

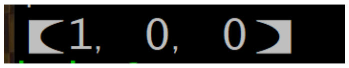
For the set triggers component in both WI system and BE system, programmers need to modify the back-end code temporarily to print the boolean list (the status of each trigger is represented by a boolean) on the terminal. Then, the programmers need to compare the status on the web page and the boolean list in the terminal in all scenarios described above. Besides, programmers also need to test whether the boolean list can control the triggers by mapping the boolean to the trigger models in the back-end code.

Once the system passes all the tests above, the programmer should also do an overall testing by mimicking a real user. The test procedure includes creating an exam, projecting the pop-up window changing trigger status, and the system should support these operations.

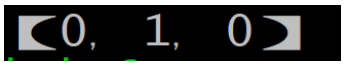
Test result

We have all three triggers activated as described above, here are the results:

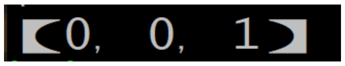
1. Each of the triggers is applied alone



```
[1, 0, 0]
```

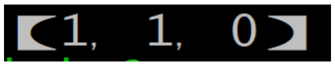


```
[0, 1, 0]
```

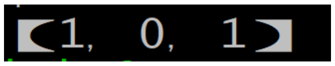


```
[0, 0, 1]
```

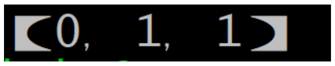
2. Any two triggers are applied



```
[1, 1, 0]
```

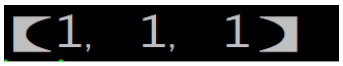


```
[1, 0, 1]
```



```
[0, 1, 1]
```

3. All triggers are applied



```
[1, 1, 1]
```

Once the programmer clicks the checkboxes on the set triggers page, the corresponding boolean list is printed in the terminal in all scenarios described above, and the boolean lists are the same as expected.

Therefore, this feature passes the test.

2.4 Generating report

Feature description

The system shall generate a report including exam information and timestamps of operations (detected suspicious behaviors, cancelled highlighted behaviors, etc.) when the exam is ended.

Test plan

This part of testing is embedded with both 2.2 and 2.3. Output text file will include the triggers applied and test information coming from 2.3, programmer needs to make sure the information in the text file is matched with the selection they've made for the test. Also the timestamps of suspicious behaviors coming from 2.2, programmer will need to manually record the timestamp for the suspicious behavior, this part needs to be matched with all the timestamps in the text file, as well as the reason that triggers the alert.

Justifications

The mechanism for this feature is that the text file generated after the exam includes all the information we needed for future usage, just in case some students have doubts on their behaviors, we can always easily find that timestamp and check the footage(not in the scope of the project), we only store the text file that includes information from 2.2 and 2.3, we don't have any feature to store the the actual footage for simplicity. We will have a proper format for all the information we need, in order to test the correctness of the information, (there might be data loss while we store the information or pass it between MM and WI), we need to manually write down or record any of the information related to the testing.

Here is a list of information should be appear on the text file:

1. Exam information: Exam name(COMPSCI 4ZP6A), start and end time(xxxx-xx-xx), triggers applied.(see 2.3 Exam Management)

2. Supervisors information: Leading supervisor name and regular supervisor.
3. Timestamp of suspicious behavior, reasons that triggers the alert.

Implementation

As for generating report result, we only have three components for this feature, datestamp, duration and reason as you can see down below, our backend machine generate a text file report using ‘|’ separates data.

Test result

Manual record:

