



INFORMATICS
INSTITUTE OF
TECHNOLOGY

UNIVERSITY OF
WESTMINSTER 

INFORMATICS INSTITUTE OF TECHNOLOGY THE ITERATIONS

“VISAGO”

**Engagement level detection system using facial
emotions**

Module: 5COSC009C.2 Software Development Group Project

Module Leader: Mr. Banuka Athuraliya

Team Members

1. Name : Thiran De Silva

Student ID : 20200829

UOW ID : W1836091

2. Name : Shiny Fernando

Student ID : 20200675

UOW ID : W1833550

3. Name : Sathnidu Weerawardhana

Student ID : 20200938

UOW ID : W1838919

4. Name : Damsith Vidanapathirana

Student ID : 2019279

UOW ID : W1809757

5. Name : Chamath Nimnaada

Student ID : 20200843

UOW ID : W1836094

Table of Contents

| | |
|--|----|
| Chapter 1 – Implementation..... | 5 |
| 1.1 Chapter Overview | 5 |
| 1.2 Overview of the prototype..... | 5 |
| 1.3 Technology selection | 5 |
| 1.3.1 Language selection..... | 5 |
| 1.3.2 Libraries/Frameworks selection..... | 6 |
| 1.4 Datasets..... | 7 |
| 1.4.1 Implementation of the data science component | 9 |
| 1.5 Implementation of the front-end component | 25 |
| 1.6 Implementation of the backend component | 30 |
| 1.8 Chapter Summary | 35 |
| 2 Testing..... | 35 |
| 2.1 Chapter introduction | 35 |
| 2.2 Testing criteria | 35 |
| 2.3 Testing functional requirements | 35 |
| 2.4 Testing non-functional requirement..... | 36 |
| 2.5 Unit testing..... | 37 |
| 2.6 Performance testing | 37 |
| 2.7 Usability testing | 40 |
| 2.8 Compatibility testing..... | 40 |
| 2.9 Chapter summary | 41 |
| Chapter 3: Evaluation | 41 |
| 3.1 Chapter overview..... | 41 |
| 3.2 Evaluation methods | 41 |
| 3.3 Quantitative evaluation | 41 |
| 3.5 Self-evaluation..... | 42 |
| 3.6 Chapter Summary | 43 |
| Chapter 4: Conclusion..... | 43 |
| 4.1 Chapter Overview | 43 |
| 4.2 Achievements of aims and objectives..... | 43 |
| 4.3 Limitations of the research | 44 |
| 4.4 Future enhancements | 44 |

| | |
|------------------|----|
| References. | 45 |
|------------------|----|

Chapter 1 – Implementation

1.1 Chapter Overview

This chapter would concentrate on the prototype's implementation, with a thorough breakdown of the programming languages, tools, implementations, and libraries that were used. Any issues that arise will be addressed, as well as the action taken to minimize and resolve them. This chapter contains code fragments that demonstrate how each system feature is implemented.

1.2 Overview of the prototype

- After comparing and processing which algorithm is most accurate, the primary backend will process into the system and analyze and discover issues indicated in reviews.
- Once the issues have been found, the system will analyze the data to further classify and categorize them in order to find maintenance issues.

1.3 Technology selection

1.3.1 Language selection.

After a review of the available programming languages, Python was chosen as the primary language for this project's implementation. For the segment on data science, Python was selected due to following advantages,

- Python is an extremely useful programming language. Python's simplicity allows the developers to concentrate on the subject at hand. They don't need to spend a lot of time learning the programming language's syntax or behavior. You write less code and accomplish more.
- Python is an interpreted language, which means that the code is executed line by line by Python. In the event of an error, it stops the program's execution and reports the error. Even if the program has several faults, Python only displays one. This helps debugging.
- Python's standard library is vast, and it contains practically all of the functions required for any work. As a result, users won't need to rely on third-party libraries. Even if you do, a Python package manager (pip) makes importing additional wonderful packages from the Python package index much easier (PyPi). There are about 200,000 packages in all.

- Users must update the code in various languages, such as C/C++, to run the program on different systems. With Python, however, this is not the case. Users only have to write it once and it may be used wherever.

1.3.2 Libraries/Frameworks selection

1.3.2.1 OpenCV

OpenCV (Open-Source Computer Vision Library) is a free software library for computer vision and machine learning. OpenCV was created to provide a common infrastructure for computer vision applications and to help commercial goods incorporate machine perception more quickly. Because OpenCV is a BSD-licensed product, it is simple for businesses to use and alter the code.

More than 2500 optimized algorithms are included in the library, which contains a comprehensive mix of both classic and cutting-edge computer vision and machine learning techniques. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high-resolution image of an entire scene, remove red eyes from images taken with flash, follow eye movements, recognize scenery, and establish markers to overlay. OpenCV has a user group of about 47 thousand members and an estimated number of downloads of over 18 million.

1.3.2.3 Firebase

The Firebase Realtime Database is a database that is hosted in the cloud. Data is saved in JSON format and synchronized in real time across all connected clients. When you use our Apple, Android, and JavaScript SDKs to create cross-platform apps, all of your clients share a single Realtime Database instance and are automatically updated with the most recent data.

By giving secure access to the database directly from client-side code, the Firebase Realtime Database allows you to create complex, collaborative apps. Data is stored locally, and Realtime events continue to trigger even when the user is offline, providing a responsive experience. When the device regains connectivity, the Realtime Database instantly merges any discrepancies between local data changes and remote updates that happened while the client was offline.

To describe how your data should be formatted and when data can be read from or written to, the Realtime Database provides a flexible, expression-based rules language called Firebase Realtime

Database Security Rules. Developers can decide who has access to what data and how they can access it when Firebase Authentication is used.

The Realtime Database is a NoSQL database, which means it differs from relational databases in terms of optimizations and functionality. Only operations that can be completed fast are allowed in the Realtime Database API. This allows you to create a fantastic real-time experience that can serve millions of users while remaining responsive. As a result, it's critical to consider how people will access your data and then organize it correctly.

1.4 Datasets

Dataset 1(MRL eye dataset)

Computer vision tasks include detecting eyes and their parts, estimating gaze, and determining the frequency of eye blinking. The team that has created the dataset have been working on these tasks in the domain of driver behavior for the past few years, which has resulted in the collection of a large amount of testing data collected under real-world situations. As a result, the developers present the MRL Eye Dataset, a large-scale collection of human eye images. This collection contains low- and high-resolution infrared photos taken under various lighting situations and by various instruments. The dataset can be used to test a variety of features or trainable classifiers. The photos are separated into numerous categories to facilitate comparing algorithms easier, and they are also excellent for training and testing classifiers.

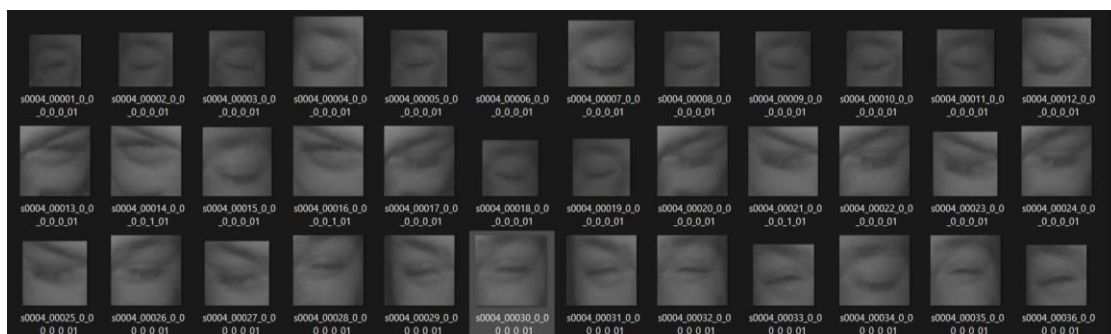


Figure 1: MRL eye dataset

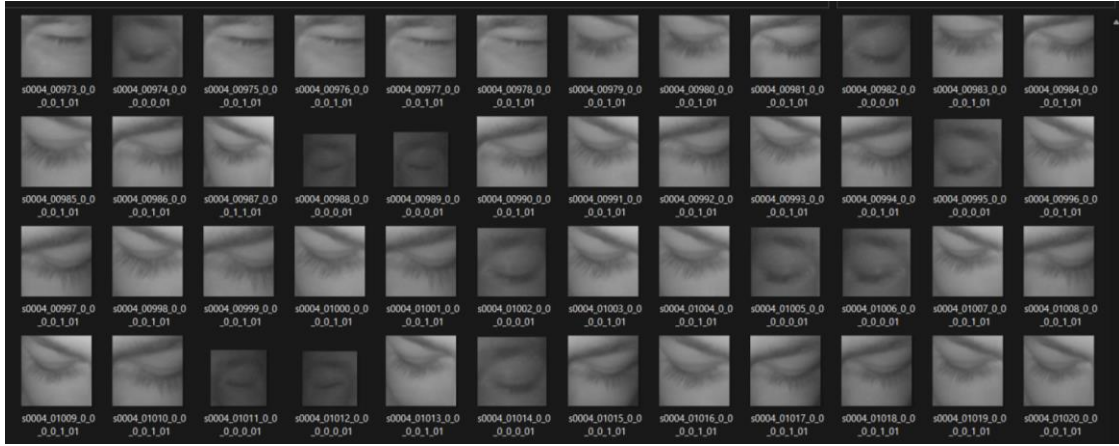


Figure 2: MRL eye dataset

Dataset 2 (fer-2013)

The data consists of grayscale images of faces at a resolution of 48x48 pixels. The faces have been automatically registered such that they are more or less centered in each image and take up around the same amount of area.

The goal is to categorize each face into one of seven categories based on the emotion expressed in the facial expression (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). There are 28,709 examples in the training set and 3,589 examples in the public test set.

There have been 143,112 views and 27,802 downloads of this dataset.

Main two folders (test and the train folder.)

| | | |
|-------|-------------------|-------------|
| test | 3/13/2022 9:49 PM | File folder |
| train | 3/13/2022 9:49 PM | File folder |

Figure 3: Fer-2013 dataset

Folders are divided as follows.

| | | |
|----------|-------------------|-------------|
| angry | 3/13/2022 9:49 PM | File folder |
| disgust | 3/13/2022 9:49 PM | File folder |
| fear | 3/13/2022 9:49 PM | File folder |
| happy | 3/13/2022 9:49 PM | File folder |
| neutral | 3/13/2022 9:49 PM | File folder |
| sad | 3/13/2022 9:49 PM | File folder |
| surprise | 3/13/2022 9:50 PM | File folder |

Figure 4: Divided folders on Fer-2013 dataset

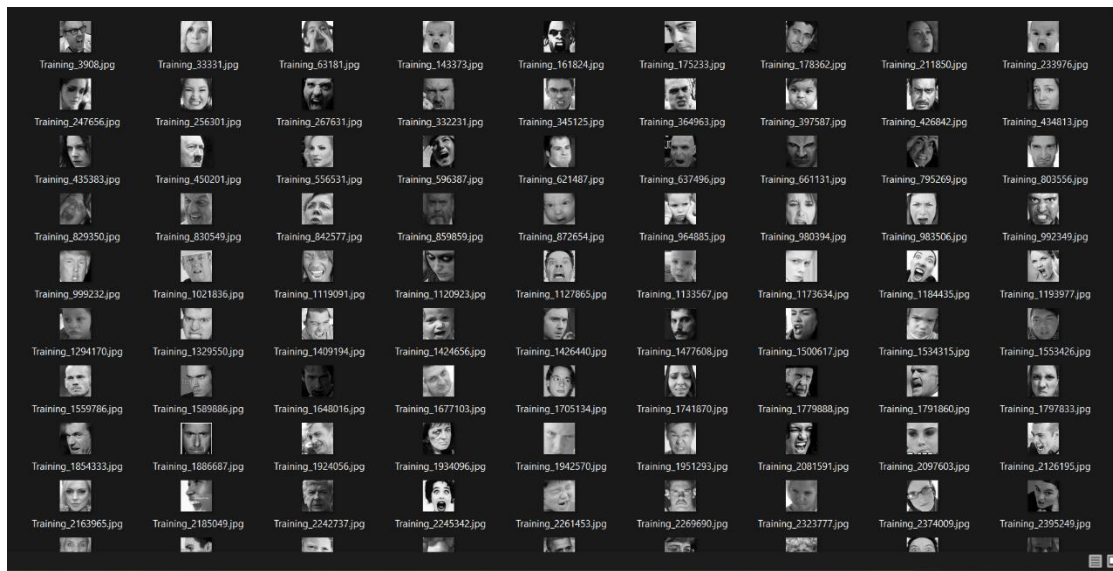


Figure 5: *Fer-2013 dataset*

1.4.1 Implementation of the data science component

1.4.1.1 *TensorFlow-Keras*

TensorFlow is an open-source machine learning platform that runs from start to finish. This class focuses on using a specific TensorFlow API to develop and train machine learning models. TensorFlow is a rich framework for managing all parts of a machine learning system; however, this class focuses on using a specific TensorFlow API to develop and train machine learning models. Complete details on the TensorFlow system can be found in the TensorFlow documentation.

