SENIOR DESIGN PROJECT

Horus

Low Level Design Report

Ufuk Palpas, Furkan Demir, Can Kılıç, Asya Doğa Özer, Muzaffer Köksal

Supervisor: Uğur Güdükbay
Jury Members: Erhan Dolak and Tağmaç Topal

February 28, 2022

TABLE OF CONTENTS

1. Introduction 1
  1.1 Design Trade-offs 2
    1.1.1 Scalability vs. Performance 2
    1.1.2 Implementation Time vs. Accuracy 2
    1.1.3 Usability vs. Functionality 2
    1.1.4 Responsiveness vs. Cost 2
  1.2 Interface Documentation Guidelines 3
  1.3 Engineering Standards (UML, IEEE) 3
  1.4 Definitions, Acronyms and Abbreviations 3

2. Packages 3
  2.1 Interface 3
  2.2 Mobile Interface 5
  2.3 Controller Component 6

3. Class Interfaces 8
  3.1 Main Menu Controller 8
  3.2 Face Detector 9
  3.3 Emotion Detector 10
  3.4 Single User Mode Controller 10
  3.5 Crowd Control Mode Controller 11
  3.6 Deception Detection Mode Controller 11
  3.7 Screen Capture Mode Controller 12
  3.8 Results Controller 12
  3.9 Result 13
  3.10 Chart 14
  3.11 Real Time Results Controller 15
  3.12 Real Time Result 16
  3.13 Mood 17
  3.14 Single User 17
  3.15 Crowd Control 18
  3.16 Screen Capture 19
  3.17 App Connection 20
  3.18 Deception Detector 20

4. References 21

1. Introduction

The goal of Horus is to recognize human emotions. Horus will detect the emotions of a single person or a group of people. The detection can be done with a camera or multiple cameras rapidly taking images/video in real-time. Horus will first detect the faces and then analyze their emotions. Horus will be able to analyze the user's emotions from saved videos or images besides real-time detection.

In the single user mode, Horus examines a saved video or real-time footage over a time period, it will report the various emotions detected from a single person. So it will attempt to give the subject's mood from long-term analysis of his emotions with respect to time. For Horus to analyze a user's face, there shouldn't be any obstacles, so if the user wears a mask, hat, etc. Horus will warn the user to remove any such obstacles. Horus will also have a crowd control mode where it will analyze the emotions of a group of people such as stand-up, movie, art activity audiences, or students in a classroom through multiple cameras or a single camera to provide feedback to show organizers, who can look at audiences' moods and decide on which content to keep and which to discard. Horus will also detect emotions on video conference calls such as Zoom. For this option, Horus will contain a screen capturing mode. It will analyze the emotions of the participants on the conference call and provide feedback to the organizers. Horus will also have a deception detection mode to tell whether a sentence said by the user is a lie according to the mimics and the tone.

The results will be shown in real-time on the desktop app with a small window or the mobile app with audio or visuals in various types of graphs that the user selects. The charts will contain detailed information about the scan, such as average, time analysis, camera statistics, etc., to give an appropriate result to the user.

Horus will have many usages. For example, a street artist can place a camera nearby and get feedback from the Horus according to the audience's emotions by using a mobile application that shows the general emotion of the crowd. Another use case can be handling tension and preventing aggression. For example, Horus can be used in a prison's common areas to keep an eye on prisoners. If Horus detects multiple nervous, angry prisoners, it will alert the guards.

This report will give an overview of the Horus' low-level architecture and its design. In the first section, the design trade-offs of Horus and engineering standards are explained. The following sections describe the packages and class interfaces to give an idea about Horus' system.

1.1 Design Trade-offs

1.1.1 Scalability vs. Performance

As the number of people being detected by Horus increases, so does the demand on the hardware. One of the main objectives of Horus is to provide accurate emotion analysis on as many people as possible in a short amount of time; however, performance will suffer as the scale gets more significant. In addition, the accuracy of the emotion detection will be adversely affected as the number of people goes above a certain threshold.