2023-03-11, 11s, look around 10+s

2023-01-11, 3s, unexpected head position

2023-01-11, 4s, look around 10+s

2023-01-11, 6s, look around 10+s

2023-01-11, 7s, look around 10+s

2023-01-11, 3s, unexpected head position

Two tigers: Activated

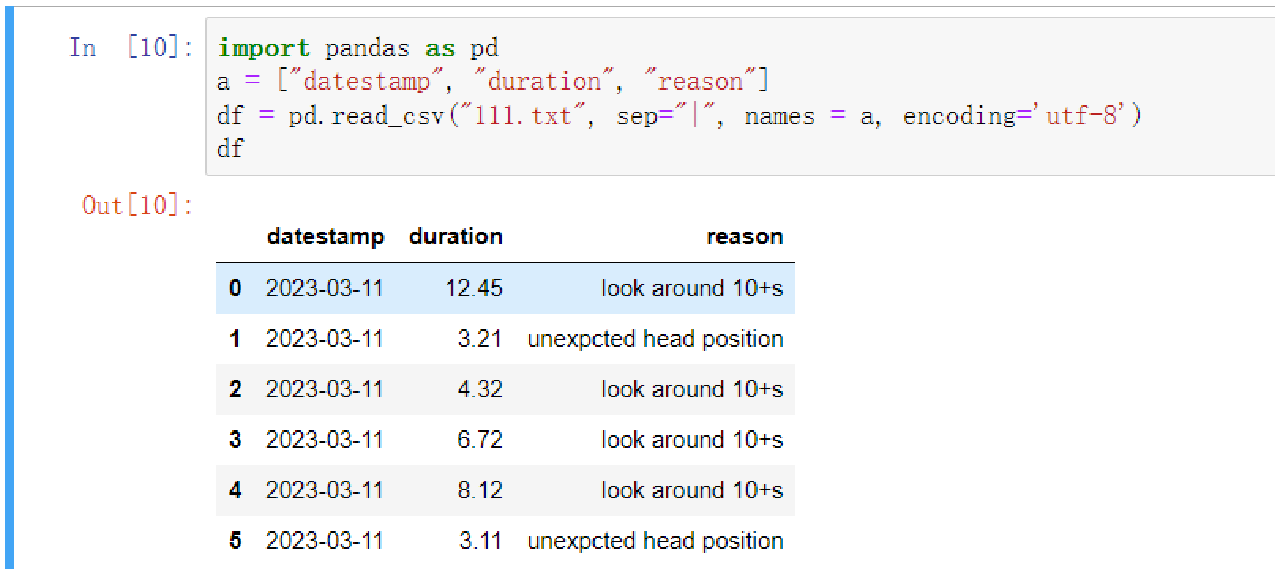
1. look around 10+s
2. unexpected head position

Supervisor:

1. Yuki Zhao
2. Victor Li

Leading supervisor:

Guoz



```
In  [10]: import pandas as pd
          a = ["datestamp", "duration", "reason"]
          df = pd.read_csv("111.txt", sep="|", names = a, encoding='utf-8')
          df
Out[10]:
```

	datestamp	duration	reason
0	2023-03-11	12.45	look around 10+s
1	2023-03-11	3.21	unexpcted head position
2	2023-03-11	4.32	look around 10+s
3	2023-03-11	6.72	look around 10+s
4	2023-03-11	8.12	look around 10+s
5	2023-03-11	3.11	unexpcted head position

As you can see from the above data, our manual record has 1s+ difference than the actual data, but the reason tigers the alert is the same, so that our generating report feature successed.
Our report does not include the triggers activated, also the names for the supervisors because of the format, cause we want the data to be read as a whole instead of two parts, but it seems we have to divide these two parts, or just read the timestamp parts first and then append the data together.

2.5 Streaming camera feeds

Feature description

The system shall display a GUI that allows supervisors to control (start and end) the stream of camera feeds to the BE and MM.

Test plan

This feature can be divided into two components: controlling and streaming, thus the test needs to be conducted on both of them separately.

For the controlling component, programmers should mimic the supervisors using the system in real-world scenarios including starting and stopping the camera feeds, and test whether the system works as expected. When the programmer clicks the start button on the GUI, the system should start streaming, and the system should end streaming when the stop button is clicked. For the streaming component, based on the correctness of the controlling component, programmers should test whether the camera feeds are sent to the back end for further analysis. The programmers need to start steaming and ensure the back end received the camera feeds.

After that, the overall testing which simulates an examination scenario for this feature should be conducted. The streaming will be started and keep working for 2-3 hours with a stable internet connection, and programmers should monitor the system every 30 min to check whether the system works properly. This testing should be conducted independently and with the suspicious detection feature.

Justifications

This feature is tested by separating two components because the streaming component depends on the controlling component, thus ensuring the controlling component works correctly should be done before testing the streaming component. By separation of concerns, the errors can be detected in the early stages and errors in one component will never affect the other component.

Since the system is designed to be used in real examination scenarios, the programmers should mimic the supervisor starting and stopping the streaming. Besides, the stability of the system should also be tested during the examination period (usually 2-3 hours). The independent testing can ensure this feature works, and the overall testing (with the suspicious detection feature) can ensure the two important systems cooperate as a whole.

Implementation

There is no code to test the streaming feature, we use manual testing instead. The programmers should test the feature as described in the test plan above, and record the test results and any other potential problems. The testing procedure should be repeated to make the result more accurate.

Test result

As described above, the GUI can control sending the camera feeds, and the video is sent to the back-end, which means the two components of the feature work correctly. As for the stability of the streaming, there was no failure or disconnection during the test periods.