TensorFlow APIs are organized in a hierarchical structure, with high-level APIs built atop lower-level APIs. Low-level APIs are used by machine learning researchers to design and test new machine learning algorithms. In this lesson, you'll define and train machine learning models, as well as make predictions, using a high-level API called `tf.keras`. The TensorFlow variation of the open-source Keras API is `tf.keras`.

1.4.1.2 *Dlib*

Dlib is a cross-platform open-source software library created in the C++ programming language for a variety of purposes. Its architecture is largely influenced by concepts from contract-based software engineering and component-based software engineering. This means it's first and

foremost a collection of self-contained software components, each with detailed documentation and debugging options.

1.4.1.3 numpy

Pandas is an open-source library designed to make it simple and natural to work with relational or labeled data. It includes a number of data structures and methods for working with numerical data and time series. This library is based on the NumPy Python library. Pandas is quick and has a high level of performance and productivity for its users.

Advantages

- For processing and evaluating data, it's quick and easy.
- It is possible to load data from various file objects.
- Missing data (expressed as NaN) in floating point and non-floating-point data is easily handled.
- Size mutability: columns in DataFrame and higher-dimensional objects can be added and removed.
- Merging and combining data sets.
- Data sets can be reshaped and pivoted in a variety of ways.
- Time-series functionality is provided.
- Split-apply-combine procedures on data sets are made easier with this powerful group by functionality.

1.4.1.4 matplotlib

matplotlib.pyplot is a set of routines that allow matplotlib to behave similarly to MATLAB. Each pyplot function modifies a figure in some way, such as creating a figure, a plotting area in a figure, charting certain lines in a plotting area, decorating the plot with labels, and so on.

Various states are retained throughout function calls in matplotlib.pyplot, allowing it to keep track of things like the current figure and plotting area, as well as direct plotting functions to the current axes

Drowsy data science component

Data preparation

Below are the codes that imported to do the data preparation part.

```
In [1]: import os
import shutil
import glob
from tqdm import tqdm
```

Figure 6: Imports for the data preparation

From the below codes all the 89000 pictures are divided into two folders.

```
In [2]: Raw_DIR= r'E:\jupyter\drowsy 2nd\mr1Eyes_2018_01'
for dirpath, dirname, filenames in os.walk(Raw_DIR):
    for i in tqdm([f for f in filenames if f.endswith('.png')]):
        if i.split('_')[4]=='0':
            shutil.copy(src=dirpath+'/'+i, dst=r'E:\jupyter\drowsy 2nd\prepared_data\closed_eye')

        elif i.split('_')[4]=='1':
            shutil.copy(src=dirpath+'/'+i, dst=r'E:\jupyter\drowsy 2nd\prepared_data\open_eye')
```

Figure 7: Data preparation for pictures

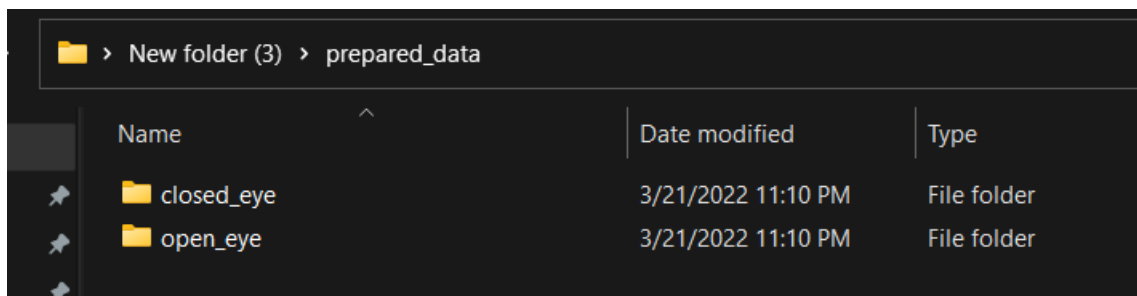


Figure 8: Data preparation folders

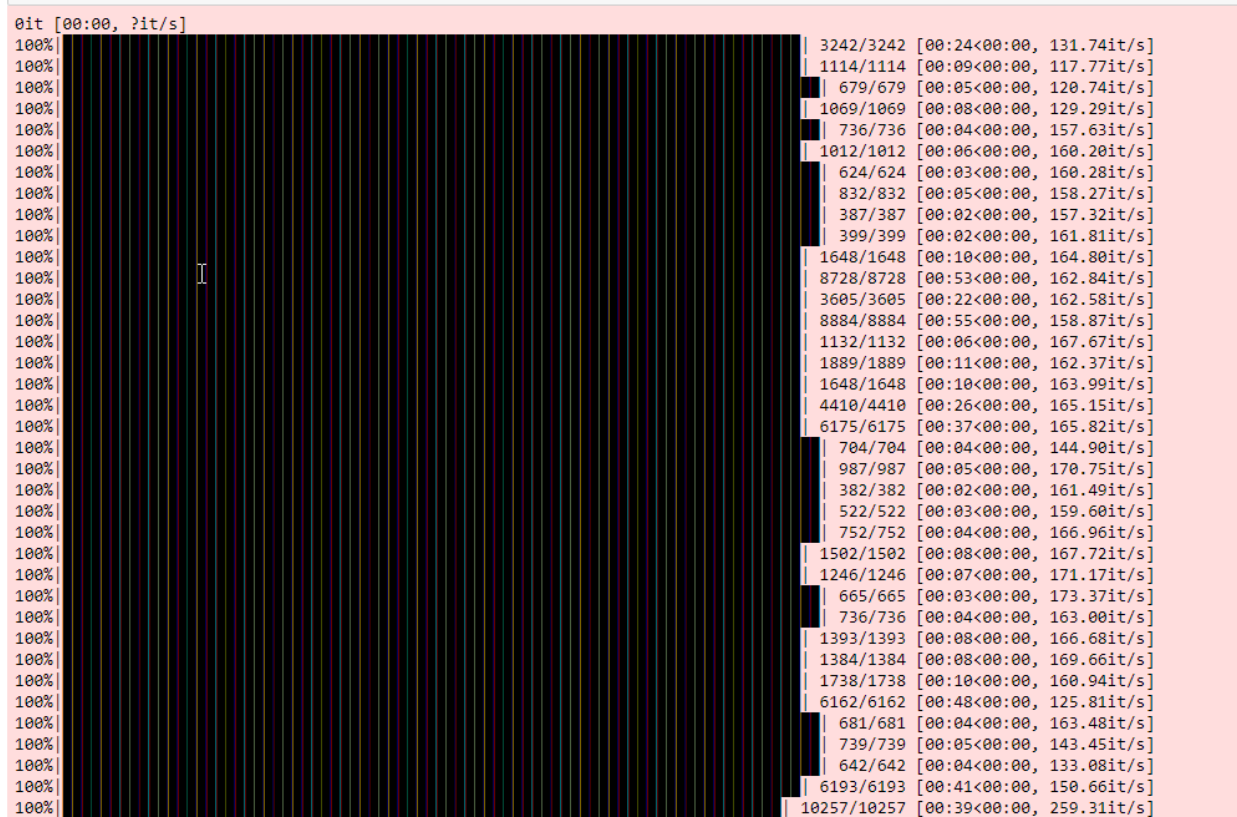


Figure 9: Progress bar of the codes

Above are the progress bars of the codes

Training the model

Below are the imports that need to train the model.

```
In [1]: import tensorflow as tf
        from tensorflow.keras.applications import InceptionV3
        from tensorflow.keras.models import Model
        from tensorflow.keras.layers import Dropout, Input, Flatten, Dense, MaxPooling2D
        from tensorflow.keras.preprocessing.image import ImageDataGenerator # Data Augmentation
```

Figure 10: Imports for the train model

The folders are divided in to train and the test folders in each of those folders there are two folders named open eye and closed eye. From the closed eye and the open eye folders the first step is to take 90% of images from the closed eye folder and 10% of the images to the validation data process and also from the open eye folder 90% of the images are taken into the training process and 10% of the images are taken to the validation data process.

The test data process is done for ensure the evaluate the performance of the model.

Train_data-80%(images), validation data-20%(images) 61122 and 15280 images respectively

Validation data process is done to make sure the process is done properly.

```
In [3]: batchsize=8

In [4]: train_datagen= ImageDataGenerator(rescale=1./255, rotation_range=0.2,shear_range=0.2,
      zoom_range=0.2,width_shift_range=0.2,
      height_shift_range=0.2, validation_split=0.2)

      train_data= train_datagen.flow_from_directory(r'C:\Users\Thiran De Silva\Desktop\SDGP Final\drowsy\prepared_data\train',
      target_size=(80,80),batch_size=batchsize,class_mode='categorical',subset='training' )

      validation_data= train_datagen.flow_from_directory(r'C:\Users\Thiran De Silva\Desktop\SDGP Final\drowsy\prepared_data\train',
      target_size=(80,80),batch_size=batchsize,class_mode='categorical', subset='validation')

Found 61122 images belonging to 2 classes.
Found 15280 images belonging to 2 classes.
```

Figure 11: Validation data process

Below is the number of images in the test_data process (10% of the images)

```
In [5]: test_datagen = ImageDataGenerator(rescale=1./255)

      test_data = test_datagen.flow_from_directory(r'C:\Users\Thiran De Silva\Desktop\SDGP Final\drowsy\prepared_data\test',
      target_size=(80,80),batch_size=batchsize,class_mode='categorical')

Found 8496 images belonging to 2 classes.
```

Figure 12: Images in the test data process

First the inceptionV3 architecture imported to train the model via transfer learning.

Advantages to use the inceptionV3 model

- It has higher efficiency
- It has a deeper network compared to the Inception V1 and V2 models, but its speed isn't compromised.
- It is computationally less expensive.
- It uses auxiliary Classifiers as regularizes.

```

'concatenate_1[0][0]',
'activation_93[0][0]']

avg_pool (GlobalAveragePooling (None, 2048)      0      ['mixed10[0][0]']
2D)

predictions (Dense)          (None, 1000)      2049000      ['avg_pool[0][0]']

=====
Total params: 23,851,784
Trainable params: 23,817,352
Non-trainable params: 34,432
=====

In [7]: bmodel = InceptionV3(include_top=False, weights='imagenet', input_tensor=Input(shape=(80,80,3)))
        hmodel = bmodel.output
        hmodel = Flatten()(hmodel)
        hmodel = Dense(64, activation='relu')(hmodel)
        hmodel = Dropout(0.5)(hmodel)
        hmodel = Dense(2,activation='softmax')(hmodel)

        model = Model(inputs=bmodel.input, outputs= hmodel)
        for layer in bmodel.layers:
            layer.trainable = False

In [8]: model.summary()

Model: "model"
-----
Layer (type)                 Output Shape              Param #   Connected to
-----
input_2 (InputLayer)         [(None, 80, 80, 3)]      0         []
conv2d_94 (Conv2D)           (None, 39, 39, 32)       864       ['input_2[0][0]']
batch_normalization_94 (Batch Normalization) (None, 39, 39, 32)       96       ['conv2d_94[0][0]']
activation_94 (Activation)    (None, 39, 39, 32)       0         ['batch_normalization_94[0][0]']
conv2d_95 (Conv2D)           (None, 37, 37, 32)       9216      ['activation_94[0][0]']
batch_normalization_95 (Batch Normalization) (None, 37, 37, 32)       96       ['conv2d_95[0][0]']

```

Figure 13: Imported inception V3 model

This originally give out 1000 classes but, in this code, it is reduced to 2 classes.

```
[9]: from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
```

Figure 14: Import main TensorFlow package

Importing above mentioned packages from the main TensorFlow package.

```

In [10]: checkpoint = ModelCheckpoint(r'C:\Users\Thiran De Silva\Desktop\SDGP Final\drowsy\my_model.h5',
                                     monitor='val_loss', save_best_only=True, verbose=3)

        earlystop = EarlyStopping(monitor = 'val_loss', patience=7, verbose= 3, restore_best_weights=True)

        learning_rate = ReduceLROnPlateau(monitor= 'val_loss', patience=3, verbose= 3, )

        callbacks=[checkpoint,earlystop,learning_rate]

```

Figure 15: Adding variables

Above are the checkpoint, earlystop, learning_rate variables.

Early stopping method.

Assume that the purpose of a training session is to reduce the loss. The statistic to be tracked would be 'loss,' and the mode would be 'min.' A model.fit() training loop will verify whether the loss is no longer decreasing at the end of each epoch, taking into account the min delta and patience if applicable. When it is discovered that it is no longer dropping, model.stop training is set to True, and the training is completed.