1.1.2 Implementation Time vs. Accuracy

High accuracy is vital for Horus since the application quality depends on how correct it computes the mood averages. If the accuracy is too low, then the application will be basically useless. As the size of the train dataset increases, the implementation time will increase but also the accuracy will increase. Training the model with a larger dataset leads to more correct results. Unfortunately, doing that will increase the time we spend while training the model.

1.1.3 Usability vs. Functionality

The main goal of Horus is to achieve high usability. The menus and usage are simple and clean; therefore, there will not be complex functionalities such as detailed search (according to gender, age, etc.), registration, accounts. It means that the functionality of Horus will be kept at a minimum: it will only include basics which are single-person mode, crowd control mode, deception detection mode, and screen capture mode. Having only simple four modes in the main menu will create easy usage for the user, and s/he will not need any assistance to use the application.

1.1.4 Responsiveness vs. Cost

The quality of the results provided by Horus is heavily dependent on the hardware used, especially its processing power and the resolution quality of the cameras used in the system. Of course, the higher the hardware quality, the higher their price. Therefore, there is a strong positive correlation between the responsiveness of Horus and the cost of the system used to run it.

1.2 Interface Documentation Guidelines

We used camel case format in the naming of our classes, attributes and methods. The class interface descriptions are explained with the format below:

Class <class-name>	
explanation of the class	
Attributes	
<visibility> <type> <example> <visibility> <type> <example>	
Methods	
<visibility> <return-type> <exampleMethod(parameters)>	method description if necessary

1.3 Engineering Standards (UML, IEEE)

UML guidelines[1] are used for each subsection of this report, and the previous ones involve designing specific diagrams, subsystem decompositions, descriptions of the class interfaces, use cases, etc. Any information gathered from outside sources is cited according to IEEE standards[2]. We generally utilized the engineering standards used and recommended by Bilkent University.

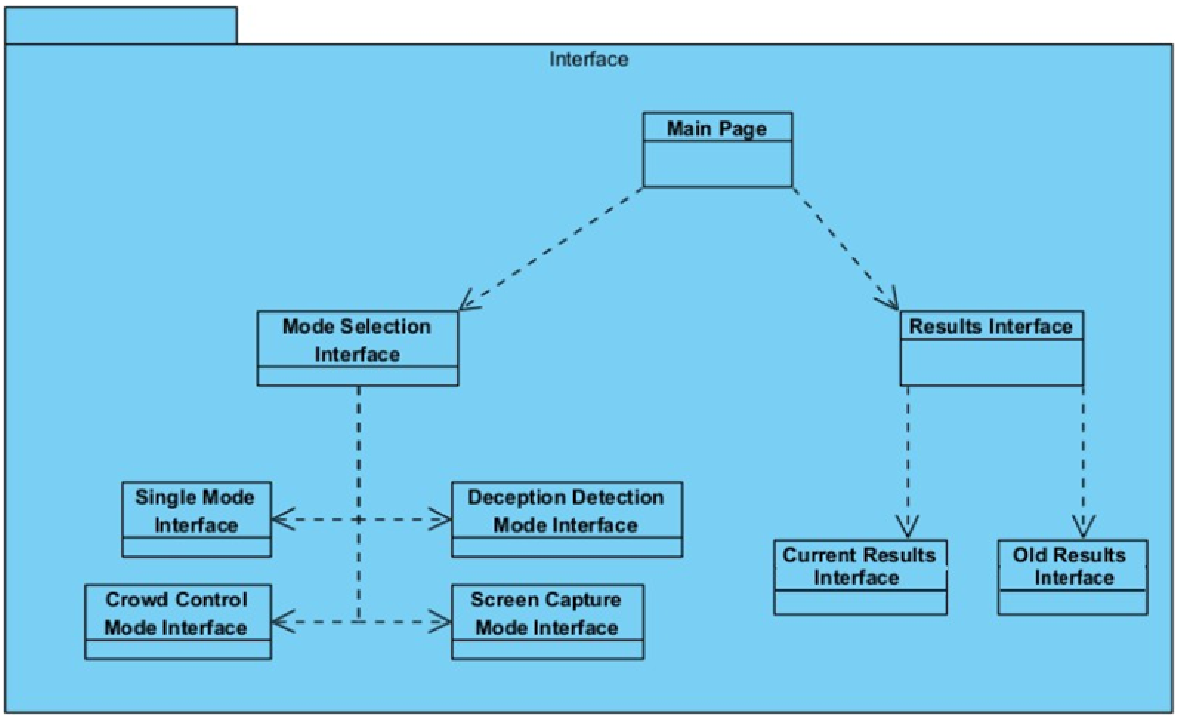
1.4 Definitions, Acronyms and Abbreviations

