

CS2310 Final Report

Intensity Adjustable Exercise System for Senior People Based on Facial Expression Recognition

Project demo video: <https://youtu.be/DTGi4Oc-eok>

Source code: https://github.com/Toobbby/SIS_Exercise_System

Fangzheng Guo
fag24@pitt.edu

Project background and motivation

Nowadays, personal electronic devices such as smartphones are widespread. More and more intelligent systems based on smartphones are invented to make people's life easier. Doing sports at home is also a new trend since coronavirus is spreading outside. However, it is reported that some senior people get hurt due to unsuitable sport intensity they set initially.

We all certainly know that the intensity of one good exercise should start at a low level, increase rapidly and finally goes down. But sometimes we are concentrating on the exercise itself, have no time to consider when should we adjust the intensity. Some people will choose to set up a sport program on treadmill before they start and finish it anyway. But for senior people, it is another story.

Senior people usually have some potential health issues and should do exercise appropriately. In other word, if they try to force themselves to finish a hard sport program, they may get hurt. What if we construct an intelligent system that monitors the user's status? Facial expression is clearly relevant to our status. If we feel exhausted, our facial expression is different from what our face looks like when we are calm. In this project, I proposed an exercise system based on SIS Server that monitors the user's facial expression and adjust the sport program's intensity accordingly.

System components

We will need a camera to capture the photo of user's face. The facial expression on the picture should be analyzed and classified (calm or not calm) by an analyzer. A processor should play the role as a core to command the sport program to adjust its intensity.

The system consists of 4 components: Camera, Facial Expression Analyzer, the Processor and the Sport Program.

Camera:

The camera should be set to capture user's face every 10 second, intuitively. It should pass the photo to the analyzer right after getting it. In practice, I use the front camera on my computer as the Camera component.

Facial Expression Analyzer:

Since a lot of machine learning methods is created for pattern recognition, the sub-problem: facial expression recognition could be solved with some of them. In this project, an opensource deep learning framework TensorFlow is used in facial expression recognition. The model is trained with FER-2013 dataset on Kaggle [1] and is able to output a set of weights (likelihood) on Angry, Disgusted, Fearful, Happy, Sad, Surprised and Neutral.

The analyzer should classify whether the facial expression in the photo is calm or not calm using the weight set. Angry, Disgusted, Fearful will directly contribute to Not Calm, Happy and Neutral will contribute to Calm. Sad and Surprised cannot be count as one state directly, but the relationship inside is figured out.

Processor:

The processor is playing the role as a decision maker. It will send messages to Sport Program to adjust the intensity basing on the patient's status.

At the first half of a program, if the patient is calm, the processor will increase the intensity of the sport program. Otherwise the processor will let the sport program stay with current intensity. On the opposite, the processor will reduce the intensity of the program at the second half if the patient is not calm. Otherwise keep the current intensity. All in all, the goal is to provide a safe and appropriate sport program for the user based on their real-time feeling.

Sport Program:

The sport program simulates a real program on a treadmill. The length of a program and the highest possible intensity is configurable. It is controlled by the processor and will adjust the intensity accordingly during a program. One program is always divided into 2 parts: the first half as **Rising Period** in which the intensity intends to go up as the user gradually warmed up and the second half as **Falling Period** in which the intensity intends to go down as the user got tired.

Figure 1 is the flow chart describes the interaction between 4 components.

