Real-time Human Attribute Analysis Web Application

CS310: Final Report

Year of Study: Third

2022-2023

Neema Raiyat
2001540

Supervisor:
Greg Watson

Department of Computer Science
University of Warwick

Abstract

Emotions form the fundamental basis for physical communication between humans. They give us a glimpse about our thought process and an insight of what is going on inside the mind. Therefore, measuring the changes in emotion induced by products is clearly an object of interest for businesses. Many existing systems that measure and monitor a human's emotion are restricted as they are unimodal, i.e. they use only a single channel of sensory input and hence are less reliable. In our work, we successfully design and develop a multimodal web application using *React.js* capable of, real-time human attribute analysis (RTHAA), that not only monitors a human's emotions using their facial expression, using deep learning, but also measures their speech-content sentiment, using sentiment analysis, resulting in a more comprehensive and robust tool. Other human attributes like age and sex are also measured as they can provide useful data on how different demographics interact with the system. User testing with people of varying ages, sexes and ethnicities showed that our application can more reliably measure a human's overall impression of a system compared to contemporary tools.

Keywords: Web Development, React.js, Computer Vision, Human Computer Interaction, Machine Learning, Facial Expression Recognition, Sentiment Analysis, User Interface Design

Acknowledgements

I would like to sincerely thank my supervisor, Dr Greg Watson, for his support over the duration of this project. I would also like to thank my second assessor, Professor Nathan Griffiths, who provided valuable feedback on my presentation as well as offered advice about what to include in the final report. Furthermore, I would like to thank my personal tutor, Ian Saunders, for his support by always stressing his availability for any help.

Lastly, I would like to thank my family members for their constant moral support over the duration of what has been a very difficult year.

Contents

List of Figures

List of Tables

Word Count: 17, 977

1 Introduction and Motivation

Computers occupy an increasingly more dominant role within the world today (Statista 2022). We are almost always interfacing with computers, whether it is during work, entertainment or even when socialising with friends and family. Consequently, it has become important for businesses to develop software which enhances the user's experience, and thus it has become essential for developers to be able to accurately measure a user's experience of a system. One very popular way of gauging user experience is through usability testing, typically via usability questionnaires or facilitated tasks. The drawbacks of such methods of usability testing are that they require facilitators to be present, which may affect the decision making of participants and can present logistical issues if the software is to be tested on a large number of participants. Furthermore, there is a lost opportunity with questionnaires and surveys as they, for example, do not capture factors such as first impressions, which Lindgaard et al. (2006) have shown to be a strong indicator as to whether users will enjoy their experience.

This project lies within the broader sphere of human computer interaction (HCI), a field that focuses on the design, evaluation and study of the interaction between humans and computer systems. HCI aims to understand how people interact with technology and how technology can be designed to be more user-friendly. Ultimately, we want to improve the way users interface with digital systems by enhancing current forms of usability testing, and in this project, we do this by developing a tool, *RTHAA*, capable of *Real-time Human Attribute Analysis*, i.e. a tool that allows us to monitor a human's impression of a digital system by analyzing their emotions via their facial expressions, sentiment via speech input and other human attributes (like age and sex) in real-time. This allows us to overcome the usual limitations of usability testing as we can use said tool in conjunction with the software being tested, to measure and track the emotions and sentiment of participants throughout the study and provide analytical data without the need for a facilitator to be present. Such a tool is not restricted to just software testing, but rather Kołakowska et al. (2013) state that it has potential applications in various other domains such general

product testing, education and video games.

It is worth providing clarity on what is meant by certain related — but not interchangeable — terms used frequently throughout this report. As popularised by Dzedzickis et al. (2020), *'emotion'* will is used in the context of referring to the psychological state deduced by the expression displayed on a human's face in response to stimuli. This generally falls into one of seven emotion classes: neutrality, happiness, sadness, anger, fear, disgust and surprise. On the other hand, we use *'expression'* similarly to how it is defined by Bettadapura (2012), where it refers to the physical movements and changes in the muscles of the face that convey emotional states. This includes frowning, smiling, raising eyebrows, wrinkling the forehead, and other facial movements that are associated with specific emotions. Additionally, *'sentiment'* is typically used in the context of human speech and refers to the general attitude or feeling towards stimuli, which is often expressed as positive, negative, or neutral. Note that we use *'speech'* to refer to the actual content of the speech, i.e. we examine the linguistic and semantic aspects of speech to identify sentiment as opposed to the acoustic features of speech, such as pitch, intensity and rhythm. Therefore, *'speech sentiment analysis'* will be made to refer to, for the purpose of brevity, sentiment analysis being conducted on the content of the speech (spoken words), not to be confused with speech emotion recognition (SER) which concerns the analysis of the acoustic features of the speech signal itself. We also differentiate *'sex'* from *'gender'* using the distinction popularised by Prince (2005) and Deaux (1985). The term *'sex'* refers to the biological and physiological differences between two distinct classes: male and female, as opposed to *'gender'*, which is related to social, cultural and psychological characteristics associated with masculinity and femininity, which can be fluid and may not necessarily align with an individual's biological sex. We adopt the term *'human attribute'* or just *'attribute'* — similarly to how it is used by Yaghoubi et al. (2020) — to encompass a person's emotional state, age or sex as aspects of human characteristics.

Many existing systems that measure and monitor a human's emotion are restricted as they are unimodal i.e. they use a single channel of sensory input, such as images alone to classify emotions. The challenge this project entails, is to develop

a multimodal web application that measures human emotions (and other human attributes) and sentiment accurately using facial expressions from video and speech inputs from audio, all measured in real-time while providing analytical data. Multimodal systems are often considered more robust (Poria et al. 2017, Garcia-Garcia et al. 2018) and accurate compared to unimodal systems, as they can capture information from multiple sources to provide a more comprehensive assessment of an individual's emotional state as opposed to just relying on, for instance, facial expressions. Section 1 will discuss in greater depth the advantages of multimodal systems over unimodal.

Since the web application would be used without the presence of a facilitator but rather the user alone, it should be able to inform the user if the environment is adequately lit so that the pre-trained models used to classify human attributes have a higher chance of classifying more accurately. This is because poor lighting conditions can reduce image quality and reduce the visibility of facial features such as eyebrows, eyes and skin colour, which pre-trained models rely on to infer human attributes.

Moreover, the majority of systems capture up to six emotions, namely happiness, sadness, anger, fear, disgust and surprise as they constitute the six basic human emotions stated by Ekman (1992) in his widely accepted theory of core emotions. In this project, we will detect a seventh emotional state, neutrality, as this provides more useful analytical data. There will also be an analysis component to this project where we compare and contrast existing pre-trained models used for emotion detection and sentiment analysis.

2 Background

While this project is primarily focused on software production, it involves many diverse areas of computer science, including machine learning, sentiment analysis, software design, web development, user interface design, and the broader field of human-computer interaction.

In this section, we will begin by discussing literature related to the general field of *'emotion recognition'*, which refers to the practice of determining a person's emotion through any means possible such as using physiological signals. We will further motivate the use of the tool this project aims to create by summarising the research done into these other methods of emotion recognition and why they are not entirely sufficient for many scenarios, hence motivating the need for emotion recognition via facial expressions — *facial expression recognition* (FER) — and speech sentiment analysis. We will then dedicate a significant amount of discussion to the research and analysis of existing machine learning (and deep learning) methods and models used in FER as this forms a large part of the decision making process when deciding which models to use in our application.

Furthermore, we will discuss different approaches to sentiment analysis, a natural language processing (NLP) technique that involves determining the sentiment expressed in a piece of text. Moreover, we will briefly discuss web development and user interface (UI) design, however, more discussion will be dedicated to this in the design and implementation section of this report. Lastly, we will compare and contrast popular, existing human attribute recognition tools and highlight how RTHAA improves upon these existing tools. These improvements consequently form the objectives of this project.

2.1 Emotion Taxonomy in Facial Expression Research

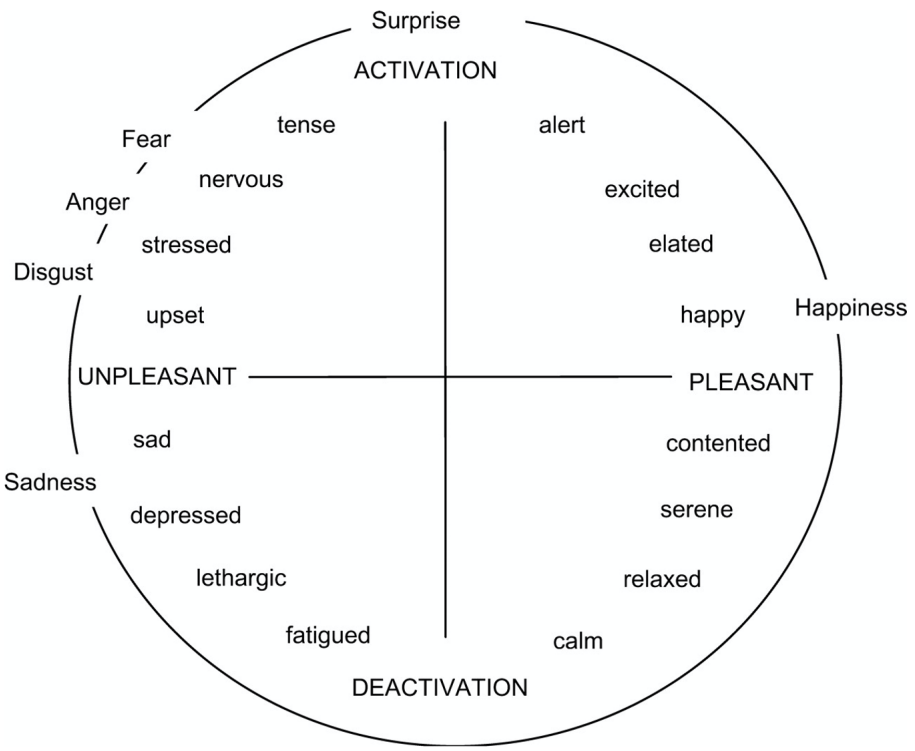


Figure 1: Ekman's discrete basic emotions (outer circle) and Russell's Circumplex Model of Affect (two-dimensional plane), a reproduction from Feldman Barrett & Russell (1998)

Psychologists have developed various taxonomies, often called *'models'*, for classifying emotions, ranging from universally displayed basic emotions to complex, culturally specific ones. Among these models, Ekman's basic set of emotions (Ekman 1992) and Russell's circumplex model of affect (Russell 1980) have been dominant in facial expression research, both of which can be seen in Figure 1. Ekman & Friesen (1971) proposed six prototypical (basic) emotions, namely: anger, disgust, fear, happiness, sadness, and surprise. These emotions are universally displayed and recognised from facial expressions. This discrete model has gained popularity as it is easy for humans to recognise and describe facial expressions associated with these basic emotions. It has been the most prevalent model for measuring emotion and has dominated studies related to facial expression recognition and hence forms the basis of the emotions *RTHAA* aims to classify. The alternative model proposed by Russell represents emotional states in a two-dimensional (pleasant-unpleasant,

activation-deactivation) bipolar space which is circular (similar to another popular emotion model defined by Plutchik (2001)), rather than specific discrete categories.

2.2 Existing Methods of Emotion Recognition

Here, we critically compare various different methods of emotion recognition by looking at their accuracy, reliability and scalability, ultimately motivating the use of facial expression recognition and speech sentiment analysis in our work.

Emotion recognition methods can be broadly categorized into two major categories. The first category involves using internal signals, specifically physiological signals, which include the electroencephalogram (EEG), electrocardiogram (ECG), galvanic skin response (GSR) and many more. These signals are less subject to intentional control and can provide insights into the individual's internal emotional state. For example, EEG signals can capture the electrical activity in the brain, providing information about neural processes associated with certain emotions. However, collecting and analyzing physiological signals may require specialized equipment and expertise, diminishing its scalability.

Furthermore, the interpretation of the signals to recognise emotions can be complex as they use a wide variety of machine learning and deep learning models. Traditional machine learning methods require carefully designed and hand-crafted features as well as feature optimisation. Contrarily, deep learning methods learn the inherent principle of the data and extract features automatically, eliminating challenging feature engineering stages of traditional methods. The whole emotion recognition framework is shown in Figure 2. Signal preprocessing, which is included both in traditional methods and deep learning methods, is adopted to eliminate the noise effects caused by the crosstalk, measuring instruments, electromagnetic interferences, etc. We will not cover the machine learning techniques employed in the processing of physiological signals, as our main focus is on the results in order to support the utilization of FER. However, we will extensively discuss the machine learning (and by extension deep learning) techniques adopted in FER.

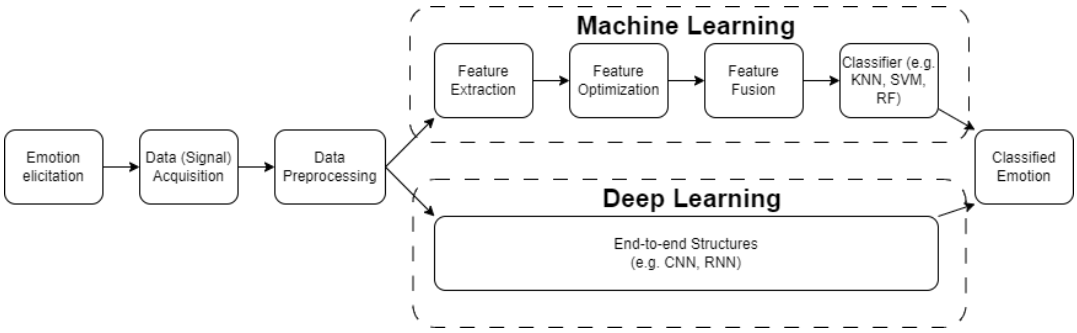


Figure 2: Emotion recognition framework using physiological signals

The second category involves using human physical signals, such as facial expressions (Zhang et al. 2016), speech signals (acoustics) (Mao et al. 2014), gestures (Ajili et al. 2019) and other overt behaviors, which are relatively easy to collect. However, the reliability of these signals may not always be guaranteed, as individuals can intentionally control their physical signals, such as masking their true emotions with a smile in a formal social setting, making it difficult to accurately identify their underlying emotions.

2.2.1 Using Physiological Signals

In this section, we discuss existing methods of emotion recognition that use physiological signals and how effective they are.

2.2.1.1 Electroencephalography (EEG)

Electroencephalography (EEG) refers to an electrophysiological, noninvasive (does not require any surgical procedures or invasive measures) technique for the recording of electrical activity arising from the human brain. Electrodes are simply placed on the scalp, making it a relatively safe and comfortable method compared to other brain imaging techniques. Wu et al. (2017) provide a method for emotion recognition using only two channels of frontal EEG signals at FP1 and FP2 (the left and right frontal pole EEG electrodes) shown in Figure 3. The experiment using a Gradient-boosting Decision Tree (GBDT) classifier validated the effectiveness of the method, where the maximum and mean classification accuracy were 76.34% and 75.18% respectively. Note that this method classifies signals into positive and negative (binary classification).

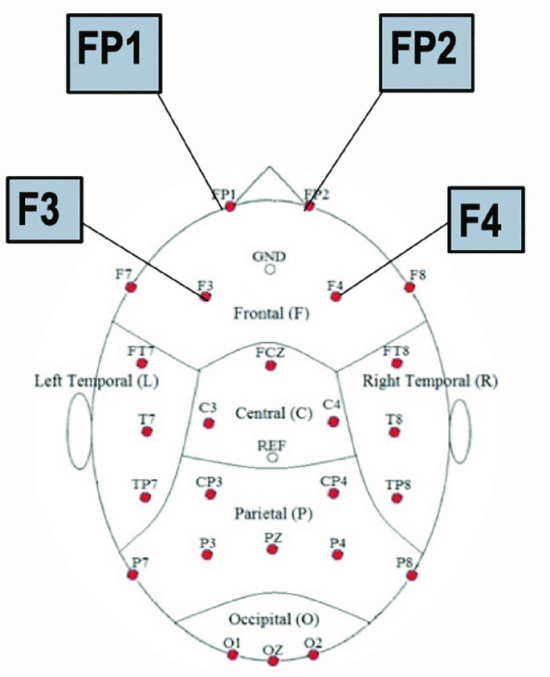


Figure 3: Position of EEG electrodes on a human head, taken from Neal et al. (2019)

Li, Tian, Shu, Xu & Hu (2018) propose a new approach which considers together the temporal, spatial, and frequency characteristics of EEG signals. This approach extracts rational asymmetry (RASM) as the feature to describe the frequency-space domain characteristics of EEG signals and constructs a Long Short-Term Memory (LSTM) network as the classifier to explore the temporal correlations of EEG signals. The results showed that the mean accuracy of emotion recognition achieved 81.10% in valence and 74.38% in arousal (activation). While EEG achieves acceptable accuracy for binary classification tasks, current solutions cannot achieve reliable results for multi-class classification tasks, for example, attempting to classify signals into the six discrete emotions classes stated by Ekman (1992), or to combine the readings from valence and arousal to deduce specific emotions stated in the Circumplex Model of Affect introduced by Russell (1980).

2.2.1.2 Electrocardiography (ECG)

Electrocardiography (ECG) refers to the noninvasive interpretation of the electrical activity of the heart in real time. Since heart activity is related with human central system ECG is useful not only in analyzing the heart's activity it can be also used

successfully for emotion recognition as demonstrated by Goshvarpour & Abbasi (2017). During an ECG, small electrodes are placed on the skin of the chest, limbs, or other specific locations on the body. These electrodes are connected to a machine that records the electrical signals generated by the heart as it beats.

Valenza et al. (2014) propose a probabilistic framework able to characterize the emotional state of a subject using a support vector machine (SVM) through the analysis of heartbeat dynamics exclusively. Results, estimating emotions each 10 seconds, achieve an overall accuracy in recognizing four emotional states (shown in Figure 4) based on the circumplex model of affect of 79.29%, with 79.15% on the valence axis, and 83.55% on the arousal axis. These results demonstrate that using ECGs for emotion recognition produce marginally more reliable results than using EEGs, however, more sources can consistently achieve results obtained by Wu et al. (2017) and Li, Tian, Shu, Xu & Hu (2018) for EEGs, whereas the source cited for ECGs (Valenza et al. 2014), seems to be the only source that achieves higher results, causing there to be less certainty in the reliability of this source.

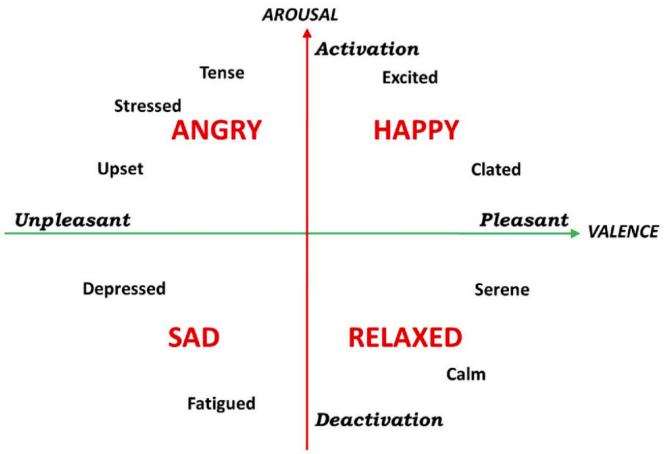


Figure 4: A four-class, discrete version of Russell's Circumplex Model used in and taken from Valenza et al. (2014)

2.2.1.3 Galvanic Skin Response (GSR)

Galvanic skin response (GSR), also known as electrodermal activity (EDA) or skin conductance (SC), refers to a physiological measurement that assesses the electrical conductance of the skin, typically on the surface of the fingers or palms, in response to emotional or physiological stimuli. GSR is commonly used as an indicator of

sympathetic nervous system activity, which is associated with emotional and physiological arousal (Wu et al. 2010, Lidberg & Wallin 1981).

Wen et al. (2014) used GSR, fingertip blood oxygen saturation and heart rate as input signals to recognise five emotions using random forests (RF). This yielded an overall accuracy rate of 74% on a database containing 477 cases. Wu et al. (2010) managed to recognise six emotions with average verifying recognition rates of surprise, fear, disgust, grief, happy and angry being 78.72%, 73.37%, 70.48%, 62.65%, 62.52% and 44.93% respectively. Das et al. (2016) combined ECG and GSR signals to recognise three emotional states: happy, sad and neutral. The classification performance for happy-sad, sad-neutral and happy-neutral emotions was 93.32%, 91.42% and 90.12% respectively.

To summarise, the foundational notion that underpins emotion recognition using physiological signals, is that humans will respond differently according to different stimuli. Signals obtained from EEGs, ECGs, GSRs, etc. can all be used to deduce emotions. Wioleta (2013) highlights the value of these biological responses as they are spontaneous and uncontrollable hence providing the most reliable data since they are involuntary. However, using physiological signals for emotion recognition poses some disadvantages. Firstly, subtle physical conditions, e.g. brain activity, can be influenced by external events, such as stress, so it can be difficult to ensure that any emotion induced in the person of interest is solely the cause of the environment being tested. While measures can be taken to mitigate the impact of external influences, it presents a logistical challenge if testing is desired to be done on a large number of people. Using human physical signals (e.g. facial expressions and speech) are much more scalable compared to using physiological signals as they do not require medical expertise nor specialist equipment such as electrodes.

In conclusion, the above methods manage to obtain reasonable accuracy in very controlled environments, but this accuracy begins to detoriorate considerably as more emotion classes are introduced. Using physiological signals for emotion recognition also presents logistical challenges and hence is not very scalable nor cost-effective. This motivates the central aim of this project — to offer a more scalable

and cost-effective alternative to using physiological signals by building a web application that automatically monitors emotions and sentiment accurately based on facial expressions and speech-content.

2.2.2 Using Human Physical Signals

2.2.2.1 Gestures

Ajili et al. (2019) show that emotion recognition can be done via the analysis of body posture (human movement). In different moods and environmental states, human beings will involuntarily undergo some posture changes as demonstrated by Lee et al. (2017) which can be seen in Table 1. Bull (2016), Coulson (2004) indicate that information such as the time and frequency of these posture changes can reveal important information about the emotions a human might be feeling. However, using human gestures for emotion recognition poses certain limitations. Many gestures have little to no emotional significance, or perhaps, have different emotional meanings in different environments. Also the actual detection of gestures themselves (not the emotions they may exude) can be made extremely challenging due to image noise and hence is a hard method to generalise to all environments according to Glowinski et al. (2008). The combination of the factors listed above make emotion recognition via gestures inadequate if high classification accuracy is desired (unlike using facial expressions) with recently developed systems achieving a recognition rate of approximately 62% (Piana et al. 2014).

Emotion	Gestures and Postures
Happiness	Body extended, shoulders up, arms lifted up or away from the body
Interest	Lateral hand and arm movement and arm stretched out frontal
Surprise	Right/left hand going to the head, two hands covering the cheeks self-touch two hands covering the mouth head shaking body shift-backing
Boredom	Raising the chin (moving the head backward), collapsed body posture, and head bent sideways, covering the face with two hands
Disgust	Shoulders forward, head downward and upper body collapsed, and arms crossed in front of the chest, hands close to the body
Anger	Lifting the shoulder, opening and closing hand, arms stretched out frontal, pointing, and shoulders squared

Table 1: Relations between emotions and body posture — inspired by Lee et al. (2017), Metri et al. (2011)

2.2.2.2 Speech Signals

Speech emotion recognition (SER) refers to the process of automatically recognizing emotions from speech signals. It involves analyzing the acoustic features of speech, such as pitch, intensity and rhythm, to identify emotional cues and infer the emotional state of the speaker. Recent carried out by Ton-That & Cao (2019) obtain good classification accuracies, specifically 74.31% and 97.29% on the German *Emo-DB* and English *SAVEE* databases respectively. However, individual differences combined with noisy environments can lead to significant variations in speech signals which ultimately brings some difficulties to recognition, and hence must be remediated with the use of a large phonetic database. Therefore, the acquisition of speech signals imposes strong requirements on the surrounding environment — as stated by You et al. (2006) — similar to how background lighting can affect the discernment of emotions from facial expressions (another challenge this project aims to overcome).

In conclusion, the above methods of using human physical signals are easier to collect than the previously mentioned physiological signals, however, their accuracy depends significanly on the absence of environmental noise. This motivates the use

of FER and speech sentiment analysis as they are less subject to environmental factors and have greater classification accuracies, which will be discussed in Section 2.3.

2.2.2.3 Sentiment Analysis

Sentiment analysis is a subfield of NLP that focuses on extracting subjective information from text. For our application we look to analyse the sentiment of the user's speech by classifying their spoken words into positive, negative or neutral. We also look to employ a method that doesn't just classify into the 3 aforementioned classes, but also gives the degree to which the user's speech is positive or negative as this provides useful auxillary data. In this section we will briefly summarise some of the approaches to sentiment analysis, however, this will be explored in more detail in Section 6.2.3.1.

One approach to sentiment analysis is by using a lexicon-based approach. This approach uses sentiment lexicons, i.e. dictionaries, which map collections of words and phrases to an associated positive, negative, or neutral sentiment. Each word in the input text is assigned a score based on its corresponding sentiment in the lexicon, and these scores are then aggregated to calculate the overall sentiment of the text.

Another approach is to use supervised machine learning models. This involves training a machine learning model on a large dataset of labelled text, where each text is labelled as positive, negative, or neutral. The model then predicts the sentiment of new, unfamiliar text. Additionally, hybrid approaches exist that combine the above approaches, such as the hybrid approach proposed by Appel et al. (2018).

2.3 Facial Expression Recognition (FER)

Here, we critically compare existing machine learning and deep learning methods and models used in FER by looking at their accuracy. We will also provide a brief explanation of the functioning of each employed model where appropriate, explaining their mechanisms and operations.

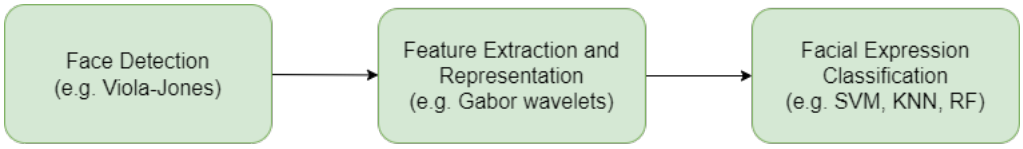


Figure 5: General Design of a Facial Expression Recognition (FER) System

FER systems can be decomposed into 3 key stages, where the output of one stage feeds into another. These stages are shown in Figure 5 and are discussed in more detail below.

2.3.1 Face Detection

The first key part of a FER system is to detect and locate human faces in the image or video. Face detection algorithms can be based on various techniques including Viola-Jones algorithm (Viola & Jones 2001) and Multi-task Cascaded Convolutional Networks (MTCNN) (Xiang & Zhu 2017). The Viola-Jones algorithm works by scanning an image using a sliding window and classifying each sub-window as either containing an object of interest or not. MTCNN is a deep learning-based algorithm that consists of three stages of deep neural networks that work in a cascading manner to detect faces in an image. These algorithms typically output the coordinates of the bounding boxes around the detected faces. Vaillant et al. (1994) designed one of the first neural networks for face localization without making any assumptions about the position, scale, or background of the face. Scale independence was achieved by scanning the input window at multiple scales. Since then, convolutional neural networks (CNNs) that use cascading approaches have become more popular (Li et al. 2015, Xiang & Zhu 2017). Commonly used datasets for model training and testing include Annotated Facial Landmarks in the Wild (AFLW) (Koestinger et al. 2011), WIDER FACE (Yang et al. 2016) and Labeled Faces in the Wild (LFW) (Huang et al. 2008). Farfade et al. (2015) fine-tune an AlexNet — a CNN architecture — and show state-of-the-art results on the AFLW dataset which has faces with varying levels of rotation, illumination and occlusion. To summarise, deep learning approaches lead in performance, specifically, Jiang & Learned-Miller (2017) demonstrate that CNNs are currently the best algorithms used in face detection, compared to traditional algorithms such as Viola-Jones achieving state-of-the-art results on the

WIDER FACE test set as well as other face detection benchmarks such as The Face Detection Data Set and Benchmark (FDDB) (Jain & Learned-Miller 2010).