IEEE - Institute of Electrical and Electronics Engineers
UML - Unified Modeling Language

2. Packages

2.1 Interface

Our outermost package is the Interface package (see Figure 3) which consists of classes that are responsible for the interface of the program. This package aims to provide a user interface that displays the results, provides the navigation, and so on.


Interface
Main Page
Mode Selection Interface
Results Interface
Single Mode Interface
Deception Detection Mode Interface
Crowd Control Mode Interface
Screen Capture Mode Interface
Current Results Interface
Old Results Interface
Figure 1: General Interface

- **Main Page:** Main Page is responsible for the welcoming page that has links to the given interfaces.

- **Mode Selection Interface:** The mode selection interface is responsible for handling the selection between mode options: “single user”, “deception detection”, “crowd control” or “screen capture”.

- **Single Mode Interface:** Single mode interface is responsible for handling the single user mode functionalities.

- **Crowd Control Mode Interface:** The crowd mode interface is responsible for handling the main crowd mode functionalities and camera selection.

- **Deception Detection Mode Interface:** The deception detection mode interface is responsible for handling the deception detection mode functionalities.

- **Screen Capture Mode Interface:** The screen capture mode interface is responsible for handling the screen capture mode functionalities.

- **Results Interface:** Results interface is responsible for “choose analysis” and “choose analyses type” pages.

- **Current Results Interface:** Current results interface is responsible for showing the current results page and showing the current results inside mode pages.

● **Old Results Interface:** Old results interface is responsible for showing the old results page.

2.2 Mobile Interface

The mobile interface package (see Figure 4) contains classes that display the face and emotion detection algorithms' data in graphs and charts within the user's mobile device. This package aims to provide similar functionality to the interface package within mobile devices.