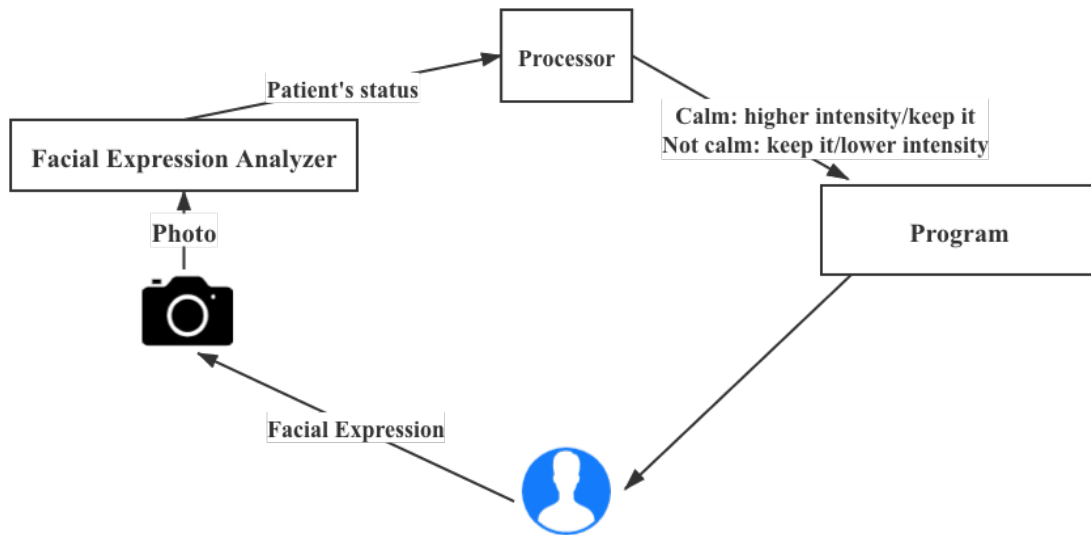


Figure 1. System Flow Chart

Facial Expression Recognition and User's Status (calm/not calm)

Facial expression helps the system to get known of the user's status while doing the sport program. Open source deep learning model was trained with FER-2013 dataset [1]. Facial expression data is classified as Angry, Disgusted, Fearful, Happy, Sad, Surprised or Neutral in the dataset, as shown in figure 2.



Figure 2 An example of FER-2013 dataset

The trained model can achieve 65% accuracy on facial expression classification. For this project, instead of knowing the exact facial expression, which is presented by the user, the status (calm/not calm) is more important.

Intuitively, Angry, Disgusted, Fearful correspond to Not Calm, Happy and Neutral correspond to Calm. Things become a little complicated on Sad and Surprised facial expression, since they are not directly corresponding to one state. Inspired by [2], I decided to not use weight vector of Surprised at the first half of the program, since it is a common reaction to program intensity

increase. In the second half, it will contribute to Not Calm state. Similarly, vector of Sad is not used in the second half. In the first half, it will contribute to Not Calm state.

The sum of the vectors which represent each state will come through a comparison. The user's statue is classified following the result.

