



Project Title

Interim Report

DT228
BSc in Computer Science

Damian Wojtowicz

C17413722

Andrea Curley

School of Computer Science
Technological University, Dublin

Date

Abstract

The goal of this project is to investigate the use facial expression recognition technology to improve software usability and user experience testing. The application will be of use to software developers who will have users complete test cases and the application will record the user's facial expressions to better detect the problem areas in the software.

This project is going to use machine learning to automatically detect the user's emotions and display the data to the developers for further analysis. The advantage of automatically detecting emotions is that video footage does not have to be watched by anyone and data is automatically collected and stored.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

Student Name

Date

Acknowledgements

Body text

Table of Contents

1. Introduction.....	7
1.1. Overview.....	7
1.2. Project Description	7
1.3. Project Aims and Objectives	8
1.4. Challenges	8
1.5. Project Scope.....	8
1.6. Thesis Roadmap.....	9
2. Research.....	10
2.1. Background Research	10
2.1.1 Emotions in Usability Testing.....	10
2.1.2 Facial Expression Recognition Research.....	13
2.1.3 Usability Testing Research.....	14
2.1.4 Machine Learning Research.....	15
2.3.2 Dataset Research	19
2.2. Alternative Existing Solutions to Your Problem	22
2.2.1 UserZoom Go.....	22
2.2.2 UserTesting.....	24
2.2.3 Loop11	26
2.3. Technologies Research	29
2.3.3 Programming Languages	29
2.3.4 Python Libraries	29
2.4. Usability Testing Technologies	31
2.4.1 Usability Testing.....	31
2.4.2 Video Uploading.....	31
2.4.3 Website Hosting	31
2.5. Existing Final Year Projects	33
2.6. Conclusions	34
3. Prototype Design.....	36
3.1 Introduction	36
3.2 Software & Data Mining Methodologies	36
Scrum Agile.....	36
CRISP-DM	38
3.3 Overview of System	40
3.4. Web Application	43
3.4.1 Front-End.....	43

3.4.2 Back-End	51
3.4.3 Database	51
3.5 Local App.....	52
3.5. Facial Expression Recognition & Machine Learning	54
CRISP-DM Methodology in Prototype Development	54
Discussion.....	55
3.6 Conclusions	56
4. Prototype Development	56
4.1. Introduction.....	57
4.2 Machine Learning.....	57
4.2.1 CSV to Image Converter	57
4.2.2 Model Training.....	60
4.2.3 Model Evaluation & Discussion	63
4.6. Conclusions	66
5. Testing and Evaluation.....	67
5.1. Introduction.....	67
5.2. Plan for Testing.....	67
5.3. Plan for Evaluation.....	70
5.4. Conclusions	70
6. Issues and Future Work	71
6.1. Introduction.....	71
6.2. Issues and Risks	71
6.3. Plans and Future Work	72
6.3.1. GANTT Chart	73
Bibliography	74

1. Introduction

1.1. Overview

The purpose of this project is to explore facial expression recognition technology as a way to improve usability testing for software. This project will focus mainly on usability testing on websites. Usability testing is a very important aspect of software development and especially crucial when the application has a large amounts of users with varying degrees of ability. Usability testing involves testing the functionality of an application, a website or digital product by observing how real people with no prior knowledge of the website attempt to navigate through it and attempt to complete tasks [\[1\]](#).

Usability testing reveals issues that are not obvious to developers, designers and people who have in-depth knowledge and understanding of the product. By observing the user, the designer can not only identify problems but also uncover opportunities to improve and learn about the target user's preferences. For a website to be successful it must allow its users to efficiently and effectively get to where they want to be and complete the tasks they want to complete without confusion and frustration.