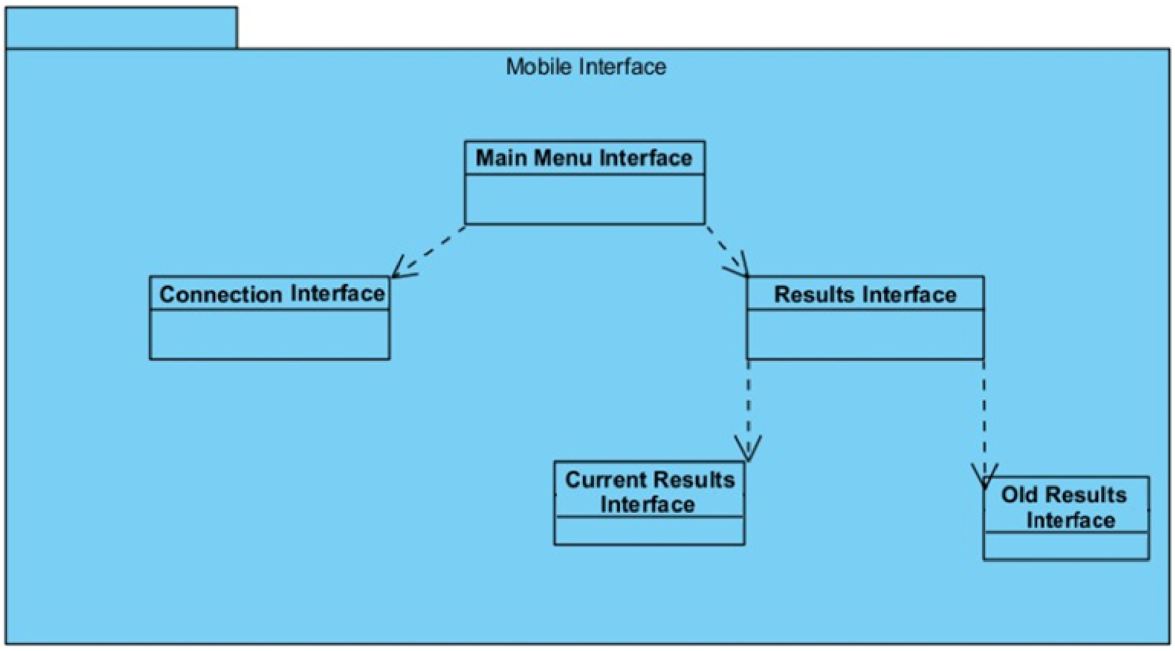


Figure 2: Mobile Interface Package

● **Main Menu Interface:** Main menu is responsible for the welcoming page that has links to the given interfaces.

● **Connection Interface:** The connection interface is responsible for showing the Bluetooth connection status.

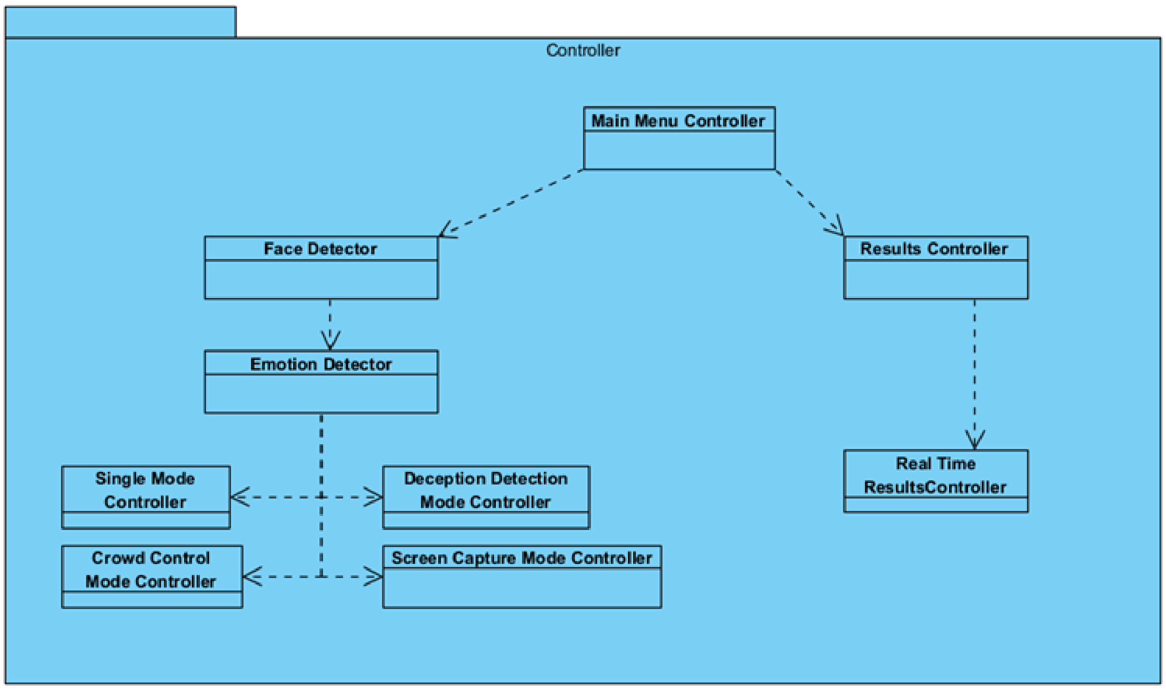
● **Results Interface:** Results interface is responsible for "choose analysis" and "choose analyses type" pages.

● **Current Results Interface:** Current results interface is responsible for showing the current results page and showing the current results inside mode pages.

● **Old Results Interface:** Old results interface is responsible for showing the old results page.

2.3 Controller Component

The Controller package (see Figure 5) consists of classes that are responsible for data processing that comes from the Interface package. These controller classes provide emotion detection algorithms and result displaying functions.



Controller
Main Menu Controller
Face Detector
Results Controller
Emotion Detector
Real Time ResultsController
Single Mode Controller
Deception Detection Mode Controller
Crowd Control Mode Controller
Screen Capture Mode Controller

Figure 3: Controller Package

● **Main Menu Controller:** Main Menu Controller is responsible for interactions that come from the corresponding interface. Furthermore, the Main Menu Controller handles the connection operation with the mobile app.