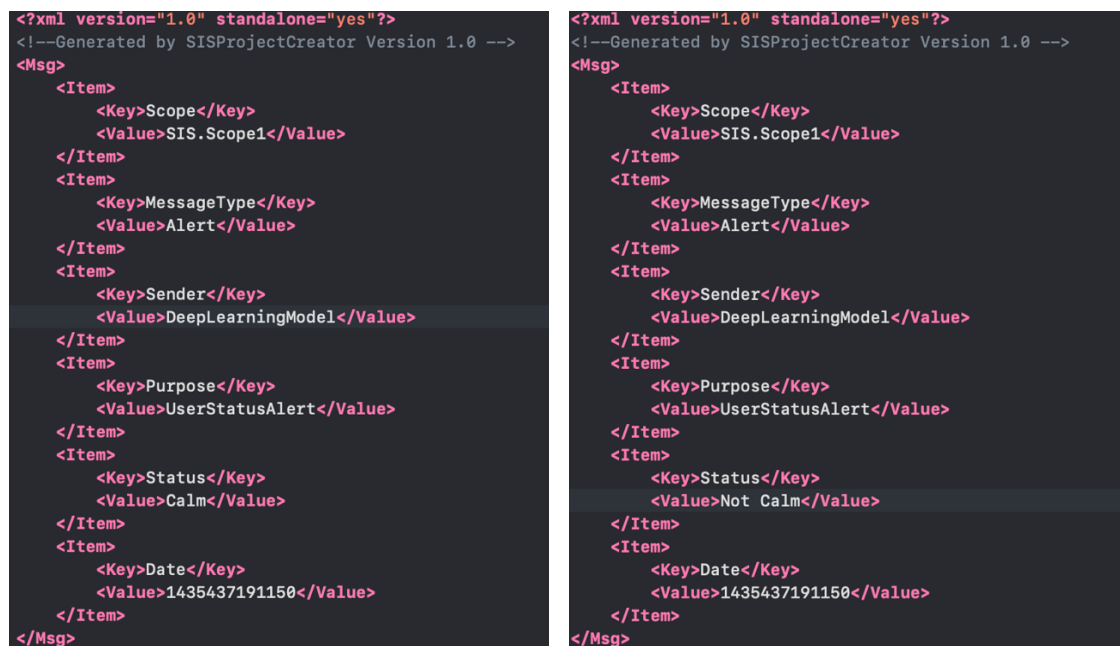
System Workflow and Messages

The components communicate with others by sending messages. A typical workflow is the camera take user's photo and store it in memory, then the processor will read the photo, detect the user's face and analyze the facial expression to determine the user's status. The processor then sends messages to the sport program according to the logic described above. The processor will receive messages from sport program identifying a program has started, the first half has finished, and the current program has finished. The state of the program helps it to make the correct decision.

The sport program will send message to processor at the start point, the middle of the program and the end point. It will receive messages from processor with command to increase or decrease the intensity. If the intensity has reached the highest possible intensity set before, it will not increase it anymore. If the intensity is reduced to 0, it will stop the program and send the message representing the program has finished to processor.

Messages designed based on scenarios:

1. photo analyzer of processor analyzed the photo of user and got the status of user.



```
<?xml version="1.0" standalone="yes"?>
<!--Generated by SISProjectCreator Version 1.0 -->
<Msg>
  <Item>
    <Key>Scope</Key>
    <Value>SIS.Scope1</Value>
  </Item>
  <Item>
    <Key>MessageType</Key>
    <Value>Alert</Value>
  </Item>
  <Item>
    <Key>Sender</Key>
    <Value>DeepLearningModel</Value>
  </Item>
  <Item>
    <Key>Purpose</Key>
    <Value>UserStatusAlert</Value>
  </Item>
  <Item>
    <Key>Status</Key>
    <Value>Calm</Value>
  </Item>
  <Item>
    <Key>Date</Key>
    <Value>1435437191150</Value>
  </Item>
</Msg>
```

```
<?xml version="1.0" standalone="yes"?>
<!--Generated by SISProjectCreator Version 1.0 -->
<Msg>
  <Item>
    <Key>Scope</Key>
    <Value>SIS.Scope1</Value>
  </Item>
  <Item>
    <Key>MessageType</Key>
    <Value>Alert</Value>
  </Item>
  <Item>
    <Key>Sender</Key>
    <Value>DeepLearningModel</Value>
  </Item>
  <Item>
    <Key>Purpose</Key>
    <Value>UserStatusAlert</Value>
  </Item>
  <Item>
    <Key>Status</Key>
    <Value>Not Calm</Value>
  </Item>
  <Item>
    <Key>Date</Key>
    <Value>1435437191150</Value>
  </Item>
</Msg>
```

Figure 3 Messages about the user's status

2. processor send the command to sport program to increase the intensity.

```
<?xml version="1.0" standalone="yes"?>
<!--Generated by SISProjectCreator Version 1.0 -->
<Msg>
  <Item>
    <Key>Scope</Key>
    <Value>SIS.Scope1</Value>
  </Item>
  <Item>
    <Key>MessageType</Key>
    <Value>Alert</Value>
  </Item>
  <Item>
    <Key>Sender</Key>
    <Value>Processor</Value>
  </Item>
  <Item>
    <Key>Receiver</Key>
    <Value>SportProgram</Value>
  </Item>
  <Item>
    <Key>Purpose</Key>
    <Value>DecreaseIntensityAlert</Value>
  </Item>
  <Item>
    <Key>Date</Key>
    <Value>1435437191150</Value>
  </Item>
</Msg>
```