2.3.2 Feature Extraction and Representation (Selection)

The next stage involves extracting a forming the correct representation of a face. Feature extraction involves extracting meaningful features from the detected faces that can be used as inputs for the expression recognition algorithm. These features could include facial landmarks, facial action units or texture information. One such algorithm for extracting meaningful features is principal component analysis (PCA), a latent factor model which transforms the original features of the data into a new set of orthogonal features called principal components, which are linear combinations of the original features. The principal components capture the most important patterns or structures in the data.

Afterwards, we must select some of the extracted features to choose in our final feature representation, this process is known as feature reduction, sometimes called dimensionality reduction. The extracted features may be high-dimensional and not all features may be relevant for expression recognition. Feature selection techniques may be applied to select the most informative features that contribute to accurate expression recognition. PCA can be used for dimensionality reduction which can reduce the computational burden of once-difficult tasks. This is done by selecting a specified number of top principal components k, which help obtain the most important features. k is a hyper-parameter that should be tuned via techniques such as K-fold cross-validation. PCA is often performed by using singular value decomposition (SVD) which computes d eigenvectors (principal components) where d is the number of features of the dataset. These eigenvectors have the same dimensionality as the original feature vectors themselves and hence can be re-arranged as matrices, commonly known as 'eigenfaces' in the field of facial image analysis, as seen in Figure 7, along with the original dataset shown in Figure 6:

Figure 6: *Labeled Faces in the Wild (LFW)* dataset containing images of famous political figures

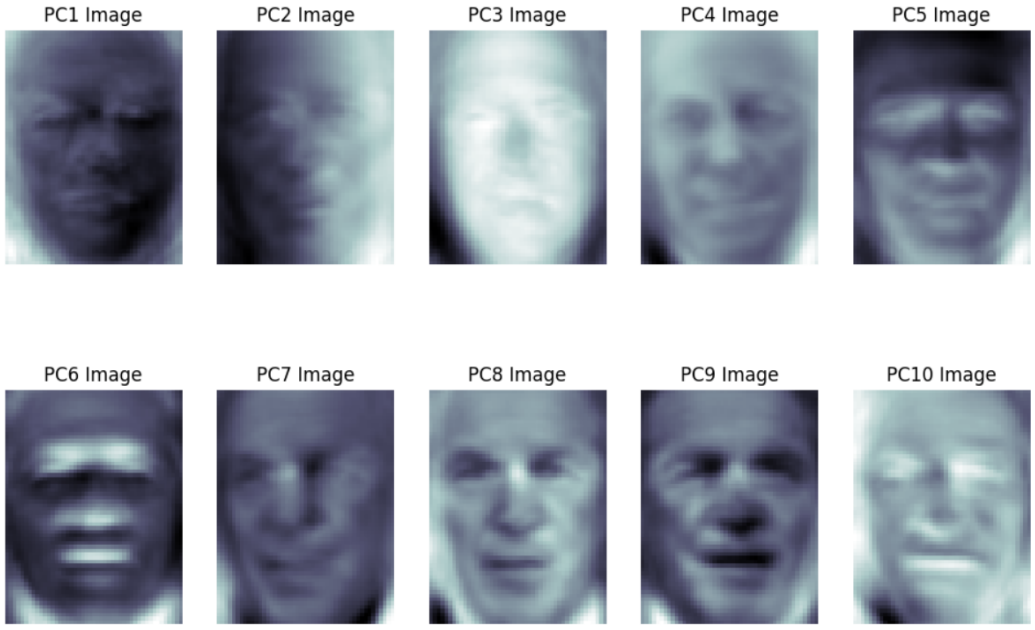


Figure 7: Principal Components (PCs)/Eigenfaces of the *LFW* dataset shown as images

Dimensionality reduction is usually a lossy conversion (information is lost) but PCA is a strong technique to alleviate that information loss. It allows for the removal of redundant information, and variations that can drastically improve the classifiability of data, such as kernel-based PCA, will likely be used prior to the classification

phase of the FER system (however this usually comes at the cost of projecting onto high-dimensional feature spaces, which has performance implications).

2.3.3 Facial Expression Classification

Finally, we select a model to classify the facial expressions that are represented in our feature space. The model is trained on the labelled dataset, which now includes the detected face images with expression labels, to learn the patterns of different expressions. Below, we outline some of the widely-used datasets as well as the most popular machine learning and deep learning methods used in FER and their results on these datasets.

2.3.3.1 Datasets

Listed below are the datasets that are commonly used in the training and evaluation of models in FER and are considered the benchmark datasets:

- **The Extended Cohn-Kanade (CK+) Dataset** (Lucey et al. 2010)
  The CK+ dataset contains grayscale images (a sample of which can be seen in Figure 8) of both male and female people for eight emotions (69% female). The CK+ dataset contains the facial expressions of 210 adults with ages ranging from 18 to 50 years, among them 81% Euro-American, 13% Afro-American, and 6% diverse groups. Unfortunately, the CK+ dataset contains an uneven number of samples for each emotion class (as shown in Figure 9), lending itself vulnerable to overfitting.

Figure 8: Preprocessed sample images of CK+ dataset from Kar et al. (2019)

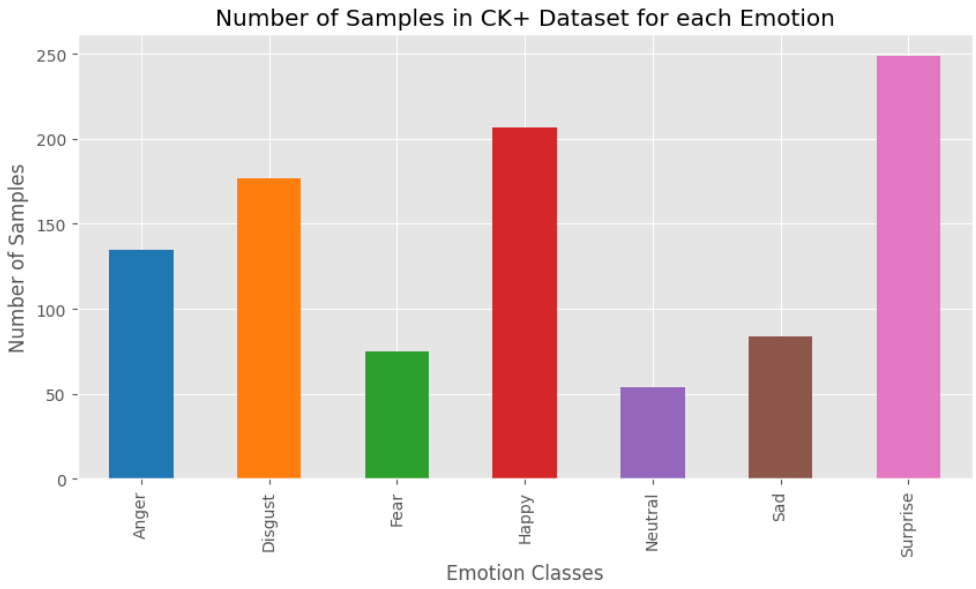


Figure 9: Number of samples for each emotion class in the CK+ dataset

- **Japanese female facial expression (JAFFE) Dataset** (Lyons et al. 1998)
  The JAFFE dataset contains 213 grayscale images (a sample of which can be seen in Figure 10) taken from 10 subjects, each exhibiting seven different emotions including happiness, neutral, sadness, anger, disgust, contempt, fear and surprise. Each emotion label has approximately the same number of images associated with it (29-31 images), mitigating the chance of overfitting. Within the dataset, each subject has a minimum of 3-4 images for each emotion label.

Figure 10: Preprocessed sample images of JAFFE dataset from Kar et al. (2019)

- **AffectNet Dataset** (Mollahosseini et al. 2017)
  The AffectNet dataset includes over 1,000,00 coloured images of faces that were gathered from the internet using 1250 emotion-related keywords in six different

languages, by querying three major search engines. Consequently, it is the largest database of facial expressions, valence, and arousal, allowing the use of two different emotion models (taxonomies): a categorical model similar to the one introduced by Ekman (1992), and continuous (dimensional) model, similar to the one developed by Russell (1980). Unfortunately, AffectNet suffers from an uneven distribution of images for each emotion category as shown in Table 2, which can lead to overfitting.

Category	**Total number of manually annotated images in the training and validation set**
Neutral	75374
Happy	134915
Sad	25959
Surprise	14590
Fear	6878
Disgust	4303
Anger	25382
Contempt	4250
None	33588
Uncertain	12145
Non-Face	82915
Total	420299

Table 2: The total number of manually annotated images in the training and validation set (excluding test set) in each category of emotion in the AffectNet dataset

- **FER-2013 Dataset** (Goodfellow et al. 2013)

  The Facial Expression Recognition 2013 dataset contains approximately 30,000 facial RGB images of different expressions of which there are seven: angry, disgust, fear, happy, sad, surprise and neutral. FER-2013 also suffers from an uneven distribution of images for each emotion label, specifically, there are significantly less images for *disgust* than there are for *happy* as demonstrated by Figure 11, allowing the potential for any model trained on this dataset to underfit on *disgust* and overfit on *happy*.

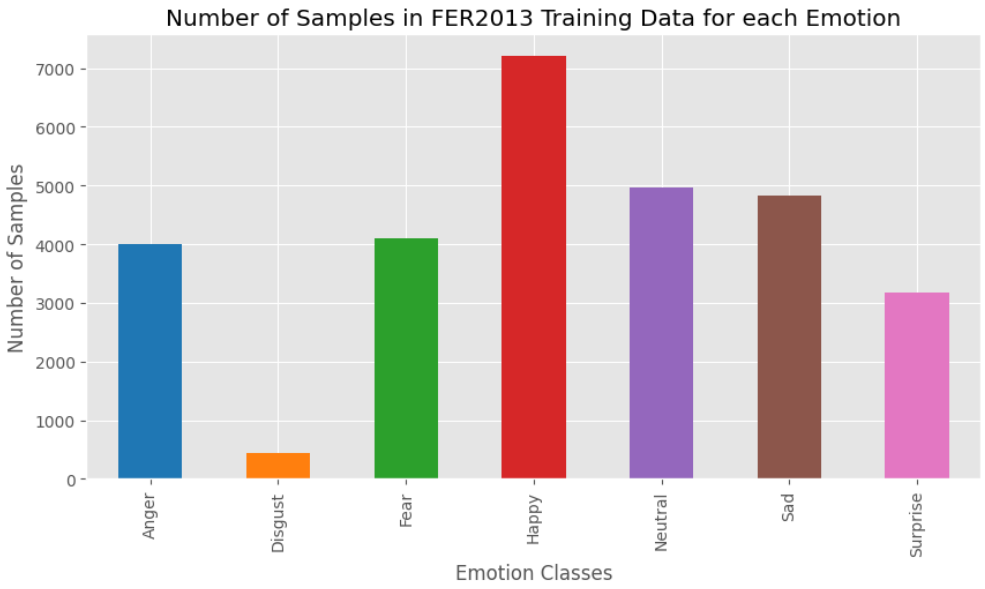


Figure 11: Number of samples in training set for each emotion class in the FER-2013 dataset

More information about each dataset can be found in the ZIP file submitted along with this report. The ZIP contains an exploratory data analysis (EDA) of the datasets mentioned above (where Figures 9 and 11 were obtained from).

2.3.3.2 Support Vector Machines

Support vector machines (SVMs) are one of the earliest, simplest and most popular techniques used in machine learning. They are supervised learning models that are principally used in regression and classification problems. The main idea behind SVMs is to find the best possible decision boundary (or hyperplane) that maximises the margin seperating the data points of different classes — the margin being defined as the distance between the hyperplane and the nearest data point of each class as shown in Figure 12.

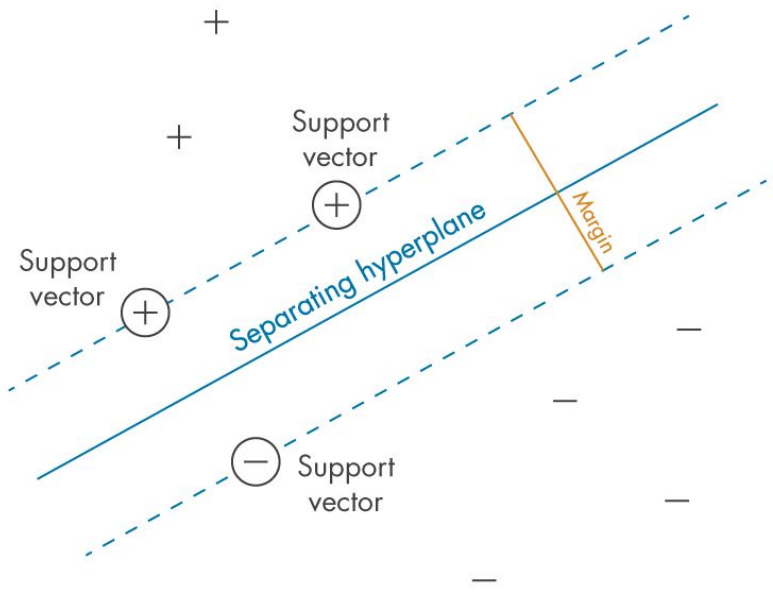


Figure 12: Support vector machine that classifies data points belonging to two classes: positive (+) and negative (-). Image taken from MathWorks (2023)

One of the earlist methods that use SVMs was proposed by Michel & El Kaliouby (2003) in the detection of facial expressions in live videos. Features are extracted using a facial tracker and are then classified using a SVM. Kazmi & Arfan Jaffar (2012) use a wavelets based method in which the classification is done by using seven SVMs in parallel. Each SVM classifies one expression which is later aggregated with others using a maximum function, achieving a classification accuracy of 81.57% on the JAFFE dataset. The approach proposed by Zhang et al. (2015) utilizes multiple kernel learning (MKL) in SVMs to combine features. In this method, the calculation of kernel weights is performed sequentially within the SVM, taking into consideration both sparse and non-sparse kernel combinations, achieving an accuracy of 93.6% on the CK+ dataset. Most recently, Kar et al. (2019) propose a scheme that works in three stages. The first stage employs ripplet transform type II (ripplet-II) to extract features from facial images because of its efficiency in representing edges and textures. The next stage uses PCA and linear discriminant analysis (LDA) to obtain a more compact and discriminative feature set. In the final stage, classification is performed using a least squares variant of SVMs (LS-SVM) with a radial basis function (RBF) kernel. The proposed system is validated on the CK+ and JAFFE datasets, achieving accuracies of 98.97% and 99.46% respectively yielding superior

performance compared to other state-of-the-art schemes.

2.3.3.3 K-Nearest Neighbors

K-nearest neighbors (K-NN) classifiers are another widely-used classifier in emotion recognition and machine learning more generally. It is a supervised, non-parametric (size of the model grows with input data size) model used for classification tasks. K-NN classifiers are based on the Lipschitzness assumption:

Lipschitzness Assumption

If two feature vectors are close to each other, then their classes are likely to be the same.

When attempting to classify new, unlabeled data points, the K-NN algorithm calculates the distance (similarity) between the new data point and the reference set of training data points. The K-NN algorithm then selects the k-nearest neighbors — the k training data points that are closest in terms of similarity to the new data point. The most frequent class in these k-nearest data points is assigned to the unlabeled data point.

Dino & Abdulrazzaq (2019) present a system which classifies images into eight basic facial expressions which are normal, happy, angry, contempt, surprise, sad, fear and disgust. This method uses the Viola-Jones algorithm (Viola & Jones 2001) for face detection and histogram of oriented gradients (HOG) is used as a descriptor for feature extraction from the images of expressive faces. HOG is a feature extraction technique that works by computing local gradient information from an image to create histograms that represent the distribution of gradient orientations in different regions of the image and intern captures local edge information. PCA is applied for the purpose of dimensionality reduction. Finally, a K-NN classifier is used for classification, resulting in an accuracy of 79.97% on the CK+ dataset. Another method that employs a similar approach by using a modified Viola-Jones algorithm and K-NN classifier proposed by Yadav & Singha (2020) achieves a classification accuracy of 98.5% and 95.15% on the JAFFE and CK+ datasets respectively. This method differs in the way that it uses a hybrid approach that combines two features,

namely HOG and local binary pattern (LBP), which are used for feature extraction.

2.3.3.4 Random forest

Random forests (RF) are an ensemble learning method in machine learning that use multiple decision trees (tree-like structures used for making decisions) to make predictions. During training, multiple trees are trained on random subsets of the data and features. During prediction, the predictions from all the trees are combined to obtain the final prediction. Pu et al. (2015) propose a framework that uses a twofold random forest (i.e. two distinct random forests), resulting in a classification accuracy of 96.38% on the CK+ dataset. Munasinghe (2018) proposes a method that uses facial landmarks which uses HOG for feature extraction and a sliding window detection scheme for detecting faces. The resulting accuracy achieved was 90% on the CK+ dataset.

2.3.3.5 Neural Networks

We now explain the notion of a neural network, as this forms essential domain knowledge needed to understand the proceeding models used in FER. Neural networks, commonly referred to as artificial neural networks (ANNs), are a class of computational models used in machine learning that are inspired by the human brain. Neural networks refer to a class of machine learning algorithms that are designed to learn patterns from data in an adaptive and hierarchical manner. Neural networks consist of nodes that are interconnected (also called neurons) and are organised into layers that are typically stacked on top of each other to form a network as shown in Figure 13.

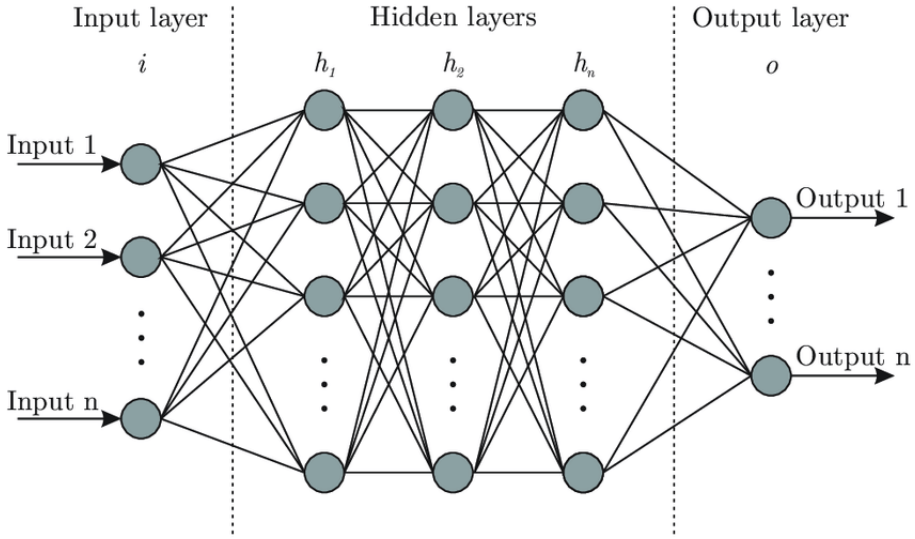


Figure 13: Example of an artificial neural network, taken from Medium (2023)

Neural networks consist of three main types of layers: the input layer, hidden layers, and output layer. The input layer receives the input data, which can be in a variety of forms (images, text, etc.), and is represented as nodes. The hidden layers process and transform the input data to learn relevant features or representations, and typically consist of multiple nodes. The number of hidden layers and nodes in each hidden layer can vary. The output layer (the final layer of the neural network) produces the output based on the learned representations. Another crucial part of neural networks are the connections between the nodes which are represented by weighted edges, which transmit information from one layer to another. While the neural network is being trained, the weights are adjusted according to a learning algorithm e.g. stochastic gradient descent (SGD). This is done iteratively to minimise the error between the predicted output and the actual output, allowing the network to learn and improve its performance.

2.3.3.6 Multi-layer perceptrons

A multi-layer perceptron (MLP) is a type of ANN used in machine learning for supervised learning tasks such as classification and regression. Information flows in one direction from the input layer through the hidden layers to the output layer. Dino & Abdulrazzaq (2019) uses a MLP to achieve a classification accuracy of 82.97% on the CK+ dataset. This method uses the Viola-Jones algorithm (Viola &

Jones 2001) for face detection as well as HOG and PCA for feature representation (before finally being classified by a MLP). Boughrara et al. (2016) propose a method that achieves 84.58% and 96.66% on the FER-2013 and CK+ datasets respectively.

2.3.3.7 Convolutional Neural Networks

Convolutional neural networks (CNNs) refer to a deep learning model that are designed to automatically learn and extract relevant features from input data, making them well-suited for tasks that require understanding and processing visual patterns i.e. they have been explicitly designed to work well with images. This is due to their ability to extract multiple layers of abstract representations, for example, the first convolutional layer might be responsible for edge or corner detection, whereas the last convolutional layer is responsible for detecting more complex patterns, a face, for example (see Figure 14). Convolutional layers perform convolution operations, which involve applying filters to input data (images in our case) in a sliding window fashion, to extract local patterns or features. Convolutional layers are typically followed by activation functions (e.g. ReLU, sigmoid, etc.) to introduce nonlinearity into the model. Poolinglayers are introduced to help reduce computational complexity (by downsampling the feature maps generated by the convolutional layers) and improve the model's ability to generalise across different spatial locations. Lastly, some layers in the network may have dropout — a regularization technique used to prevent overfitting by randomly setting a fraction of the neurons to zero during training, forcing the network to learn more robust representations.

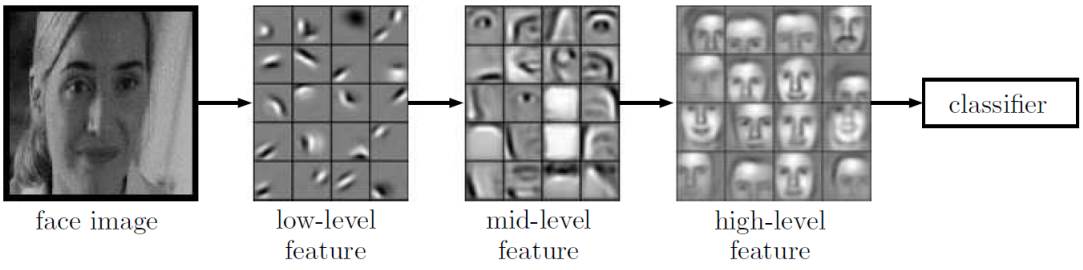


Figure 14: Hierarchical features in CNNs, taken from Nguyen et al. (2019)

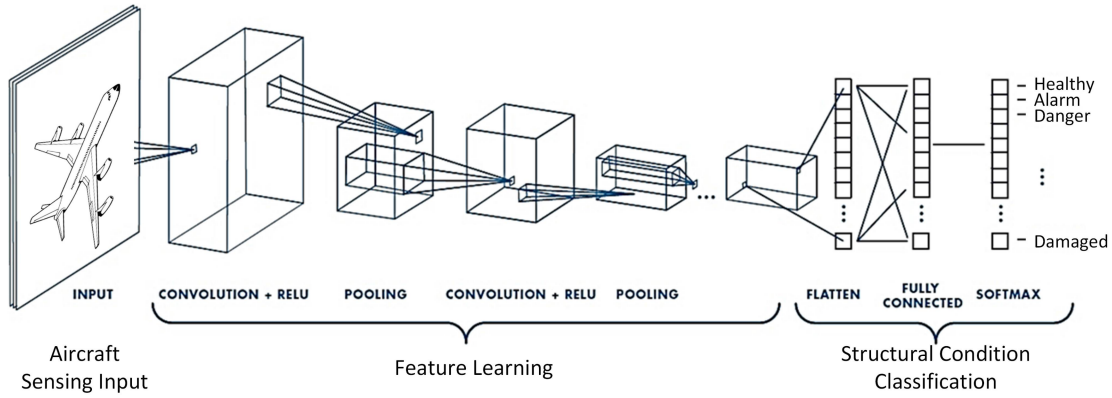


Figure 15: Generic architecture of a CNN, taken from Tabian et al. (2019)

Xie & Hu (2017) propose a feature redundancy-reduced convolutional neural network (FRR-CNN) which is a variant of the traditional CNN architecture that results in less redundant features being generated during feature extraction, yielding a more compact representation of an image. This leads to a more efficient and effective deep learning model. This CNN aims to classify images into the taxonomy introduced by Ekman (1992), and achieves an accuracy of 92.06% on the CK+ dataset. Table 3 shows the confusion matrix of the FRR-CNN on the CK+ dataset.

Predicted Label → Actual Label ↓	**AN**	**SA**	**HA**	**DI**	**FE**	**SU**
AN	86.11	11.11	0	2.78	0	0
SA	23.46	69.14	0	3.70	0	3.70
HA	0	0	95.06	2.47	2.47	0
DI	3.09	0	1.85	95.06	0	0
FE	16.67	0	1.85	0	75.93	5.56
SU	0	0	0	0	0	100

Table 3: Normalised confusion matrix of FRR-CNN on CK+ dataset (%) used in Xie & Hu (2017) where AN, SA, HA, DI, FE and SU correspond to anger, sadness, happiness, disgust, fear and surprise respectively

Li, Zeng, Shan & Chen (2018) propose a CNN architecture that achieves an accuracy of 97.03% on the CK+ dataset. The CNN proposed also attempts to cope with occlusions in images as well, achieving an impressive accuracy of 54.84% on the modified AffectNet dataset (modified to include occlusions in each image), the congusion matrix of which can be seen in Figure 16.

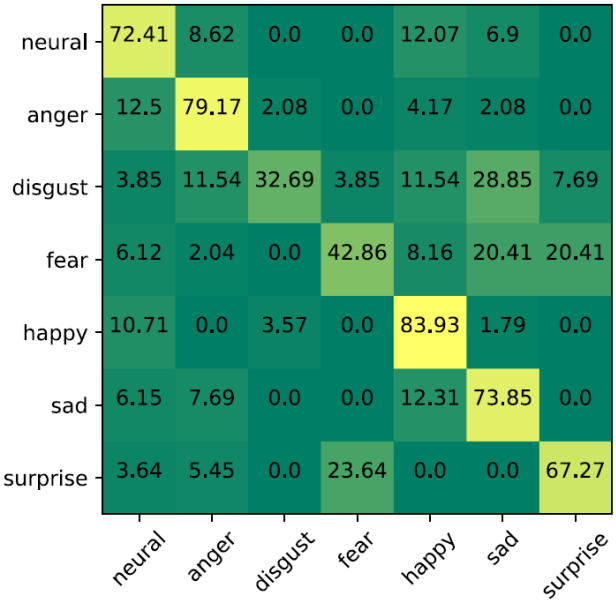


Figure 16: Normalised confusion matrix of CNN proposed by Li, Zeng, Shan & Chen (2018) on AffectNet (merged with Real-world Affective Faces (RAF-DB) dataset)

Table 4 summarises the discussion above and shows that, deep learning approaches generally perform marginally better on CK+ (the most common benchmark dataset used in almost all the methods discussed, and hence most appropriate to compare with) unless computationally expensive feature extraction techniques, such as ripplet-II transform, are used before classification. As a result, we will be leaning more to utilising deep learning approaches in our application as they achieve a good balance of classification accuracy and also computational cost. Since our application we want to develop must classify faces in real-time, ensuring our model is fast is an important consideration.