Research has been conducted on the number of participants needed to reveal the most problems with the best returns. In a study conducted by Virzi (1990, 1992) random samples of different sizes of participants were created, it was found that on average, four or five participants would reveal about 80% of the usability problems uncovered by all the participants, with diminishing returns with additional participants [\[2\]](#). Faulkner (2003) also concluded that testers may want to include more than five participants. With 100 random samples of five participants, she found that on average, the five-person samples uncovered 85% of the problems [\[2\]](#). However, the results by in research paper by Bastien *et. al* (2003) [\[3\]](#) indicated that the first 5 participants only uncovered 35% of the usability problems.

The purpose of this project is to implement a website usability testing tool and investigate if usability testing can be improved upon with the use of machine learning and more specifically facial expression recognition.

1.2. Project Description

UsabCheck is usability testing application and consists of two applications. The first application is a web application that will be used by the researcher, they could be the developer or designer of their website. The researcher will access the *UsabCheck* web application to create and configure their usability study. The researcher will provide the link to the website they want to conduct the usability test on and provide instructions for the tasks they want the participant to complete and questionnaires and questions they want the user to fill out at any point within the study.

The *UsabCheck* web application will provide statistics on how participants have performed in the tasks and their responses to questions and other forms. The researcher will have the option to view the screen recording and a camera recording of their face while they are completing the tasks. This includes the audio of the participant speaking. The researcher will also be provided with a journey map of the user completing the task and their facial expressions to help better identify the problem areas.

The second UsabCheck application will run locally on the computer the participant is using and will be recording their screen and monitoring their facial expressions. After the usability test is conducted the data is uploaded to the cloud. This data includes video's and screen captures.

1.3. Project Aims and Objectives

The aims and objectives that should be achieved by the end of the project include the following.

1. Provide a usability testing website that allows the researchers to create usability study tests.
2. Provide the researchers to view the results of the tests on the website. This includes viewing of the footage, answers to questions and questionnaires, viewing a journey map of facial expressions.
3. Provide a local usability testing application that will record the screen, record the test participants face from the camera, show the participant the test instructions, and upload the data to the cloud.
4. For the above aims to be achieved, smaller goals and objectives have to be met. This includes a machine learning algorithm that will classify facial expressions.
5. The machine learning algorithms and dataset have to be researched and implemented.
6. Cloud storage has to be obtained in some way, whether it be creating my own or paying for a service to store the videos for streaming on the website.

1.4. Challenges

1. The dataset for the face expression recognition will be a very important factor in the success of the classification.
2. The various algorithms for face classification need to be researched and considered as this will have an impact on the accuracy of face expression classification.
3. The usability testing aspect has to be researched well for the application to be useful. The scope of this project is big in that regard.
4. There are various ways of going about cloud storage, the pros and cons of each will have to be considered.
5. Time management is a challenge that is very important to the success of the project.

1.5. Project Scope

The scope of this project involves designing and implementing a usability application for desktop applications. More specifically the focus will be on website usability testing. The application will not be intended for usability testing on mobile applications. The application will involve the use of machine learning to detect facial expressions of the participant who is completing the tests created by the researchers or developers. The project will involve two applications; a web application and a local application.

1.6.Thesis Roadmap

Will be added at the very end

2. Research

2.1. Background Research

2.1.1 Emotions in Usability Testing

Landowska (2015) [4] investigated the use of emotion analysis in usability testing. They have evaluated the methods of obtaining the emotion data and the models for representing that data. The means of obtaining the data which include questionnaires, facial expression analysis and sentiment analysis were evaluated by the accuracy, robustness to disturbances, independence on human will, and interference with usability testing procedures. The method of obtaining the emotion recognition data that is relevant to this project is facial expression analysis. The accuracy of this method is medium to high and does not interfere with the usability testing procedure. However, the robustness to disturbances is low, meaning that illumination and occlusion of the face reduces the accuracy of the method.