Figure 4 Message about the increase command

3. processor send the command to sport program to decrease the intensity.

```
<?xml version="1.0" standalone="yes"?>
<!--Generated by SISProjectCreator Version 1.0 -->
<Msg>
  <Item>
    <Key>Scope</Key>
    <Value>SIS.Scope1</Value>
  </Item>
  <Item>
    <Key>MessageType</Key>
    <Value>Alert</Value>
  </Item>
  <Item>
    <Key>Sender</Key>
    <Value>Processor</Value>
  </Item>
  <Item>
    <Key>Receiver</Key>
    <Value>SportProgram</Value>
  </Item>
  <Item>
    <Key>Purpose</Key>
    <Value>IncreaseIntensityAlert</Value>
  </Item>
  <Item>
    <Key>Date</Key>
    <Value>1435437191150</Value>
  </Item>
</Msg>
```

Figure 5 Message about the decrease command

4. sport program notify the processor the program has started.

```
<?xml version="1.0" standalone="yes"?>
<!--Generated by SISProjectCreator Version 1.0 -->
<Msg>
  <Item>
    <Key>Scope</Key>
    <Value>SIS.Scope1</Value>
  </Item>
  <Item>
    <Key>MessageType</Key>
    <Value>Alert</Value>
  </Item>
  <Item>
    <Key>Sender</Key>
    <Value>SportProgram</Value>
  </Item>
  <Item>
    <Key>Receiver</Key>
    <Value>Processor</Value>
  </Item>
  <Item>
    <Key>Purpose</Key>
    <Value>StartAlert</Value>
  </Item>
  <Item>
    <Key>Date</Key>
    <Value>1435437191150</Value>
  </Item>
</Msg>
```

Figure 6 Message about the start of the program

5. sport program notify the processor the first half of the program has finished.

```
<?xml version="1.0" standalone="yes"?>
<!--Generated by SISProjectCreator Version 1.0 -->
<Msg>
  <Item>
    <Key>Scope</Key>
    <Value>SIS.Scope1</Value>
  </Item>
  <Item>
    <Key>MessageType</Key>
    <Value>Alert</Value>
  </Item>
  <Item>
    <Key>Sender</Key>
    <Value>SportProgram</Value>
  </Item>
  <Item>
    <Key>Receiver</Key>
    <Value>Processor</Value>
  </Item>
  <Item>
    <Key>Purpose</Key>
    <Value>MiddleAlert</Value>
  </Item>
  <Item>
    <Key>Date</Key>
    <Value>1435437191150</Value>
  </Item>
</Msg>
```

Figure 7 Message about the program's status change

6. sport program notify the processor the program has finished.

```
<?xml version="1.0" standalone="yes"?>
<!--Generated by SISProjectCreator Version 1.0 -->
<Msg>
  <Item>
    <Key>Scope</Key>
    <Value>SIS.Scope1</Value>
  </Item>
  <Item>
    <Key>MessageType</Key>
    <Value>Alert</Value>
  </Item>
  <Item>
    <Key>Sender</Key>
    <Value>SportProgram</Value>
  </Item>
  <Item>
    <Key>Receiver</Key>
    <Value>Processor</Value>
  </Item>
  <Item>
    <Key>Purpose</Key>
    <Value>FinishAlert</Value>
  </Item>
  <Item>
    <Key>Date</Key>
    <Value>1435437191150</Value>
  </Item>
</Msg>
```

Figure 8 Message about the end point of the program

Implementation and Operating of Components

Step 1: run SIS Server

```
# fguo @ FangzhengdeMacBook-Pro in ~/Desktop/SISv5/NewSISServer [21:29:54]
$ java SISServer .
SISServer starts, waiting for new components
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
```

Figure 9 screenshot of SIS Server

The SIS Serve helps to transfer and record the messages between components.