Method	**Dataset**	**Accuracy (%)**	**Reference**
SVM (wavelets)	JAFFE	81.57	Kazmi & Arfan Jaffar (2012)
SVM (MKL)	CK+	93.60	Zhang et al. (2015)
SVM (ripplet-II)	CK+, JAFFE	98.97, 99.46	Kar et al. (2019)
K-NN (Viola-Jones)	CK+	79.97	Dino & Abdulrazzaq (2019)
RF (twofold)	CK+	96.38	Pu et al. (2015)
RF (HOG)	CK+	90	Munasinghe (2018)
MLP	CK+	82.97	Dino & Abdulrazzaq (2019)
MLP	CK+, FER-2013	96.66, 84.58	Boughrara et al. (2016)
CNN	CK+	92.06	Xie & Hu (2017)
CNN	CK+, AffectNet (with occlusions)	97.03, 54.84	Li, Zeng, Shan & Chen (2018)

Table 4: Summary of the main methods used in FER and their results on widely-used datasets

2.4 Application Development and User Interface Design

This section of the report is going to explore the ways a tool like *RTHAA* could be possibly be developed. We will justify why a web application is more suitable than a desktop application or a mobile application for our use case. We will also briefly discuss some of the common considerations when it comes to user interface design in relation to web application development.

Web applications offer many advantages compared to desktop applications and mobile applications, the most notable of which is its accessibility. Since our application will be used for usability testing, we want to be able to use it on as many devices as possible to allow the user the greatest flexibility in how and what they can test. Web applications can be accessed through web browsers on a wide range of devices such as desktop computers, laptops, tablets and smartphones without the need for installing any software locally. This ultimately makes web applications more accessible as they can be accessed from anywhere with an internet connection

and a compatible web browser. Furthermore, web applications are easier to update and maintain from the user's perspective as they can access the latest version of the application without needing to manually update software on their local devices that also takes up local memory on their device.

However, web applications do pose certain limitations that neither desktop nor mobile applications pose. Web applications tend to have greater performance limitations as they run in web browsers and are subject to browser constraints and are reliant on the user's internet connection. They may also have limited offline functionality, as they require an internet connection to be accessed. Additionally, web applications may have security considerations, as they are accessible over the internet and may be vulnerable to web-based attacks. Despite these limitations, our application remains unhindered as it processes everything locally on the user's device, meaning that once the page has loaded, users will not necessarily need an internet connection to be able to use the application as everything happens in the user's browser and requires no further communication with a remote server.

User interface (UI) design in web development entails creating visual and interactive elements for users to interact with effectively and efficiently. A primary consideration in the design of user interfaces in web applications is accessibility for users with disabilities, such as visual or hearing impairments. We will be looking to follow accessibility guidelines and standards, such as the Web Content Accessibility Guidelines (WCAG) developed by the World Wide Web Consortium (W3C) (2018). This helps ensure that the interface is inclusive and can be used by all users. Widely-used UI design principles such as Nielsen's usability principles (Nielsen 1994), will also be employed where appropriate as they provide a framework for evaluation and help avoid common mistakes that can negatively impact the user experience. UI design will be discussed in greater detail in Section 5 of this report.

2.5 Existing Solutions

Affective computing (automated emotion recognition) is a relatively nascent field, consequently resulting in only a few number of tools being available large number of

tools exist for human attribute analysis, the most popular of which include: Noldus (2023), MorphCast (2023) and iMotions (2023). We will discuss the strengths and weaknesses of each tool and how our work improves on these tools, forming the objectives of this project.

2.5.1 Noldus

Noldus Information Technology is a company specializing in behavioral research and data analysis. Their facial expression recognition software is known as *FaceReader*. *FaceReader* is a unimodal web application tool that works via image upload and hence is incapable of real-time analytics, therefore being more restricted in its use cases. However, the tool is capable of predicting other human attributes such as age and sex. Figure 44 in Appendix A shows a screenshot of Noldus' *FaceReader*.

2.5.2 MorphCast

Morphcast have developed a unimodal client-side (i.e. works entirely in the browser) web application. Making the application client-side is more preferable for privacy and ethical purposes as it removes any chance for any emotion data of an individual to be stored remotely. However, this does mean that users with devices that have less computational resources (e.g. users with older mobile phones) will not be able to comfortably use the tool as the pre-trained models work in the browser, as opposed to being processed in a remote server. MorphCast predicts other human attributes such as age and sex and also processes facial data in real-time, causing it to be considered as the most comprehensive tool out there. Figure 45 in Appendix A shows a screenshot of *Morphcast*.

2.5.3 iMotions

iMotions provide a unimodal desktop application that is capable of only predicting human emotions, not other attributes like age and sex. Since it is a desktop application, it is less accessible and is generally less preferable for the reasons mentioned in the web development section. Figure 46 in Appendix A shows a screenshot of *iMotions*.

2.5.4 RTHAA - Our Work

RTHAA improves upon the aforementioned tools in many ways, the most significant of which, is the fact that *RTHAA* is multimodal. Multimodality is important as it means that our tool is more likely to accurately measure user experience according to Poria et al. (2017) and Garcia-Garcia et al. (2018) as they can capture information from multiple sources to provide a more comprehensive assessment of an individual's emotional state as opposed to just relying on a single channel of sensory input. For example, *RTHAA* has been shown to be able to better detect more obscure emotions, as we can see whether the facial expressions of a user align with the sentiment of what their saying, for example, a user expressing sarcasm may have a positive sentiment in their speech, but a neutral facial expression — both of which can be measured by our tool — and hence we can begin to infer that theres a good chance the user is being sarcastic.

Furthermore, *RTHAA* is capable of deducing the age and gender of the user, unlike *iMotions*. Additionally, *RTHAA* is real-time and hence is more versatile. *RTHAA* also provides more emotion data compared to the existing solutions mentioned above, for example, RTHAA provides a live pie chart showing the total proportion of the test a user expressed a particular emotion. Since *RTHAA* will be used to measure user experience, and ideally, users should be able to set up the tool by themselves, *RTHAA* informs the user of the quality of their lighting conditions, so that the user can adjust their environment in order for the pre-trained models to be able to more effectively operate. We can confidently say that *RTHAA* is a more comprehensive and accurate testing suite for measuring human emotions and attributes. Table 5 summarises the existing solutions and how *RTHAA* compares to them. Figure 58 and 59 in Appendix D show screenshots of *RTHAA* in light and dark mode respectively.

	Noldus	**MorphCast**	**iMotions**	**RTHAA**
Modality	Unimodal	Unimodal	Unimodal	Multimodal
Application type	Web application	Web application	Desktop application	Web application
Processing	Image upload	Real-time	Real-time	Real-time
Attributes Detected	Age, sex	Age, sex	-	Age, sex, speech sentiment
Data provided	Emotion data of static image	Current emotion, emotion data over time	Emotion data over time	Current emotion, emotion data over time, average emotion (proportion of each emotion)
Additional Features	Identifies facial landmarks in static image	-	-	Detects quality of lighting condition and provides more analytical data

Table 5: Comparison of *RTHAA* with *Noldus*, *Morphcast* and *iMotions*

3 Objectives and Requirements

In order to develop RTHAA in an effective way such that it is clear to see how the project improves upon contemporary tools, the project has been broken down into several constituent objectives (along with sub-objectives) listed below. These objectives form the high-level targets that this project aims to achieve.

Stated in this section are also the requirements for the web application itself, split into functional and non-functional requirements. Since the project employs an agile methodology, the requirements can better stomach changes and so will be be flexible to adapt with the project's progression. Each objective and requirement has been assigned a priority according to the *MoSCoW method* (Must-haves, Should-haves, Could-haves and Won't-haves):

- **Must** — Essential system functionality. The minimum capability that should be implemented. These are requirements that are critical to the success of the project and must be delivered for the project to be considered a success. If any of these must-have requirements are not met, the project may be considered a failure.
- **Should** — These are requirements that are important but not critical to the success of the project but are considered important enough to include in the project if sufficient resources and time remain.
- **Could** — Requirements that do not form part of the core functionality of the system, but are still within the scope of the project. They could bring value to users but will be considered lower priority than the must-have and should-have requirements and can be deferred.
- **Won't** — These are requirements that will not be included in the current phase or release of the project.

3.1 Objectives

Below are the project's objectives which form the high-level targets that this project aims to achieve. Objectives should be completed in the order specified.

1. Read literature about existing methods of emotion recognition to establish the feasibility of various aspects of this project and potential changes that need to be made (**MUST**).
   (a) We scope the task by creating an initial reading list that contains information about current similar systems and the challenges they face and ways they can be improved.
   (b) Writing up and summarising the relevant findings.
   (c) Critical comparison of options available.
2. Analyse existing methods for FER that span both machine learning and deep learning, to determine the best method for this project to employ. We will determine the best algorithm by how well it performs in terms of accuracy and speed (**MUST**).
   (a) Read related literature to find the current methods and algorithms used in FER.
   (b) Assess each algorithm individually by analysing their speed and accuracy.
   (c) Critically compare each algorithm to one another in order to establish which is most suitable for our application.
3. The application's FER model should be able to detect at least the 6 emotions stated by Ekman (1992) to a suitable degree of accuracy (this will be more precisely defined in the functional requirements) (**MUST**).
   (a) Review and trial similar systems and record their performance.
   (b) Compare the performance of those existing systems with the applications performance.

4. The application's model for sentiment analysis should be able to classify a piece of text (gathered from speech) into positive, neutral and negative sentiment (**MUST**).
   (a) Review and trial similar systems and record their performance.
   (b) Compare the performance of those existing systems with the applications performance.
5. The application should provide analytical data on how the participants emotions changed throughout the usage of the software being tested (**MUST**).
   (a) The model should receive periodic images of the participants face, and record their emotions at a set frequency, to determine how their emotions vary throughout the test.
6. The application should determine if the individual has set up the test environment correctly by gauging whether they have adequate lighting conditions (**SHOULD**).
7. The application should be able to detect a wider range of emotions, for example neutral (**SHOULD**).
   (a) This will require more reading of current literature to determine feasibility as well as research into finding efficient models that detect neutral but still can accurately classify the other emotions.
8. The application could also detect age and sex, so that once the data of many participants are collected, potential trends could be detected between age groups and sex categories (**COULD**).

3.2 Requirements

Requirements define what a system needs to achieve by specifying capabilities or features that must be present in order for the system to meet its intended purpose. Below are the requirements *RTHAA* intends to fulfil and they are split into

functional (Table 6) and non-functional (Table 7) requirements. Non-functional requirements specify the qualities or characteristics that the system should possess, such as performance, whereas functional requirements define what the system should do and how it should behave in response to specific inputs or actions.

3.2.1 Functional Requirements

Requirement ID	Description	Priority
FR.1	If a backend is used, it will use Flask (a python web framework).	Must
FR.2	If a backend is used, the WebSocket API will be used due to its efficiency.	Must
FR.3	The application must be able to classify user's emotions from their face into at least 6 emotion categories defined by Ekman (1992).	Must
FR.4	The application must be able to classify the user's speech into neutral, positive or negative sentiment	Must
FR.5	The application could be able to detect age and gender	Could
FR.6	Users must be able to see the current emotion they're dislpaying	Must
FR.7	Users must be able to see how their emotions changed over time	Must
FR.8	Users should be able to see the proportion of each emotion they showed throughout the experiment	Should
FR.9	Users should be able to see the transcript of their speech	Should
FR.10	Users must be able to see the sentiment of their speech	Must
FR.11	The application could be able to tell users if their lighting conditions are adequeate or not	Could

Table 6: *RTHAA*'s functional requirements

3.2.2 Non-functional Requirements

Requirement ID	Description	Priority
NFR.1	The application must support any chromium-based browser (such as *Google Chrome* and *Microsoft Edge*).	Must
NFR.2	The application will be written using *JavaScript* and *React.js*.	Must
NFR.3	The application's user interface will be built using *Material UI* and *CSS*.	Must
NFR.4	The application must be multimodal.	Must
NFR.5	The application must be able to access the user's webcam.	Must
NFR.6	The application must be able to access the user's microphone.	Must
NFR.7	The application should be production ready so that it can be built using Node.js' `npm build` command.	Should
NFR.8	The application should be real-time, i.e. processes data in under 2 seconds.	Should
NFR.9	The application should follow WCAG2.1 guidelines.	Should
NFR.10	The application should be user-friendly and easy to use, with an intuitive user interface.	Should
NFR.11	The application should be able to accurately measure emotions from facial expressions, within 15% classification accuracy of contemporary tools.	Should
NFR.12	The system could have a light and dark mode to enhance user experience by offering different contrast options.	Could
NFR.13	Users could be able to print the data collected from the application.	Could
NFR.14	Users should be able to access help via a button that toggles a tutorial.	Could
NFR.15	The application could support non-chromium-based browsers, such as FireFox.	Could

Table 7: *RTHAA*'s non-functional requirements

4 Methodology

The approach this project adopted was split into two major phases, a research phase — to be completed during the first term of the academic year — and a development phase — to be completed during the second term. This was done as the author of the project had very little knowledge of the field of machine learning or emotion recognition, as well as had limited experience in web development. As there was a lot of uncertainty in what exactly this project was going to entail, an agile approach was adopted so that the project could better withstand changing requirements. The research phase of this project served the purpose of informing the project aims and had many objectives:

- Gain a deeper insight into the general field of HCI
- Understand the various approaches used in the field of emotion recognition (e.g. using physiological signals)
- Understand the various machine learning and deep learning models used in emotion recognition via facial expressions (FER)
- Compare and contrast the work done by multiple authors to find the best methods and models to be able to accurately classify facial expressions into emotion categories
- Understand and compare the various methods employed in sentiment analysis
- Find ways current emotion recognition software could be improved by comparing existing solutions available online
- Compare and contrast existing web development technologies to decide the best way to develop the desired application

This research enabled us to approach the development phase knowning exactly how and what to develop. This research also clarified the initial project objectives which changed considerably throughout the project, however we were able to cope with this due to the agile approach we adopted as previously mentioned.

4.1 Software Development Methodology

There are primarily two main schools of thought when it comes to software development methodologies, namely, plan-driven and agile. Plan-driven methodologies such as the waterfall model have a sequential, linear approach to software development and they emphasise the importance of planning process activities in advance. Furthermore, there is a fixed specification before development commences, and there is a large emphasis on producing comprehensive documentation. Consequently, plan-driven methodologies are often used for large-scale, long-term projects with clear, well-defined requirements.

As mentioned previously, the requirements of this project were unstable, motivating the use of an agile approach. Agile approaches focus on iterative and incremental development with flexible and adaptive planning, and hence is clearly the more appropriate methodology to employ for this project. Moreover, since this project has a relatively short time frame, using a plan-driven approach such as waterfall would be very high risk, as a single change in requirements would force the entire project to be revised due to the sequential approach waterfall employs. However, some plan-driven ideas were still utilized, namely reuse-oriented software development. This is so that we can increase productivity, improve quality, and reduce development timeby leveraging existing software assets, such as UI libraries.

Consequently, the project adopted the scrum agile methodology as it focuses on iterative development and delivering working software in short sprints. This suits our needs since the application has many distinct, discrete features features (emotion, age, gender, lighting etc.), lending itself to being developed incrementally and so an iterative methodology would allow for the production of prototypes and minimum viable products (MVPs), allowing us to incorporate user feedback earlier into the development process. Another agile approach under consideration for this project was extreme programming (XP), however scrum was chosen in the end as it is more flexible than XP as it allows for changes to be made throughout the development process. Furthermore, XP has a fixed set of practices that must be followed. However, XP has shorter development cycles, increasing the extent to

which user feedback can be incorporated into the development process.

Another popular agile methodology is test-driven development (TDD), which involves writing tests before the actual application code is written so that the code attempts to pass those tests. However, as the author is the only developer for this project, there is a risk that the developer may not be able to identify all possible scenarios or may have a bias towards their own code. Consequently, for TDD to be effective, code should be reviewed by other developers, which cannot be done as this is an individual project. Therefore, TDD will not be employed in our agile practice, however, extensive testing will take place and will be discussed further in Section 7.

4.2 Project Management Methodology

Although related, project management methodology and software development methodology are not the same. Software development methodologies such as scrum or XP, define a set of processes that guide the software development process — they focus on how software is built. However, project management methodologies specify how the overall project is to be managed, focusing on how it will be planned, executed and controlled. It is important for both of these frameworks to be used in tandem in order to successfully complete a project.

Scrumban — a marriage of two other commonly used frameworks, namely, scrum and kanban — was chosen as the project management methodology as it allows us to employ a combination of ideas from the agile and kanban philosophy. More specifically, we select the most relevant principles and practices depending on the project's demands:

- **Kanban board** (lean) — A backlog in the form of a kanban board was maintained.
  - Serves as a way to visualise the product backlog and help manage and prioritise work. The work items are categorised as either 'Todo', 'In Progress' and 'Done'.
  - This limits the number of tasks, or tickets as they're called in kanban, being worked on and hence reduces task-switching overhead.

  - It serves as a pull-system, meaning we only complete what is required by prioritising tickets, preventing wasteful scope creep.

- **Mininmum viable products** (agile) — Each 2-week sprint would produce a working product by completing an item off of the backlog.
  - This allowed the integration of user feedback early into the development process which proved very useful when it came to validating the UI.
  - Minimised risk as each MVP's core functionality could be reviewed by supervisor.

Figure 47 in Appendix B.1 shows a screenshot of the kanban board towards the end of the project. The screenshots in Appendix D shows the MVPs developed after each sprint during the development phase of the project.

4.2.1 Risk Management

Due to the strict deadlines this project entails, as well as the sizeable amount of documentation needed to be produced, the biggest risk identified was running out of time. As we employed a kanban board, our tasks were prioritised. All tasks with the 'must' priority were aimed to be completed by latest, week 8 of term 2. Figure 17 shows the project's risk management plan in the form of a risk matrix, indicating the risk-averse appetite this project adopted, as the matrix's leading diagonal contains mostly 'High' risk problems. A quantifiable risk analysis would be preferred but they can give a false sense of precision (Raftery 2003) and also consume a considerable amount of project time, contradicting our agile philosophy of avoiding excessive planning.

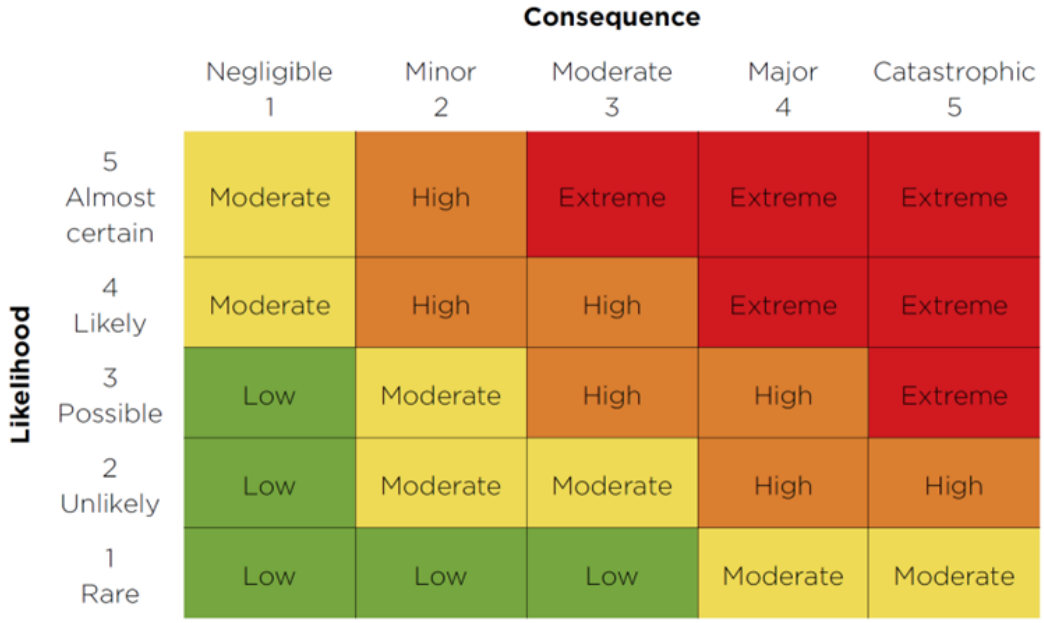


Figure 17: Risk matrix of the project, showing a risk-averse appetite

4.2.1.1 Version Control

Version control is essential in software development to backup and manage the source code of a project and keep track of the history of changes made to the codebase. *Git* was chosen as the version control system to be adopted since it is free, easy-to-use and also provides a cloud-based hosting service for managing repositories known as *GitHub*. Using version control improves efficiency as commits are made after every change, ensuring less time will be used to backtrack the introduction of bugs and errors since the change in each file is clearly recorded and can be reverted back to.

4.2.1.2 Risk Action Plan

Table 8 displays several risks associated with the project, their risk level — measured via the risk matrix in Figure 17 — and their corresponding action plan to remediate the risk.

Risk	**Likelihood**	**Consequence**	**Risk Level**	**Action**
Project data being lost, for example, due to hard drive corruption, misplacement of laptop or accidental deletion.	Unlikely	Catastrophic	High	*GitHub* was used to maintain a remote repository and 2 remote backup locations (external hard drives) were also utilised.
Datasets needed for testing and training models not being available — some need permission and take a long time to gain access to, especially *AffectNet*.	Possible	Major	High	The decision was made to request these datasets early in term 1 during the research phase of the project so that by term 2, all the datasets needed were available.
Requirements of the project change.	Possible	Minor	Moderate	By employing an agile methodology, we significantly mitigate the impact of such an event as we are much more adaptable to changing requirements.
Project is not completed on time, for example, due to illness.	Possible	Major	High	The project's deadline was made to be week 8 of term 2, to increase the sense of urgency to complete the project. If this cannot be achieved a discussion should be had with the project supervisor to discuss contingency plans.
Lack of user involvement — If users are not involved in the design and testing of the software, specifically the UI, the final product may not meet their needs.	Rare	Moderate	Low	By adopting an iterative methodology, users will be involved at the end of each sprint.
Imbalances in datasets used to train models may lead to bias, for example, FER-2013 is heavily biased towards 'happiness'.	Likely	Moderate	High	Techniques such as data augmentation could be employed to help address class imbalance by creating additional synthetic data points for the minority class, thereby balancing the number of samples in each class.

Table 8: Project's risk action plan

4.2.2 Time Management

As mentioned previously, this project was split into two phases, a research phase and a development phase. In the initial stages of the project, we decomposed the project into smaller, more manageable components and created a Gantt chart to schedule the tasks, establish dependencies between tasks and visualize the project timeline which can be seen in Figure 48 in Appendix B.2.

Unfortunately, this initial project timeline was unrealistically optimistic, especially for term 1, as we did not account for various external factors affecting the project, for example, applying for internships or other modules — such as the machine learning module — that needed to be studied before certain aspects of this project could be executed properly. However, despite this optimism, we outlined in the specification that it is highly probable that all research tasks scheduled in term 1 will not be completed and so any uncompleted tasks will be rescheduled into the Christmas holiday period, which was previously slack. This did not hinder the quality of the project as we adopted an agile methodology that can better stomach change and uncertainty. Certain tasks, however, are essential for the development phase of the project; requesting datasets early on is critical because large parts of the application development process are dependent on it. Therefore, this task was given very high priority and was completed on time in term 1.

Figure 49 in Appendix B.2 shows the actual executed timeline of the project which displays some tasks that have been rescheduled into the *'Christmas Holidays'* period, shown by a black range bar. Term 2 was less busy than term 1 as there were no dependencies on term 2 modules. Therefore, many of the tasks scheduled for term 2 were not only completed on time, but some tasks were actually finished before their allotted deadline. As mentioned before, all high priority tasks were scheduled to be completed by the end of week 8, which was achieved, allowing more time to be dedicated to improving the UI via user testing.

5 Design

In this section, we will explore the overarching design considerations that guide the project. Initially, we will define the key components of the project, what exactly we want to achieve and its rationale. This will be followed by an examination of the two primary approaches under consideration for constructing the application: one that employs all pre-trained models in a python backend and another that performs all computations within the browser. Subsequently, we will scrutinize the current technologies adopted in each approach and justify our selection of technologies. Finally, we will discuss and justify the UI design choices in our application and how we aim to evaluate the design.

The application we intend to design has multiple components to it that are worth clarifying. We aim to design a web application that accesses the user's webcam and records their emotions through their facial expresions. Our objective is to have a model that can categorize facial expressions into a minimum of six distinct emotion classes, utilizing the taxonomy popularised by Ekman (1992) as well as a supplementary emotion class — 'neutral' — as it is the state of no discernible emotional expression. Moreover, we aim to retrieve other human attributes from the user's webcam, namely, age and sex. As discussed in Section 2.5.4 and stated by NFR.4 in Section 3.2.2, ensuring our application is multimodal is essential; we will be looking not only to access the user's webcam, but their microphone too. This is so that the user's spoken words can be converted to text, via speech-to-text technology, which can then be analysed for its sentiment using sentiment analysis. We will also be looking to implement a feature that can inform users whether their lighting conditions are adequate or not, as stated by FR.11 in Section 3.2.1. Each of these components can be seen in the activity diagram — a type of unified modeling language (UML) diagram — shown in Figure 18, which shows the tasks that must be done in parallel as the user interacts with the web application.

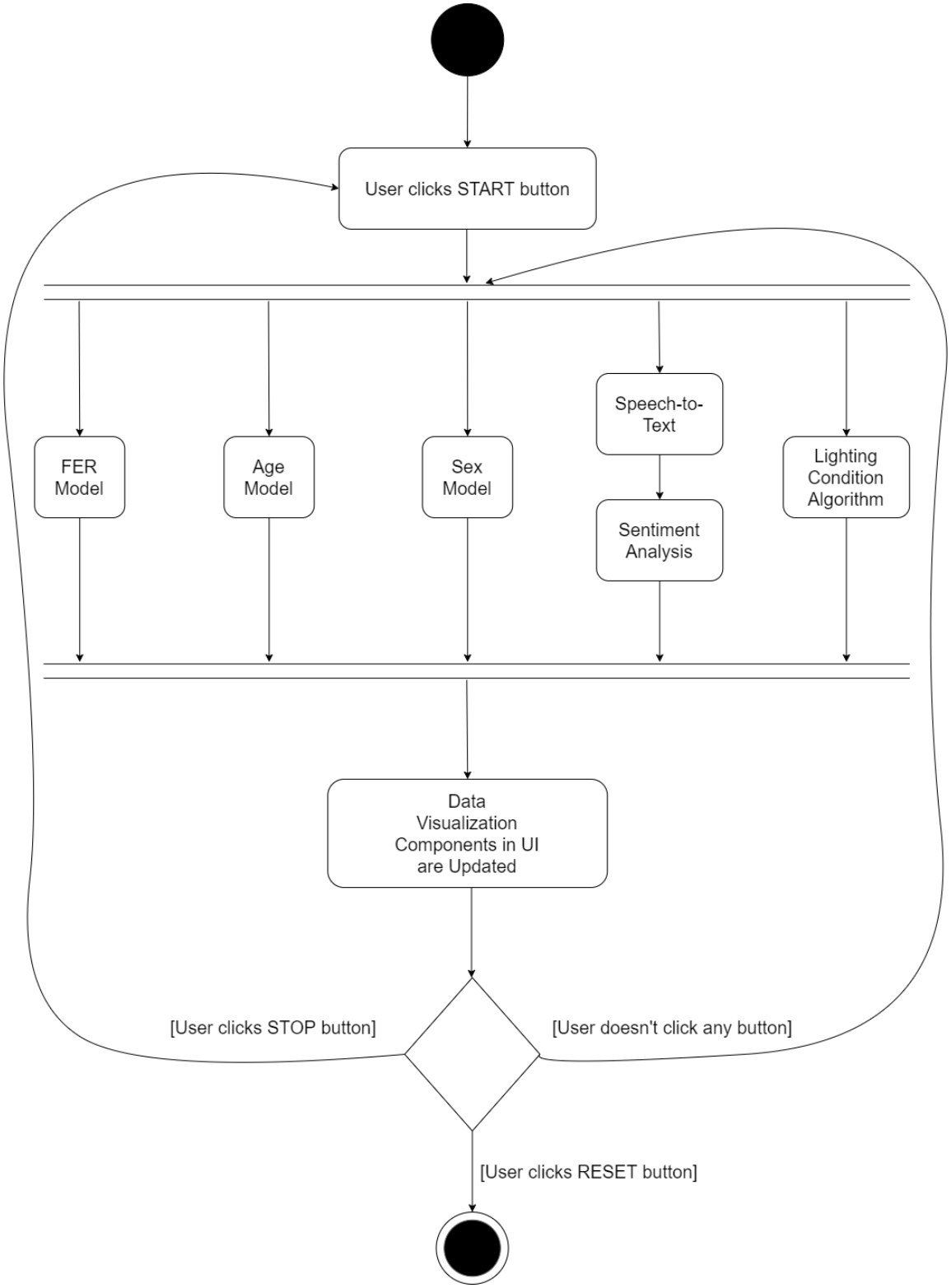


Figure 18: UML activity diagram of *RTHAA*

The data visualization components that we aim to implement, stated by NFR.6, NFR.7 and NFR.8 in Section 3.2.1, are:

- Current emotion status (Figure 19) — Shows the predicted emotion, as well as the predicted percentage chance of other emotions in the form of a bar chart.