ReduceLROnplateau method

Reduce learning rate when a metric has stopped improving. Models often benefit from reducing the learning rate by a factor of 2-10 once learning stagnates. This scheduler reads a metrics quantity and if no improvement is seen for a 'patience' number of epochs, the learning rate is reduced

ModelCheckpoint method

The ModelCheckpoint callback is used in conjunction with model.fit() training to save a model or weights (in a checkpoint file) at a predetermined interval, so that the model or weights can be loaded later to resume training from the saved state.

```
In [11]: model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['accuracy'])  
  
history = model.fit_generator(train_data, steps_per_epoch=train_data.samples//batchsize,  
                             validation_data=validation_data,  
                             validation_steps=validation_data.samples//batchsize,  
                             callbacks=callbacks,  
                             epochs=25)
```

Figure 16: Model checkpoint method

In Python, we may utilize the keras.fit() and keras.fit_generator() deep learning libraries to train our machine learning and deep learning models. Both of these functions can perform the same purpose, but the difficulty is when to employ which one.

Below is the training part of the model.

```

7640/7640 [=====] - ETA: 0s - loss: 0.2102 - accuracy: 0.9169
Epoch 1: val_loss improved from inf to 0.22723, saving model to C:\Users\Thiran De Silva\Desktop\SDGP Final\drowsy\my_model.h5
7640/7640 [=====] - 607s 79ms/step - loss: 0.2102 - accuracy: 0.9169 - val_loss: 0.2272 - val_accu-
racy: 0.8988 - lr: 0.0010
Epoch 2/25
7639/7640 [=====>.] - ETA: 0s - loss: 0.1821 - accuracy: 0.9296
Epoch 2: val_loss improved from 0.22723 to 0.20684, saving model to C:\Users\Thiran De Silva\Desktop\SDGP Final\drowsy\my_mode-
l.h5
7640/7640 [=====] - 324s 42ms/step - loss: 0.1821 - accuracy: 0.9296 - val_loss: 0.2068 - val_accu-
racy: 0.9116 - lr: 0.0010
Epoch 3/25
7639/7640 [=====>.] - ETA: 0s - loss: 0.1713 - accuracy: 0.9339
Epoch 3: val_loss did not improve from 0.20684
7640/7640 [=====] - 310s 41ms/step - loss: 0.1713 - accuracy: 0.9339 - val_loss: 0.2098 - val_accu-
racy: 0.9135 - lr: 0.0010
Epoch 4/25
7639/7640 [=====>.] - ETA: 0s - loss: 0.1672 - accuracy: 0.9358
Epoch 4: val_loss improved from 0.20684 to 0.20359, saving model to C:\Users\Thiran De Silva\Desktop\SDGP Final\drowsy\my_mode-
l.h5
7640/7640 [=====] - 311s 41ms/step - loss: 0.1672 - accuracy: 0.9358 - val_loss: 0.2036 - val_accu-
racy: 0.9177 - lr: 0.0010
Epoch 5/25
7640/7640 [=====] - ETA: 0s - loss: 0.1625 - accuracy: 0.9385
Epoch 5: val_loss did not improve from 0.20359
7640/7640 [=====] - 316s 41ms/step - loss: 0.1625 - accuracy: 0.9385 - val_loss: 0.2094 - val_accu-
racy: 0.9146 - lr: 0.0010
Epoch 6/25
7639/7640 [=====>.] - ETA: 0s - loss: 0.1599 - accuracy: 0.9383

```

Figure 17: Training part of the model

```

Epoch 6: val_loss did not improve from 0.20359
7640/7640 [=====] - 321s 42ms/step - loss: 0.1599 - accuracy: 0.9383 - val_loss: 0.2095 - val_accu-
racy: 0.9187 - lr: 0.0010
Epoch 7/25
7640/7640 [=====] - ETA: 0s - loss: 0.1594 - accuracy: 0.9404
Epoch 7: val_loss did not improve from 0.20359

Epoch 7: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
7640/7640 [=====] - 330s 43ms/step - loss: 0.1594 - accuracy: 0.9404 - val_loss: 0.2060 - val_accu-
racy: 0.9156 - lr: 0.0010
Epoch 8/25
7640/7640 [=====] - ETA: 0s - loss: 0.1465 - accuracy: 0.9443
Epoch 8: val_loss improved from 0.20359 to 0.19975, saving model to C:\Users\Thiran De Silva\Desktop\SDGP Final\drowsy\my_mode-
l.h5
7640/7640 [=====] - 334s 44ms/step - loss: 0.1465 - accuracy: 0.9443 - val_loss: 0.1997 - val_accu-
racy: 0.9217 - lr: 1.0000e-04
Epoch 9/25
7639/7640 [=====>.] - ETA: 0s - loss: 0.1434 - accuracy: 0.9444
Epoch 9: val_loss did not improve from 0.19975
7640/7640 [=====] - 337s 44ms/step - loss: 0.1434 - accuracy: 0.9444 - val_loss: 0.2077 - val_accu-
racy: 0.9173 - lr: 1.0000e-04
Epoch 10/25
7639/7640 [=====>.] - ETA: 0s - loss: 0.1421 - accuracy: 0.9464
Epoch 10: val_loss did not improve from 0.19975
7640/7640 [=====] - 336s 44ms/step - loss: 0.1421 - accuracy: 0.9464 - val_loss: 0.2042 - val_accu-
racy: 0.9208 - lr: 1.0000e-04
Epoch 11/25
7640/7640 [=====] - ETA: 0s - loss: 0.1420 - accuracy: 0.9467
Epoch 11: val_loss improved from 0.19975 to 0.19679, saving model to C:\Users\Thiran De Silva\Desktop\SDGP Final\drowsy\my_mode-
l.h5
7640/7640 [=====] - 338s 44ms/step - loss: 0.1420 - accuracy: 0.9467 - val_loss: 0.1968 - val_accu-
racy: 0.9233 - lr: 1.0000e-04
Epoch 12/25
7640/7640 [=====] - ETA: 0s - loss: 0.1400 - accuracy: 0.9472

```



```

Epoch 12/25
7640/7640 [=====] - ETA: 0s - loss: 0.1400 - accuracy: 0.9472
Epoch 12: val_loss did not improve from 0.19679
7640/7640 [=====] - 338s 44ms/step - loss: 0.1400 - accuracy: 0.9472 - val_loss: 0.1989 - val_accu-
racy: 0.9226 - lr: 1.0000e-04
Epoch 13/25
7639/7640 [=====>.] - ETA: 0s - loss: 0.1415 - accuracy: 0.9462
Epoch 13: val_loss did not improve from 0.19679
7640/7640 [=====] - 337s 44ms/step - loss: 0.1414 - accuracy: 0.9462 - val_loss: 0.2023 - val_accu-
racy: 0.9211 - lr: 1.0000e-04
Epoch 14/25
7640/7640 [=====] - ETA: 0s - loss: 0.1390 - accuracy: 0.9477
Epoch 14: val_loss did not improve from 0.19679

Epoch 14: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
7640/7640 [=====] - 340s 44ms/step - loss: 0.1390 - accuracy: 0.9477 - val_loss: 0.2010 - val_accu-
racy: 0.9217 - lr: 1.0000e-04
Epoch 15/25
7639/7640 [=====>.] - ETA: 0s - loss: 0.1374 - accuracy: 0.9484
Epoch 15: val_loss improved from 0.19679 to 0.19612, saving model to C:\Users\Thiran De Silva\Desktop\SDGP Final\drowsy\my_mode-
l.h5
7640/7640 [=====] - 345s 45ms/step - loss: 0.1374 - accuracy: 0.9484 - val_loss: 0.1961 - val_accu-
racy: 0.9231 - lr: 1.0000e-05
Epoch 16/25
7639/7640 [=====>.] - ETA: 0s - loss: 0.1372 - accuracy: 0.9484
Epoch 16: val_loss did not improve from 0.19612
7640/7640 [=====] - 346s 45ms/step - loss: 0.1372 - accuracy: 0.9484 - val_loss: 0.1999 - val_accu-
racy: 0.9229 - lr: 1.0000e-05
Epoch 17/25
7639/7640 [=====>.] - ETA: 0s - loss: 0.1365 - accuracy: 0.9489
Epoch 17: val_loss improved from 0.19612 to 0.19605, saving model to C:\Users\Thiran De Silva\Desktop\SDGP Final\drowsy\my_mode-
l.h5
7640/7640 [=====] - 349s 46ms/step - loss: 0.1365 - accuracy: 0.9489 - val_loss: 0.1961 - val_accu-
racy: 0.9235 - lr: 1.0000e-05

```

Figure 18: Training part of the model

```

Epoch 18/25
7639/7640 [=====>.] - ETA: 0s - loss: 0.1361 - accuracy: 0.9478
Epoch 18: val_loss did not improve from 0.19605

Epoch 18: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
7640/7640 [=====] - 353s 46ms/step - loss: 0.1361 - accuracy: 0.9478 - val_loss: 0.2068 - val_accu-
racy: 0.9172 - lr: 1.0000e-05
Epoch 19/25
7640/7640 [=====] - ETA: 0s - loss: 0.1372 - accuracy: 0.9485
Epoch 19: val_loss did not improve from 0.19605
7640/7640 [=====] - 330s 43ms/step - loss: 0.1372 - accuracy: 0.9485 - val_loss: 0.2075 - val_accu-
racy: 0.9199 - lr: 1.0000e-06
Epoch 20/25
7640/7640 [=====] - ETA: 0s - loss: 0.1375 - accuracy: 0.9493
Epoch 20: val_loss did not improve from 0.19605
7640/7640 [=====] - 336s 44ms/step - loss: 0.1375 - accuracy: 0.9493 - val_loss: 0.1963 - val_accu-
racy: 0.9240 - lr: 1.0000e-06
Epoch 21/25
7640/7640 [=====] - ETA: 0s - loss: 0.1382 - accuracy: 0.9480
Epoch 21: val_loss did not improve from 0.19605

Epoch 21: ReduceLROnPlateau reducing learning rate to 1.000000111620805e-07.
7640/7640 [=====] - 341s 45ms/step - loss: 0.1382 - accuracy: 0.9480 - val_loss: 0.2062 - val_accu-
racy: 0.9198 - lr: 1.0000e-06
Epoch 22/25
7639/7640 [=====>.] - ETA: 0s - loss: 0.1373 - accuracy: 0.9470
Epoch 22: val_loss did not improve from 0.19605
7640/7640 [=====] - 345s 45ms/step - loss: 0.1373 - accuracy: 0.9470 - val_loss: 0.2034 - val_accu-
racy: 0.9218 - lr: 1.0000e-07
Epoch 23/25
7640/7640 [=====] - ETA: 0s - loss: 0.1377 - accuracy: 0.9482
Epoch 23: val_loss did not improve from 0.19605
7640/7640 [=====] - 341s 45ms/step - loss: 0.1377 - accuracy: 0.9482 - val_loss: 0.2009 - val_accu-
racy: 0.9215 - lr: 1.0000e-07

```

```

Epoch 24/25
7640/7640 [=====] - ETA: 0s - loss: 0.1368 - accuracy: 0.9482
Epoch 24: val_loss did not improve from 0.19605
Restoring model weights from the end of the best epoch: 17.

Epoch 24: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-08.
7640/7640 [=====] - 345s 45ms/step - loss: 0.1368 - accuracy: 0.9482 - val_loss: 0.2000 - val_accu-
racy: 0.9220 - lr: 1.0000e-07
Epoch 24: early stopping

```

Figure 19: Training part of the model

Emotion recognition

Folders are as follows

Test and the train folder



| | | |
|---|-------------------|-------------|
|  test | 3/30/2022 2:54 PM | File folder |
|  train | 3/30/2022 2:55 PM | File folder |

Figure 20: Test and train folder

Main 6 folders in the test and the train folder.








| | | |
|--|-------------------|-------------|
|  angry | 3/30/2022 2:54 PM | File folder |
|  disgust | 3/30/2022 2:54 PM | File folder |
|  fear | 3/30/2022 2:54 PM | File folder |
|  happy | 3/30/2022 2:54 PM | File folder |
|  neutral | 3/30/2022 2:54 PM | File folder |
|  sad | 3/30/2022 2:54 PM | File folder |
|  surprise | 3/30/2022 2:54 PM | File folder |

Figure 21: Folders in the main training folder

Training the model

Below are the imports that need to train the model

```

In [1]: import tensorflow as tf
        from tensorflow.keras.models import Model
        from tensorflow.keras.layers import Dropout, Input, Flatten, Dense, MaxPooling2D
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
        import os

```

Figure 22: Imports in the training model

As the previous model this model also uses the same method to the train and the test method.

```
In [2]: batchsize=32

In [3]: train_datagen= ImageDataGenerator(rescale=1./255, rotation_range=0.2,shear_range=0.2,
      zoom_range=0.2,width_shift_range=0.2,
      height_shift_range=0.2, validation_split=0.2)

train_data= train_datagen.flow_from_directory(r'E:\jupyter\emotion recognition\fer 2013\train',
      target_size=(224,224),batch_size=batchsize,class_mode='categorical',subset='training' )

validation_data= train_datagen.flow_from_directory(r'E:\jupyter\emotion recognition\fer 2013\train',
      target_size=(224,224),batch_size=batchsize,class_mode='categorical', subset='validation')

Found 22968 images belonging to 7 classes.
Found 5741 images belonging to 7 classes.
```

Figure 23: Train and test method

Batch size in this training code is set to 32

The ImageDataGenerator is used for to do the data augmentation part.