Step 2: Initialize all components in the system

```
fguo@FangzhengdeMacBook-Pro: ~/desktop/SISv5/Scripts
# fguo @ FangzhengdeMacBook-Pro in ~/desktop/SISv5 [22:53:20]
$ cd Scripts
# fguo @ FangzhengdeMacBook-Pro in ~/desktop/SISv5/Scripts [22:53:25]
$ ls
runIndividualComp  runInitializer.sh  runserver.sh
runInitializer.bat runserver.bat
# fguo @ FangzhengdeMacBook-Pro in ~/desktop/SISv5/Scripts [22:53:29]
$ /Users/fguo/Desktop/SISv5/Scripts/runInitializer.sh
Registration Fail: null
Registration Attempt: Analyzer
Registration Success: Analyzer
Registration Attempt: GUI
Registration Success: GUI
Registration Attempt: Program
Registration Success: Program
Registration Attempt: Processor
Registration Success: Processor
Registration Attempt: Camera
Registration Success: Camera
```

Figure 10 screenshot of initialization

Step 3: start Camera component and Photo Analyzer

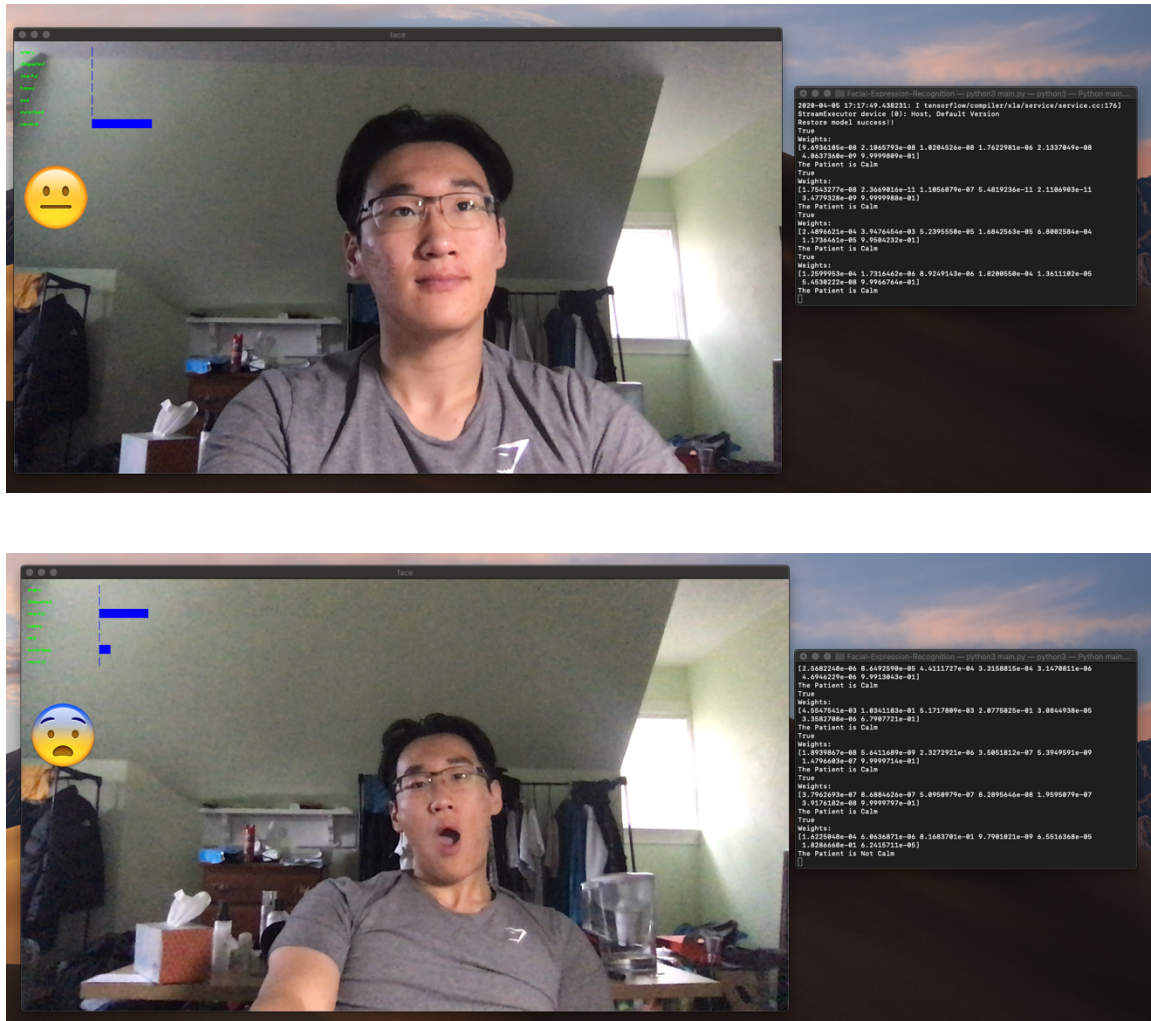


Figure 11 Analyzer based on deep learning model output classification result for captured photos

The photo will be taken every 5 seconds. As shown in the figure above, the facial expression analyzer will output a set of weight vector representing the likelihood between user's emotion and Angry, Disgusted, Fearful, Happy, Sad, Surprised and Neutral. The processor then classified the user's status based on the weight vectors. The emoji representing the most likely emotion of the user in the photo is printed on screen, helps us to justify the correctness of the facial expression recognition.

```
java SISServer...
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
=====
Role : Basic
Scope : SIS.Scope1
MessageType : Connect