They have investigated the use of Ekman's six basic emotion model (See Figure 1 – Ekman's Six Basic Emotion Model) as a method for representing the emotional state. Examples of emotional states include frustration, boredom, excitement, and empowerment. These emotional states consist of one or more of the more basic emotions such as disgust, anger, surprise, fear, sadness and joy. They have found that with this model certain emotional states are difficult to illustrate with the Ekman's six basic emotion model, for example, boredom can be expressed as subtle sadness which is not the most intuitive representation of the emotional state. Whissel's wheel emotion model (See Figure 2) has been investigated. Dimensional models such as Whissel's wheel can be easily interpreted by computer systems, however it is more difficult for humans to understand. Further research on this model can be done to automatically detect more complex emotions in the *UsabCheck* application.

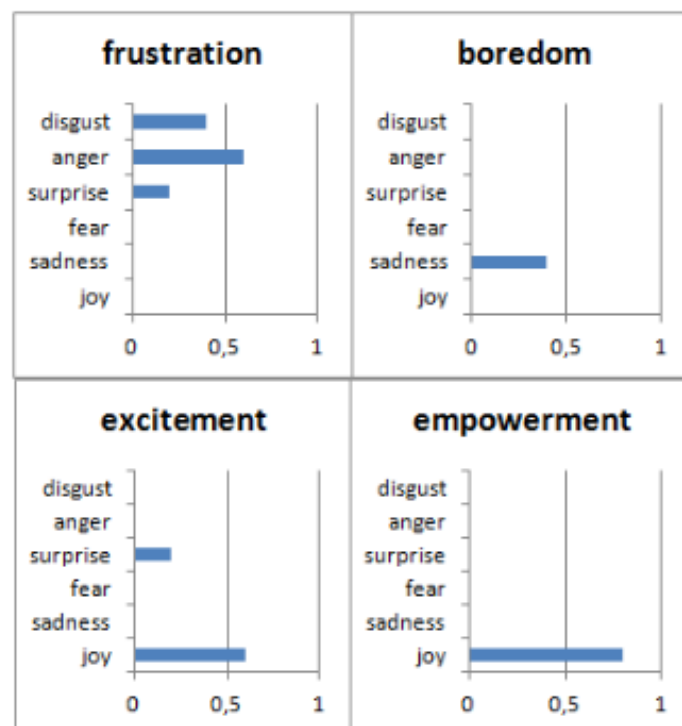


Figure 1 – Ekman's Six Basic Emotion Model

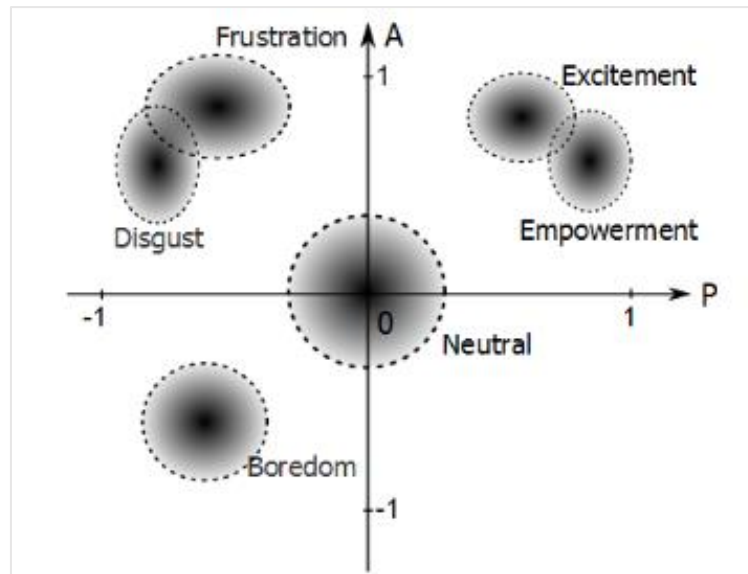


Figure 2 – Whissel's Wheel Emotion Model

An additional study which involved post hoc analysis of recordings from usability tests was performed. This was to assess the practical applicability of the emotion recognition techniques in a real-world environment. The tests varied in that some participants used a mouse and keyboard and other participants used a touch screen. The instructions were provided on paper and the participants would look away from the camera to look at the instructions. This disturbance meant that the facial expression recognition with the use of the camera was not effective during those times. Some participants covered their face with their hand as much as 33 percent of the time which interfered with the facial expression recognition. The camera was located below the screen.