Train_data-80%(images), validation data-20%(images) 61122 and 15280 images respectively

Validation data process is done to make sure the process is done properly.

Below is the number of images in the test data process (10% of the images)

```
In [4]: test_datagen = ImageDataGenerator(rescale=1./255)

test_data = test_datagen.flow_from_directory(r'E:\jupyter\emotion recognition\fer 2013\test',
      target_size=(224,224),batch_size=batchsize,class_mode='categorical')

Found 7178 images belonging to 7 classes.

In [6]: model = tf.keras.applications.MobileNetV2()

In [7]: base_input = model.layers[0].input
base_output = model.layers[-2].output

In [8]: base_output

Out[8]: <KerasTensor: shape=(None, 1280) dtype=float32 (created by layer 'global_average_pooling2d')>

In [9]: final_output = layers.Dense(128)(base_output) # adding new layer after the output
final_output = layers.Activation('relu')(final_output)
final_output = layers.Dense(64)(final_output)
final_output = layers.Activation('relu')(final_output)
final_output = layers.Dense(7, activation='softmax')(final_output)
```

Figure 24: Import the MobileNetV2 version

In the above image the MobileNetV2 version is imported and in the original Mobilenet version there are 1000 layers given in the start and in this image the layers are reduced to 7.

```

In [12]: new_model.summary()

global_average_pooling2d (Glob (None, 1280)      0      ['out_relu[0][0]']
alAveragePooling2D)

dense (Dense)          (None, 128)      163968    ['global_average_pooling2d[0][0]']

activation (Activation) (None, 128)      0      ['dense[0][0]']

dense_1 (Dense)         (None, 64)      8256     ['activation[0][0]']

activation_1 (Activation) (None, 64)      0      ['dense_1[0][0]']

dense_2 (Dense)         (None, 7)       455      ['activation_1[0][0]']

=====
Total params: 2,430,663
Trainable params: 2,396,551
Non-trainable params: 34,112

In [13]: from tensorflow.keras.callbacks import ModelCheckpoint,EarlyStopping, ReduceLROnPlateau

In [14]: checkpoint = ModelCheckpoint(r'E:\jupyter\emotion_recognition\my_newModel.h5',
                                     monitor='val_loss',save_best_only=True,verbose=3)

         earllystop = EarlyStopping(monitor = 'val_loss', patience=7, verbose= 3, restore_best_weights=True)

         learning_rate = ReduceLROnPlateau(monitor= 'val_loss', patience=3, verbose= 3, )

         callbacks=[checkpoint,earllystop,learning_rate]

```

Figure 25: Import image the layers

From the tensorflow.keras.callbacks ModelCheckpoint, EarlyStopping and ReduceLROnplateau packages are imported.

Early stopping method.

Assume that the purpose of a training session is to reduce the loss. The statistic to be tracked would be 'loss,' and the mode would be 'min.' A model.fit() training loop will verify whether the loss is no longer decreasing at the end of each epoch, taking into account the min delta and patience if applicable. When it is discovered that it is no longer dropping, model.stop training is set to True, and the training is completed.

ReduceLROnplateau method

Reduce learning rate when a metric has stopped improving. Models often benefit from reducing the learning rate by a factor of 2-10 once learning stagnates. This scheduler reads a metrics quantity and if no improvement is seen for a 'patience' number of epochs, the learning rate is reduced

ModelCheckpoint method

The ModelCheckpoint callback is used in conjunction with model.fit() training to save a model or weights (in a checkpoint file) at a predetermined interval, so that the model or weights can be loaded later to resume training from the saved state.

```
In [ ]: new_model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['accuracy'])
new_model.fit_generator(train_data, steps_per_epoch=train_data.samples//batchsize,
                        validation_data=validation_data,
                        validation_steps=validation_data.samples//batchsize,
                        callbacks=callbacks,
                        epochs=2)
```

Figure 26: Model checkpoint method

Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments.

```
Epoch 1/40
594/593 [=====] - 18s 31ms/step - loss: 1.4183 - accuracy: 0.4491 - val_loss: 1.0883 - val_accuracy:
0.5424
Epoch 2/40
594/593 [=====] - 17s 29ms/step - loss: 1.1447 - accuracy: 0.5305 - val_loss: 1.1000 - val_accuracy:
0.5139
Epoch 3/40
594/593 [=====] - 16s 28ms/step - loss: 1.0011 - accuracy: 0.5787 - val_loss: 1.1149 - val_accuracy:
0.5612
Epoch 4/40
594/593 [=====] - 17s 28ms/step - loss: 0.9507 - accuracy: 0.6089 - val_loss: 1.0474 - val_accuracy:
0.6079
Epoch 5/40
594/593 [=====] - 17s 28ms/step - loss: 0.9134 - accuracy: 0.6270 - val_loss: 1.3852 - val_accuracy:
0.5327
Epoch 6/40
594/593 [=====] - 17s 29ms/step - loss: 0.8948 - accuracy: 0.6395 - val_loss: 0.8634 - val_accuracy:
0.6461
Epoch 7/40
594/593 [=====] - 16s 28ms/step - loss: 0.8632 - accuracy: 0.6548 - val_loss: 0.9289 - val_accuracy:
0.6382
Epoch 8/40
594/593 [=====] - 16s 28ms/step - loss: 0.8469 - accuracy: 0.6627 - val_loss: 1.0066 - val_accuracy:
0.6376
Epoch 9/40
594/593 [=====] - 17s 29ms/step - loss: 0.8270 - accuracy: 0.6707 - val_loss: 1.0263 - val_accuracy:
0.5976
Epoch 10/40
592/593 [=====>.] - ETA: 0s - loss: 0.8137 - accuracy: 0.6777
Epoch 00010: ReduceLROnPlateau reducing learning rate to 0.002499999999441206455.
594/593 [=====] - 16s 28ms/step - loss: 0.8134 - accuracy: 0.6776 - val_loss: 0.9359 - val_accuracy:
0.6309
```

Figure 27: Adam optimization

```

Epoch 11/40
594/593 [=====] - 16s 27ms/step - loss: 0.7366 - accuracy: 0.7116 - val_loss: 0.7550 - val_accuracy:
0.7073
Epoch 12/40
594/593 [=====] - 17s 28ms/step - loss: 0.7166 - accuracy: 0.7169 - val_loss: 0.7298 - val_accuracy:
0.7018
Epoch 13/40
594/593 [=====] - 17s 29ms/step - loss: 0.7005 - accuracy: 0.7234 - val_loss: 0.7178 - val_accuracy:
0.7248
Epoch 14/40
594/593 [=====] - 16s 28ms/step - loss: 0.6861 - accuracy: 0.7296 - val_loss: 0.6920 - val_accuracy:
0.7273
Epoch 15/40
594/593 [=====] - 16s 27ms/step - loss: 0.6814 - accuracy: 0.7367 - val_loss: 0.6949 - val_accuracy:
0.7236
Epoch 16/40
594/593 [=====] - 17s 29ms/step - loss: 0.6806 - accuracy: 0.7364 - val_loss: 0.7011 - val_accuracy:
0.7212
Epoch 17/40
594/593 [=====] - 17s 28ms/step - loss: 0.6655 - accuracy: 0.7387 - val_loss: 0.6795 - val_accuracy:
0.7267
Epoch 18/40
594/593 [=====] - 16s 28ms/step - loss: 0.6626 - accuracy: 0.7385 - val_loss: 0.6869 - val_accuracy:
0.7315
Epoch 19/40
594/593 [=====] - 16s 27ms/step - loss: 0.6497 - accuracy: 0.7462 - val_loss: 0.6806 - val_accuracy:
0.7358
Epoch 20/40
594/593 [=====] - 17s 29ms/step - loss: 0.6477 - accuracy: 0.7482 - val_loss: 0.7156 - val_accuracy:
0.7242
Epoch 21/40
594/593 [=====] - 16s 28ms/step - loss: 0.6450 - accuracy: 0.7489 - val_loss: 0.6749 - val_accuracy:
0.7388

```

Figure 28: Adam optimization

```

Epoch 22/40
594/593 [=====] - 16s 28ms/step - loss: 0.6364 - accuracy: 0.7502 - val_loss: 0.6930 - val_accuracy:
0.7412
Epoch 23/40
594/593 [=====] - 17s 28ms/step - loss: 0.6343 - accuracy: 0.7530 - val_loss: 0.6764 - val_accuracy:
0.7358
Epoch 24/40
594/593 [=====] - 17s 29ms/step - loss: 0.6294 - accuracy: 0.7583 - val_loss: 0.6717 - val_accuracy:
0.7418
Epoch 25/40
594/593 [=====] - 17s 28ms/step - loss: 0.6182 - accuracy: 0.7574 - val_loss: 0.6857 - val_accuracy:
0.7248
Epoch 26/40
594/593 [=====] - 17s 28ms/step - loss: 0.6132 - accuracy: 0.7597 - val_loss: 0.6807 - val_accuracy:
0.7339
Epoch 27/40
594/593 [=====] - 18s 30ms/step - loss: 0.6149 - accuracy: 0.7577 - val_loss: 0.6879 - val_accuracy:
0.7309
Epoch 28/40
593/593 [=====>] - ETA: 0s - loss: 0.6109 - accuracy: 0.7660
Epoch 00028: ReduceLROnPlateau reducing learning rate to 0.0006249999860301614.
594/593 [=====] - 17s 28ms/step - loss: 0.6107 - accuracy: 0.7660 - val_loss: 0.7048 - val_accuracy:
0.7212
Epoch 29/40
594/593 [=====] - 16s 27ms/step - loss: 0.5837 - accuracy: 0.7733 - val_loss: 0.6670 - val_accuracy:
0.7352
Epoch 30/40
594/593 [=====] - 17s 29ms/step - loss: 0.5736 - accuracy: 0.7771 - val_loss: 0.6825 - val_accuracy:
0.7273
Epoch 31/40
594/593 [=====] - 17s 29ms/step - loss: 0.5664 - accuracy: 0.7813 - val_loss: 0.6737 - val_accuracy:
0.7309

```

Figure 29: Adam optimization

```

Epoch 32/40
593/593 [=====>] - ETA: 0s - loss: 0.5634 - accuracy: 0.7811
Epoch 00032: ReduceLROnPlateau reducing learning rate to 0.00015624999650754035.
594/593 [=====] - 17s 28ms/step - loss: 0.5640 - accuracy: 0.7810 - val_loss: 0.6751 - val_accuracy:
0.7370
Epoch 33/40
594/593 [=====] - 16s 27ms/step - loss: 0.5546 - accuracy: 0.7845 - val_loss: 0.6679 - val_accuracy:
0.7406
Epoch 34/40
594/593 [=====] - 18s 30ms/step - loss: 0.5587 - accuracy: 0.7868 - val_loss: 0.6729 - val_accuracy:
0.7327
Epoch 35/40
592/593 [=====>] - ETA: 0s - loss: 0.5477 - accuracy: 0.7862Restoring model weights from the end of the
best epoch.
594/593 [=====] - 17s 28ms/step - loss: 0.5482 - accuracy: 0.7859 - val_loss: 0.6707 - val_accuracy:
0.7358
Epoch 00035: early stopping

```

Figure 30: Import Adam optimization

The early stopping method is executed in the end because after the 35th epoch because the learning rate is not improving. Because the learning rate is not improving there is no use of running epochs anymore.