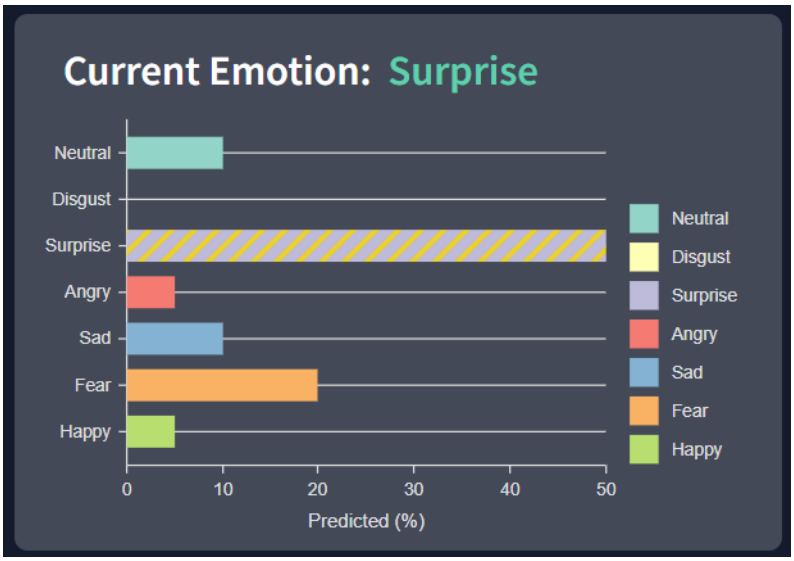


Figure 19: RTHAA's *'CurrentEmotion'* component

This is a useful data visualization component to have as it shows not only the dominant emotion being displayed, but also the probability of other emotions, providing more useful data.

- Emotion-time graph (Figure 20) — Shows how the emotions of a user changed over time.

Figure 20: RTHAA's *'EmotionTime'* component

This is a useful data visualization component as one can see how a user's

emotions have changed over time and can therefore map the change in emotions to certain stimuli in whatever environment the user is testing.

- Average emotion displayed during test (Figure 21) — Shows the proportion of each emotion the user displayed during the entire test in the form of a pie chart.

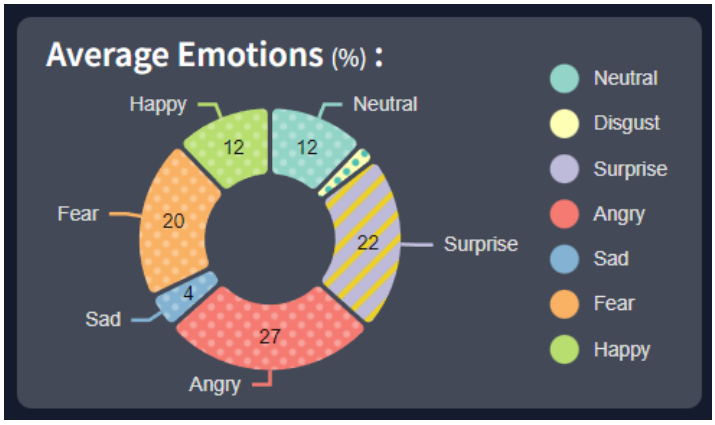


Figure 21: RTHAA's *'AverageEmotion'* component

This is a useful data visualization component as one can quickly determine the user's overall experience of a system, for example, if the pie chart contains 75% happy and 25% neutral, we can deduce that the user had a positive overall experience using the system.

5.1 System Architecture

When developing the web application, two primary system architectures were evaluated. The first architecture involves utilizing the webcam located on the frontend of the application, which captures individual frames at a specified frequency, along with the transcript of the user's speech — which has been retrieved by converting the user's speech to text — and sends them to a backend. In this design, the frames are processed by pre-trained models located in the backend, which classify the emotion, age and sex predicted in the received frame, as well as the sentiment of the user's speech. The output generated by the pre-trained models is then transmitted back to the frontend, where it is presented to the user. Alternatively, the second system architecture under consideration was simpler and involved processing all the

data within the frontend, without the use of a backend server.

Each architecture has its own set of advantages and disadvantages. Using a backend means that the processing load is offloaded to a backend server, which can handle more complex computations more efficiently. This means users with weaker devices, such as older generation phones, can still use the application the same way someone on a powerful laptop would be able to, as the bulk of the processing is done remotely. Furthermore, having the computation on the backend separate from the frontend, means that the backend can be easily updated and improved without affecting the frontend code, however, due to the software architectural pattern we employ, which will be discussed in Section 6.2.1, the data logic will be abstracted away from the UI code anyway, regardless of the system architecture used. However, using a backend poses some disadvantages too. Firstly, the application relies heavily on internet connectivity, as it is constantly sending images across a network which is expensive and slow, causing there to be a delay between the capture of the frames and the output displayed, making the application less real-time.

In addition, there are privacy implications that need to be considered as users may feel uneasy about sending their personal information, such as their age and sex, to a remote server where it could potentially be stored and misused if adequate safeguarding procedures are not implemented. The second architecture proposed where all data processing is done in the frontend, has the advantage of being simpler to develop and maintain, as it does not require a backend server. Moreover, this approach can offer faster performance as there is no delay caused by transmitting images to a backend and waiting for a response. Additionally, the application can operate without a constant internet connection after the initial page is retrieved, as all processing can be performed locally, which also nullifies any potential privacy issues. Figures 50 and 51 in Appendix C show the high-level design of the two system architectures under consideration. Their technologies are justified in Section 5.2.

The decision was made to implement an architecture that utilises a remote server, to provide users with the convenience of remote processing. However, in hindsight, it

was realized that this decision may not have been the best course of action. Despite the processing being done remotely, the user is still required to transmit a stream of images for an extended period of time, which puts significant computational stress on the user's device, more so than what the pre-trained models working locally would have. This will be discussed more in Section 6, where we compare the performance of the implementations of both architectures. Despite implementing the architecture that uses a remote server, we still had time to implement the frontend-processing architecture as we were more familiar with the tools and technologies involved in the process and we had chosen a more development-oriented methodology.

5.2 Tools, Frameworks and Libraries

In this section, we justify the selection of the languages, frameworks and libraries employed in the development process of both architectures. As we are looking to develop an interactive, dynamic web application, as opposed to a static web page, JavaScript must inevitably be used. Dynamic web applications have interactive and real-time user interfaces. JavaScript adds dynamic functionality to web pages, such as live data visualizations, without requiring a full page reload.

5.2.1 Selecting a JavaScript Framework

Building complex dynamic pages using raw JavaScript can be tedious as it has no standard library and has confusing syntax as there are multiple ways of defining semantics (Reales 2022). Consequently, JavaScript frameworks that aim to ease and speed up the development process have grown substantially in popularity (Delcev & Draskovic 2018). As the JavaScript framework determines what the user sees and how they interact with the web application, selecting an appropriate framework is pivotal. As stated by Saks (2019), the current, most popular JavaScript frameworks in web development are React.js, AngularJS and Vue.js.

Of the three aforementioned JavaScript frameworks, React.js was ultimately chosen for many reasons:

- React.js is more suitable for single-page applications compared to AngularJS

and Vue.js as its syntax centres around the idea of reusable components (Daityari 2023) — chunks of UI that constitute the single page application that can be used any number of times — which simplifies development and makes the code easier to manage and maintain.

- React.js has a larger community and ecosystem with a wealth of third-party libraries, more so than AngularJS and Vue.js (Daityari 2023). This reduces development overhead because it allows for us to take advantage of reuse-oriented software development as mentioned in Section 4.1. Consequently, we can use libraries like Material UI and Nivo which significantly cut down on writing unnecessary code by offering reusable UI components.

- Since our application has a lot of dynamic content, such as live graphs and pie charts, the page will be re-rendering very frequently. React.js is known for its excellent performance, particularly in terms of rendering speed. This is mostly due to React.js having the most efficient virtual document object model (DOM) implementation, compared to AngularJS and Vue.js (x team 2021). The DOM is a programming interface for web documents which represents a document as a tree structure, shown in Figure 22, where each node in the tree corresponds to a document object. The virtual DOM refers to a lightweight, in-memory representation of the actual DOM tree that is used to optimise the process of updating the UI by updating only the parts of the actual DOM tree that need to change, hence resulting in faster rendering.

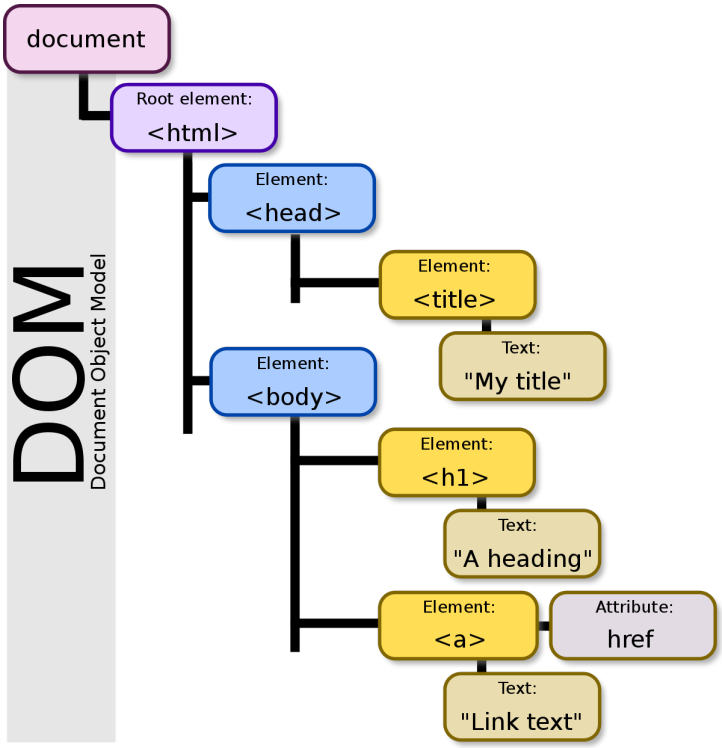


Figure 22: Visual representation of the DOM, taken from Wikipedia contributors (2023)

5.2.2 Other Technologies Used

Apart from React.js, other technologies were also utilized. These include Python and Flask, for use on the backend of the application. TensorFlow, Keras and matplotlip will also be used to help evaluate the performance of pre-trained models, as well as for conducting EDAs on various datasets. The justification of each tool is as follows:

- **Python** — Python has an extensive range of libraries, including NumPy, Pandas, Matplotlib, Scikit-learn and TensorFlow, that can help to simplify complex operations in machine learning and hence will be used when assessing pre-trained models to be used for the web application.

- **Flask** — Flask is a lightweight web framework used to build full-stack web applications that have Python on the backend. Another popular Python web framework is Django, however, this was not chosen due to it being more for database-driven applications (Johns & Singh Khatri 2023).

- **TensorFlow & Keras** — TensorFlow is an open-source library for creating and training machine learning models. Keras is an open-source deep learning

framework built on top of TensorFlow and provides a high-level application programming interface (API) for building deep neural networks. PyTorch was also considered as an alternative machine learning framework to TensorFlow, but was not chosen because of its smaller community and ecosystem (O'Connor 2023).

- **Matplotlib** — Matplotlib is a data visualization library for creating plots, graphs, charts, histograms and other types of visualizations from data in Python. It will mainly be used in the process of evaluating the performance of pre-trained models as well as for conducting EDAs on datasets.
- **Nivo** — Nivo is a React.js library that provides a rich set of data visualization components and will be used for displaying the graphs, pie charts and bar charts in our application.
- **Material UI** — Material UI is a React.js library that offers a comprehensive suite of tools that help develop UIs faster.

5.3 UI Design

Designing a good user interface is essential because it directly affects how users interact and experience the application. A well-designed UI can make a great first impression on the user, starting their user experience with a positive tone. A primary consideration in the design of user interfaces in web applications is accessibility for users with disabilities, such as visual or hearing impairments. We will be looking to follow accessibility guidelines and standards, such as the WCAG, discussed in Section 2.4, to ensure that the interface is inclusive and can be used by all users.

Initial drafts and high-level design considerations of the UI were developed by considering the content of the application, for example, having a symmetrical layout. The window showing the user's webcam was centred on the page, as this creates a more balanced and visually appealing aesthetic. Furthermore, when the webcam window is in the centre of the page, the user can look at their own face while staring straight ahead. If the webcam window was on the side of the page, the user would have to look to the side to see themselves, and in the webcam footage, they would

be looking to the side instead of straight ahead. The webcam footage was also made to be horizontally inverted as it can be quite distracting when a user, for example, moves their head right, but in the webcam window, their head turns left. The combination of these two design ideas leads to the user's movements being more intuitively represented on screen. Figure 23 shows the very first draft created for the design of the UI. Figure 24 shows the centred webcam window, with all the data visualization components surrounding it in the final design of RTHAA.

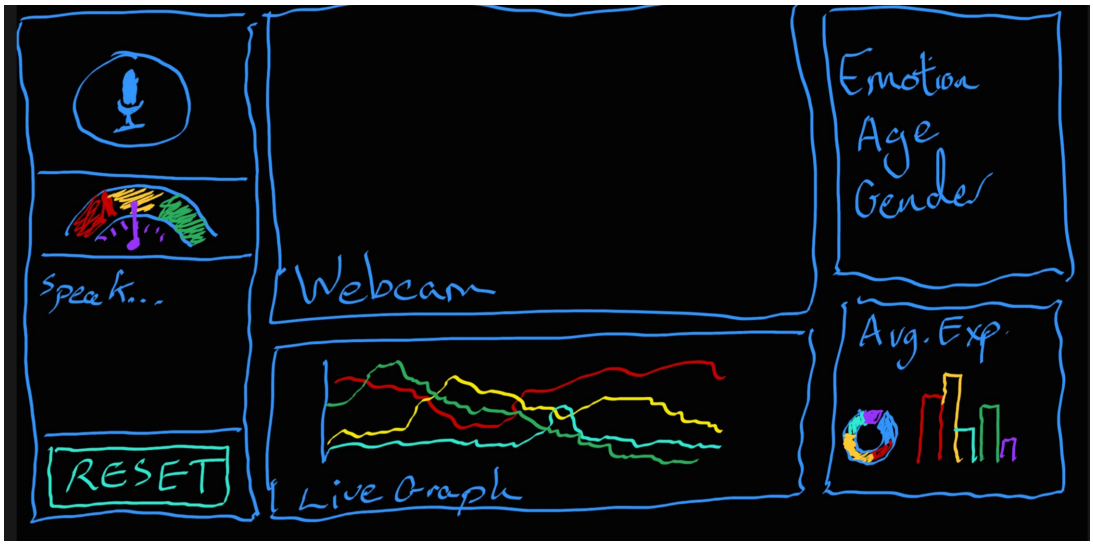


Figure 23: Very first draft of UI design

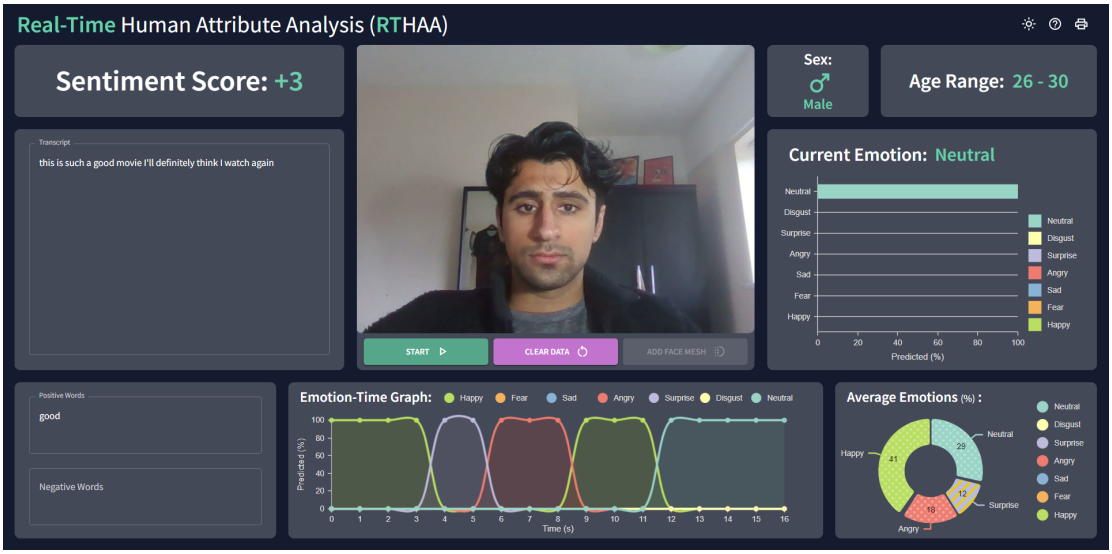


Figure 24: Screenshot of the UI of the finished application

As shown by Figures 23 and 24, the UI design hasn't changed significantly since its initial inception, due to care being taken by ensuring that common UI design

practices were being followed. However, the design has still undergone some change as a result of user feedback at the end of each development sprint. The primary aim of the design was for it to be clean, simple and intuitive. All the data visualization components were made to be put on a single dashboard for the purpose of navigation, so that it becomes very easy for users to obtain the information they require. However, one concern was that the UI may become too clogged. To combat this, a minimal UI design was created, with a non-intrusive colour scheme. The colour scheme employs a dark blue color for the background and uses lighter, grayish hues for the boxes that contain text and data visualization elements. Additionally, white and a deep-sea green colour is used for text. The use of dark colors for the backdrop and white text creates a high contrast, increasing accessibility and contributing to a sleek and minimal design that conveys a sense of professionalism.

Two central design principles were adopted to guide the UI design process, namely, the Gestalt laws of perceptual organisation (Todorovic 2008) and Nielsen's usability principles (Nielsen 1994). The Gestalt principles were employed as they are the widely-accepted general theory of perception in psychology, as stated by Mather (2006). The principles are not just restricted to interfacing with a computer, but how we as human beings process visual information more generally. Nielsen's usability principles were also adopted in the design process as they are regarded as the standard methodology for user interface design in web development (Panchaud 2021). More detail as to how these principles were implemented will be discussed in Section 6.2.6.

6 Implementation

Developing a real-time web application that is capable of detecting emotion, speech sentiment, age, sex and lighting conditions involves piecing together many different components. In this section, we will first discuss the implementation of the architecture which uses a remote server, focusing on the backend, and why it was not chosen as the final system architecture. We will then give a high-level overview of how the frontend-processing architectutre was implemented.

6.1 Remote Server Architecture Implementation

In this section we discuss how the first architecture proposed in Section 5.1, which utilises a remote server, was implemented. It is worth nothing that the frontend for this architecture, primarily consisting of the UI, was not fully implemented, since once the main functionality of the application was deemed too slow, there was no point in developing the UI for it. This decision allowed more time for the development of the more performant, frontend-processing architecture.

As mentioned in Section 5.2.2, Flask was the python web framework used on the backend. The flow of the data would be as follows: the frontend retrieves individual frames from the user's webcam, as well as the transcript of their speech, and sends it to the Flask backend for it to be processed. The backend would then emit the outputs of the models back to the frontend for it to be displayed.

6.1.1 WebSockets

In order to speed up the transmission between the backend and frontend, WebSockets were used instead of traditional HTTP requests. A socket is one endpoint of a two-way communication link between two programs on the network. WebSockets are a type of communication protocol that allow for real-time, bidirectional communication between a client and a server over a single, long-lasting connection, through the interface of a socket. They allow for real-time data exchange without the need for continuous HTTP requests, reducing network overhead and conserving bandwidth. As the communication is full-duplex, i.e. bidirectional, both the client and

server can send data to each other simultaneously. In summary, WebSockets were employed as they provide a much faster and more efficient way of exchanging data compared to HTTP requests, which can help reduce latency and improve overall performance in real-time applications. Figure 25 shows a code snippet of how the communication between the backend and frontend was established in Flask using the `Flask-SocketIO` library which implements the WebSocket protocol.

Figure 25: Code snippet of how Flask's implementation of WebSockets were used to establish the connection between the backend and frontend

In line 6 of Figure 25, you can see the keyword arguement `cors_allowed_origins` being used. Cross-Origin Resource Sharing (CORS) is simply a security mechanism that allows a web page to request and access resources from a different domain than to the one the page itself came from, a feature which is blocked by default on most browsers. In our case, CORS needs to be used because the page itself is being hosted on a server with port 3000, as shown by `REACT_JS_PORT` in line 1 of Figure 25. However, the page attempts to access a different port on a different server, specifically port 8000, as shown by `WEBSOCKET_PORT` in line 2 of Figure 25, which is the server that actually processes the data. By using CORS, our web application can override the browser's default settings and can securely access multiple servers.

An example of how the backend interfaces with the frontend is seen in Figure 26, which shows how the backend server receives the transcript of the users speech, conducts sentiment analysis on it, and emits the results back to the frontend. The server listens on the socket `"text"` and once it receives the transcript from the frontend, which will be stored in the function arguement `text`, it conducts sentiment analysis on `text` using the `TextBlob` library, a simple NLP library. Once the sen-

timent data of the text has been computed, a dictionary containing the sentiment data is emitted back to the frontend to be displayed to the user.

```python
@socketio.on("text")
def handle_text(text):
    sentiment = TextBlob(text).sentiment
    sentiment_polarity = sentiment.polarity
    sentiment_subjectivity = sentiment.subjectivity
    data = {
        'sentiment_polarity': sentiment_polarity,
        'sentiment_subjectivity': sentiment_subjectivity
    }
    emit("text_data", data, broadcast=False)
```

Figure 26: Code snippet showing sentiment analysis of the user's speech on the backend

Ultimately, however, the system architecture which uses this remote backend was not used in the end as it was deemed to slow, violating the non-functional requirement, NFR.8, stated in Section 3.2.2. Table 9 shows the response times of the backend-processing architecture compared to the frontend-processing architecture when processing the frames retrieved from the users webcam at a frequency of 1 frame per second.

Frame of webcam	**Remote Server Average Response Time (seconds)**	**Frontend-processing Average Response Time (seconds)**
Frame 1	3.29	0.23
Frame 2	3.97	0.43
Frame 3	5.61	0.18
Frame 4	9.86	0.21
Frame 5	17.43	0.37

Table 9: Comparison of response times between the two architectures proposed when processing the frames retrieved from the user's webcam at a frequency of 1 frame per second

The results in Table 9 were obtained by conducting a test which defined a function that would wait in intervals of 1 second, sample a frame from the webcam

to send to the backend, and wait until the response was retrieved, recording the time period from when the frame was sent. Evidently, the results indicate that the performance of using a backend is too slow for our real-time purposes, and in fact becomes increasingly worse, simply because it can be easily overwhelmed and hence would not be a scalable architecture. This motivated the architecture where all the processing of data is done entirely in the frontend, i.e. the frontend-processing architecture, removing the latency and computational burden of image transmission. In Section 6, we will focus on discussing the frontend-processing architecture, which was the final architecture used. While the implementation of the backend architecture is available in the submitted ZIP file, it will not be further discussed as it was not the architecture ultimately employed.

6.2 Final Implementation

In this section, we will discuss the final implementation of *RTHAA*, which uses the frontend-processing architecture. We will begin by explaining some basic concepts in *React.js*, as it forms important background knowledge for understanding the basics of the software involved.

- **Components** — In React.js, a *component* is a reusable piece of UI that can be used to build complex UIs. A component is typically a JavaScript function that returns a piece of UI in the form of JSX, which is a syntax extension for JavaScript that allows you to write HTML-like syntax. It is possible to nest components and use *props* to enable communication between a component and its subcomponents.

- **Props** — *Props* are a mechanism to pass data from a parent component to a child component. Props are read-only objects that contain data needed by the child component in order to render.

- **State** — In React.js, *state* is a feature that allows a component to store and manage its own internal data. It represents the current condition of a component, which can be changed. One can think of state as a live variable whose value determines what is dynamically rendered by a component that is

dependent on said variable.

6.2.1 Software Architectural Pattern

In this section, we will explore and justify the software architectural pattern employed in the development of *RTHAA*, as it provides a high-level overview of the application's organisation and functionality by breaking *RTHAA* down into its components.

The software architectural pattern adopted for organising the codebase of *RTHAA*, was the model-view-controller (MVC) pattern. MVC separates the user interface (the view) part of the code from the data processing (the controller) part, meaning the UI can be easily changed without affecting the data logic, or the data itself(the model). This provides a nice abstraction that separates implementation components, helping to reduce code complexity and bugs. Furthermore, the separation of the model, view and controller makes it easier to write tests for each component as the interface to each component is very clear. Consequently, modification to the application is made easier. Since the requirements of the project were unstable at the start, ensuring modification is as easy as possible, would be very beneficial as it allows us to better stomach changes in requirements, compared to other, more rigid, architectural patterns, such as *pipe and filter*, which are more monolithic in structure.

Figure 52 in Appendix C shows how MVC was implemented in *RTHAA*. The user interface, the view, is rendered by the `<Dashboard>` component, inside of which are multiple subcomponents corresponding to the data visualization components in the screenshot of the final UI, shown in Figure 59 in Appendix D. When the user clicks the start button, the application begins loading the models. This is achieved via the use of state variables, the most important of which are shown in Figure 27. When the user clicks start, the `processing` state variable is toggled, causing a function dependent on `processing` to load the models as shown in Figure 52 in Appendix C. This forms the controller part of the application. After the controller loads the models that process the webcam footage, the output of the models, i.e.

the emotions of the user, their sex, etc. are stored in state variables. These state variables are passed down as props through to the corresponding components. As these state variables change, the components immediately re-render, displaying new data.

```
// State variables to store user data
    const [gender, setGender]       = useState('')
    const [age, setAge]             = useState('')
    const [lighting, setLighting]   = useState('')

    const [emotion, setEmotion]     = useState('')

    const [neutral, setNeutral]     = useState(0)
    const [happy, setHappy]         = useState(0)
    const [surprise, setSurprise]   = useState(0)
    const [angry, setAngry]         = useState(0)
    const [sad, setSad]             = useState(0)
    const [fear, setFear]           = useState(0)
    const [disgust, setDisgust]     = useState(0)

    const [sentimentScore, setSentimentScore] = useState(null);
```

Figure 27: Main state variables storing user data

6.2.2 Speech Recognition

Speech recognition is a technology that enables a computer to recognise and transcribe spoken words into text. Here we describe how the user's speech was converted to text, so that a transcript of the user's speech could be obtained, allowing us to conduct sentiment analysis on it.