In conclusion, this research paper provided insight into the various means of detecting emotions and representing the data. The facial expression recognition technique of obtaining data is vulnerable to interference, however, it provides accuracy in detecting the emotion. The 6 basic emotions can be combined in various ways to form more complex emotions such as frustration and boredom. This is certainly useful in providing the researchers who will use the *UsabCheck* application with more intuitive data.

Esterwood (2018) [5] discussed how facial expression recognition technology can be applied to usability testing. They looked at journey maps as a way to display emotion data. Journey maps (See Figure 3) track the user as they move through the task and shows the user's emotions along with the task they are trying to complete. In Figure 4, negative emotions of anger and sadness were assigned a -1 value and surprise and happiness were assigned +1 value on the y-axis. The time is plotted on the x-axis and the tasks are separated using a dotted line. The chart plots the emotions in a way that is either positive or negative which is useful. However, it does not display exactly what emotion was experienced which makes this approach debatable. Surprise was marked with a +1 value which is the same as happiness and carries the assumption that the user's surprise is a positive thing. One might argue that for a system to be usable the user should not much experience surprise and user surprise is a result of bad design.

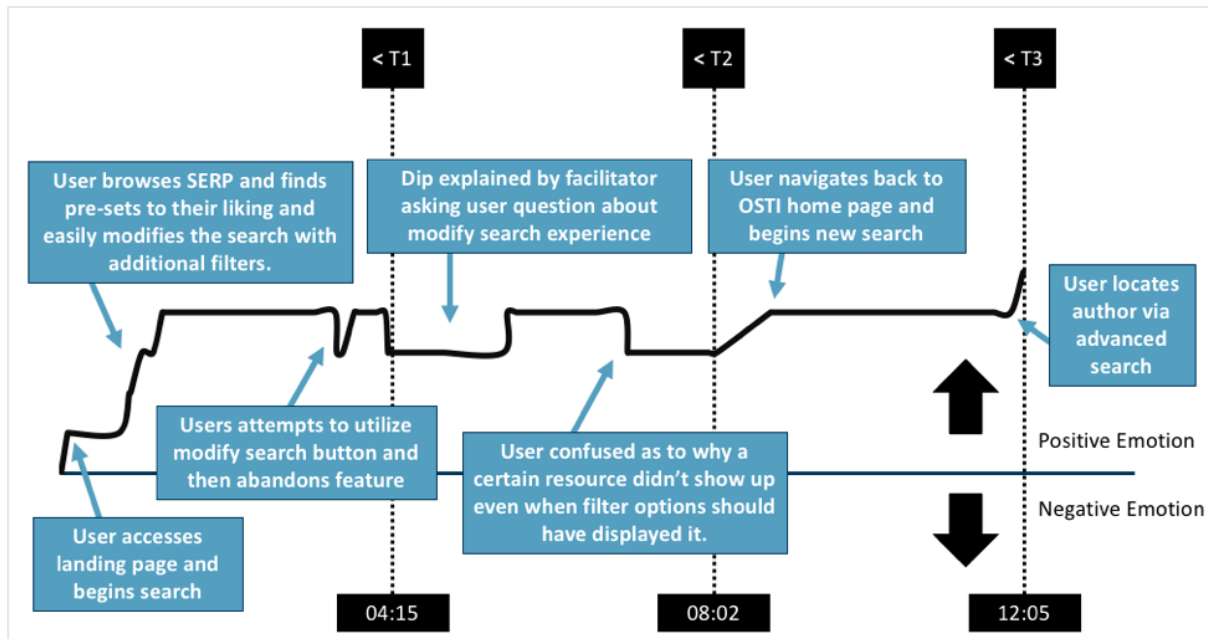


Figure 3 - Example Journey Map

The recommendations by the study for future studies is to ensure that users are limited to a strict time limit as a way to allow for averaging of emotional states and combining them into one journey map. Another recommendation is to have a second camera exclusively for the face as cropping decreases the resolution. In conclusion this study has provided insight into displaying the emotion data in the context of usability testing that is easy to understand. However, the simplicity might be a disadvantage as the chart only works with positive and negative values and the value each emotion should be assigned is up for debate.