● **Face Detector:** Face Detector detects faces within the image and frames them for the emotion detector to process. It can detect multiple faces within an image.

● **Emotion Detector:** Emotion Detector individually detects emotions of the faces provided by the face detector.

● **Single Mode Controller:** Single Mode Controller is using the Emotion Detector class for emotion detection algorithm on a single user. Then, forward the output data to be displayed on the selected type of graph.

● **Crowd Control Mode Controller:** Crowd Controller Mode Controller is using the Emotion Detector class for the emotion detection algorithm on multiple faces. It examines all the faces captured and generates an average output. Then, forward the output data to be displayed on the selected type of graph.

● **Deception Detection Mode Controller:** Deception Detection Mode Controller is using the Emotion Detector class for emotion detection algorithm on a single user and alters it to detect any kind of deception.

● **Screen Capture Mode Controller:** Screen Capture Mode Controller is using the Emotion Detector class for the emotion detection algorithm on multiple faces that are present on the screen via video conference call. It examines all the faces captured and generates an average output. Then, forward the output data to be displayed on the selected type of graph.

● **Results Controller:** Results Controller takes the data that has been generated by another controller and generates a graph of the selected type.
● **Real Time Results Controller:** Real Time Results Controller takes the data that has been generated by another controller and updates it frequently with the new data coming in from the controllers and displays the graph of the selected type.

3. Class Interfaces

3.1 Main Menu Controller

Class MainMenuController	
Main Menu Controller is responsible for interactions that come from the corresponding interface. Furthermore, the Main Menu Controller handles the connection operation with the mobile app.	
Attributes	
private ResultsController resultsMenu private AppConnection appConnection private RealTimeController realTimeController private DeceptionDetectionModeController deceptionDetection ModeController private SingleUserModeController singleUserModeController private CrowdControlModeController crowdControlModeController private ScreenCaptureModeController ScreenCaptureModeController	
Methods	
public AppConnection getAppConnection()	Creates an AppConnection instance
public ResultsController getResultsMenu()	Creates a ResultsController instance
public void sendResultsToApp()	Sends the results of the scan to the mobile app via AppConnection
public RealTimeController createRealTimeController()	Creates a RealTimeController instance
public DeceptionDetectionModeController createDeceptionDetectionModeController()	Creates a DeceptionDetectionModeController instance
public SingleUserModeController createSingleUserModeController()	Creates a SingleUserModeController instance
public CrowdControlModeController createCrowdControlModeControllerMenu()	Creates a CrowdControlModeController instance
public ScreenCaptureModeController createScreenCaptureModeControllerMenu()	Creates a ScreenCaptureModeController instance
public void downloadResult()	Downloads the result to the device

3.2 Face Detector

Class FaceDetector	
Face Detector detects faces within the image and frames them for the emotion detector to process. It can detect multiple faces within an image.	
Attributes	
private int[][][] image private int[] coordinates	
Methods	
public Model model()	Creates a Face Detector model.
public int[] detectFaces()	Detects faces and returns the coordinates of the corners of the box around them. It uses the model() function.
public int[][][] getImage()	Returns the full image in RGB format.
public boolean faceNotDetected()	Returns whether a face was successfully detected or not.

3.3 Emotion Detector

Class EmotionDetector	
Emotion Detector individually detects emotions of the faces provided by the face detector.	
Attributes	
private int topLeft private int topRight private int bottomLeft private int bottomRight private FaceDetector faceDetect	
Methods	
public Mood model()	Creates an Emotion Detector model.
public Mood detectMood()	Detects the emotion of the given face using the model() function.
public void setCoordinates(int[])	Sets the the coordinates of the corners of the box around the face that's emotion will be detected .
public FaceDetector createFaceDetector()	Creates a FaceDetector object.

3.4 Single User Mode Controller

Class SingleUserModeController	
This class is using the SingleUser class and its methods to apply the Single User Mode.	
Attributes	
private SingleUser singleUser	
Methods	
public SingleUser createSingleUser()	Creates an SingleUser instance.