Name : Analyzer
=====
Fetch messages from WebGUI
Messages fetched (if any)
[]
```

Figure 12 Analyzer as a SIS component

The duty of the analyzer is to recognize the user's facial expression from photo and classify the user's status. However, it is also responsible to send message about the user's status to the processor. Since then, I also created a module with Java for analyzer to send messages.

Step 4: start the processor

```
java SISServer...
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
=====
Role : Basic
Scope : SIS.Scope1
OutgoingMessages : OUT   Connect|Alert
MessageType : Connect
IncomingMessages : IN   Confirm|Setting:Kill||Alert:UserStatusAlert, ProgramAlert
Name : Processor
=====
Fetch messages from WebGUI
Messages fetched (if any)
[]
```

Figure 13 Processor start

The processor is able to receive UserStatusAlert from the analyzer and ProgramAlert from the sport program, as mentioned in the previous section.

Step 5: start the sport program

```
java SISServer .
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
=====
Role : Basic
Scope : SIS.Scope1
OutgoingMessages : OUT    Connect|Emergency
MessageType : Connect
IncomingMessages : IN     Confirm|Setting|Kill||Alert:PatientStatusAlert
Name : Program
=====
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
```

Figure 14 Sport Program as a component

There is a countdown mechanism inside the program components and the length of a program is set to 200 seconds for demo use.

Step 6: The photo of the user is taken by camera, analyzed by analyzer. The result is sent to processor. Then processor will perform corresponding actions on the sport program.