The recommendation to limit the amount of time the user is allowed to spend on a task as a way of better averaging the emotional states is very insightful. Users of varying degrees of ability will spend different amount of times on each task and this will certainly be an issue in pinpointing the exact stage the user is during the usability test. Reducing the amount of time, the user can spend on a task can help with that. However, that length of time is subjective to the researcher conducting the usability test.

2.1.2 Facial Expression Recognition Research

Halder et al. (2016) [6] proposed a prototype system which classifies the six basic emotions such as happiness, sadness, anger, fear, surprise, and disgust. They explored a new approach to the emotion classification from facial expression by combining a neural network-based approach with image processing. The system includes face detection and feature extraction for emotion classification. The feature extraction involved locating the eyebrows, eyes and mouth regions. The image processing techniques included the use of edge detection, for example, Sobel edge detection which finds edges by the gradient changes in an image. These features are points and these points are processed to obtain inputs for the neural network. The results of the classification were left out of the research paper since the results were being tested. Despite the lack of results the research has provided insight into facial expression recognition classification and shown that image processing techniques can be useful.

Gaurav (2018) [8] wrote a case study that discussed real-time facial expression recognition with deep learning. The case study outlined the objectives and constraints associated with facial expression recognition. Accuracy is a constraint in deep learning models and the higher the accuracy the better it will perform in the application and the less errors are made. The case study also outlined three performance metrics that can be used in the evaluation of the deep learning model. These are the *Multi-Class Log-loss*, *Accuracy*, and *Confusion Matrix*. These performance metrics will be discussed on their own in another section of the research document. The model was trained on both human faces and faces of animated characters with an approximate ratio of 1:9. The cross-validation accuracy on animated characters was 100% and 87% on human images. The VGG-16 convolutional neural network was used, this architecture will be discussed in detail in another section of the report. Three human face expression datasets were used as some datasets were not large enough. In total there was approximately 1,500 human images and 9,000 animated images.

The below confusion matrix (Figure 4) displays the results tested on the human data. The predicted class is plotted on the x-axis and the true values are plotted on the y-axis. E.g. The “Angry” class was predicted 25% of the time when the actual class was “Disgust”. The model predicted “Angry” when presented with negative face expressions such as disgust and sadness.

In conclusion this case study has provided a lot of value to this project as it introduced many aspects of deep learning which includes the datasets, training, validating and testing the model, performance metrics for evaluating the model and the neural network architecture. These served as a starting point for further research. This case study also proved that animated images may be used to help create a facial expression recognition model that can classify facial expressions of real people.