In order to convert the user's speech into text, React.js's `react-speech-recognition` library was used. It provides state variables that give us access to a transcript of speech picked up from the user's microphone. Under the hood, it uses Web Speech API, an API which enables us to incorporate voice data into our application. This library was chosen as the speech recognition library as it is widely used by many commonly used applications such as *Duolingo*, *Google Translate*, and for voice search

in *Google* (Mozilla 2023).

6.2.3 Sentiment Analysis

In this section, we discuss the implementation of how sentiment analysis was conducted on the transcript of the user's speech. The transcript is stored in a state variable, which is updated every time a new word is recognised from the speech recognition library discussed in Section 6.2.2. Consequently, this allows us to dynamically conduct sentiment analysis on the user's speech, as the transcript is updated with new words. This was achieved via a `useEffect` hook that was listening on the state variable containing the transcript. A `useEffect` hook allows us to run a specificed function when the state of a state variable changes. In this case, the function conducts sentiment analysis on the transcript state variable. Figure 28 shows the arguements of the `useEffect` hook, the first being the function to execute once the variable the hook is listening to changes, and the second being the list of state variables the hook listens to, in this case, just the `transcript` state variable.

```
// Sentiment is calculated everytime the transcript is updated
useEffect(() => {
    setSentimentScore(sentiment.analyze(transcript));
}, [transcript])
```

Figure 28: Code snippet showing how a *useEffect* hook is used to conduct sentiment analysis on the transcript everytime it is updated by verbal input

6.2.3.1 Selecting an Appropriate Model

Given that the transcript is updated every time a single word is spoken, conducting sentiment analysis every time the transcript is updated imposes a large computational burden on the user's device, especially considering that the application has other pre-trained models running in the background. Therefore, it is imperative that the model chosen is very efficient since it will be used very frequently. This was the

primary reason why a lexicon-based approach, discussed in Section 2.2.2.3, was chosen as opposed to a machine learning approach. However, lexicon-based approaches generally perform worse compared to machine learning approaches (Psomakelis et al. 2015), an example of which is shown in Section 7.1. This is becuase they operate using a pre-defined dictionary, and calculate the overall sentiment of a text by aggregating the scores of its constituent words. This means their performance can suffer when dealing with ambiguous texts as they may not capture the nuances of the language or the context, i.e. they cannot account for the semantic relationships between words, whereas a machine learning approach potentially could.

The chosen model was an AFINN-based (affective norms for english words) sentiment analyser, which uses the AFINN-165 lexicon (Nielsen 2011) as the dictionary. AFINN-165 is a dictionary where a list of English words are assigned sentiment scores ranging from -5 to +5, where negative scores indicate negative sentiment, positive scores indicate positive sentiment and a score of 0 indicates neutral sentiment. The lexicon contains over 3,000 words and phrases that have been manually assigned sentiment scores based on their emotional impact. Each word in a text is assigned its corresponding AFINN score and the scores are then aggregated to calculate the overall sentiment of the text. Al-Shabi (2020) demonstrates that an AFINN-based sentiment model achieves a reasonably good classification accuracy of 65% on the *Stanford Twitter Sentiment* dataset, a standard benchmark dataset used in sentiment analysis.

An example of the weakness of using a lexicon-based approach can be seen in Figures 29 and 30. Figure 29 shows the spoken words of the user: “this sentence is very good”. However, Figure 30 shows how the lexicon perceives ‘sentence’ as a bad word, when in this context, ‘sentence’ refers to an actual written sentence as opposed to a criminal punishment. Consequently, the resulting sentiment score is only just positive, despite the sentence itself clearly being very positive.

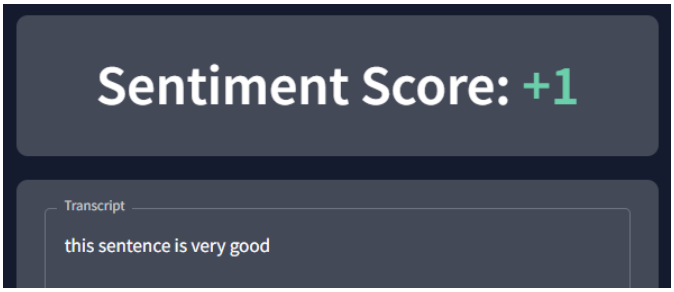


Figure 29: Sentiment score computed using the AFINN-165 lexicon

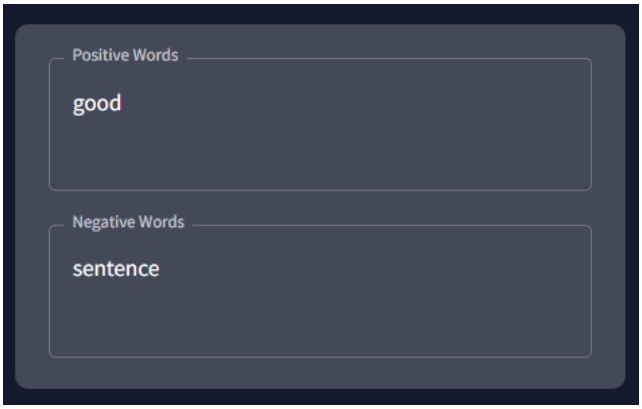


Figure 30: Perceived positive and negative words using the AFINN-165 lexicon

Despite the weaknesses of the lexicon-based approach, its very high speed and efficiency makes it the most appropriate model to use in our real-time application, satisfying NFR.8 stated in Section 3.2.2.

6.2.4 Facial Attribute Classification

As established in Section 2.3.3, deep learning models perform the best for image recognition tasks, such as FER, age recognition and sex recognition. Consequently, the pre-trained models we employed for our attribute classification tasks, were deep learning models implemented by a widely-used API for facial image recognition tasks called *face-api.js* developed by Mühler (2023). Face-api.js is a JavaScript API built on top of *tensorflow.js*, which comes with several models to choose from for your image recognition task. We explain and evaluate each model we have used below for face detection, FER, age and sex recognition.

6.2.4.1 Face Detection

For face detection, three models are offered by face-api.js:

- **SSD Mobilenet V1** — This is a neural network which implements a single-shot multibox detector (SSD) algorithm. It works by dividing the input image into multiple grids and predicting the presence of objects using a set of default bounding boxes and their offsets. As the SSD algorithm is a single-shot detection approach, it only requires a single forward pass through the neural network to make predictions, making it relatively fast and accurate. It will compute the locations of each face in an image and will return the bounding boxes together with it's probability for each face. This approach generally aims toward obtaining high accuracy in detecting face bounding boxes instead of low inference time; with the size of the quantised model being approximately 5.4 megabytes. The implementation of this model has been trained and tested on the WIDER FACE dataset previously discussed in Section 2.3.1, achieving a strong, but not quite state-of-the-art, classification accuracy of 85.20%.

- **Tiny Face Detector** — Tiny Face Detector is a high-performance CNN that operates in real-time that has been trained on a custom dataset of approximately 14,000 images labelled with bounding boxes. It is more efficient in terms of speed, size and resource consumption compared to the *SSD Mobilenet V1* face detector. Furthermore, it generally produces marginally better classification results than *SSD Mobilenet V1* on larger faces, achieving 86.93% on the WIDER FACE dataset, however, it is less effective when detecting smaller faces. This is compensated however, by the model being very fast and resource efficient, with the size of the quantised model being approximately 190 kilobytes.

- **MTCNN** — MTCNN is an implementation of the model discussed in Section 2.3.1 proposed by Xiang & Zhu (2017). Despite it achieving state-of-the-art classification results on datasets such as WIDER FACE and AFLW, it is very slow as it is involves three cascading CNNs. The model size is 2 megabytes.

Out of the three face detection models discussed above, *Tiny Face Detector* was chosen due to its speed and small size, hence being the most suitable for real-time applications, while still maintaining a good classification accuracy.

6.2.4.2 Attribute Classification

For attribute classification, face-api.js offers two deep learning models, namely, *faceExpressionNet* and *ageGenderNet*. *faceExpressionNet* is a CNN that classifies the face in an image into the emotion categories defined by Ekman (1992) and a neutral emotion category. It has been trained on multiple datasets including FER-2013 and AffectNet, achieving classification accuracies of 86.42% and 82.56% respectively. These accuracies are sufficient for our purposes, as we are not looking to necessarily use the most accurate model, but rather one that achieves similar results to contemporary models as stated by NFR.11 in Section 3.2.2. Furthermore, faceExpressionNet is only 310 kilobytes, meaning it is very resource efficient and suitable for our real-time application.

The model offered by face-api.js for detecting age and sex is *ageGenderNet*. This model uses a CNN architecture called VGG-19, shown in Figure 31, which consists of 19 convolutional layers. The model has been trained on the *'Asian Face Age Dataset'*, AFAD. The model, compared to contemporary models, achieves a strong accuracy of 97.01% for predicting sex and a mean absolute error of 3.67 for predicting age on the AFAD dataset.

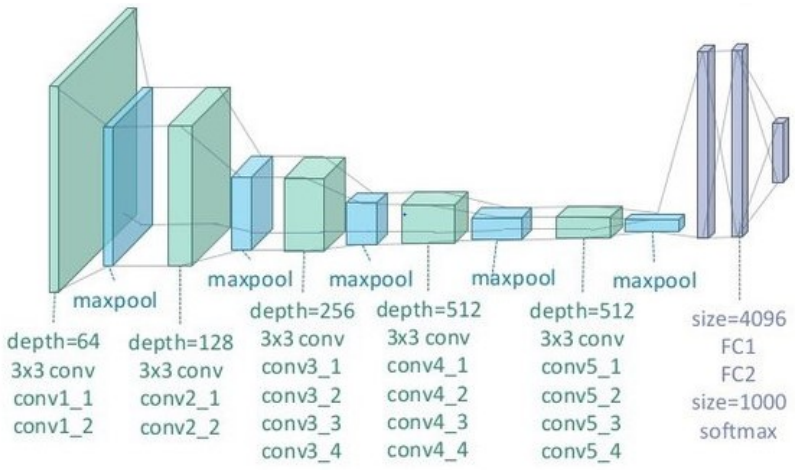


Figure 31: VGG-19 neural network, a standard CNN architecture consisting of 19 convolutional layers, taken from Rusin (2023)

6.2.4.3 Using Attribute Models

As shown in Figure 52 in Appendix C, when the user clicks the start button, i.e. a user event occurs, a state variable is changed, which is listened to using a `useEffect` hook. The function ran by the `useEffect` hook is the `loadModels()` function shown in Figure 32.

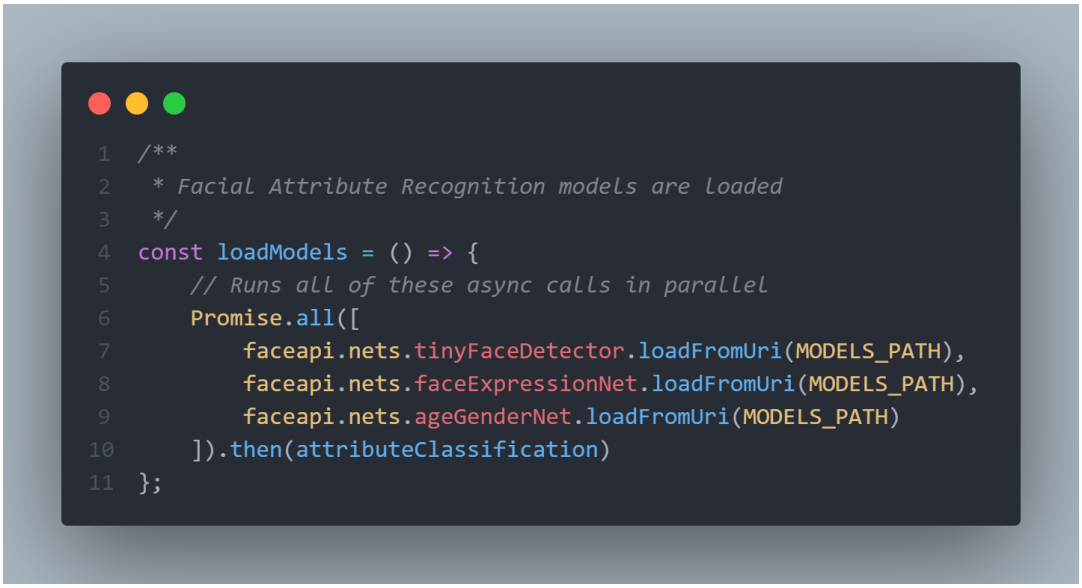


Figure 32: Image showing how each pre-trained model is loaded

Lines 7-9 in Figure 32 show how the face detection model, `tinyFaceDetector`, the FER model, `faceExpressionNet` and the age and sex model, `ageGenderNet` are loaded. As loading pre-trained models is not an instantaneous operation, and hence is an asynchronous task, promises are used so that the execution of the program is not blocked. In JavaScript, a *promise* is an object that represents the eventual completion, or failure, of an asynchronous operation. Line 6 in Figure 32 shows how the `Promise.all()` function is used so that multiple asynchronous calls can be made in parallel, saving the user time and offering a more fluid experience. JavaScript's `then()` method is then used to handle the fulfilment of a promise object. In line 10 of Figure 32, the function address of `attributeClassification` is given as an arguement to `then()` so that once the models have finished loading, attribute classification can begin.

```
// Classification
const detections = await faceapi.detectAllFaces
    (videoRef.current, new faceapi.TinyFaceDetectorOptions())
        .withFaceExpressions()
        .withAgeAndGender();
```

Figure 33: Code snippet of a part of the *attributeClassification* function showing how the models are used to classify attributes

Figure 33 shows how once the `tinyFaceDetector` model has detected a face, we then run the FER, age and sex model. The JavaScript keyword `await` is used as the attribute models require a face to be found before classifying, which is an asynchronous operation, and hence we must wait for the promise of finding a face to be fulfilled, i.e. we must wait for the `detectAllFaces()` function to finish executing before resuming with the program. `videoRef.current` is given as input to the `detectAllFaces()` function as it is a reference to the user's webcam video stream element in the DOM.

6.2.5 Lighting Conditions

In this section, we will discuss the implementation of how our web application manages to gauge lighting conditions which was a low priority feature, as stated by FR.11 in Section 3.2.1, and hence little research was done into existing approaches, as it was proposed as an extension to our application and doesn't form an important part of it. As a lot of computational burden is already put on the user's device due to the pre-traned models being used by the web application, we attempted to find a computationally inexpensive method of gauging lighting conditions. The two most common approaches involve calculating the *luminance* of each pixel or alternatively, the *perceived brightness*. In order to calculate perceived brightness, luminance must first be calculated, and is hence a more computationally expensive method and was therefore not chosen. Luminance is a linear measure of light, whereas perceived brightness is not.

In order to gauge the brightness of the lighting conditions of the user's webcam, we first must be able to retrieve the RGB, red-green-blue, colour values of each pixel in the webcam's current frame. This was done by drawing a canvas HTML element on top of the webcam window in the page, so rather than retrieving the frame directly from the webcam, we obtain it via the graphical, webcam window rendered on the page itself. After we have retrieved the frame's RGB values for each pixel, we then compute the luminance of the image by summing the luminance of each pixel in the canvas, and dividing it by the number of pixels in the image to form an average. The luminance of a pixel can be calculated using Equation 1, where L is the calculated luminance of the pixel and R, G, and B are the red, green and blue colour channels of the pixel, respectively.

$$L = 0.2126R + 0.7152G + 0.0722B \tag{1}$$

If the average luminance of the image is greater than a pre-defined threshold, which we define as `BRIGHTNESS_THRESH` in the code, then we can determine whether the user has sufficiently bright lighting conditions. The code snippet for how luminance is calculated is shown in Figure 60 in Appendix E.

6.2.6 UI Implementation

As discussed and justified in Section 5.3, the two main design principles we adopted were the Gestalt laws of perceptual organisation (Todorovic 2008) and Nielsen's usability principles (Nielsen 1994). In this section, we discuss how these principles were implemented. Some examples of how the principles were implemented are as follows:

- **Gestalt similarity principle** — This principle states that we as human beings assume that if two things look similar, then their functionality will be similar; form informs function. One of the ways this principle was implemented for our application, was in the context of colour. Red is usually used in design for terminal features, such as 'STOP', 'END' or 'CANCEL', whereas green is usually employed to initiate something, for example, 'GO', 'START' and 'BEGIN' (Colley et al. 2021). The 'START' button was made to be green, and

the 'STOP' button was made to be red, so the user, from previous experience of using other applications, is assisted in the process of determining the functions of the button by being able to map the functionality of green buttons they have seen in their past experience.

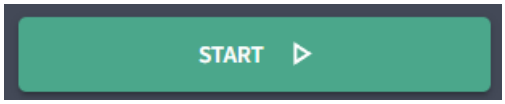


Figure 34: RTHAA's 'START' button

Figure 35: RTHAA's 'STOP' button

- **Gestalt proximity principle** — This principle states that if we see objects close together, we assume that they must be related. Figure 36 shows how UI components associated with speech, are grouped on the left side of the UI, while the components associated with facial data, i.e. emotion, sex, etc. are grouped on the right side of the UI. Furthermore, 37 shows how related buttons are grouped together. Buttons to do with high-level actions, such as logging the collected data or toggling the colour theme are placed together in the header, while buttons to do with the actual processing of data are grouped together at the centre of the UI, below the webcam window.

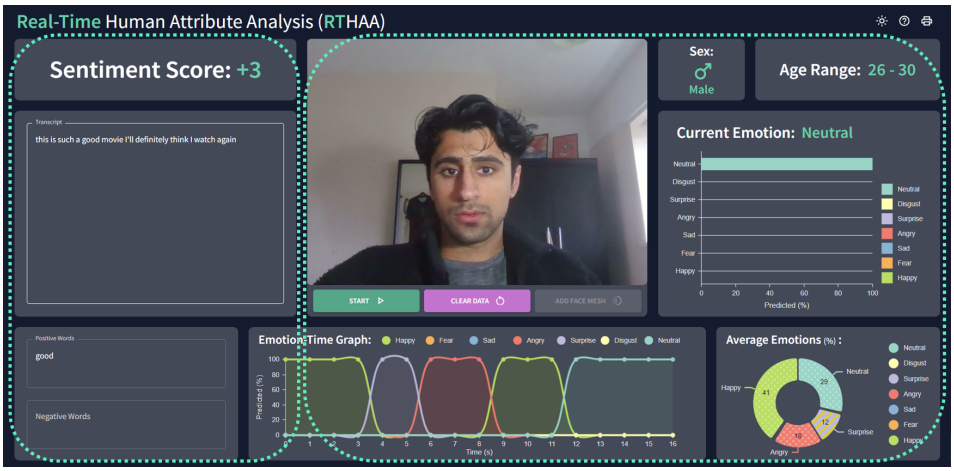


Figure 36: Screenshot of RTHAA outlining how related components are grouped together, specifically, components related to speech and components related to facial data, such as emotion and age

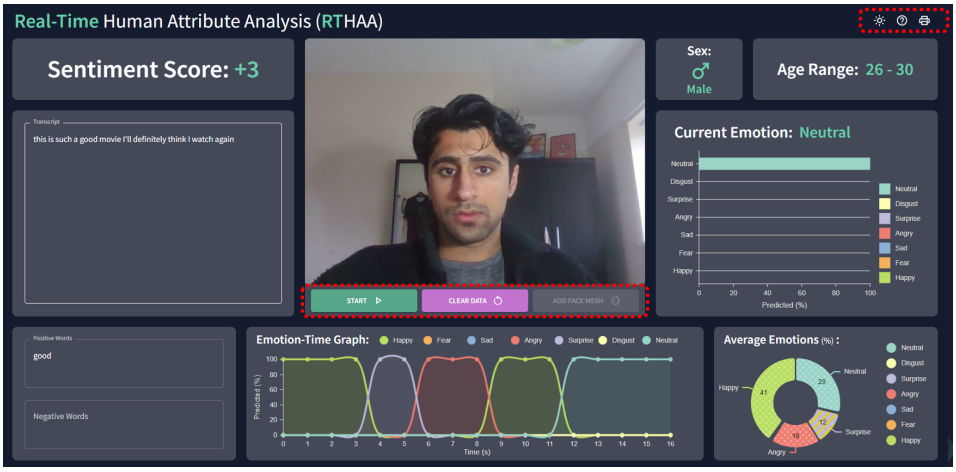


Figure 37: Screenshot of RTHAA outlining how related buttons are grouped together

- **Nielsen's 'visibility of system status' principle** — This principle emphasies the importance of keeping users informed of the status of the system. One way we have implemented this principle, is through the use of the 'START' button. As opposed to having two separate 'START' and 'STOP' buttons, we employ the use of one button, which changes upon clicking, i.e. when the 'START' button is clicked, the button itself changes to become the 'STOP' button, informing the user that the application has started processing data.

- **Nielsen's 'match between the system and real-world' principle** — Similar to the Gestalt similatiry principle, this principle states how we as humans tend to map things from the system to the real-world. In this case, we use specific colours related to certain emotions, for example, in the real-world, red is often associated with anger, and consequently red is used for displaying 'anger' on all data visualization components in the UI. Conversely, green is used for 'happy'.

- **Nielsen's icon design principles** — Icons were used to assist the user in determining the functionality of certain buttons. Figure 38 shows how symbolism, i.e. a high-level abstraction, was used as the icon for toggling the light and dark theme of the page, represented by a sun to indicate brightness. Figure 39 uses an analogous image, in this case, the button prints the collected user data and hence we use a printer icon.

Figure 38: Toggle light/dark theme button

Figure 39: Print collected user data button

7 Testing

As mentioned in Section 4.1, TDD was not employed since this is an individual project, and our tests are subject to the same bias as our application code. However, other forms of testing were conducted and will be discussed in this section.

7.1 Unit Testing

Unit testing helps to identify defects early in the development cycle by isolating problems before they become conflated with other pieces of code. Unit testing was conducted by thoroughly testing individual React.js components, such as each data visualization component. As previously mentioned in Section 6.2, components inherit props from parent components. Therefore, when it came to unit testing an individual component, the behaviour of the component was validated by testing it on all possible states of the props that it inherits. After the unit tests written for components were passed, the component would then be integrated into the application, allowing for integration testing to be conducted on higher-level components to see how multiple components interact with each other. This was done via the same method as unit testing; all possible values for a prop would be inputted into parent components, and these props would trickle down to their child components enabling us to assess the behaviour of the interaction between components.

Throughout this project, unit testing was done on many of the pre-trained models and methods before employing them into the application. One example of such testing, was comparing the accuracy of different sentiment analysis models in Python. We compared two models, one which uses a lexicon, called *VADER*, and one which uses machine learning, called *RoBERTa*. Figure 40 shows the performance of the two models on the *'Amazon Fine Food Reviews'* dataset, a common benchmark dataset used in sentiment analysis. The output of each model consists of three normalised values that indicate the degree of positivity, negativity, or neutrality of the input text. The test, shown in Figure 40, concluded that the *RoBERTa* pre-trained model performs better as it classifies each data point more accurately, compared to the *VADER* model.

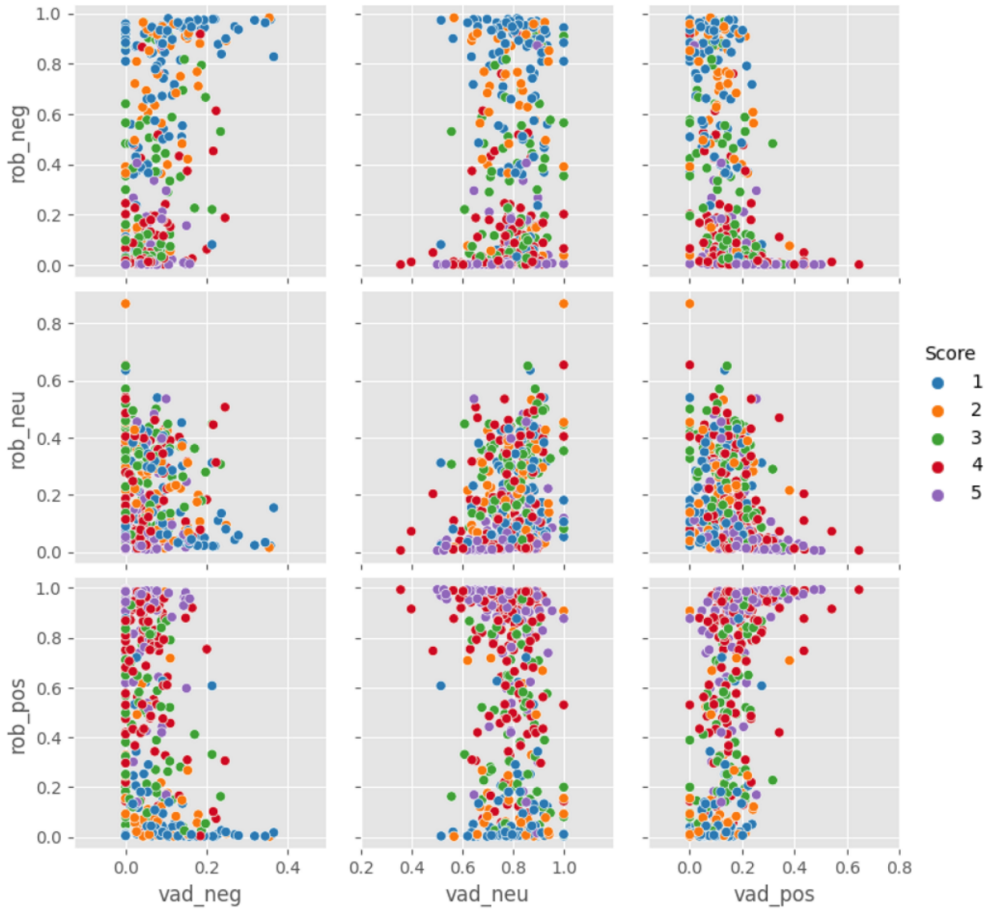


Figure 40: Performance of VADER model (x-axis) compared to RoBERTa model (y-axis) on the *'Amazon Fine Food Reviews'* dataset

7.2 User Testing

User testing is important as it provides valuable feedback on how users interact with our application. Observing these interactions enables us to gain insights into user behavior, and make informed decisions on how to improve the user experience. Alpha testing methods were adopted in order to help detect defects in the application early on in the development process. One such example, was when the supervisor for this project identified that the webcam window should be horizontally inverted for a better user experience, so that user movements are more intuitively shown. Furthermore, the supervisor also stated that the contrast of colours in certain buttons should be changed to improve visibility.

Beta testing methods were also employed towards the end of the project as the main features of the application were developed. In the penultimate sprint, a user,

who was red-green colour-blind, stated that the coloured lines on the data visualization components were "impossible to distinguish" in light mode. Consequently, the line colours were updated to a more distinct set of colours, allowing for them to be more distinguishable in light mode, as shown in Figure 41. Figure 42 shows the view for a user who is not colour-blind. Data visualization components were also made to have hover features, so that when the user hovers on them, a small box of text appears displaying what data they're hovering over.