Gaze tracking Data science component

Below are the imports related to the gaze tracking component

```
1 from __future__ import division
2 import os
3 import cv2
4 import dlib
5 from .eye import Eye
6 from .calibration import Calibration
```

Figure 31: import gaze tracking component

The user's gaze is tracked by the below class. It gives useful data such as the position of the eyes and pupils, as well as the ability to determine if the eyes are open or closed.

```
class GazeTracking(object):
```

Figure 31: Gaze tracking method

The following method locates the pupil in the picture.

```
def pupils_located(self):
    try:
        int(self.eye_left.pupil.x)
        int(self.eye_left.pupil.y)
        int(self.eye_right.pupil.x)
        int(self.eye_right.pupil.y)
        return True
    except Exception:
        return False
```

Figure 32: Method to locate pupil

The below method and the code detect the face and initialize eye objects

```

def _analyze(self):
    frame = cv2.cvtColor(self.frame, cv2.COLOR_BGR2GRAY)
    faces = self._face_detector(frame)

    try:
        landmarks = self._predictor(frame, faces[0])
        self.eye_left = Eye(frame, landmarks, 0, self.calibration)
        self.eye_right = Eye(frame, landmarks, 1, self.calibration)

    except IndexError:
        self.eye_left = None
        self.eye_right = None

```

Figure 33: Code to detect the face

```

def is_right(self):
    """Returns true if the user is looking to the right"""
    if self.pupils_located:
        return self.horizontal_ratio() <= 0.35

def is_left(self):
    """Returns true if the user is looking to the left"""
    if self.pupils_located:
        return self.horizontal_ratio() >= 0.65

def is_center(self):
    """Returns true if the user is looking to the center"""
    if self.pupils_located:
        return self.is_right() is not True and self.is_left() is not True

def is_blinking(self):
    """Returns true if the user closes his eyes"""
    if self.pupils_located:
        blinking_ratio = (self.eye_left.blinking + self.eye_right.blinking) / 2
        return blinking_ratio > 3.8

def annotated_frame(self):
    """Returns the main frame with pupils highlighted"""
    frame = self.frame.copy()

    if self.pupils_located:
        color = (0, 255, 0)
        x_left, y_left = self.pupil_left_coords()
        x_right, y_right = self.pupil_right_coords()
        cv2.line(frame, (x_left - 5, y_left), (x_left + 5, y_left), color)
        cv2.line(frame, (x_left, y_left - 5), (x_left, y_left + 5), color)
        cv2.line(frame, (x_right - 5, y_right), (x_right + 5, y_right), color)
        cv2.line(frame, (x_right, y_right - 5), (x_right, y_right + 5), color)

```

Figure 34: Code to initialize eye objects


```
gaze = GazeTracking()
webcam = cv2.VideoCapture(0,cv2.CAP_DSHOW)
```

Figure 35: Video capture method

Above is the video capture method

```
if gaze.is_blinking():
    text = "Blinking"
    if eyeStatus != text:
        eyeStatus = text
        print(text)
elif gaze.is_right():
    text = "Looking right"
    if eyeStatus != text:
        eyeStatus = text
        print(text)
elif gaze.is_left():
    text = "Looking left"
    if eyeStatus != text:
        eyeStatus = text
        print(text)
elif gaze.is_center():
    text = "Looking center"
    if eyeStatus != text:
        eyeStatus = text
        print(text)

cv2.putText(frame, text, (90, 60), cv2.FONT_HERSHEY_DUPLEX, 1.6, (147, 58, 31), 2)
```

Figure 36: Method to detect head position

1.5 Implementation of the front-end component

Home page

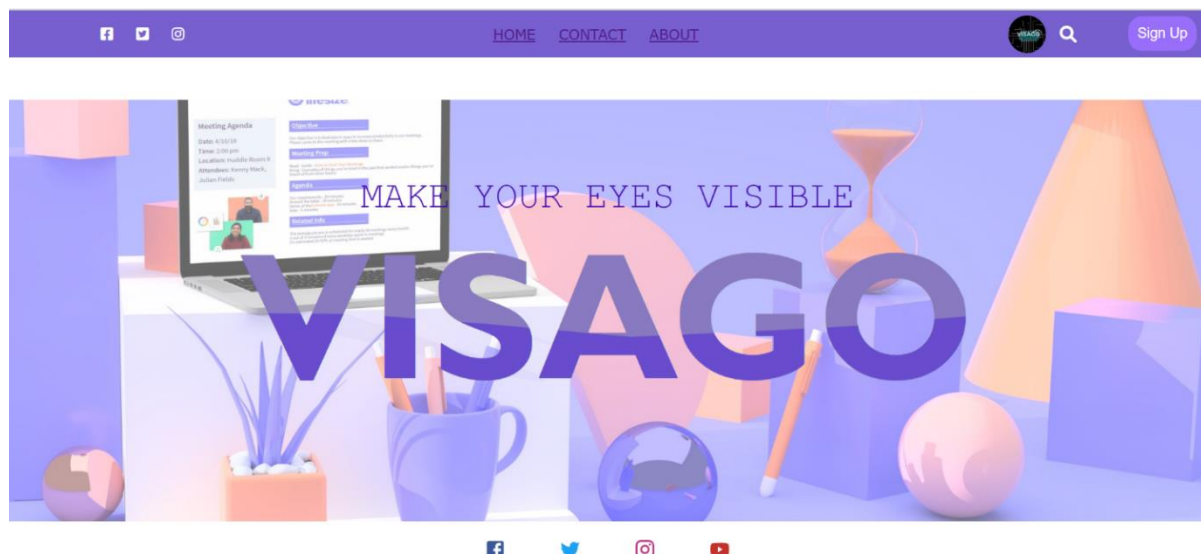


Figure 37: Home page of the front-end

About page

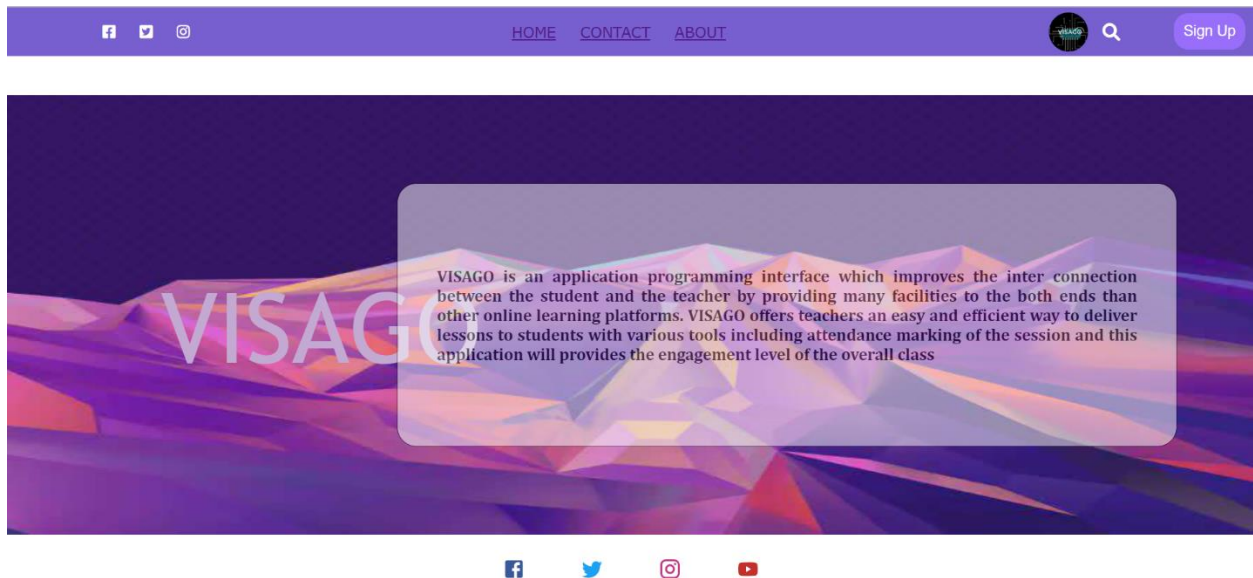


Figure 38: About page of the front-end

Contact page

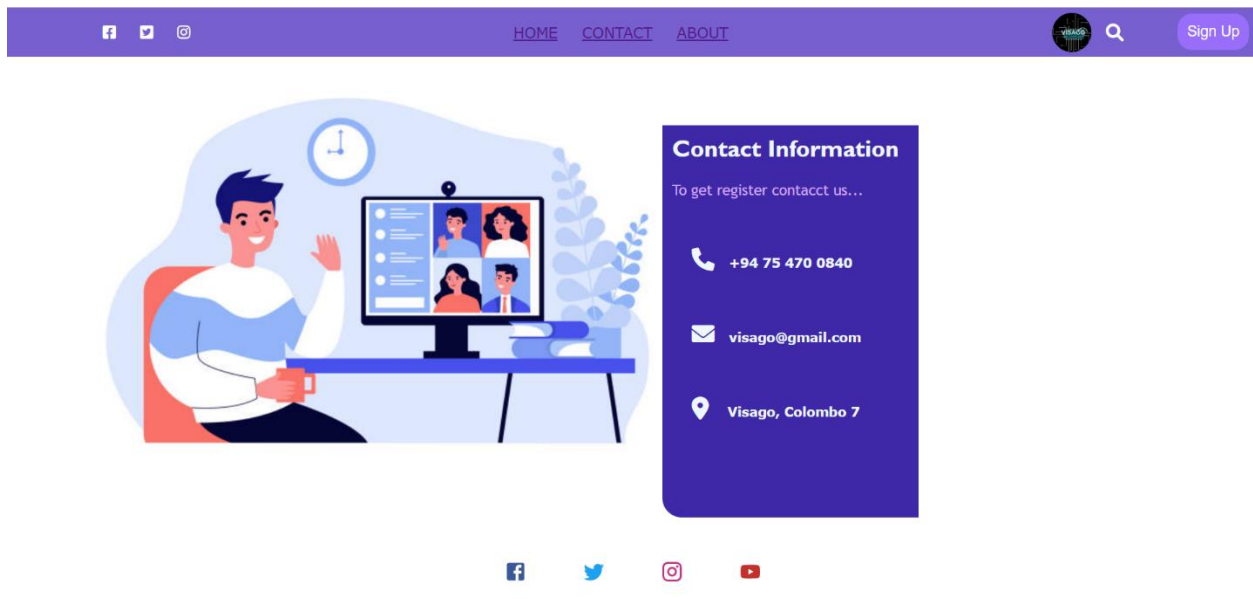
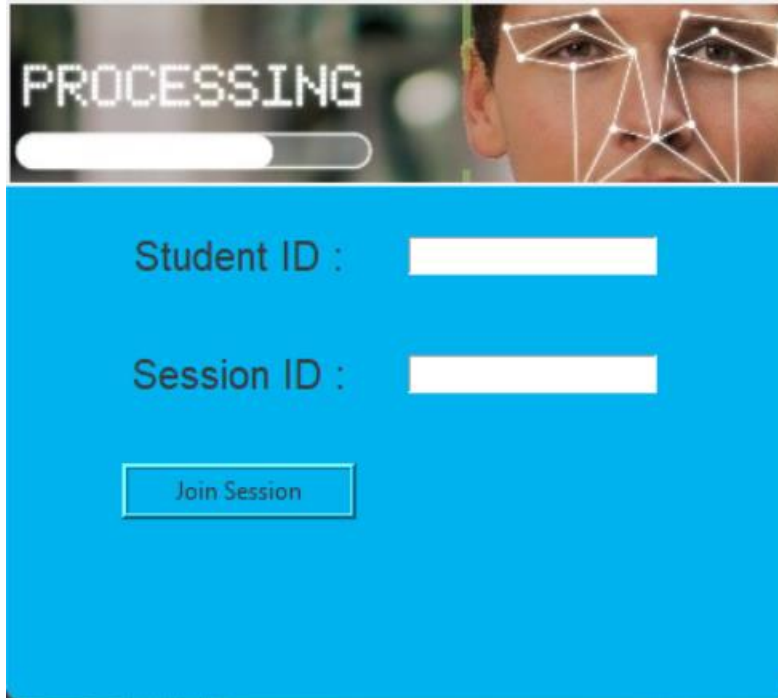


Figure 39: Contact page of the front-end

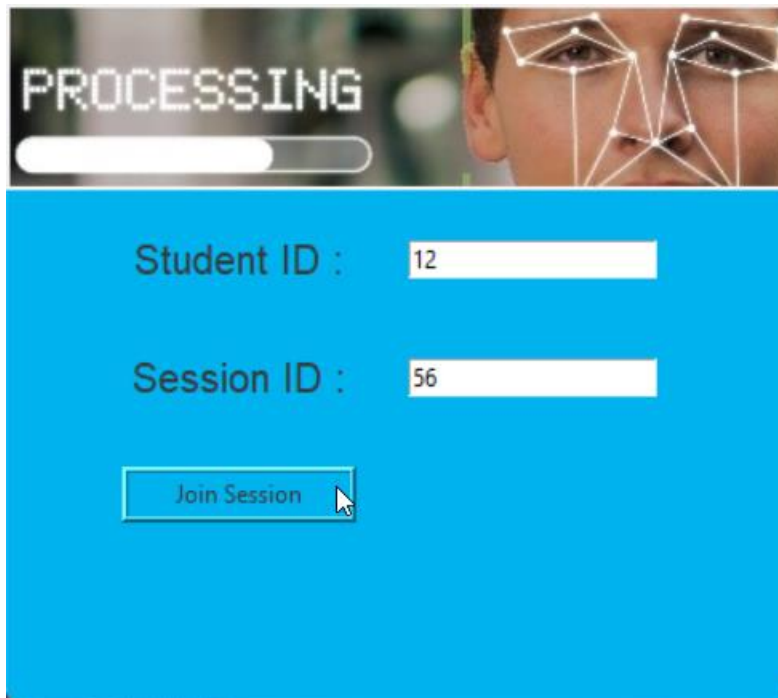
Student part



The image shows a user interface for a student. At the top, there is a header with the word "PROCESSING" in a pixelated font, followed by a progress bar that is approximately 25% full. Below this, the background is a blue gradient. There are two input fields: "Student ID :" and "Session ID :", both of which are empty. Below these fields is a button labeled "Join Session".

Figure 40: Student's user interface

The Moderator will be giving the session id to students as the below image.



The image shows the same user interface as Figure 40, but with the input fields filled. The "Student ID :" field now contains the number "12", and the "Session ID :" field now contains the number "56". The "Join Session" button is still present, and a mouse cursor is hovering over it.