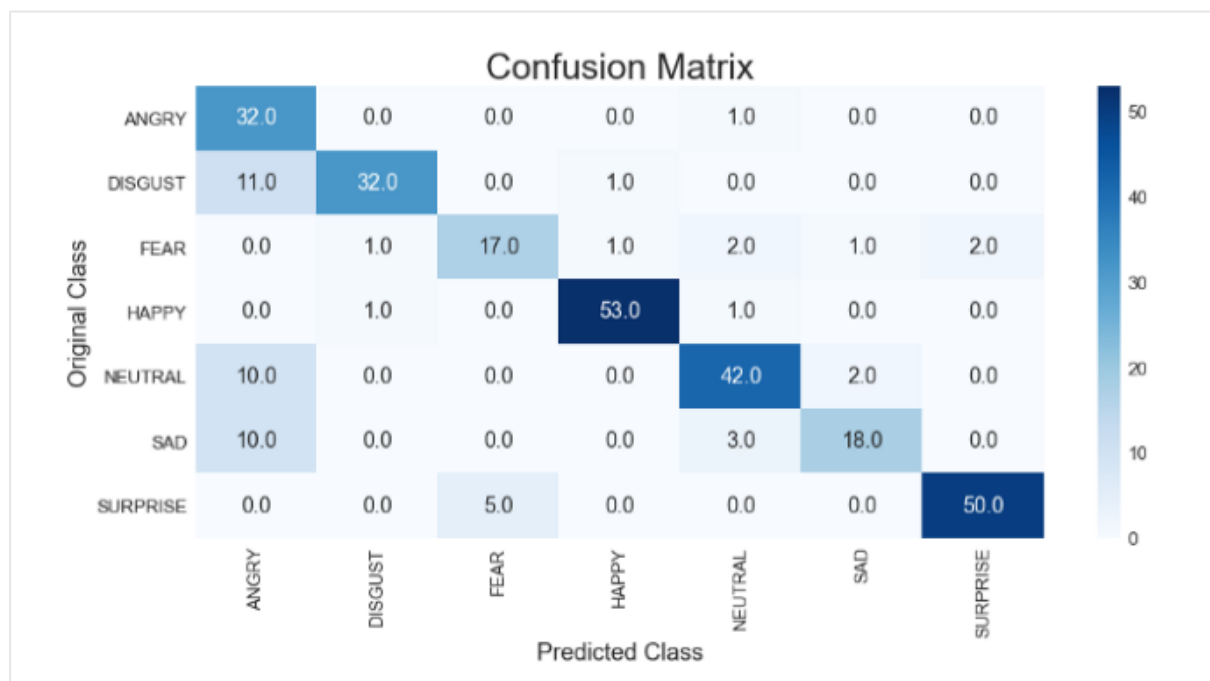


Figure 4 - Confusion Matrix Example

2.1.3 Usability Testing Research

Bastien (2009) [7] reviewed the methods and procedures in usability testing. The usability testing applications the research is based on are in the field of medical and health care informatics. *UsabCheck* will be designed to allow usability testing on virtually all desktop applications, meaning this research is relevant. They discussed the research on remote usability testing. Usability testing in a lot of cases involves the researcher/evaluator inviting the participant to their research facility and the participant completing the tasks in a test room. This room would contain the equipment to record the participant completing the tasks. For example, the recordings can be visual or/and audio. In remote usability testing the participant and the researcher are not in the same location.

Remote usability evaluation can be synchronous, meaning that researcher can manage the usability test in real-time and the interaction between researcher and test participants can be facilitated with conferencing applications. Usability evaluation can also be asynchronous, meaning that the observers are not present during the test and cannot interact with the participant as they are completing the tasks. The advantage of remote asynchronous evaluation is many usability tests to be conducted at one time in parallel and recordings and data collected can be evaluated at another time. For example, the participant will download software onto their machine that will give them instructions for conducting the usability test.

The disadvantage is that the user's equipment such as the microphone and camera need to be of a certain standard to collect the data. Variables such as lighting conditions can reduce the accuracy of the facial expression recognition algorithm.

This research has provided valuable insight into conducting usability tests and raised questions to be discussed when designing the *UsabCheck* application with regards to remote testing.

2.1.4 Machine Learning Research

In the previous section evaluation metrics were mentioned which included the confusion matrix. In this section the confusion matrix and other related machine learning model evaluation metrics will be discussed in detail. The article by Narkhede (2018) [9] explains the confusion matrix and how the values can be used to calculate precision, recall, accuracy, specificity and the AUC-ROC curve. The cell where the row and column with the same label match is the "True Positive" value. This is illustrated in Figure 5. In other words, the accuracy of the model guessing correctly. The "True Negative" value is when the model has predicted correctly that the result is negative. The "False Negative" value is when the model wrongly predicted negative when it should have predicted positive. The cells in the same row show the "False Positives" where the model wrongly predicted a value as positive when it should have predicted a value as negative.