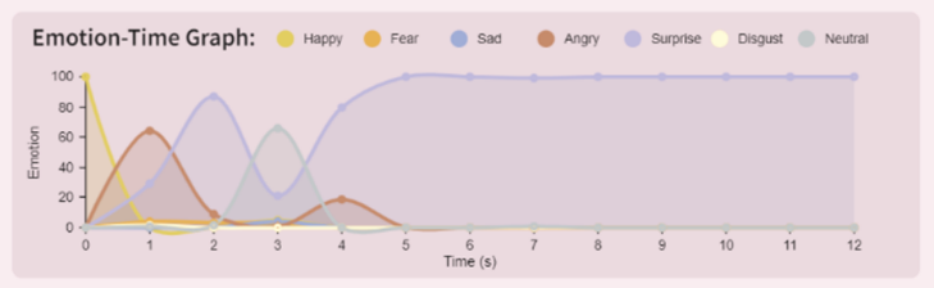


Figure 41: Updated view of the *'EmotionTime'* component, where line colours are more distinguishable for a user who has deuteranomaly, a type of red-green colour-blindness

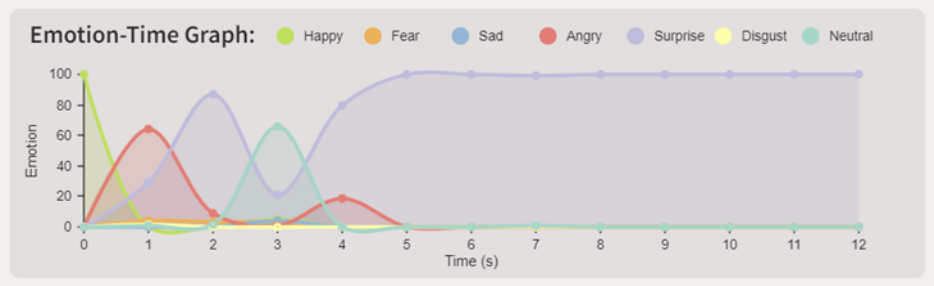


Figure 42: Updated view of the *'EmotionTime'* component for a user who is not colour-blind, i.e. a trichromatic view

User testing after each sprint allowed us to ensure that the final application offers a positive, accessible user experience. As a result, we managed to achieve an 'A' conformance level to the WCAG2.1 (World Wide Web Consortium (W3C) 2018) guidelines, where 'AAA' is the highest level of accessibility, and 'A' is the minimum level of accessibility expected for web content.

8 Evaluation

In this section, we evaluate the success of the project, based on how well we met the requirements stated in Section 3.2. Additionally, we also evaluate the success of the project by exploring the results of a survey conducted on 12 people who tested the final application, as well as the other contemporary applications discussed in Section 2.5.

8.1 Requirements

All functional requirements stated in Section 3.2.1, including 'Should's and 'Could's, were able to be completed, whereas not all non-functional requirements, stated in Section 3.2.2, were able to be completed, specifically NFR.14 and NFR.15. These two requirements were low priority 'Could' requirements and therefore only have a minor impact on the success of the project. Consequently, by looking at how well we met the requirements, we can say this project was a huge success overall.

8.2 Survey

In order to further evaluate the success of the project, a survey was conducted on 12 people who tested RTHAA, Noldus, MorphCast and iMotions. This evaluation method was chosen because surveys allows us to gauge the performance of more subjective aspects of the project, such as overall user experience. Care was taken to ensure that a diverse pool of participants were selected to mitigate the impact of any bias a particular group may have. This involved selecting participants of various ages, sexes and ethnicities. Since most of the participants selected were close friends or family, care was taken to ensure participants weren't aware which application was which, to minimise any personal bias. Ideally, a larger pool of participants would be preferable and yield more accurate results. After participants used all four tools, they were given a survey containing the following questions:

1. Which application do you feel best identified your attributes, i.e. how accurately did it predict your emotions, age and sex?

2. Which application had the better user experience?

3. Which of the four applications do you prefer overall?

4. What were the strengths and weaknesses of each application? Please elaborate.

Figure 43 shows the results of the survey for questions 1, 2 and 3. It clearly depicts RTHAA having a significantly superior user experience and being the most preferred application of the four. Furthermore, the results show that RTHAA was rated higher when it came to predicting the attributes of the participants. This is likely due to the significant amount of research and evaluation done into selecting appropriate pre-trained models, hence resulting in better performance, as well as RTHAA being multimodal and able to detect a wider range of human attributes compared to the other tools.

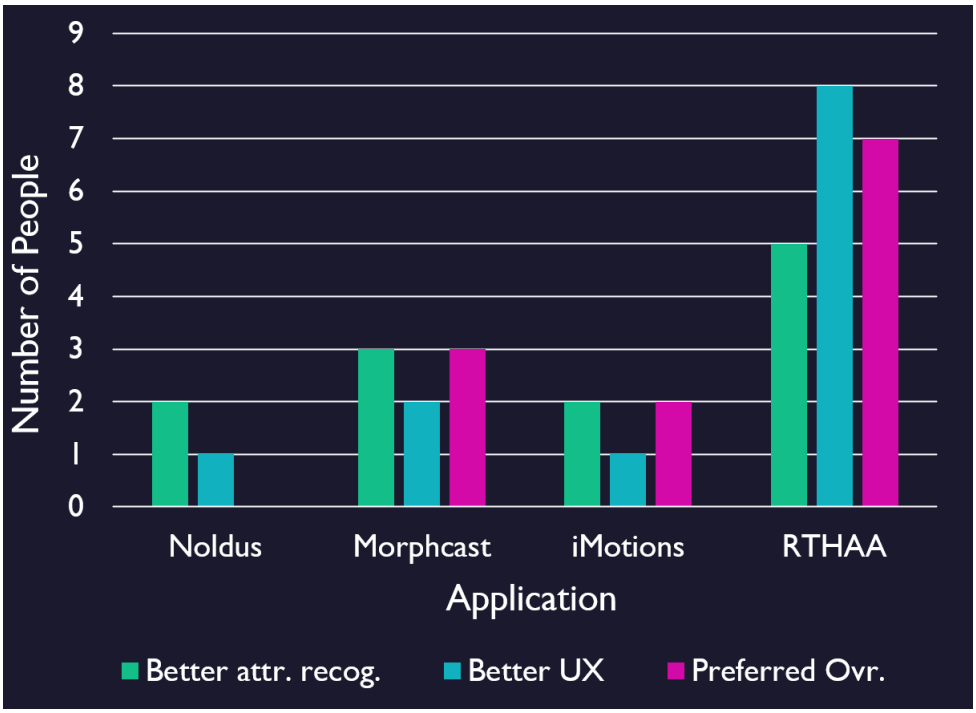


Figure 43: Results of the survey conducted on 12 participants

The responses to question 4 relating to RTHAA complemented the accuracy of the FER model employed, as well as the age and sex model. However, some of the weaknesses stated by users were that the application could often misclassify the sentiment of certain words, however, the overall sentiment score was mostly correct. This was an issue, discussed in Section 6.2.3.1, and is a fundamental issue of

employing a lexicon-based model. One participant also stated that the predicted age would sometimes increase in response to a new facial expression. This is potentially due to certain expressions causing more wrinkling of the skin and hence causing the age model to predict a higher age, however, as only one participant said this, we cannot be sure if this is a persistent issue, once again showing how this survey would benefit from testing on more people.

9 Conclusion and Future Work

To summarise, this project has successfully developed a well-designed web application, capable of analysing a person's emotions, speech sentiment and other attributes. Based on user testing, the application was found to provide a more accurate assessment of a user's system experience compared to existing tools. In addition, this application distinguishes itself as the only one of its kind that is both multimodal and accessible on the internet, requiring no installation of software. Moreover, it boasts a wide range of features, including auxiliary features like age, sex and lighting detection, making it the most comprehensive tool available. The final product is an intuitive, well-designed application which fulfils all of the core objectives outlined at the start of the project. Furthermore, there was a large research component to this project which delved into other methods of emotion recognition, such as through physiological signals. This project managed to overcome many unforseen challenges, most notably of which was switching architecture design late in development, from a backend-processing architecture to a frontend. Another difficulty this project entailed, was how many different areas of computer science it had to combine, namely, full-stack web development, machine learning, sentiment analysis and many more in order to develop the final application.

The major lessons learned in this project mainly concern the tools employed. Various tools that are built on top of React.js could have been utilised to save development time, such as TypeScript, which provides a statically-typed interface to React.js. Redux could've also been employed to help state management in React.js.

9.1 Future Work

As with any product, limitations are inevitable. In this section we discuss the limitations of the application and how it forms the basis for future work and improvement.

One limitation of the application is the accuracy of the sentiment analysis model employed. As mentioned before, it uses a lexicon-based approach which has low classification accuracy, but is high speed. Possible future work could involve developing a model which uses a machine learning approach, but is still fast enough for real-time purposes. Another limitation of the application is that its speech recognition capability is not supported on non-chromium-based browsers, such as FireFox, meaning sentiment analysis cannot be conducted on the user's speech. This is due to the speech recognition library used, which is not natively supported by FireFox. Possible future work may involve solving this issue by using a polyfill, i.e. we host our own speech recognition service which implements the web speech API discussed in Section 6.2.2 and therefore allows speech to be converted to text on non-chromium-based browsers.

10 Ethical Considerations

Various potential ethical concerns were identified during the development of the application, the most notable of which was attempting ethnicity detection, i.e. we predict the user's ethnicity based off of their face. However, this feature was removed as it could warrant racially motivated behaviour. Furthermore, the model used for ethnicity detection has the risk of being biased, which could result in unfair or discriminatory outcomes. Privacy concerns were also raised when using the backend-processing architecture. Users may feel uneasy about sending their personal information, such as their age and sex, to a remote server where it could potentially be stored and misused if adequate safeguarding procedures are not implemented. Moreover, in accordance with the UK's Data Protection Act (UK 2018), which is the UK's implementation of the General Data Protection Regulation (GDPR), obtaining user consent would be necessary for storing this information. Additionally, complete transparency regarding the processing of the data would also be required.

11 Author's Assessment of the Project

- What is the (technical) contribution of this project?

  This project contributes a well-designed web application, capable of analysing a person's emotions, speech sentiment and other attributes. Based on user testing, the application was found to provide a more accurate assessment of a user's system experience compared to existing tools. In addition, this application distinguishes itself as the only one of its kind that is both multimodal and accessible on the internet, requiring no installation of software. Moreover, it boasts a wide range of features, including auxiliary features like age, sex and lighting detection, making it the most comprehensive tool available.

- Why should this contribution be considered relevant and important for the subject of your degree?

  This project's contribution should be considered important as it paves the way for businesses to be able to test their products more scalably and accurately, by analysing the user experience of their customers. The applications scalability comes from it being a web application, available to anyone with an internet connection and a browser. User testing also showed that it gauges user experience more accurately than existing solutions.

- How can others make use of the work in this project?

  Others can use this project's software output not only for their own use, but also as a basis for creating even more advanced human attribute analysis tools, which gauge user experience using even more modalities, models and methods.

- Why should this project be considered an achievement?

  This project should be considered an achievement not only because all core requirements were fulfilled, but more importantly, because it genuinely improves upon existing tools that are available for analysing human attributes.

- What are the limitations of this project?

  As discussed in Section 9.1, the model used for sentiment analysis could be improved as the users in our survey highlighted this as a weakness of the application, so perhaps it is worth using a slower, more accurate model instead. Furthermore, while the application is supported on many popular browsers such as Google Chrome, Microsoft Edge and Safari, its speech-to-text feature is not supported on non-chromium-based browsers, such as FireFox, meaning sentiment analysis cannot be conducted on the user's speech on those particular browsers.

References

Ajili, I., Mallem, M. & Didier, J.-Y. (2019), 'Human motions and emotions recognition inspired by lma qualities', *The Visual Computer* **35**(10), 1411–1426.

Al-Shabi, M. (2020), 'Evaluating the performance of the most important lexicons used to sentiment analysis and opinions mining', *IJCSNS* **20**(1), 1.

Appel, O., Chiclana, F., Carter, J. & Fujita, H. (2018), 'Successes and challenges in developing a hybrid approach to sentiment analysis', *Applied Intelligence* **48**, 1176–1188.

Bettadapura, V. (2012), 'Face expression recognition and analysis: the state of the art', *arXiv preprint arXiv:1203.6722* .

Boughrara, H., Chtourou, M., Ben Amar, C. & Chen, L. (2016), 'Facial expression recognition based on a mlp neural network using constructive training algorithm', *Multimedia Tools and Applications* **75**, 709–731.

Bull, P. E. (2016), *Posture & gesture*, Vol. 16, Elsevier.

Colley, A., Genç, Ç., Löchtefeld, M., Mueller, H., Jensen, W. & Häkkilä, J. (2021), Exploring button design for low contrast user interfaces, *in* 'Human-Computer Interaction–INTERACT 2021: 18th IFIP TC 13 International Conference, Bari, Italy, August 30–September 3, 2021, Proceedings, Part V', Springer, pp. 411–415.

Coulson, M. (2004), 'Attributing emotion to static body postures: Recognition accuracy, confusions, and viewpoint dependence', *Journal of nonverbal behavior* **28**(2), 117–139.

Daityari, S. (2023), 'Angular vs react vs vue: Which framework to choose', `https://www.codeinwp.com/blog/angular-vs-vue-vs-react/#gref`. [Online] Accessed 23 April 2023.

Das, P., Khasnobish, A. & Tibarewala, D. (2016), Emotion recognition employing ecg and gsr signals as markers of ans, *in* '2016 Conference on Advances in Signal Processing (CASP)', IEEE, pp. 37–42.

Deaux, K. (1985), 'Sex and gender', *Annual review of psychology* **36**(1), 49–81.

Delcev, S. & Draskovic, D. (2018), Modern javascript frameworks: A survey study, *in* '2018 Zooming Innovation in Consumer Technologies Conference (ZINC)', IEEE, pp. 106–109.

Dino, H. I. & Abdulrazzaq, M. B. (2019), Facial expression classification based on svm, knn and mlp classifiers, *in* '2019 International Conference on Advanced Science and Engineering (ICOASE)', IEEE, pp. 70–75.

Dzedzickis, A., Kaklauskas, A. & Bucinskas, V. (2020), 'Human emotion recognition: Review of sensors and methods', *Sensors* **20**(3), 592.

Ekman, P. (1992), 'An argument for basic emotions', *Cognition and Emotion* **6**(3-4), 169–200.

Ekman, P. & Friesen, W. V. (1971), 'Constants across cultures in the face and emotion.', *Journal of personality and social psychology* **17**(2), 124.

Farfade, S. S., Saberian, M. J. & Li, L.-J. (2015), Multi-view face detection using deep convolutional neural networks, *in* 'Proceedings of the 5th ACM on International Conference on Multimedia Retrieval', pp. 643–650.

Feldman Barrett, L. & Russell, J. A. (1998), 'Independence and bipolarity in the structure of current affect.', *Journal of personality and social psychology* **74**(4), 967.

Garcia-Garcia, J. M., Penichet, V. M., Lozano, M. D., Garrido, J. E. & Law, E. L.-C. (2018), 'Multimodal affective computing to enhance the user experience of educational software applications', *Mobile Information Systems* **2018**.

Glowinski, D., Camurri, A., Volpe, G., Dael, N. & Scherer, K. (2008), Technique for automatic emotion recognition by body gesture analysis, *in* '2008 IEEE Computer society conference on computer vision and pattern recognition workshops', IEEE, pp. 1–6.

Goodfellow, I. J., Erhan, D., Carrier, P. L., Courville, A., Mirza, M., Hamner, B., Cukierski, W., Tang, Y., Thaler, D., Lee, D.-H. et al. (2013), Challenges in

representation learning: A report on three machine learning contests, *in* 'Neural Information Processing: 20th International Conference, ICONIP 2013, Daegu, Korea, November 3-7, 2013. Proceedings, Part III 20', Springer, pp. 117–124.

Goshvarpour, A. & Abbasi, A. (2017), 'An emotion recognition approach based on wavelet transform and second-order difference plot of ecg', *Journal of AI and Data Mining* **5**(2), 211–221.

Huang, G. B., Mattar, M., Berg, T. & Learned-Miller, E. (2008), Labeled faces in the wild: A database forstudying face recognition in unconstrained environments, *in* 'Workshop on faces in'Real-Life'Images: detection, alignment, and recognition'.

iMotions (2023), 'Facial expression analysis module', `https://imotions.com/products/imotions-lab/modules/fea-facial-expression-analysis/`. [Online] Accessed 17 April 2023.

Jain, V. & Learned-Miller, E. (2010), Fddb: A benchmark for face detection in unconstrained settings, Technical Report UM-CS-2010-009, University of Massachusetts, Amherst.

Jiang, H. & Learned-Miller, E. (2017), Face detection with the faster r-cnn, *in* '2017 12th IEEE international conference on automatic face & gesture recognition (FG 2017)', IEEE, pp. 650–657.

Johns, R. & Singh Khatri, V. (2023), 'Flask vs django', `https://hackr.io/blog/flask-vs-django`. [Online] Accessed 23 April 2023.

Kar, N. B., Babu, K. S., Sangaiah, A. K. & Bakshi, S. (2019), 'Face expression recognition system based on ripplet transform type ii and least square svm', *Multimedia Tools and Applications* **78**, 4789–4812.

Kazmi, S. B. & Arfan Jaffar, M. (2012), 'Wavelets-based facial expression recognition using a bank of support vector machines', *Soft Computing* **16**, 369–379.

Koestinger, M., Wohlhart, P., Roth, P. M. & Bischof, H. (2011), Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark

localization, *in* '2011 IEEE international conference on computer vision workshops (ICCV workshops)', IEEE, pp. 2144–2151.

Kołakowska, A., Landowska, A., Szwoch, M., Szwoch, W. & Wróbel, M. R. (2013), Emotion recognition and its application in software engineering, *in* '2013 6th International Conference on Human System Interactions (HSI)', pp. 532–539.

Lee, S. K., Bae, M., Lee, W. & Kim, H. (2017), 'Cepp: Perceiving the emotional state of the user based on body posture', *Applied Sciences* **7**(10), 978.

Li, H., Lin, Z., Shen, X., Brandt, J. & Hua, G. (2015), A convolutional neural network cascade for face detection, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 5325–5334.

Li, Y., Zeng, J., Shan, S. & Chen, X. (2018), 'Occlusion aware facial expression recognition using cnn with attention mechanism', *IEEE Transactions on Image Processing* **28**(5), 2439–2450.

Li, Z., Tian, X., Shu, L., Xu, X. & Hu, B. (2018), Emotion recognition from eeg using rasm and lstm, *in* 'Internet Multimedia Computing and Service: 9th International Conference, ICIMCS 2017, Qingdao, China, August 23-25, 2017, Revised Selected Papers 9', Springer, pp. 310–318.

Lidberg, L. & Wallin, B. G. (1981), 'Sympathetic skin nerve discharges in relation to amplitude of skin resistance responses', *Psychophysiology* **18**(3), 268–270.

Lindgaard, G., Fernandes, G., Dudek, C. & Brown, J. (2006), 'Attention web designers: You have 50 milliseconds to make a good first impression!', *Behaviour & Information Technology* **25**(2), 115–126.

Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z. & Matthews, I. (2010), The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression, *in* '2010 ieee computer society conference on computer vision and pattern recognition-workshops', IEEE, pp. 94–101.

Lyons, M., Akamatsu, S., Kamachi, M. & Gyoba, J. (1998), Coding facial expressions with gabor wavelets, *in* 'Proceedings Third IEEE international conference on automatic face and gesture recognition', IEEE, pp. 200–205.

Mao, Q., Dong, M., Huang, Z. & Zhan, Y. (2014), 'Learning salient features for speech emotion recognition using convolutional neural networks', *IEEE transactions on multimedia* **16**(8), 2203–2213.

Mather, G. (2006), *Foundations of perception*, Taylor & Francis.

MathWorks (2023), 'Support vector machine (svm)', `https://uk.mathworks.com/discovery/support-vector-machine.html`. [Online] Accessed 15 April 2023.

Medium (2023), 'What are neural networks?', `https://iaviral.medium.com/what-are-neural-networks-an-introduction-to-machine-learning-algorithms-6b73383c9089`. [Online] Accessed 16 April 2023.

Metri, P., Ghorpade, J. & Butalia, A. (2011), 'Facial emotion recognition using context based multimodal approach'.

Michel, P. & El Kaliouby, R. (2003), Real time facial expression recognition in video using support vector machines, *in* 'Proceedings of the 5th international conference on Multimodal interfaces', pp. 258–264.

Mollahosseini, A., Hasani, B. & Mahoor, M. H. (2017), 'Affectnet: A database for facial expression, valence, and arousal computing in the wild', *IEEE Transactions on Affective Computing* **10**(1), 18–31.

MorphCast (2023), 'Client side emotion ai software', `https://www.morphcast.com/`. [Online] Accessed 17 April 2023.

Mozilla (2023), 'Web speech api - speech recognition', `https://wiki.mozilla.org/Web_Speech_API_-_Speech_Recognition#:~:text=The%20speech%20recognition%20part%20of,com%20(for%20voice%20search).` [Online] Accessed 24 April 2023.

Munasinghe, M. (2018), Facial expression recognition using facial landmarks and random forest classifier, *in* '2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)', IEEE, pp. 423–427.

Mühler, V. (2023), 'face-api.js', `https://justadudewhohacks.github.io/face-api.js/docs/index.html`. [Online] Accessed 25 April 2023.

Neal, J., Strothkamp, S., Bedingar, E., Cordero, P., Wagner, B., Vagnini, V. & Jiang, Y. (2019), 'Discriminating fake from true brain injury using latency of left frontal neural responses during old/new memory recognition', *Frontiers in Neuroscience* **13**, 988.

Nguyen, H.-D., Yeom, S., Lee, G.-S., Yang, H.-J., Na, I.-S. & Kim, S.-H. (2019), 'Facial emotion recognition using an ensemble of multi-level convolutional neural networks', *International Journal of Pattern Recognition and Artificial Intelligence* **33**(11), 1940015.

Nielsen, F. Å. (2011), 'Afinn'.
URL: *http://www2.compute.dtu.dk/pubdb/pubs/6010-full.html*

Nielsen, J. (1994), *Usability engineering*, Morgan Kaufmann.

Noldus (2023), 'Facereader - measure your emotions', `https://www.noldus.com/facereader/measure-your-emotions`. [Online] Accessed 16 April 2023.

O'Connor, R. (2023), 'Pytorch vs tensorflow in 2023', `https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2023/`. [Online] Accessed 23 April 2023.

Panchaud, K. (2021), 'Nielsen's 10 usability heuristics & their importance', `https://bootcamp.uxdesign.cc/jakobs-10-usability-heuristics-their-importance-6f0ddec8c938`. [Online] Accessed 24 April 2023.

Piana, S., Stagliano, A., Odone, F., Verri, A. & Camurri, A. (2014), 'Real-time automatic emotion recognition from body gestures', *arXiv preprint arXiv:1402.5047* .

Plutchik, R. (2001), 'The nature of emotions: Human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice', *American scientist* **89**(4), 344–350.

Poria, S., Cambria, E., Bajpai, R. & Hussain, A. (2017), 'A review of affective computing: From unimodal analysis to multimodal fusion', *Information fusion* **37**, 98–125.

Prince, V. (2005), 'Sex vs. gender', *International Journal of Transgenderism* **8**(4), 29–32.

Psomakelis, E., Tserpes, K., Anagnostopoulos, D. & Varvarigou, T. (2015), 'Comparing methods for twitter sentiment analysis', *arXiv preprint arXiv:1505.02973* .

Pu, X., Fan, K., Chen, X., Ji, L. & Zhou, Z. (2015), 'Facial expression recognition from image sequences using twofold random forest classifier', *Neurocomputing* **168**, 1173–1180.

Raftery, J. (2003), *Risk analysis in project management*, Routledge.

Reales, A. (2022), '15 bad things about javascript that none tells you', `https://www.becomebetterprogrammer.com/bad-things-about-javascript/`. [Online] Accessed 23 April 2023.

Rusin, M. (2023), 'What is the vgg-19 neural network?', `https://www.quora.com/What-is-the-VGG-19-neural-network`. [Online] Accessed 25 April 2023.

Russell, J. A. (1980), 'A circumplex model of affect.', *Journal of personality and social psychology* **39**(6), 1161.

Saks, E. (2019), 'Javascript frameworks: Angular vs react vs vue.'.

Statista (2022), 'Number of social media users worldwide from 2017 to 2027 (in billions)', `https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/`. [Online] Accessed 11 April 2023.

Tabian, I., Fu, H. & Sharif Khodaei, Z. (2019), 'A convolutional neural network for impact detection and characterization of complex composite structures', *Sensors* **19**(22), 4933.

Todorovic, D. (2008), 'Gestalt principles', *Scholarpedia* **3**(12), 5345.

Ton-That, A. H. & Cao, N. T. (2019), 'Speech emotion recognition using a fuzzy approach', *Journal of Intelligent & Fuzzy Systems* **36**(2), 1587–1597.

UK (2018), 'The data protection act', `https://www.gov.uk/data-protection`. [Online] Accessed 26 April 2023.

Vaillant, R., Monrocq, C. & Le Cun, Y. (1994), 'Original approach for the localisation of objects in images', *IEE Proceedings-Vision, Image and Signal Processing* **141**(4), 245–250.

Valenza, G., Citi, L., Lanatá, A., Scilingo, E. P. & Barbieri, R. (2014), 'Revealing real-time emotional responses: a personalized assessment based on heartbeat dynamics', *Scientific reports* **4**(1), 1–13.

Viola, P. & Jones, M. (2001), Rapid object detection using a boosted cascade of simple features, *in* 'Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001', Vol. 1, Ieee, pp. I–I.

Wen, W., Liu, G., Cheng, N., Wei, J., Shangguan, P. & Huang, W. (2014), 'Emotion recognition based on multi-variant correlation of physiological signals', *IEEE Transactions on Affective Computing* **5**(2), 126–140.

Wikipedia contributors (2023), 'Document object model — Wikipedia, the free encyclopedia', `https://en.wikipedia.org/w/index.php?title=Document_Object_Model&oldid=1149830415`. [Online; accessed 23-April-2023].

Wioleta, S. (2013), Using physiological signals for emotion recognition, *in* '2013 6th International Conference on Human System Interactions (HSI)', pp. 556–561.

World Wide Web Consortium (W3C) (2018), 'Web content accessibility guidelines (WCAG) 2.1', `https://www.w3.org/TR/WCAG21/`. Accessed: ¡Access Date¿.

Wu, G., Liu, G. & Hao, M. (2010), The analysis of emotion recognition from gsr based on pso, *in* '2010 International symposium on intelligence information processing and trusted computing', IEEE, pp. 360–363.

Wu, S., Xu, X., Shu, L. & Hu, B. (2017), Estimation of valence of emotion using two frontal eeg channels, *in* '2017 IEEE international conference on bioinformatics and biomedicine (BIBM)', IEEE, pp. 1127–1130.

x team (2021), 'React vs angular: Their biggest differences', `https://x-team.com/blog/react-vs-angular/`. [Online] Accessed 23 April 2023.

Xiang, J. & Zhu, G. (2017), Joint face detection and facial expression recognition with mtcnn, *in* '2017 4th international conference on information science and control engineering (ICISCE)', IEEE, pp. 424–427.

Xie, S. & Hu, H. (2017), 'Facial expression recognition with frr-cnn', *Electronics Letters* **53**(4), 235–237.

Yadav, K. S. & Singha, J. (2020), 'Facial expression recognition using modified viola-john's algorithm and knn classifier', *Multimedia Tools and Applications* **79**(19-20), 13089–13107.

Yaghoubi, E., Khezeli, F., Borza, D., Kumar, S. A., Neves, J. & Proença, H. (2020), 'Human attribute recognition—a comprehensive survey', *Applied Sciences* **10**(16), 5608.

Yang, S., Luo, P., Loy, C.-C. & Tang, X. (2016), Wider face: A face detection benchmark, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 5525–5533.