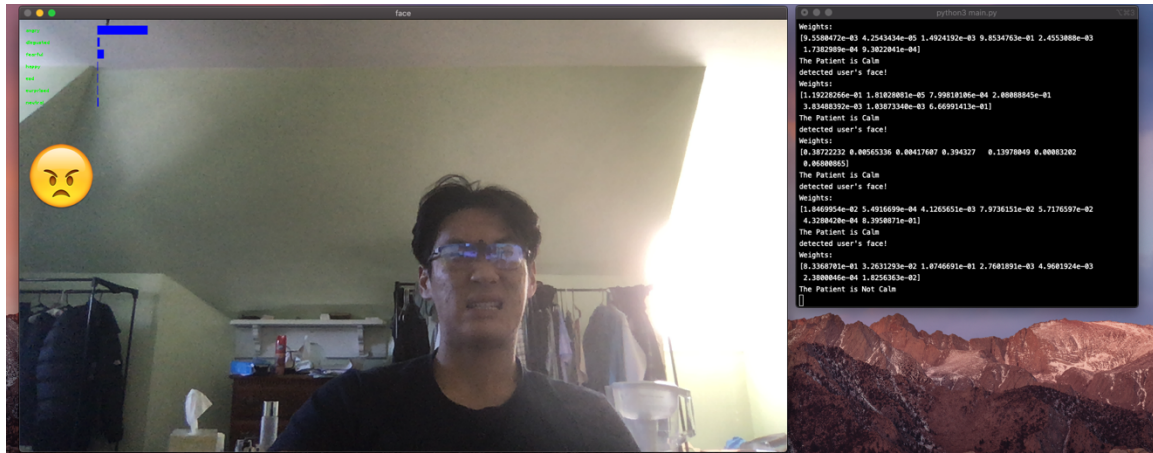
All possible scenarios based on full logic is presented below.

Scenario 1: The program is in the first half and the user is calm, the intensity of the program will be increased.



The user is calm

Scenario 3: The program is in the first half and the user is not calm, the intensity of the program will be kept.



The user is not calm.

```

Sender : Analyzer
Status : false
Scope : SIS.Scope1
Receiver : Processor
Purpose : UserStatusAlert
MessageType : Alert
Date : 1587417667207

=====Handling Alert=====
Sender: Analyzer
=====

Sender : Processor
Status : keep
Scope : SIS.Scope1
Receiver : Program
Purpose : AdjustWorkload
MessageType : Alert
Date : 1587417667208

=====Handling Alert=====
Sender: Processor

```

The processor let sport program keep the intensity.

```

Scope : SIS.Scope1
Receiver : Program
Purpose : AdjustWorkload
MessageType : Alert
Date : 1587417872662

Sender : Processor
Status : keep
Scope : SIS.Scope1
Receiver : Program
Purpose : AdjustWorkload
MessageType : Alert
Date : 1587417877666

Sender : Processor
Status : keep
Scope : SIS.Scope1
Receiver : Program
Purpose : AdjustWorkload
MessageType : Alert
Date : 1587417882671

```

The program received the command, no action on intensity (keep it).

Scenario 4: The program is in the second half and the user is calm, the intensity of the program will be kept.

```
# fguo @ FangzhengdeMacBook-Pro in ~/Desktop/SISv5/Components/camera [17:32:33]
C:130
$ java CreateAnalyzer
Calm
Calm
Calm
[]
```

Analyzer classified the user's status as calm.

```
java SISServer .
Scope : SIS.Scope1
Receiver : Processor
Purpose : UserStatusAlert
MessageType : Alert
Date : 1587418419938

=====Handling Alert=====
Sender: Analyzer

Sender : Processor
Status : keep
Scope : SIS.Scope1
Receiver : Program
Purpose : AdjustWorkload
MessageType : Alert
Date : 1587418419940

=====Handling Alert=====
Sender: Processor
Fetch messages from WebGUI
Messages fetched (if any)
[]
```

Processor let the sport program keep the intensity.

```
java CreateProcessor r.
Alert received, start processing...
===== Send out message to Sport Program =====
keep

Sender : Processor
Status : keep
Scope : SIS.Scope1
Receiver : Program
Purpose : AdjustWorkload
MessageType : Alert
Date : 1587418414939

Alert received, start processing...
===== Send out message to Sport Program =====
keep

Sender : Processor
Status : keep
Scope : SIS.Scope1
Receiver : Program
Purpose : AdjustWorkload
MessageType : Alert
Date : 1587418419940
[]
```

The sport program stayed with the same intensity.