3.5 Crowd Control Mode Controller

Class CrowdControlModeController	
This class is using the CrowdControl class and its methods to apply the Crowd Control Mode.	
Attributes	
private CrowdControl crowdControl	
Methods	
public CrowdControl createCrowdControl()	Creates a CrowdControl instance.

3.6 Deception Detection Mode Controller

Class DeceptionDetectionModeController	
This class is using the DeceptionDetection class and its methods to apply the Deception Detection Mode.	
Attributes	
private DeceptionDetection deceptionDetection	
Methods	
public DeceptionDetection createDeceptionDetection()	Creates a DeceptionDetection instance.

3.7 Screen Capture Mode Controller

Class ScreenCaptureModeController	
This class is using the ScreenCapture class and its methods to apply the Screen Capture Mode.	
Attributes	
private ScreenCapture screenCapture	
Methods	
public ScreenCapture createScreenCapture()	Creates a ScreenCapture instance.

3.8 Results Controller

Class ResultsController	
Results Controller takes the data that has been generated by another controller and generates a graph of the selected type.	
Attributes	
private Result[] savedResults	
Methods	
public Result[] getSavedResults()	Returns the savedResults as an array.
public boolean saveResult(Result)	It saves the Result to the savedResults array and returns true.

3.9 Result

Class Result	
This class keeps general information about the analyzed data such as date, camera count, duration, mood counts, face counts, etc. This class will be used for various purposes such as sorting the analyzed data, using the information in different analyses.	
Attributes	
private string date private int CameraCount private string outputType private string time private int duration private int faceCount private int[][] moodCounts	
Methods	
public int getFaceCount()	Returns the faceCount in the analyzed data.
public void setFaceCount(int)	Sets the faceCount.
public int[][] getMoodCounts()	Returns the count of the moods in an array according to time/frame.
public void setMoodCounts(int[][])	Sets the MoodCounts.
public void addMood(Mood, string)	Adds a mood to the MoodCounts array.
public int getDuration()	Returns the duration of the data.
public void setDuration(int)	Sets the duration of the data.
public string getTime()	Returns the time of the analysis.
public void setTime(String)	Sets the time of the analysis.
public string getOutputType()	Returns the output type of the result.
public void setOutputType(string)	Sets the output type of the result.
public int getCameraCount()	Returns the camera count.
public void setCameraCount(int)	Sets the camera count.
public string getDate()	Returns the date of the analysis.
public void setDate(String)	Sets the date of the analysis.
public Chart createChart()	Creates a Chart object.

3.10 Chart

Class Chart	
This class is used to create charts after the analysis of data. Information kept in this class will be used to create various types of charts. This class is also used for creating charts which will be shown to the user.	
Attributes	
private Mood[] moods private int[] percentages private int[][][][] colors	
Methods	
public int[] getPercentages()	Returns the percentages of each mood in an array.
public void setPercentages(int[])	Sets the percentages of each mood in an array.
public Mood[] getMoods()	Returns the counts of each mood in an array.
public void setMoods(Mood[])	Sets the counts of each mood in an array.
public int[][][][] getColors()	Returns the colors for the moods in RGB format.
public void setColors(int[][][][])	Sets the colors for the moods in RGB format.

3.11 Real Time Results Controller

Class RealTimeResultsController	
Real Time Results Controller takes the data that has been generated by another controller and updates it frequently with the new data coming in from the controllers and displays the graph of the selected type. Results of the analysis can be achieved by using this class too.	
Attributes	
private RealTimeResult[] cumulativeResults private RealTimeResult currentResult	
Methods	
public RealTimeResult[] getCumulativeResults()	Returns the cumulativeResults array.
public void addToCumulativeResults(RealTimeResult)	Adds a RealTimeResult to the cumulativeResults array.
public RealTimeResult computeResult()	Updates the currentResult.
public RealTimeResult getCurrentResult()	Returns the currentResult.