You, M., Chen, C., Bu, J., Liu, J. & Tao, J. (2006), Emotion recognition from noisy speech, *in* '2006 IEEE International Conference on Multimedia and Expo', IEEE, pp. 1653–1656.

Zhang, X., Mahoor, M. H. & Mavadati, S. M. (2015), 'Facial expression recognition using lp-norm mkl multiclass-svm', *Machine Vision and Applications* **26**(4), 467–483.

Zhang, Y.-D., Yang, Z.-J., Lu, H.-M., Zhou, X.-X., Phillips, P., Liu, Q.-M. & Wang, S.-H. (2016), 'Facial emotion recognition based on biorthogonal wavelet entropy, fuzzy support vector machine, and stratified cross validation', *IEEE Access* **4**, 8375–8385.

Appendices

Appendix A Existing Solutions Screenshots

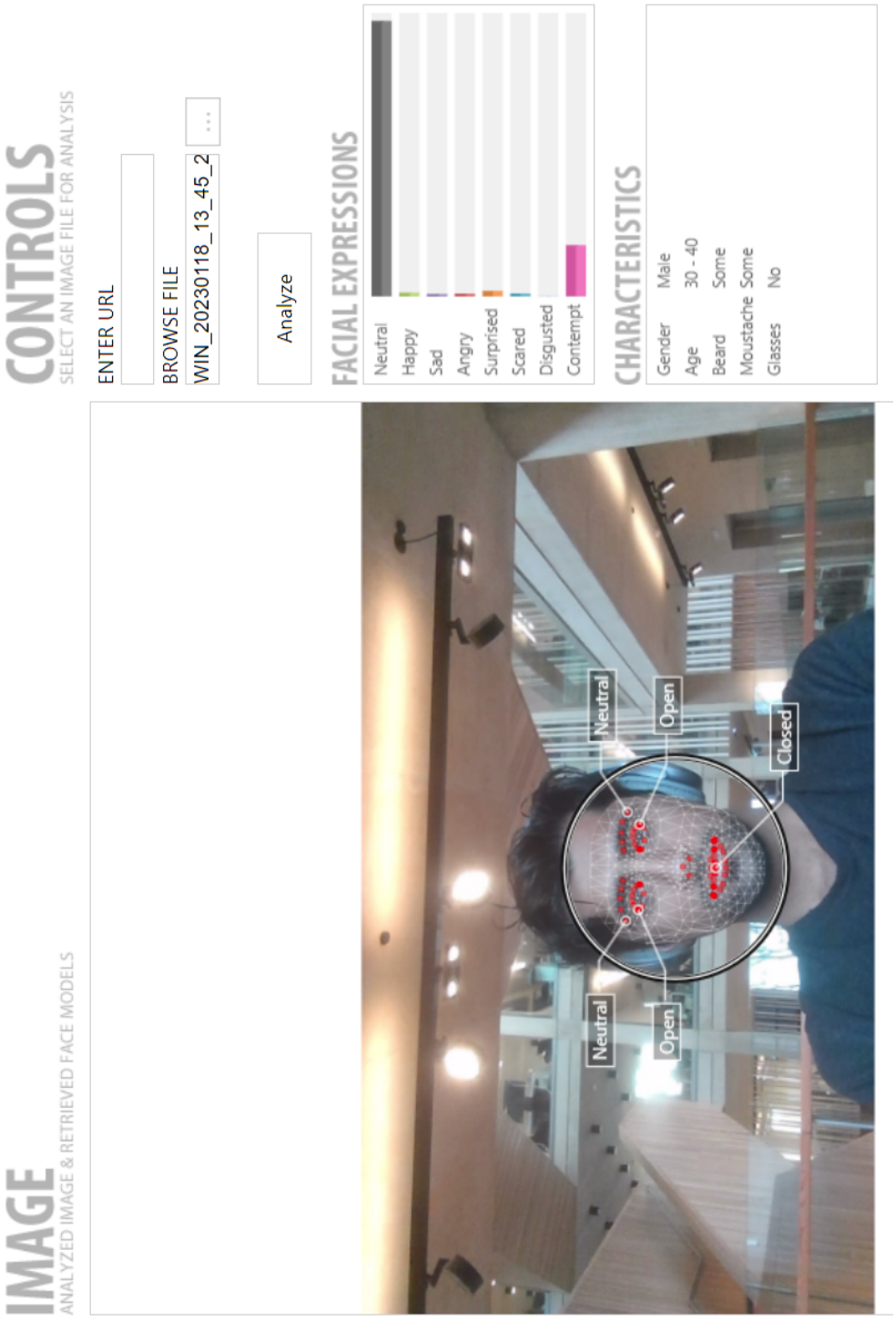


Figure 44: A screenshot of Noldus (2023)

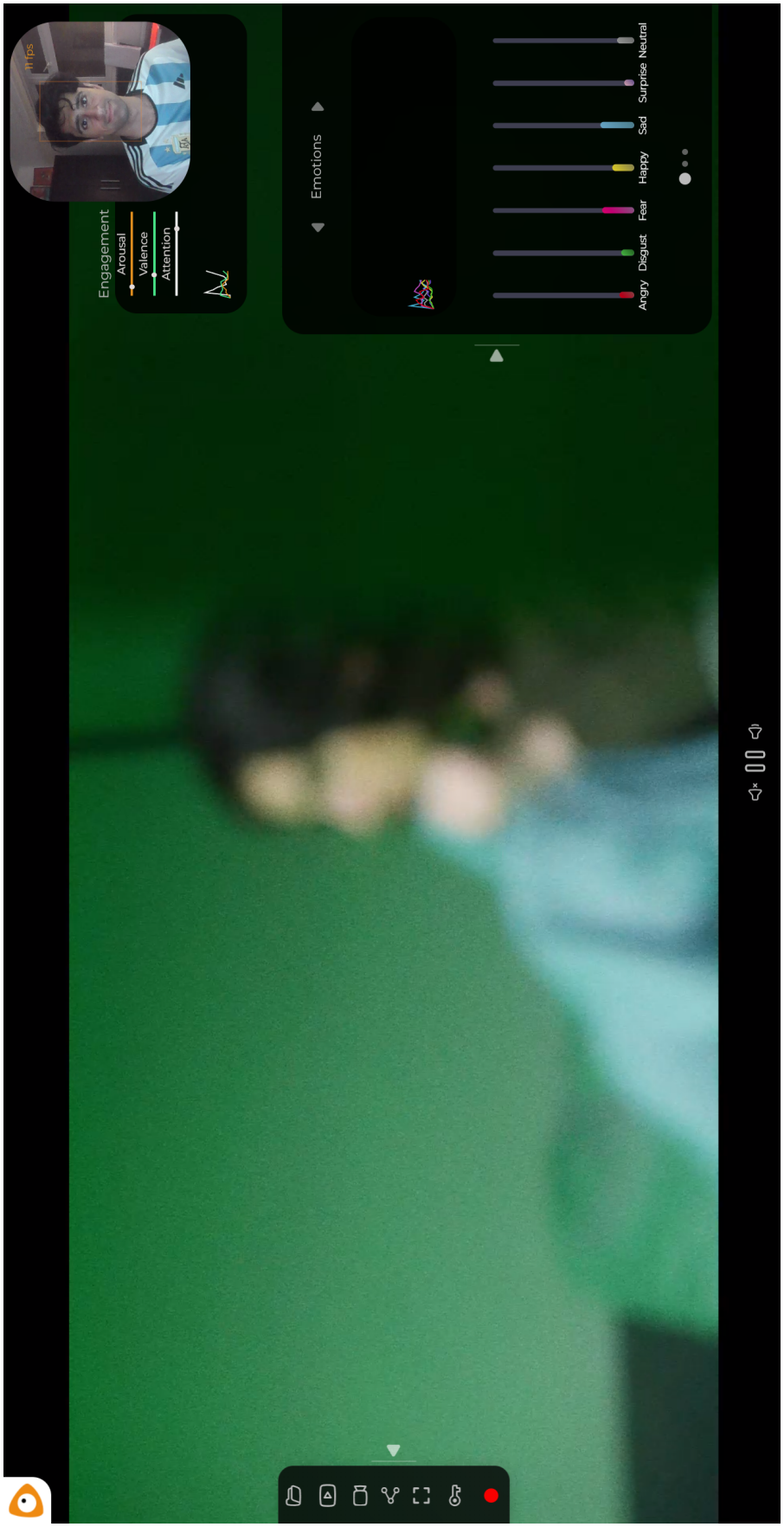


Figure 45: A screenshot of MorphCast (2023)

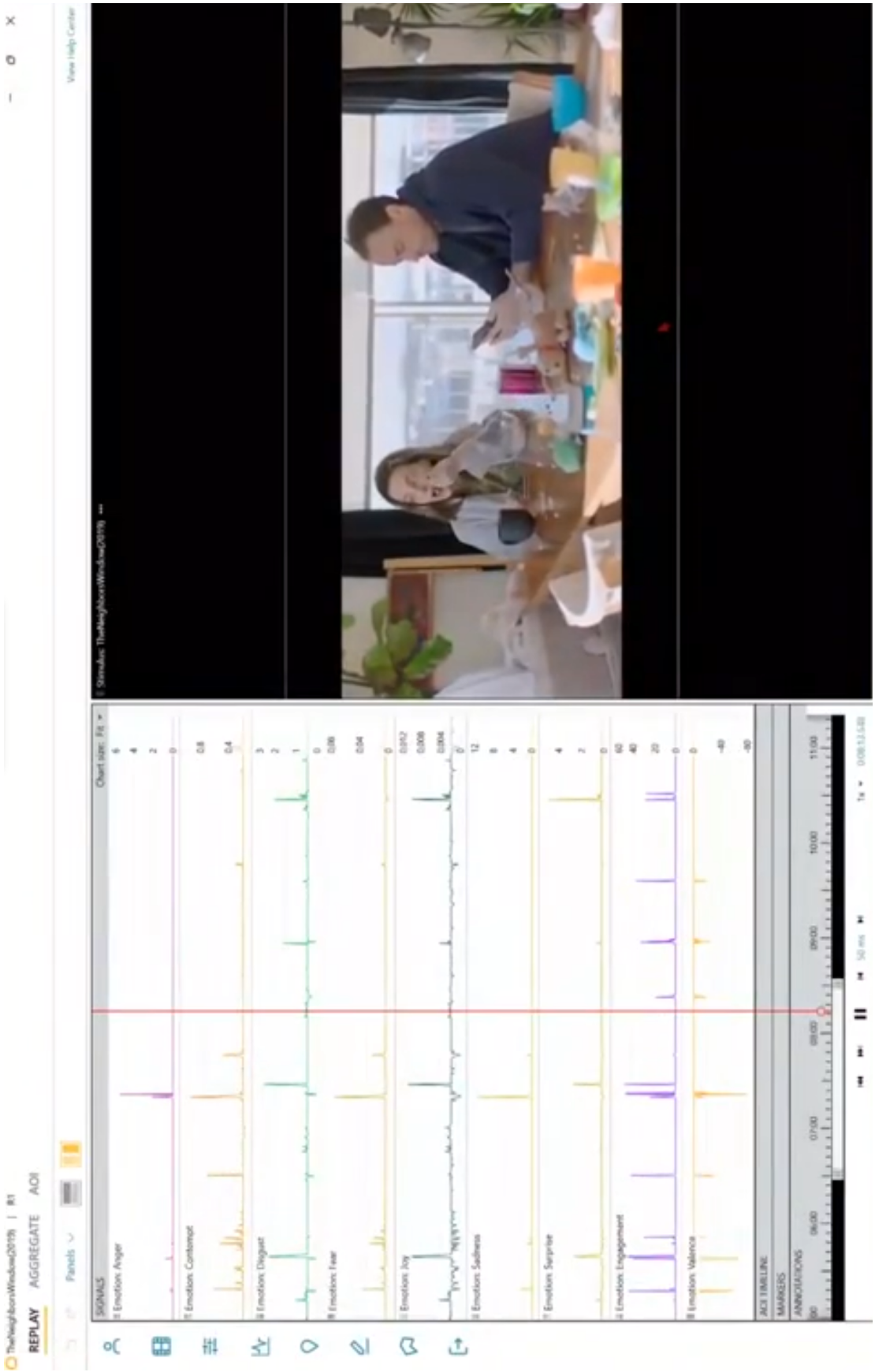


Figure 46: A screenshot of iMotions (2023)

Appendix B Project Methodology Screenshots

B.1 Kanban Board

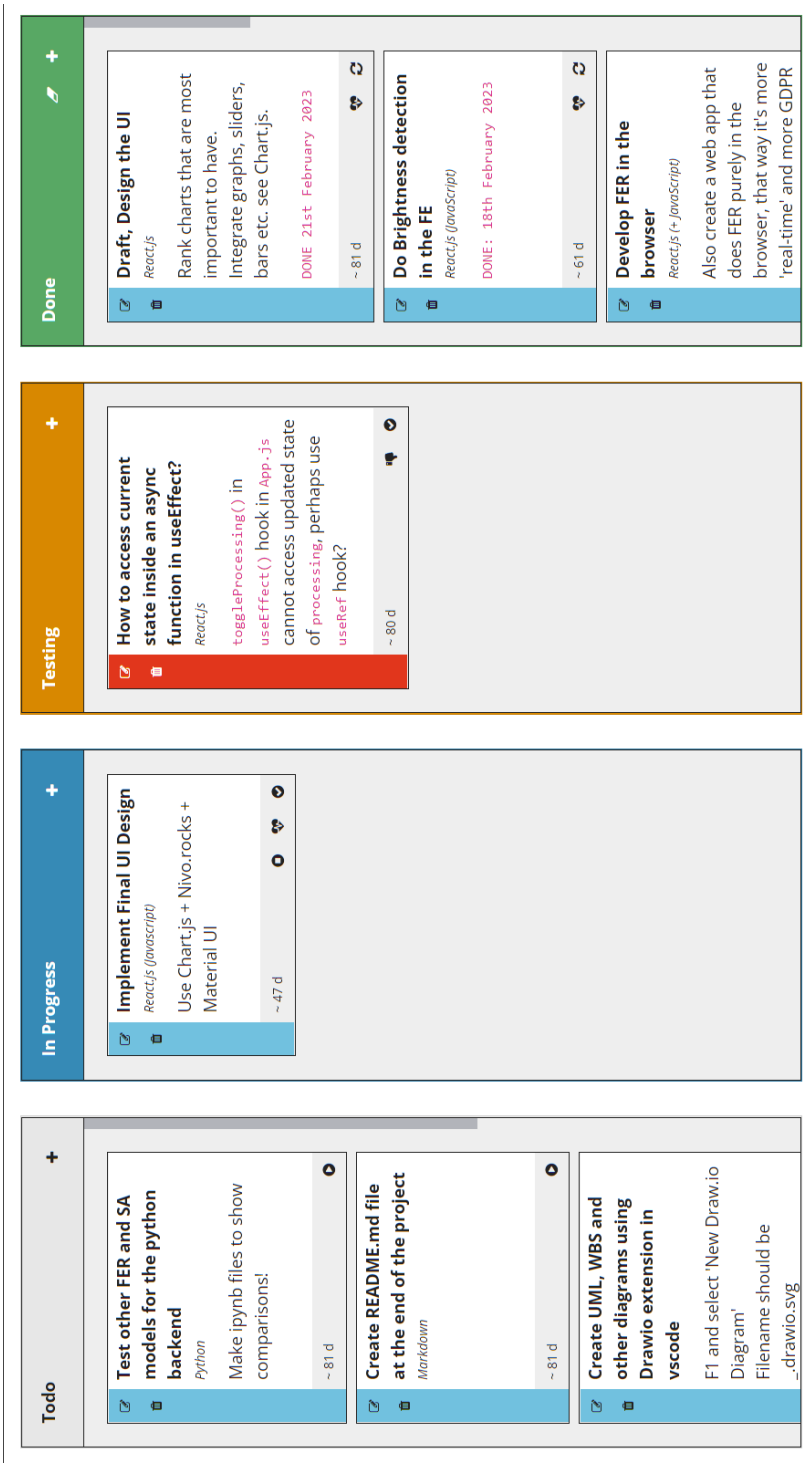


Figure 47: A screenshot of the kanban board used in this project

B.2 Gantt Charts

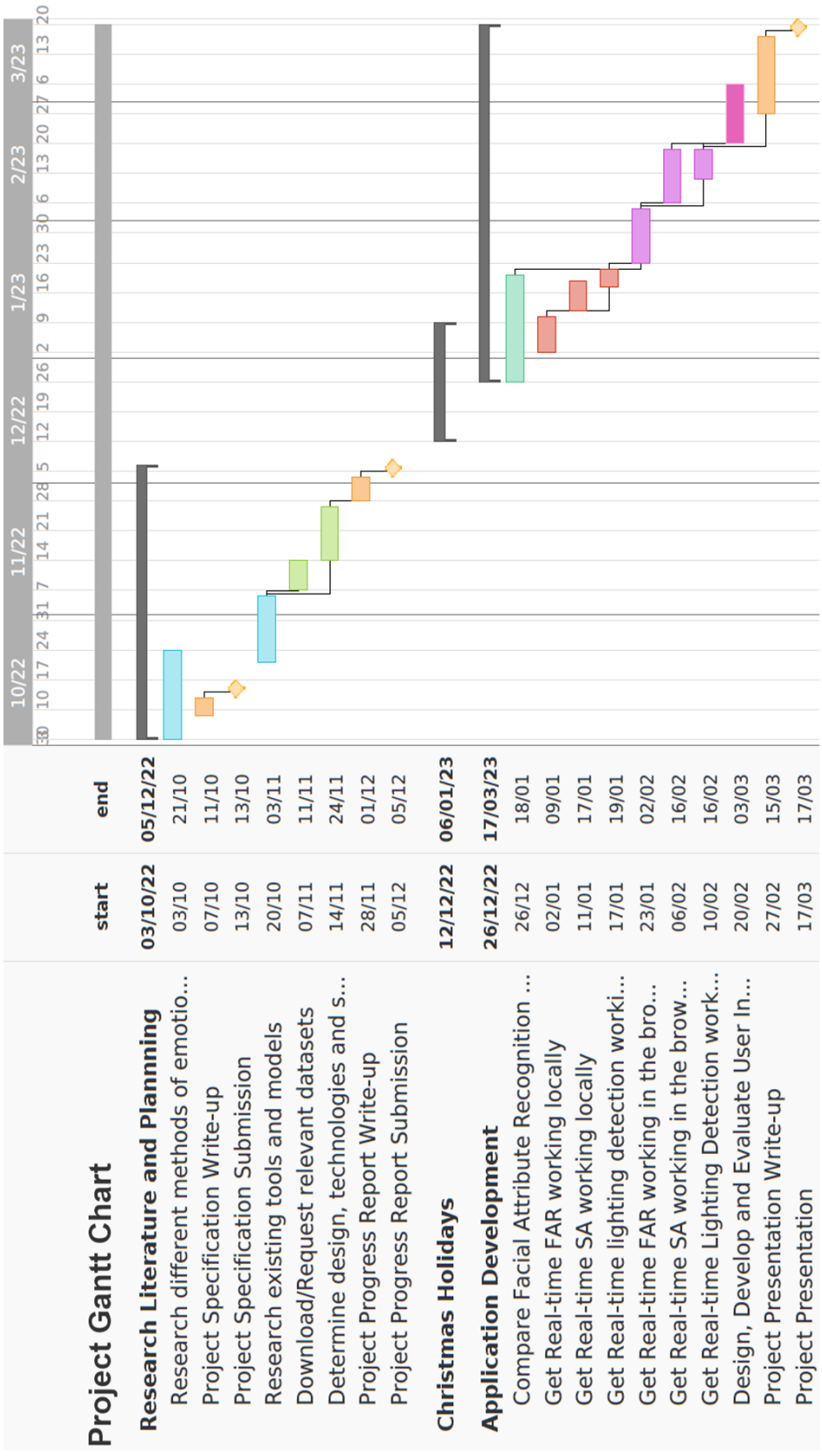


Figure 48: Gantt chart showing the planned schedule of the project

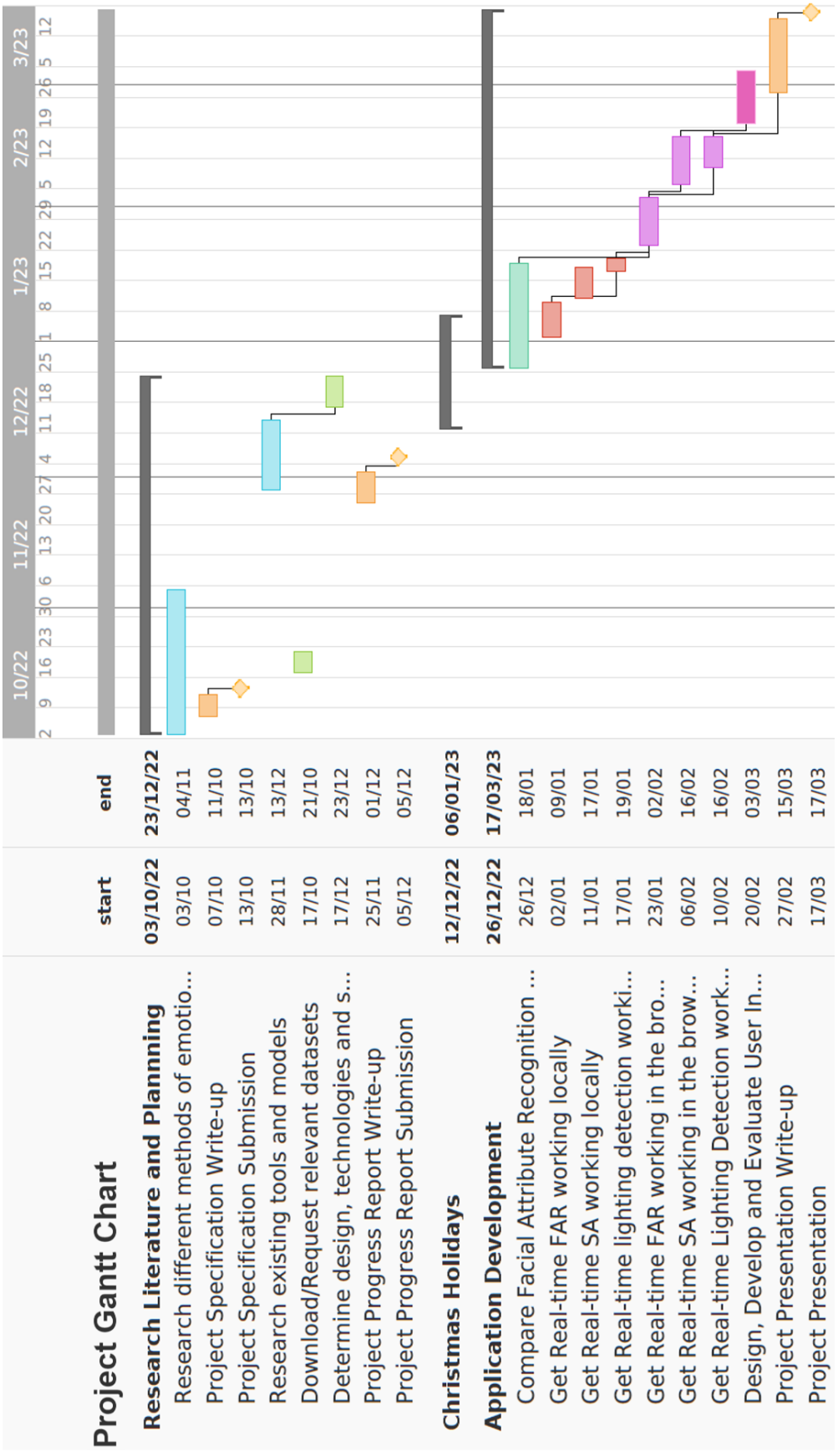


Figure 49: Gantt chart showing the executed timeline of the project

Appendix C System Diagrams

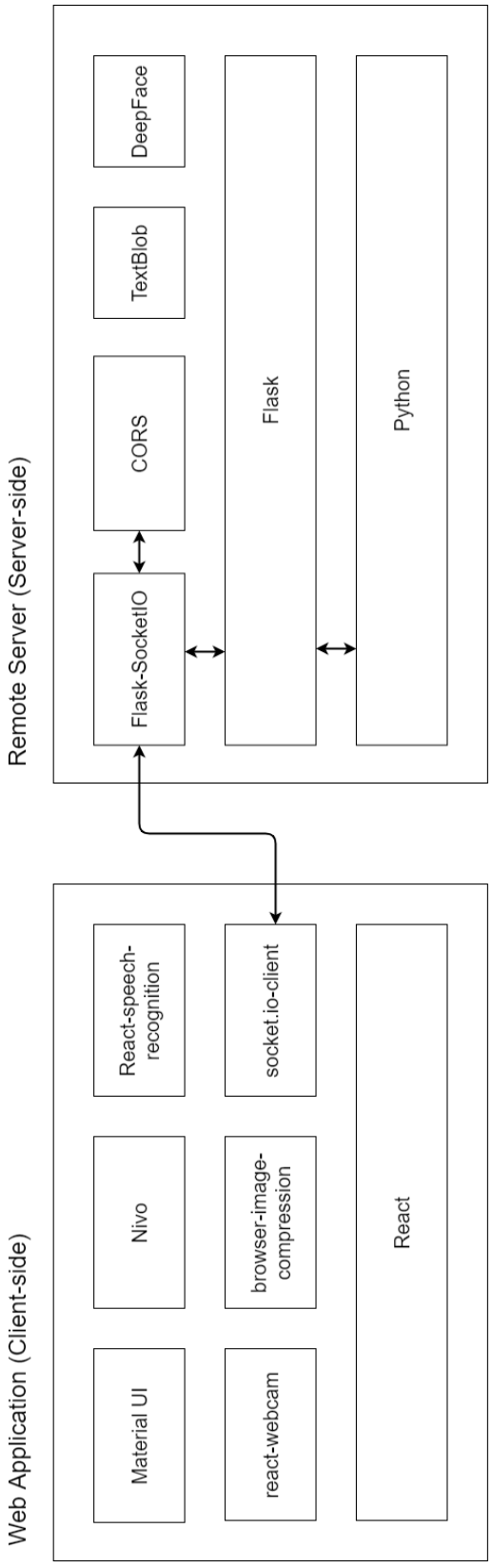


Figure 50: System architecture diagram showing key components of architecture that uses a remote server

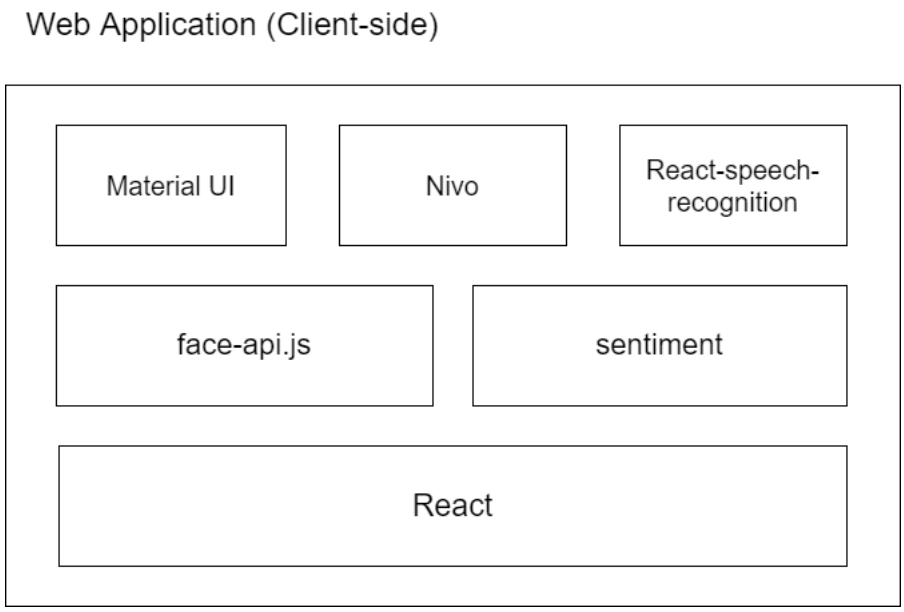


Figure 51: System architecture diagram showing key components of architecture that processes all data in the frontend