Scenario 5: The program is in the second half and the user is not calm, the intensity of the program will be decreased.

```
# fguo @ FangzhengdeMacBook-Pro in ~/Desktop/SISv5/Components/camera [17:33:50]
C:130
$ java CreateAnalyzer
Not Calm
Not Calm
Not Calm
Not Calm
Not Calm
Not Calm
Not Calm
Not Calm
[]
```

Analyzer classified the user's status as calm.

```
java CreateProcessor r.
Alert received, start processing...
===== Send out message to Sport Program =====
decrease

Sender : Processor
Status : decrease
Scope : SIS.Scope1
Receiver : Program
Purpose : AdjustWorkload
MessageType : Alert
Date : 1587418896737

Alert received, start processing...
===== Send out message to Sport Program =====
decrease

Sender : Processor
Status : decrease
Scope : SIS.Scope1
Receiver : Program
Purpose : AdjustWorkload
MessageType : Alert
Date : 1587418901738
```

The processor then send message to the sport program to adjust the intensity (decrease).

```
java CreateProgram teProgram.
Receiver : Program
MessageType : Confirm

Connect to SISServer successful.

Sender : Processor
Status : decrease
Scope : SIS.Scope1
Receiver : Program
Purpose : AdjustWorkload
MessageType : Alert
Date : 1587418901738

The patient's workload has been reduced, currently is 9

Sender : Processor
Status : decrease
Scope : SIS.Scope1
Receiver : Program
Purpose : AdjustWorkload
MessageType : Alert
Date : 1587418906745

The patient's workload has been reduced, currently is 8
```

The sport program received the command, the intensity is reduced.

Scenario 6: The intensity is decreased to zero, the program will stop at this point.

```
fguo@FangzhengdeMacBook-Pro: ~/desktop/sisv5/Components/progr... ㄿ5
Sender : Processor
Status : decrease
Scope : SIS.Scope1
Receiver : Program
Purpose : AdjustWorkload
MessageType : Alert
Date : 1587418941763

The patient's workload has been reduced, currently is 1

Sender : Processor
Status : decrease
Scope : SIS.Scope1
Receiver : Program
Purpose : AdjustWorkload
MessageType : Alert
Date : 1587418946766

The patient's workload has been reduced, currently is 0
the program has finished

# fguo @ FangzhengdeMacBook-Pro in ~/desktop/sisv5/Components/program [17:42:26]
$
```

The intensity has been reduced to 0. The user is exhausted, the program should stop anyway.

Scenario 7: The program time out and finished.

```
fguo@FangzhengdeMacBook-Pro: ~/desktop/sisv5/Components/progr... ㄿ5
Sender : Processor
Status : decrease
Scope : SIS.Scope1
Receiver : Program
Purpose : AdjustWorkload
MessageType : Alert
Date : 1587420324134

The patient's workload has been reduced, currently is 7

Sender : Processor
Status : decrease
Scope : SIS.Scope1
Receiver : Program
Purpose : AdjustWorkload
MessageType : Alert
Date : 1587420329136

The patient's workload has been reduced, currently is 6
the program has finished

# fguo @ FangzhengdeMacBook-Pro in ~/desktop/sisv5/Components/program [18:05:29]
$
```

One successful work out has been finished.

Conclusions and future expectations

This project focused on senior people's demand, built up an intelligent system which utilized the SIS Server and latest machine learning techniques, also helped me get better understanding on component-based pattern design. At next step, I should provide proper front-end access for the user, to make the system easier to use in practice. Modern deep learning methods should be optimized with this certain problem in order to get better accuracy on user's status classification. More patterns and components could be added into the system to make the system more robust.

After learning the course, I realized most of the software engineering design techniques is really useful. The idea of **divide big problem into several small problems** helps me a lot. In future work and study, I will make good use of what I learned.

References

- [1] I. J. Goodfellow, D. Erhan et al. (2015) 'Challenges in representation learning: A report on three machine learning contests', Neural Networks, 64: 59-63.
- [2] Barrett, L. F. et al. (2019) 'Emotional Expressions Reconsidered: Challenges to Inferring Emotion From Human Facial Movements', Psychological Science in the Public Interest, 20(1), pp. 1-68.