Figure 41: Student's user interface after filling data requirements

Moderator part

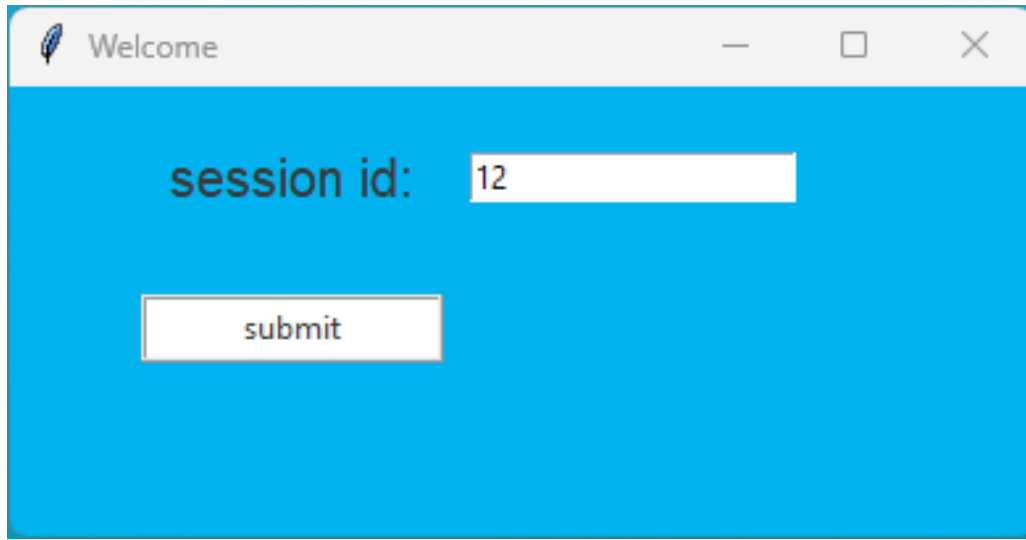


Figure 42: Moderator's user interface

In the moderator side the moderator needs to input the session that the moderator has given to the student as above.

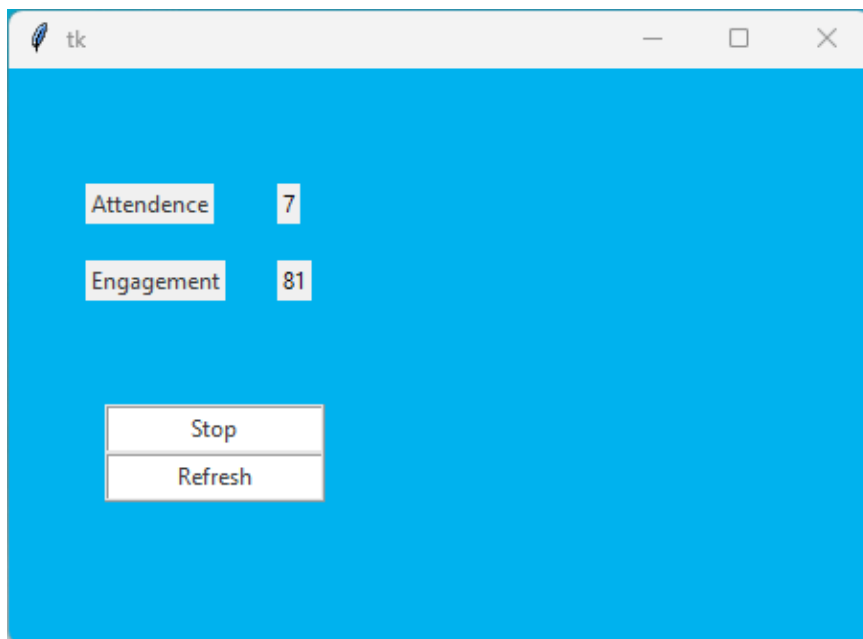


Figure 43: Moderator's user interface with the engagement level

In the above image represents the attendance of the students and the engagement level of the whole class.

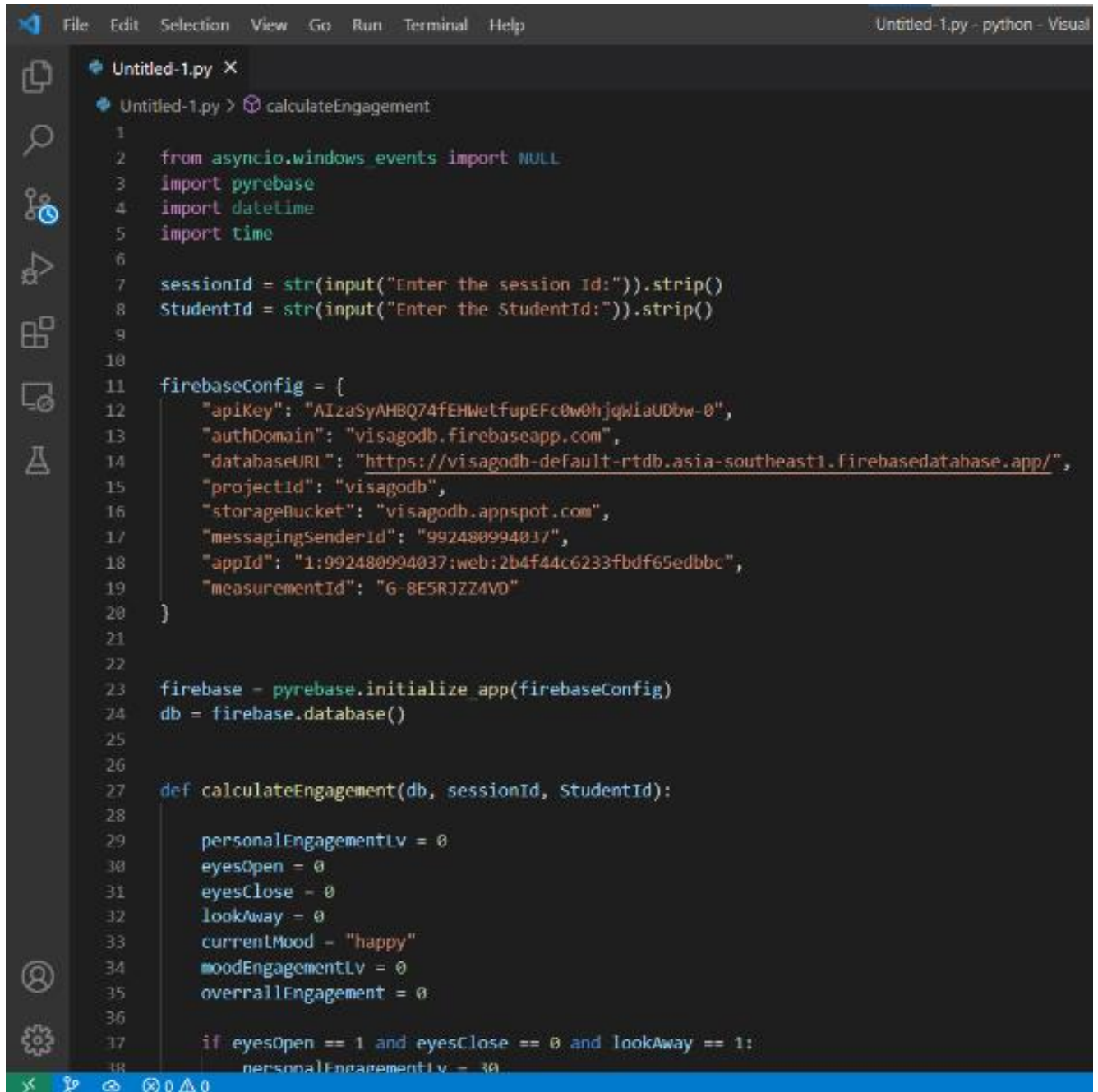
From the refresh button the programme will renew the engagement level and the attendance.

```
1 import tkinter as tk
2 from PIL import ImageTk, Image
3
4 master = tk.Tk()
5 master.geometry("390x350")
6 sessionId = ''
7 StudentId = ''
8
9
10 def login():
11     global sessionId, StudentId
12     sessionId = lblusername.get()
13     StudentId = lblpassword.get()
14
15 def data_get():
16     return sessionId, StudentId
17
18
19 bg_color = "DeepSkyBlue2"
20 fg_color = "#383a39"
21 master.configure(background=bg_color)
22 master.title("Welcome")
23 # ---heading image
24 photo = ImageTk.PhotoImage(Image.open("emo.jpg"))
25 tk.Label(master, image=photo).grid(rowspan=3, columnspan=5, row=0, column=0)
26 # -----Session id
27 tk.Label(master, text="Student ID :", fg=fg_color, bg=bg_color, font=("Helvetica", 15)).grid(row=8, padx=(50, 0),
28                                                                                          pady=(20, 10))
29 lblusername = tk.Entry(master)
30 lblusername.grid(row=8, column=1, padx=(10, 10), pady=(20, 10))
31
32 # ----Student id
33 tk.Label(master, text="Session ID :", fg=fg_color, bg=bg_color, font=("Helvetica", 15)).grid(row=9, padx=(50, 0),
34                                                                                          pady=(20, 10))
35 lblpassword = tk.Entry(master)
36 lblpassword.grid(row=9, column=1, padx=(10, 10), pady=(20, 10))
37
38 # -----button
39 tk.Button(master, text="Join Session", borderwidth=3, relief='ridge', fg=fg_color, bg=bg_color, width=15,
40           command=lambda: [login(), master.destroy()]).grid(row=10, padx=(50, 0), pady=(20, 10))
```

Figure 44: Data stored to the firebase

Above are the codes of the GUI part. In this part there are two text field inputs and a button. The first text field is used to get the session id from the moderator and the second text field is used to get the student id and the button is used to submit the session id and the student id to store data into the firebase.

1.6 Implementation of the backend component Database



```
File Edit Selection View Go Run Terminal Help
Untitled-1.py - python - Visual

Untitled-1.py X
Untitled-1.py > calculateEngagement
1
2 from asyncio.windows_events import NULL
3 import pyrebase
4 import datetime
5 import time
6
7 sessionId = str(input("Enter the session Id:")).strip()
8 StudentId = str(input("Enter the StudentId:")).strip()
9
10
11 firebaseConfig = {
12     "apiKey": "AIzaSyAHBQ74fEHwEtfupEFc0w0hjQwIaUDbw-0",
13     "authDomain": "visagodb.firebaseio.com",
14     "databaseURL": "https://visagodb-default-rtdb.asia-southeast1.firebaseio.com/",
15     "projectId": "visagodb",
16     "storageBucket": "visagodb.appspot.com",
17     "messagingSenderId": "992480994037",
18     "appId": "1:992480994037:web:2b4f44c6233fbdf65edbbc",
19     "measurementId": "G-8E5R3ZZ4VD"
20 }
21
22
23 firebase = pyrebase.initialize_app(firebaseConfig)
24 db = firebase.database()
25
26
27 def calculateEngagement(db, sessionId, StudentId):
28
29     personalEngagementLv = 0
30     eyesOpen = 0
31     eyesClose = 0
32     lookAway = 0
33     currentMood = "happy"
34     moodEngagementLv = 0
35     overallEngagement = 0
36
37     if eyesOpen == 1 and eyesClose == 0 and lookAway == 1:
38         personalEngagementLv = 30
```

Figure 45: Database section with firebase

```

File Edit Selection View Go Run Terminal Help
Untitled-1.py - python - Visual

Untitled-1.py X
Untitled-1.py > calculateEngagement

36
37     if eyesOpen == 1 and eyesClose == 0 and lookAway == 1:
38         personalEngagementLv = 30
39     elif eyesOpen == 0 and eyesClose == 1 and lookAway == 1:
40         personalEngagementLv = 0
41     elif eyesOpen == 1 and eyesClose == 0 and lookAway == 0:
42         personalEngagementLv = 40
43     elif eyesOpen == 0 and eyesClose == 1 and lookAway == 0:
44         personalEngagementLv = 35
45
46     if currentMood == "happy":
47         moodEngagementLv = 60
48     elif currentMood == "sad":
49         moodEngagementLv = 50
50     elif currentMood == "angry":
51         moodEngagementLv = 50
52     elif currentMood == "disgust":
53         moodEngagementLv = 50
54     elif currentMood == "fear":
55         moodEngagementLv = 50
56     elif currentMood == "surprise":
57         moodEngagementLv = 50
58     elif currentMood == "neutral":
59         moodEngagementLv = 50
60
61     overallEngagement = personalEngagementLv+moodEngagementLv
62
63     # dictionary using to pass the engagement level to the db
64     overallEngagementUpdate = {"overall Engagement": overallEngagement}
65
66     engagement_update(db, sessionId, StudentId, overallEngagementUpdate)
67
68
69 def engagement_update(db, sessionId, StudentId, engagement):
70     db.child(sessionId).child(StudentId).update(engagement)
71
72
73 while True:

```

Figure 4610: Calculate engagement level

```

72
73 while True:
74
75     calculateEngagement(db=db, sessionId=sessionId, StudentId=StudentId)
76     print("Updated")
77     time.sleep(5)
78

```

Figure 47: Calculate engagement level

In the above code snippet, it describes the pushing the data into the firebase database. First it saves the two inputs into the variables (student ID, session ID). And then it saves the relevant emotion as a string variable. Line number 37 to 59 validations calculate the relevant student engagement level. In line number 69 method use to update the student engagement in the database. And in line number 73 while block uses to call the engagement calculation method in every 5 seconds continuously.