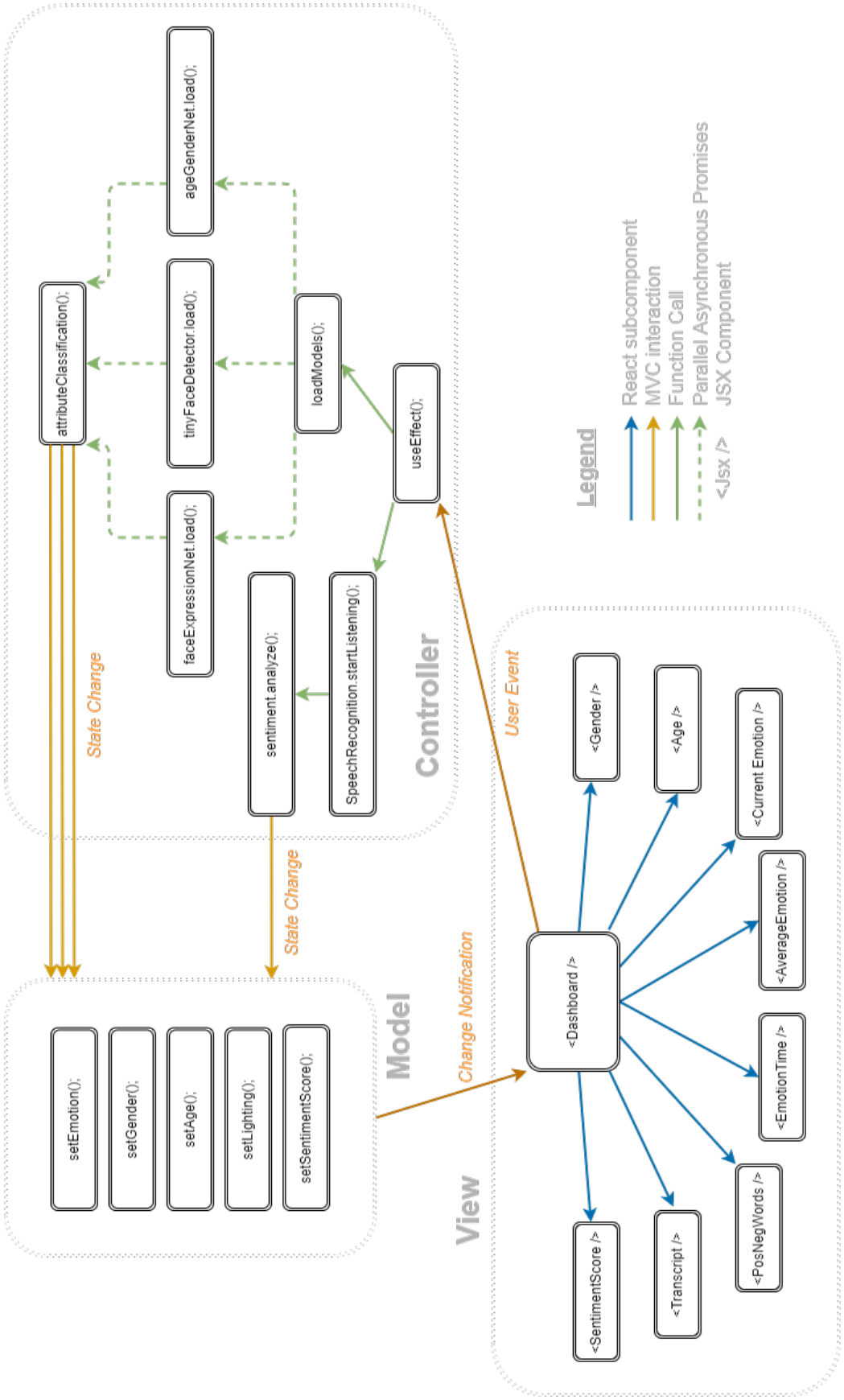


Figure 52: Simplified diagram showing how the MVC architectural pattern was implemented in *RTHAA*

Appendix D Application Screenshots

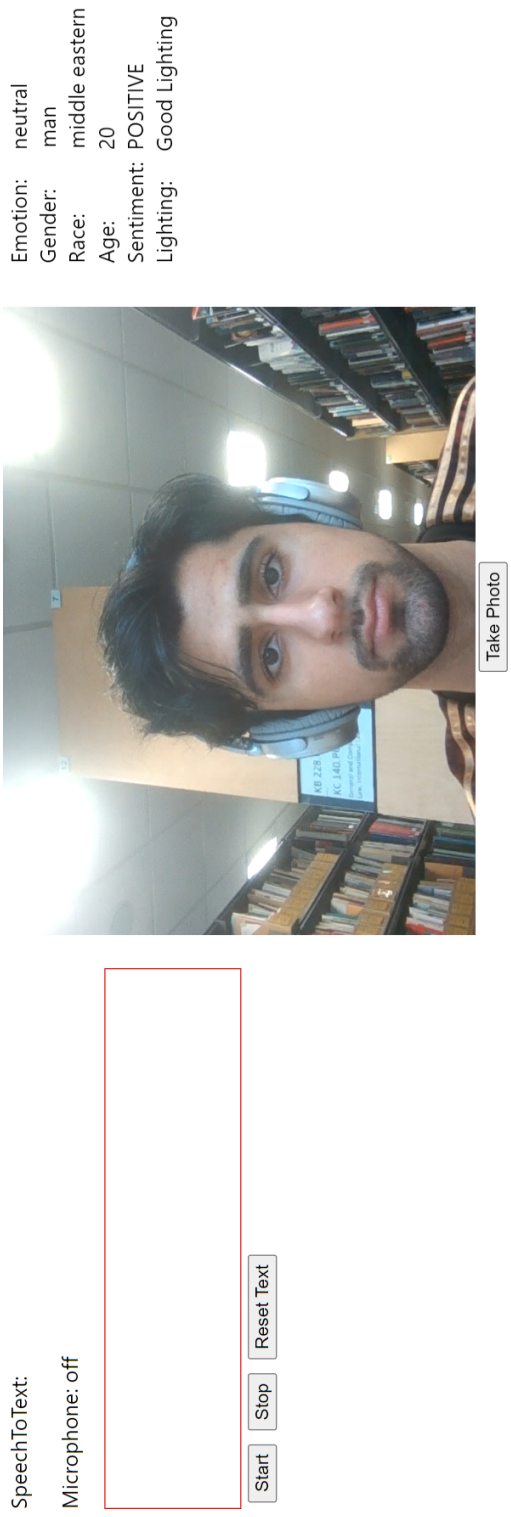


Figure 53: MVP 1 of RTHAA dashboard

Figure 54: MVP 2 of RTHAA dashboard

Figure 55: MVP 3 of RTHAA dashboard, using face-api.js

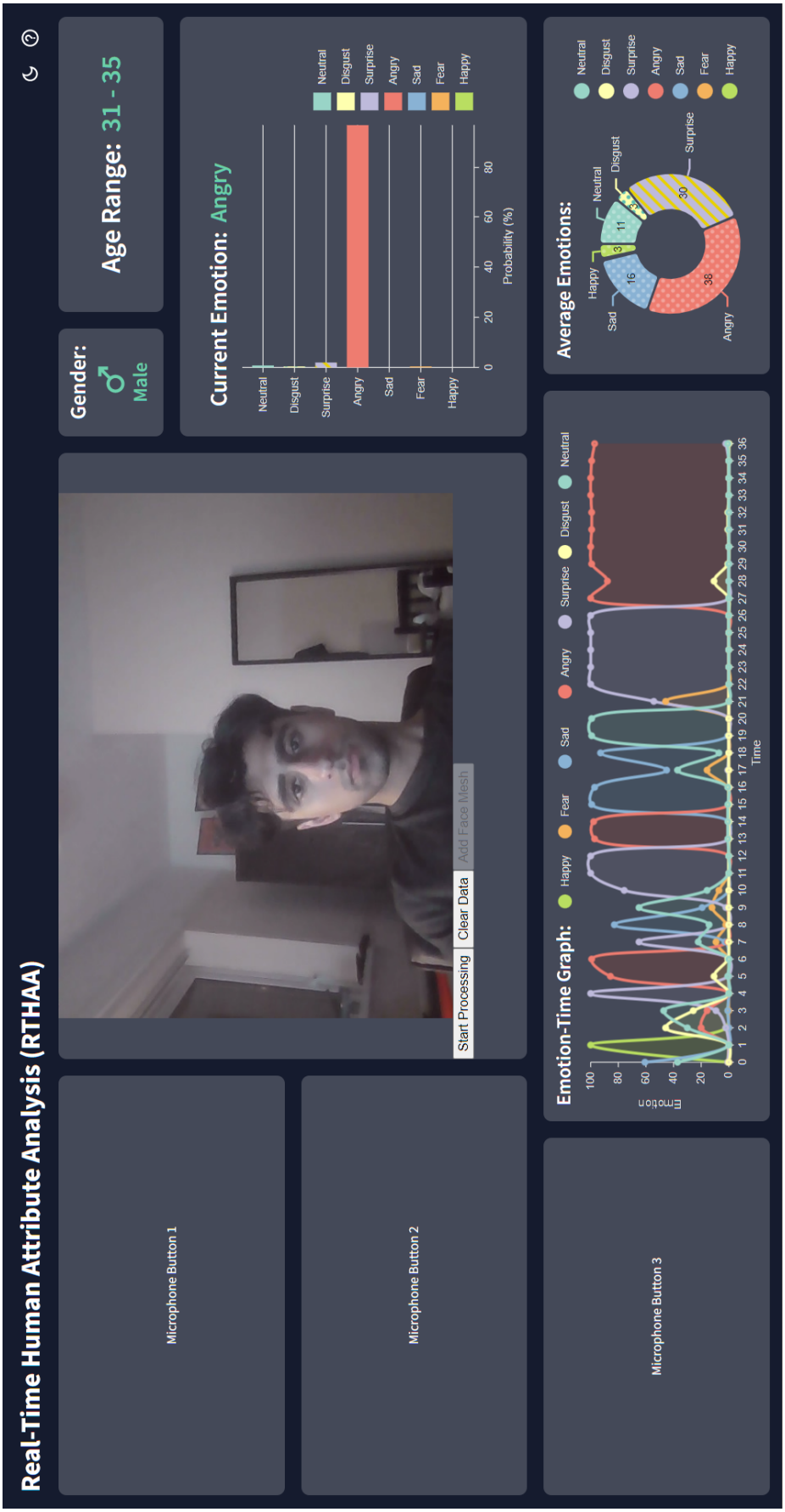


Figure 56: MVP 4 of RTHAA dashboard

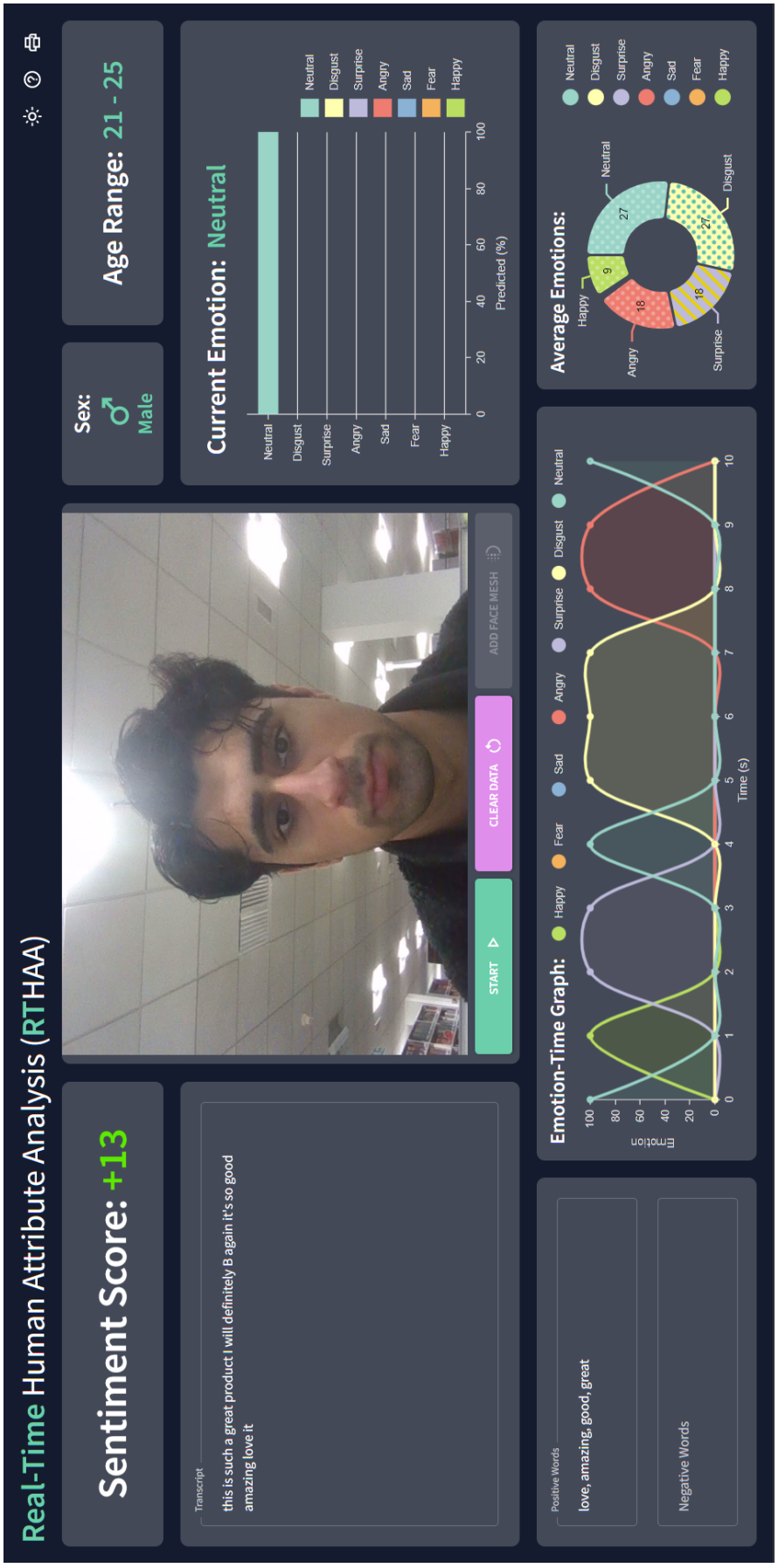


Figure 57: MVP 5 ofRTHAA dashboard

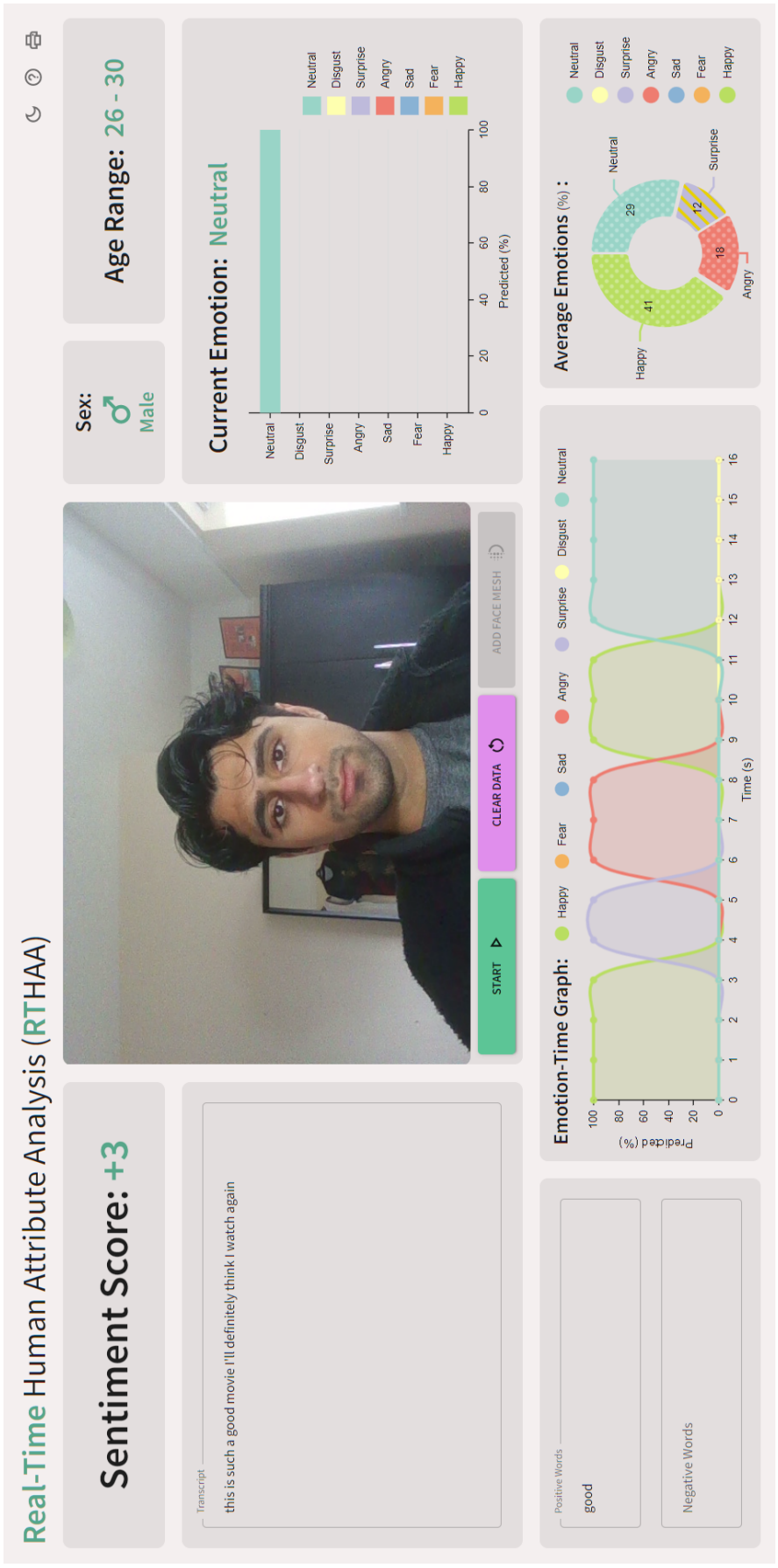


Figure 58: Finished RTHAA dashboard in light mode

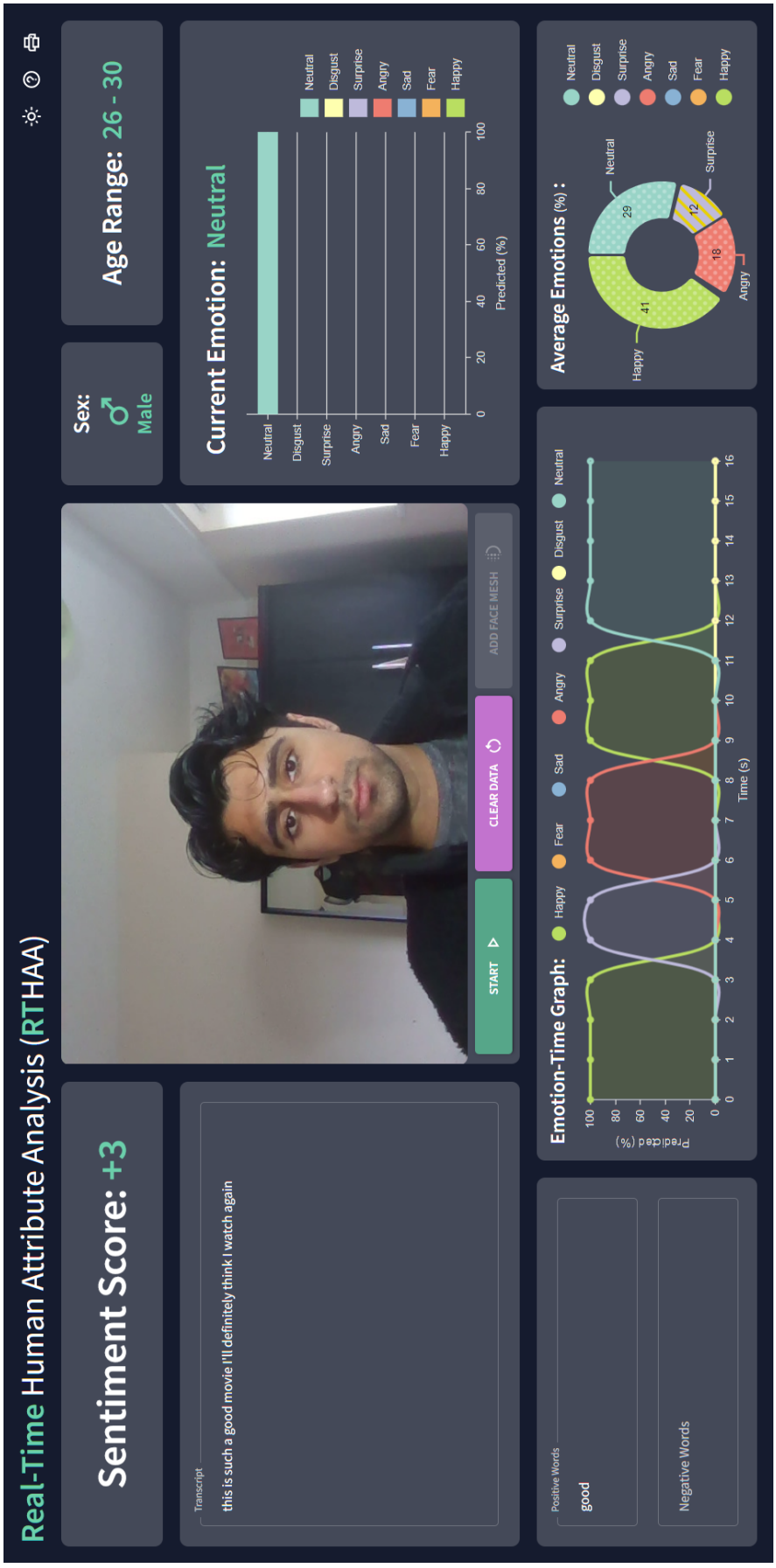
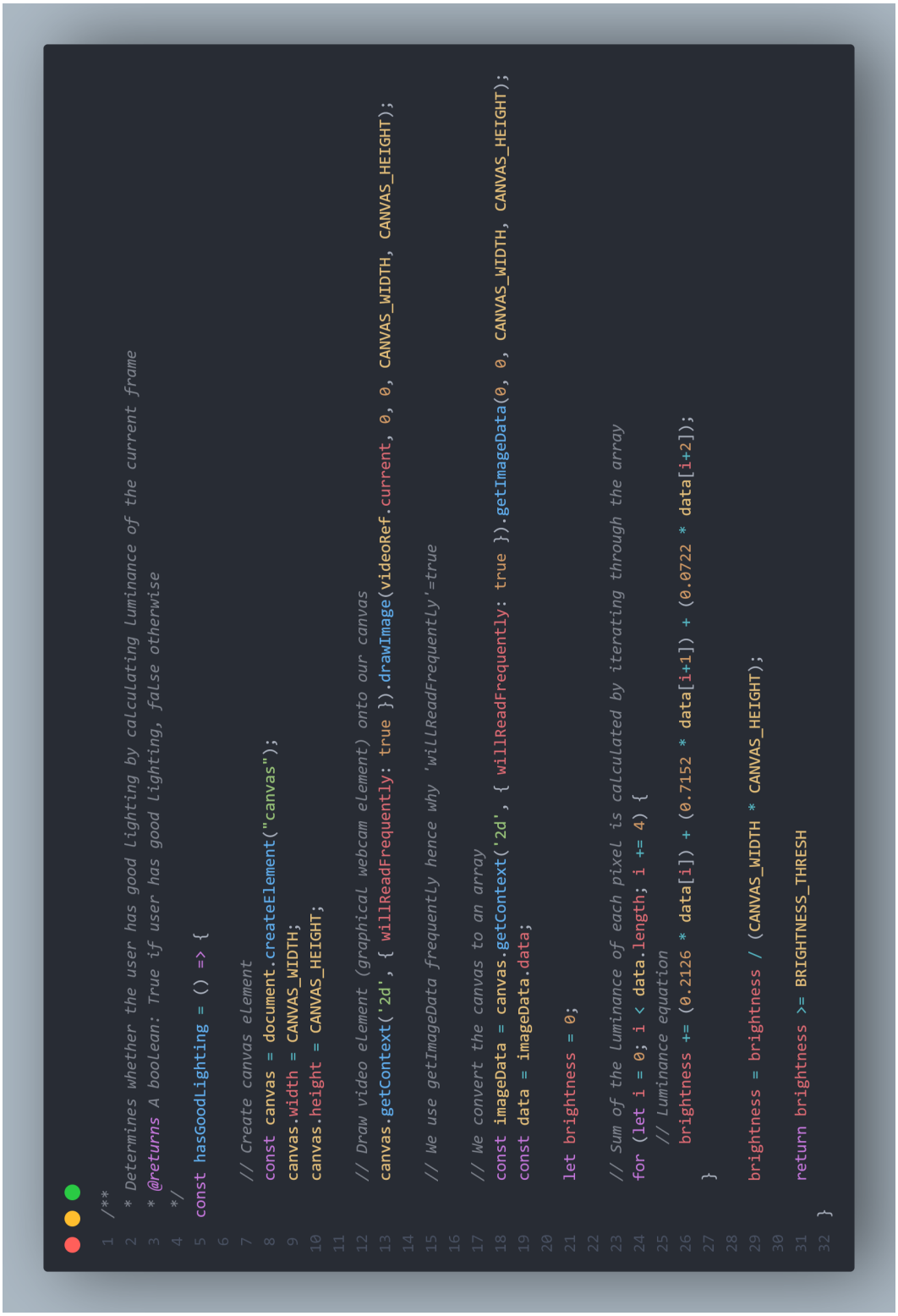


Figure 59: Finished RTHAA dashboard in dark mode

Appendix E Code Snippets



```
/**
 * Determines whether the user has good lighting by calculating luminance of the current frame
 * @returns A boolean: True if user has good lighting, false otherwise
 */
const hasGoodLighting = () => {

    // Create canvas element
    const canvas = document.createElement("canvas");
    canvas.width = CANVAS_WIDTH;
    canvas.height = CANVAS_HEIGHT;

    // Draw video element (graphical webcam element) onto our canvas
    canvas.getContext('2d', { willReadFrequently: true }).drawImage(videoRef.current, 0, 0, CANVAS_WIDTH, CANVAS_HEIGHT);

    // We use getImageData frequently hence why 'willReadFrequently'=true

    // We convert the canvas to an array
    const imageData = canvas.getContext('2d', { willReadFrequently: true }).getImageData(0, 0, CANVAS_WIDTH, CANVAS_HEIGHT);
    const data = imageData.data;

    let brightness = 0;

    // Sum of the luminance of each pixel is calculated by iterating through the array
    for (let i = 0; i < data.length; i += 4) {
        // Luminance equation
        brightness += (0.2126 * data[i]) + (0.7152 * data[i+1]) + (0.0722 * data[i+2]);
    }

    brightness = brightness / (CANVAS_WIDTH * CANVAS_HEIGHT);

    return brightness >= BRIGHTNESS_THRESH
}
```

Figure 60: Code snippet showing how the lighting conditions of the user are determined