To put the theory into context of emotion classification we look at "Angry" column in Figure 6. Outlined in green is the correct predictions made by the model. The x-axis label and y-axis label for that cell are both "Angry". The cells within the column that are outlined in red and shows the number of times "Angry" has been predicted when the actual label was not "Angry", for example, "Disgust". The cells outlined in orange show the times when the model wrongly classified "Angry" as another class. E.g. The emotions was classified as "Neutral" when it was "Angry".

		Predicted Values	
		Positive (1)	Negative (0)
Actual Values	Positive (1)	True Positive	False Negative
	Negative (0)	False Positive	True Negative

Figure 5 – Confusion Matrix Labels

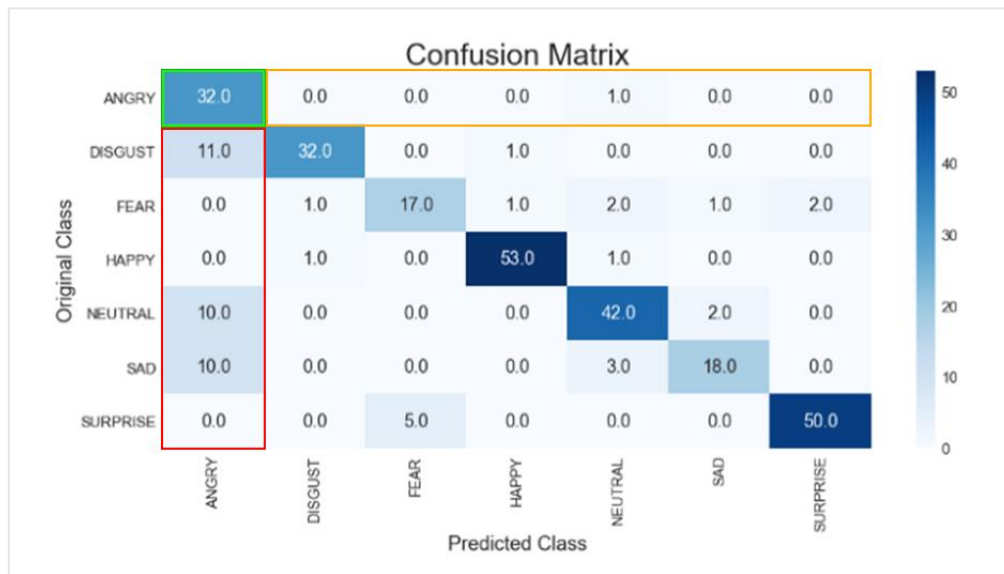


Figure 6 – Confusion Matrix

The article by Koehrsen (2018) [10] explains in detail the classification metrics of recall, precision, accuracy and the F-measure. Accuracy on its own is not a good enough metric to determine if the model is useful if the sample size is large and the ratio of the classes is extremely uneven. This is known as the imbalance classification problem. Increasing the recall means decreasing the precision and vice versa. Depending on the type of application, the design decision might be to increase precision over recall or recall over precision and this will be considered in the design of the *UsabCheck* application.

The *Recall* is the ability of the model to correctly predict.

$$\text{Recall} = \frac{TP}{TP + FN}$$

The *Precision* calculates how many positive classes were actually positive.

$$\text{Precision} = \frac{TP}{TP + FP}$$

The *Accuracy* is how much out of all the classes have been predicted correctly.

The *F-measure* combined the recall and precision values to more easily compare two models with high recall and low precision and vice versa.

$$\mathbf{F - measure = \frac{2*Recall*Precision}{Recall + Precision}}$$

2.3.1.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a type of Deep Neural Network (DNN) and can efficiently and effectively be used to perform numerous tasks that involve pattern and image recognition. These tasks include object recognition, hand gesture recognition [Citation] and face recognition [Citation], to name a few. CNNs may be used for image segmentation, classification and retrieval with high accuracy. For example, a CNN trained on the MNIST (handwritten digits dataset) has achieved a detection rate of 99.77% [Citation] which shows how effective CNNs in image classification.