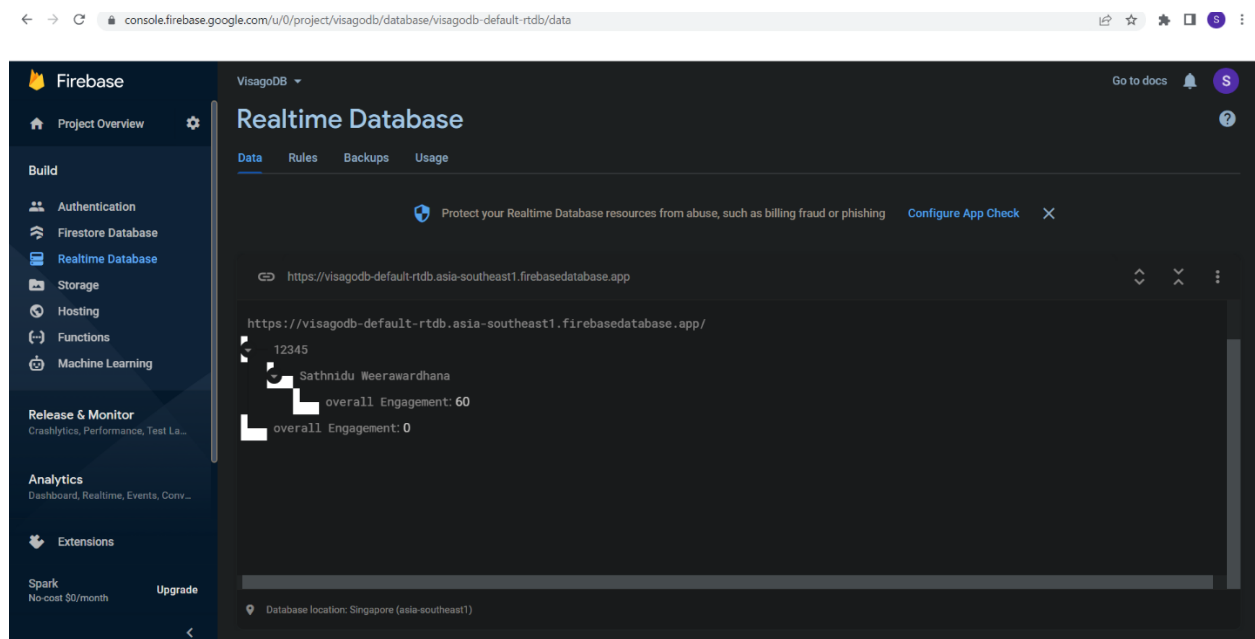
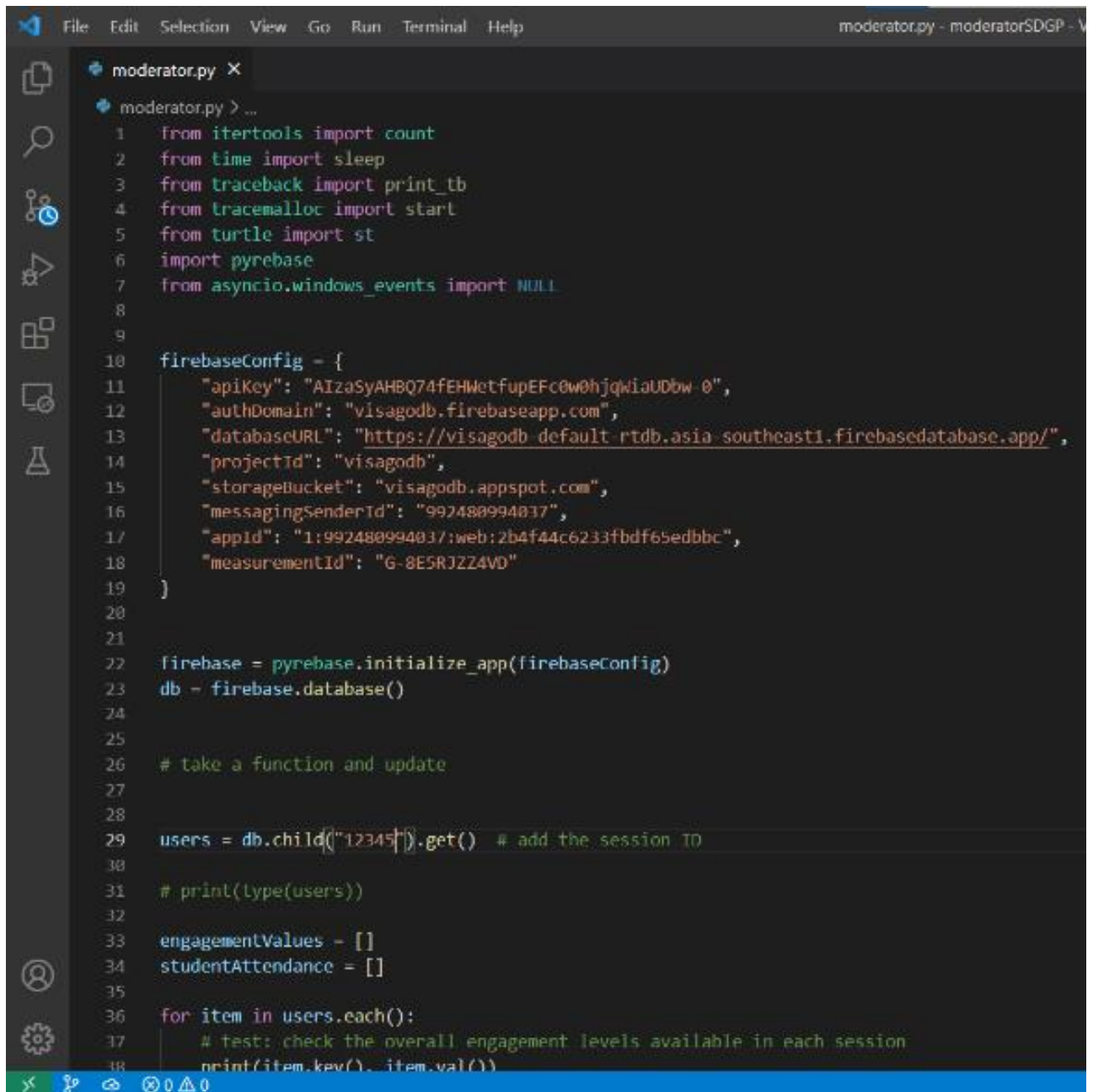


Figure 48: Pushed data into a real-time database

The above screenshot displays how the data is pushed into the real-time database. Only the individual student engagement levels and the names will be there relevant to each session. Every 5 seconds the database will get updated with the new engagement levels of the students. Session ID will be the key to every record.



```
File Edit Selection View Go Run Terminal Help moderator.py - moderatorSDGP - V
moderator.py X
moderator.py > ...
1 from itertools import count
2 from time import sleep
3 from traceback import print_tb
4 from tracemalloc import start
5 from turtle import st
6 import pyrebase
7 from asyncio.windows_events import NULL
8
9
10 firebaseConfig = {
11     "apiKey": "AIzaSyAHBQ74fEHWetfupEFc0w0hjQwiaUDbw-0",
12     "authDomain": "visagodb.firebaseio.com",
13     "databaseURL": "https://visagodb-default-rtdb.asia-southeast1.firebaseio.com/",
14     "projectId": "visagodb",
15     "storageBucket": "visagodb.appspot.com",
16     "messagingSenderId": "992480994037",
17     "appId": "1:992480994037:web:2b4f44c6233fbdf65edbbc",
18     "measurementId": "G-8E5RJZZ4VD"
19 }
20
21
22 firebase = pyrebase.initialize_app(firebaseConfig)
23 db = firebase.database()
24
25
26 # take a function and update
27
28
29 users = db.child("12345").get() # add the session ID
30
31 # print(type(users))
32
33 engagementValues = []
34 studentAttendance = []
35
36 for item in users.each():
37     # test: check the overall engagement levels available in each session
38     print(item.key(), item.val())
```

Figure 49: Configuration of the firebase

```

15
36 for item in users.each():
37     # test: check the overall engagement levels available in each session
38     print(item.key(), item.val())
39     engagementValues += [item.val()["overall Engagement"]]
40     studentAttendance += [item.key()]
41
42
43 print("Engagement Values =", engagementValues)
44 print("Student Id's=", studentAttendance)
45
46 totalEngagement = 0
47 count = 0
48
49 for value in engagementValues:
50     totalEngagement = totalEngagement+value
51     count = count+1
52
53 overallEngagement = totalEngagement/count
54 print("Class engagement level:", overallEngagement)
55

```

Figure 50: Data assigning into the database

The above code snippet, it's related to the moderator side of the project. Basically, it takes all the engagement levels of the students relevant to that session and the names of every student from the database and assigns them into 2 arrays. Moderator only has to enter the session id in line number 29 to get the above-mentioned data. With the use of for loop adding each index value into one variable and dividing it by the count of the students to get the overall engagement level of the whole class.

1.7 GIT Repository

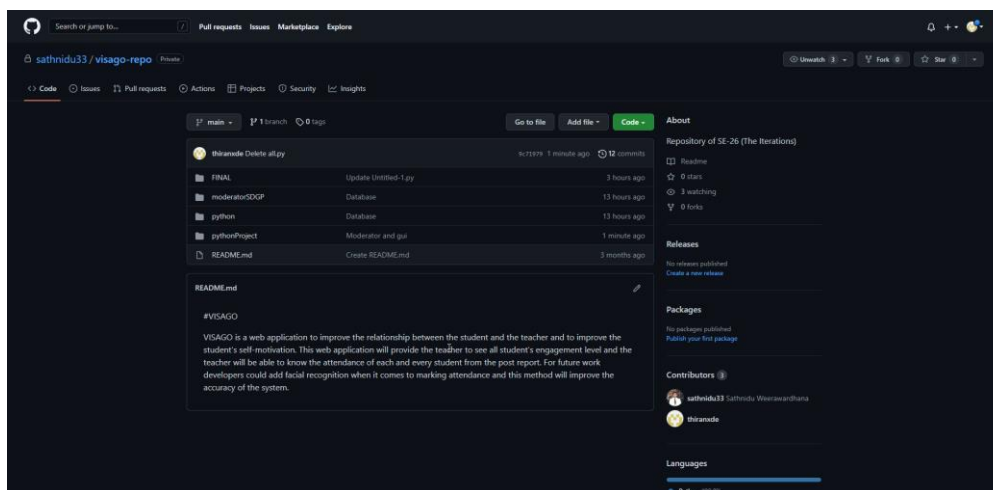


Figure 51: GIT repository

1.8 Chapter Summary

This chapter discussed a number of design decisions taken in connection with the prototype's implementation for this report. This chapter explains the approaches that were utilized to create the suggested framework. After the reasons were provided, the instruments were picked. Python was chosen as the development language for the proposed framework. The application was created with the help of the OpenCV and Firebase libraries. The following chapter will focus on testing and how the applied architecture was validated. It will include information on the experimental methods employed, as well as test cases and ultimate results.

2 Testing

2.1 Chapter introduction

This chapter is mainly focusing on the testing section. The testing criteria will be discussed after the goals and objectives of testing have been discussed. It will next examine testing the functional and non-functional requirements to ensure that it meets all of the requirements. This chapter also include a depth description concerning the constraints faced during the testing period.

2.2 Testing criteria

Testing is a method of determining whether or not a system meets its functional and non-functional requirements. Testing criteria are used to assess the system's functional and structural quality.

Structural quality-This technique focuses primarily on non-functional criteria while also paying attention to the execution of functional needs. Functional quality-More focus on development features that meet the system's technical needs. The system's functional requirement is used to assess the method's quality.