3.12 Real Time Result

Class RealTimeResult	
This class keeps the real time analysis results. This class is used by both the RealTimeController and Chart classes. It also contains detailed information about ongoing analysis.	
Attributes	
private int time private int[] moodCounts private int faceCount private int frameCount private Chart outputType	
Methods	
public int getFrameCount()	Returns the number of the current frame.
public void setFrameCount(int)	Sets the number of the current frame.
public int getFaceCount()	Returns the faceCount.
public void setFaceCount(int)	Sets the faceCount.
public int[] getMoodCounts()	Returns the count of each mood in an array.
public void setMoodCounts(int[])	Sets the count of each mood in an array.
public void addMood(Mood)	Adds a mood or increases the one if available to the moodCounts array.
public int getTime()	Returns the time of the analysis.
public void setTime(int)	Sets the time of the analysis.
public Chart getOutputType()	Returns the output type.
public void setOutputType(Chart)	Sets the output type.

3.13 Mood

Class Mood
This is an enumeration which declares the moods: Happy, Sad, Neutral, Bored, Interested, Anxious, Angry, Excited. These moods will be used in every part of the analysis.
Attributes
enumeration: Happy enumeration: Sad enumeration: Neutral enumeration: Bored enumeration: Interested enumeration: Anxious enumeration: Angry enumeration: Excited

3.14 Single User

Class SingleUser	
This class contains the emotions detectors itself for single user mode. Facial emotion detection will be held under this class. Trained data will be used by this class.	
Attributes	
private EmotionDetector emotionDetector	
Methods	
public EmotionDetector createEmotionDetector()	Initializes the emotion detector class.

3.15 Crowd Control

Class CrowdControl	
This class is using the Emotion Detector class for the emotion detection algorithm on multiple faces. It examines all the faces captured and generates an average output. Then, forward the output data to be displayed on the selected type of graph. It stores all the required information about crowd control mode such as face count and camera count.	
Attributes	
private EmotionDetector[] emotions private int faceCount private int cameraCount	
Methods	
public int getFaceCount()	Returns the count of the faces.
public void setFaceCount(int)	Sets the count of the faces.
public int getCameraCount()	Returns the count of the cameras.
public void setCameraCount(int)	Sets the count of the cameras.
public EmotionDetector[] getEmotions()	Returns the moods of faces using the EmotionDetector class.
public float calcAverage()	Calculates the average emotion value provided from all faces.

3.16 Screen Capture

This class is using the Emotion Detector class for the emotion detection algorithm on multiple faces that are present on the screen via video conference call. It examines all the faces captured and generates an average output. Then, forward the output data to be displayed on the selected type of graph.	
Attributes	
private EmotionDetector emotionDetector private int[][][][] inputSequence	
Methods	
public int getFaceCount()	Returns the count of the faces.
public void setFaceCount(int)	Sets the count of the faces.
public void EmotionDetector[] getEmotions()	Returns the moods of faces using the EmotionDetector class.
public float calcAverage()	Calculates the average emotion value provided from all faces.

3.17 App Connection

Class AppConnection	
This class manages the connection between the desktop application and the mobile application. It arranges connection and data transfer between applications	
Attributes	
private boolean isConnected private String connectedDevice	
Methods	
public boolean getIsConnected()	Returns true if the connection is established.
public void setIsConnected(boolean)	Sets the connect status.
public string getConnectedDevice()	Returns the connected mobile device.
public void setConnectedDevice(string)	Sets the connected mobile device.
public void showResults()	Shows the sent data on mobile application
public void connect()	Used to establish a connection with a mobile device

3.18 Deception Detector

Class DeceptionDetector	
This class is using the Emotion Detector class for an emotion detection algorithm on a single user and alters it to detect any kind of deception. The results of the deception detector can be accessed from this class.	
Attributes	
private EmotionDetector emotionDetector private int[][][][] inputSequence	
Methods	
public boolean deceptionDetected()	Returns whether or not deception was detected.
public boolean sentenceDetected()	Returns whether or not a visible change in the facial structure is detected.

4. References

[1] IBM, "UML - Basics," June 2003. [Online]. Available: http://www.ibm.com/developerworks/ rational/library/769.html.[Accessed 25-Feb-2022].

[2] IEEE, "IEEE Citation Reference," September 2009. [Online]. Available: https://m.ieee.org/ documents/ieeecitationref.pdf.[Accessed 25-Feb-2022].