2.3 Testing functional requirements

| Functional requirement | Expected results | Actual result | Status |
|---|---|---|---------------|
| Login to the class | Student/Moderator login to the class | Student/Moderator login to the class | Success |
| Identify if the student camera is turned on | Identifies the student's camera status (ON/OFF) | Identifies the student's camera status (ON/OFF) | Success |

| | | | |
|---|--|--|---------|
| Shows the moderator all engagement levels as a percentage or a number | Moderator receives the percentage of the engagement levels | Moderator receives the percentage of the engagement levels | Success |
| Students can see their own engagement level | Students receive their own engagement level | Students receive their own engagement level | Success |
| The Moderator can get a post report of student engagement levels | Moderator receives a post report of student engagement level | Moderator receives a post report of student engagement level | Fail |
| The moderator can get an attendance report | Moderator receives the attendance report | Moderator receives the attendance report | Success |

2.4 Testing non-functional requirement

Different scenarios can be used to test the system's non-functional needs, and these scenarios will aid in identifying any issues with the non-functional requirements. The system's design is visible when it is opened, and it is a user-friendly design. When login into the system it requests your student id and session id (created by the moderator). So, it ensures that each student login to the related session and covers the accessibility requirement. Getting a prediction from the system demonstrates the system's performance and accuracy. The system's accuracy and efficiency are regarded the most significant requirements. To be accurate, the system should be able to deliver a minimum of 60% success rate.

| Scenario | Expected result | Actual result | Status |
|----------------------|-----------------------------------|-----------------------------------|---------|
| Open the application | Successfully open the system | Successfully open the system | Success |
| Login to the session | Successfully login to the session | Successfully login to the session | Success |

| | | | |
|--------------------------------|--|--|---------|
| Check the camera status | Successfully provide the status | Successfully provide the status | Success |
| Calculate the engagement level | Successfully calculated the engagement level | Successfully calculated the engagement level | Success |
| Logout from the session | Successfully logout from the session | Successfully logout from the session | Success |

2.5 Unit testing

The unit is tasked for testing particular modules or components of a certain application. Because a unit is the smallest tested portion of an application, this is unique. The primary purpose of unit testing is to ensure that the application is free of bugs when in use. In this project, web application is developed by the team. It checks student's Id and avoid mistakes and wrong data entry. If the student enters incorrect or invalid data, student will move on to a wrong session or will not move on to any session. As a result, unit testing is critical for avoiding defects, as well as providing additional benefits such as code reusability, ease of development, code dependability, and ease of debugging.

2.6 Performance testing

Performance testing is included in this section. The practice of determining an application's performance, processing speed and transfer rate, correctness, efficiency, and response to a workload is known as performance testing. Performance testing is vital since it allows you to evaluate the performance, accuracy, and uptime of numerous devices running the same application, as well as establish if a task is feasible. When the application is overloaded, it slows down or crashes.

Validation and training the importance of accuracy testing in performance testing cannot be overstated. The pace of application is affected when training accuracy is great. High precision aids the user in obtaining a precise final outcome.

Model evaluation

```
In [12]: # Model Evaluation

In [13]: acc_tr, loss_tr = model.evaluate(train_data)
          print(acc_tr)
          print(loss_tr)

7641/7641 [=====] - 287s 38ms/step - loss: 0.1485 - accuracy: 0.9431
0.1484653800725937
0.9431137442588806

In [14]: acc_vr, loss_vr = model.evaluate(validation_data)
          print(acc_vr)
          print(loss_vr)

1910/1910 [=====] - 71s 37ms/step - loss: 0.1909 - accuracy: 0.9228
0.19088369607925415
0.922840297221375

In [15]: acc_test, loss_test = model.evaluate(test_data)
          print(acc_test)
          print(loss_test)

1062/1062 [=====] - 39s 36ms/step - loss: 0.3257 - accuracy: 0.8872
0.32571738958358765
0.8872410655021667

In [16]: h = history.history
          h.keys()

Out[16]: dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy', 'lr'])
```

Figure 52: Model evaluation

Graphs of the drowsy model training

Training loss and the validation loss

```
In [18]: import matplotlib.pyplot as plt
          loss_train = history.history['loss']
          loss_val = history.history['val_loss']
          epochs = range(1,11)
          plt.plot(epochs, loss_train, 'g', label='Training loss')
          plt.plot(epochs, loss_val, 'b', label='validation loss')
          plt.title('Training and Validation loss')
          plt.xlabel('Epochs')
          plt.ylabel('Loss')
          plt.legend()
          plt.show()
```

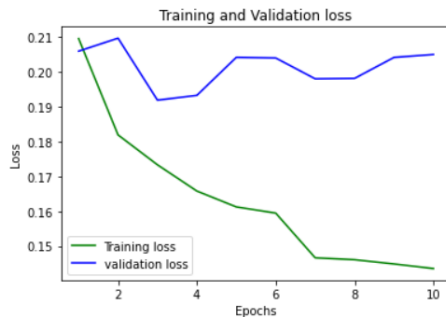


Figure 53: Training loss and the validation loss

Training accuracy and the validation accuracy

```
In [19]: loss_train = history.history['accuracy']
loss_val = history.history['val_accuracy']
epochs = range(1,11)
plt.plot(epochs, loss_train, 'g', label='Training accuracy')
plt.plot(epochs, loss_val, 'b', label='validation accuracy')
plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

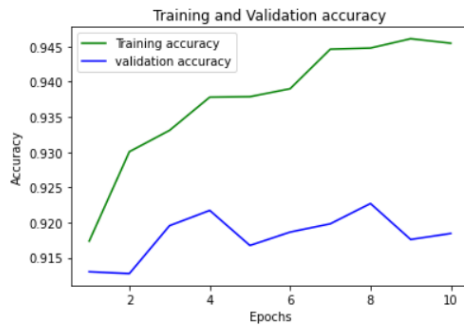


Figure 54: Training accuracy and validation accuracy

Training accuracy and the training loss

```
In [20]: loss_train = history.history['accuracy']
loss_val = history.history['loss']
epochs = range(1,11)
plt.plot(epochs, loss_train, 'g', label='Training accuracy')
plt.plot(epochs, loss_val, 'b', label='Training loss')
plt.title('Training accuracy and loss')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

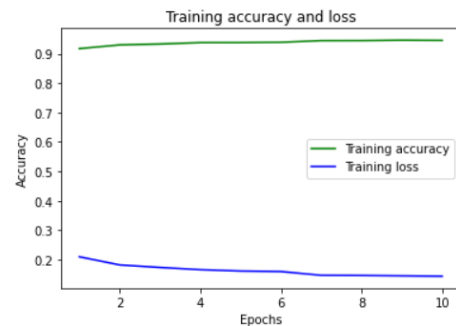


Figure 55: Training accuracy and training loss

Validation accuracy and the validation loss

```
In [21]: loss_train = history.history['val_accuracy']
loss_val = history.history['val_loss']
epochs = range(1,11)
plt.plot(epochs, loss_train, 'g', label='validation accuracy')
plt.plot(epochs, loss_val, 'b', label='validation loss')
plt.title('validation accuracy and loss')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

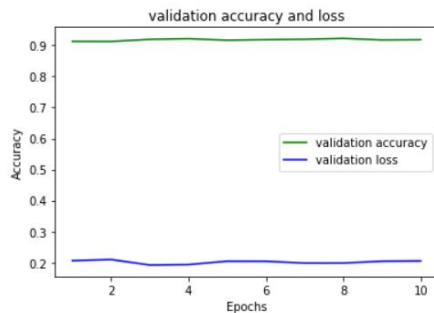


Figure 56: Validation accuracy and the validation loss

As a result, having a high level of exactness for the chosen method implies that proficiency, preparation time, and other variables are all acceptable. The customer can then benefit from the aid of a lightning-fast application.

2.7 Usability testing

This section explores into the specifics the process of the conducted usability testing of this application. Essentially, convenience testing is a means of evaluating an item by putting it to the test on real people.

There are few benefits of testing the usability of the application.

- Enhances the user experience
- The user's perception on the application.

In this application, the main goal is to calculate the student's engagement level and present it as a percentage. As a result of this testing, it helped to discover certain issues in the program and also usability testing assisted in resolving those errors.

2.8 Compatibility testing

This section contains that how the team has done the compatibility testing in application. This testing is a part of non-functional testing to ensure the designed application is compatible with

different devices, hardware and software etc. Basically, this application was tested in different devices and operating systems. This is how compatibility testing is done for this application.

2.9 Chapter summary

Testing is a main stage in the system development life cycle. This section starts with the testing goals and objectives. It includes the testing locations and objectives in this application. In this chapter it confirms the functional and non-functional requirements of uses and check whether necessity is come up short or pass.

Chapter 3: Evaluation

3.1 Chapter overview

This chapter briefly discuss the evaluation criteria, evaluator selection, qualitative evaluations, comments from end users, domain experts, and industry experts in this chapter. This chapter also includes justifications, self-evaluation, and a review from the product's authors.

3.2 Evaluation methods

This project included a review of the methodology and prototype. The methodology used during the implementation stage was utilized to assess the process. The prototype was assessed to see if it included all of the essential characteristics and if it performed properly.

3.3 Quantitative evaluation

Because this is a newly designed application, input and assessments from industry, domain experts, and users are critical to the project's success. A few of their comments are included here.

| Question | |
|--|---|
| What do you think about the scope of the project | |
| Person | Feedback |
| Imseh Ekanayake | Smart idea which focused into a global issue |
| Malin Wickramasinghe | A good way to improve the connection between the students and the teacher in online education system. |
| Miran De Silva | An efficient way to conduct lectures and the sessions through online platform |

| | |
|--|---|
| Question | |
| What do you think about the scope of the project | |
| Person | Feedback |
| Imseh Ekanayake | A really great idea which benefits both the parties (student and the moderator) |
| Malin Wickramasinghe | The first impression is really surprising |
| Miran De Silva | A really good project which can improve the online education platforms. |

| | |
|---|---|
| Question | |
| What do you think about the overall accuracy and the performance of this system | |
| Person | Feedback |
| Imseh Ekanayake | Really impressive with the overall accuracy of the project |
| Malin Wickramasinghe | Stable and accuracy calculated outputs |
| Miran De Silva | Done impressively considering the current situation in the country. |

3.5 Self-evaluation

| Statements | Thiran | Chamath | Sathnidu | Shiny | Damsith |
|---|----------------|----------------|----------------|----------------|---------|
| Contributed ideas to the project | Strongly agree | Strongly agree | agree | agree | |
| Listened and respected the team mates | agree | agree | Strongly agree | Strongly agree | |
| Cooperated with the group members | Strongly agree | Strongly agree | Strongly agree | Strongly agree | |
| Did the fair share of work for this project | Agree | Strongly agree | Agree | Agree | |
| Helped the group to solve problems | Agree | Agree | Agree | Agree | |
| Encourage the other members positively | Strongly agree | Strongly agree | Agree | Agree | |

3.6 Chapter Summary

We discussed the evaluation criteria, how we picked evaluators for this project, subjective assessment, and contributions and opinions from professionals and clients in this area. To handle the examinations, assessment standards were learned. The direction of assessment models covers the creator's self-evaluation. The creator's perspective clarified the undertaking's ideas, scope, implementation, correctness, and potential enhancements.

Chapter 4: Conclusion

4.1 Chapter Overview

This chapter summarizes the information presented in the previous chapters. The major purpose is to double-check each and every phase of the project development process. In addition, what experiences, abilities, and technical knowledge have been obtained, as well as the issues and obstacles that our team members have faced over the development period, will be briefly detailed in this chapter.

4.2 Achievements of aims and objectives

Main aim of this project is to detect the student's engagement level during in an online classroom.

| Objective | Description | Status |
|---|--|---------|
| Identify productive methods and strategies to apply in the prototype implementation by gathering data on related work in this domain. | Worked according to the literature review on previous work and find accurate pre-built models. | Success |
| Maintaining the model's accuracy at a level of at least 75% | Proved everything through diagrams and screenshots in Implementation chapter. | Success |
| Conveniently implementing all functional requirements | Described in the Testing chapter. | Success |

| | | |
|---|---------------------------------|---------|
| Collaborate with industry professionals and other subject specialists to develop a project and prototype. | Described in Evaluation chapter | Success |
|---|---------------------------------|---------|

4.3 Limitations of the research

- Only supports for English language- Currently Visago supports only the English language, since it is a Global language and Visago will support for multi languages in the future.
- Functional requirement limitations-Providing student's own engagement level couldn't produce due to the short time allotted for its implementation. Except this requirement, other requirements are done successfully developed.

4.4 Future enhancements

The time constraint was barely enough to complete the project's employment, and the project isn't completely bug-free and accurate. With the help of a few assessors and a few alternative concepts, project performance is frequently improved in the future.

- Developing a mobile application in Android and iOS platform
- Increase the accuracy of the project up to 100%

References.

- [1] Dlib.net. 2022. *dlib C++ Library - Introduction*. [online] Available at: <<http://dlib.net/intro.html>> [Accessed 1 May 2022].
- [2] Firebase. 2022. *Firebase Realtime Database | Firebase Documentation*. [online] Available at: <<https://firebase.google.com/docs/database>> [Accessed 1 May 2022].
- [3] GeeksforGeeks. 2022. *keras.fit() and keras.fit_generator() - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/keras-fit-and-keras-fit_generator/> [Accessed 1 May 2022].
- [4] Opencv.org. 2022. *About - OpenCV*. [online] Available at: <<https://opencv.org/about/>> [Accessed 1 May 2022].
- [5] Tech Vidvan. 2022. *Python Advantages and Disadvantages - step in the right direction*. [online] Available at: <<https://techvidvan.com/tutorials/python-advantages-and-disadvantages/>> [Accessed 1 May 2022].
- [6] TensorFlow. 2022. *Machine learning education | TensorFlow*. [online] Available at: <<https://www.tensorflow.org/resources/learn-ml>> [Accessed 1 May 2022].