A CNN consists of layers and these layers include, the input layer, convolutional layers, pooling layers, rectified linear unit layer and a fully connected layer. The input layer is the first layer that takes in the input data. The convolutional layer applies filters to extract the features from the data and a CNN may involve the use of many convolutional layers. The extracted features are passed to the pooling layer which reduces the size of the images. This preserves the most prominent features which is extracted. The Rectified Linear Unit Layer (ReLU) is an activation function that replaces any negative number in the pooling layer with zero. This keeps the CNN mathematically stable and prevents the vanishing gradient problem when the number of layers increases [Citation]. The fully connected layer is the final layer and uses the results from the previous layer to classify an image [Citation] [Citation].

2.3.1.2 VGG Model

The VGG-16 Model [Citation] is a famous CNN model that achieved a 92.7% test accuracy in the ImageNet dataset [Citation], making it the winner of the ILSVRC (ImageNet Large Scale Visual Recognition Competition) in 2014 [Citation]. The dataset the model was trained on consisted of 14 million labelled images belonging to 1000 classes [Citation]. This model can be used to train a facial expression recognition model as shown by Gede Putra Kusuma *et. al* 2020 [Citation]. They used a fine-tuned VGG-16 model on the FER2013 dataset and achieving an accuracy of 69.40%, outperforming most standalone-based model results.

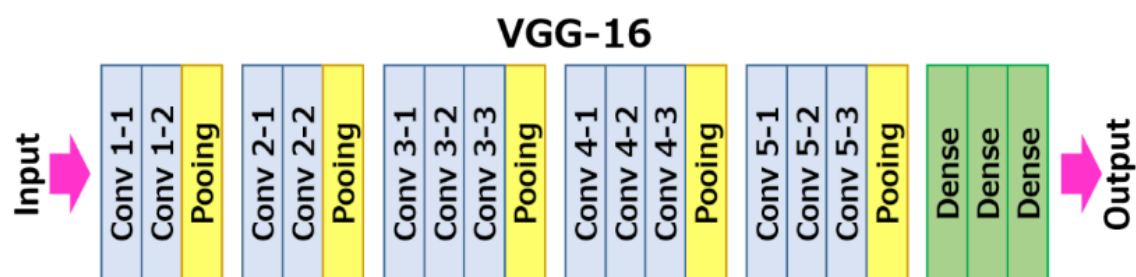


Figure 7 – VGG-16 Architecture

2.3.1.3 Support Vector Machines (SVM)

Support Vector Machines (SVMs) can be used for both regression and classification tasks. The algorithm works by finding a hyperplane in an N-Dimensional space where N represents the number of features. The hyperplanes separate the data points and the optimal hyperplane has a maximum margin between the data points of the classes. The accuracy of the algorithm depends on the hyperplane to separate the classes. In other words, the smaller the margin the more difficult it is to classify the data points as belonging more to one class than the others [\[Citation\]](#).

In the scenario that the data points are not linearly separable, the SVM kernel trick can be used. The concept behind the kernel trick is to map the dataset into a higher dimensional space which enables the algorithm to find a hyperplane that can separate the data [\[Citation\]](#) [\[Citation\]](#).

2.3.1.4 Bag of Visual Words (BOVW)

Bag of Visual Words is commonly used in image classification. The concept is similar to the Bag of Words (BOW) in natural language processing. In BOW the frequency of the words is obtained and can be displayed on a histogram. In BOVW, instead of counting the frequency of words in a given text, the image features are counted and can be displayed in a similar fashion using a histogram. Image features consist of a keypoint and a descriptor. Keypoints are segments of the image that stand out and this can be determined by the corners and edges in the image. These keypoints do not change even if the image is resized or rotated. The descriptor is the description of the keypoint.

The descriptors are clustered with a clustering algorithm. An example of a clustering algorithm is K-Means clustering and the center of each cluster is the visual “word”. A histogram is generated for each training images and to classify an image the features are first extracted, then plotted on a histogram and compared with the visual words’ histogram from the training set [\[Citation\]](#) [\[Citation\]](#).

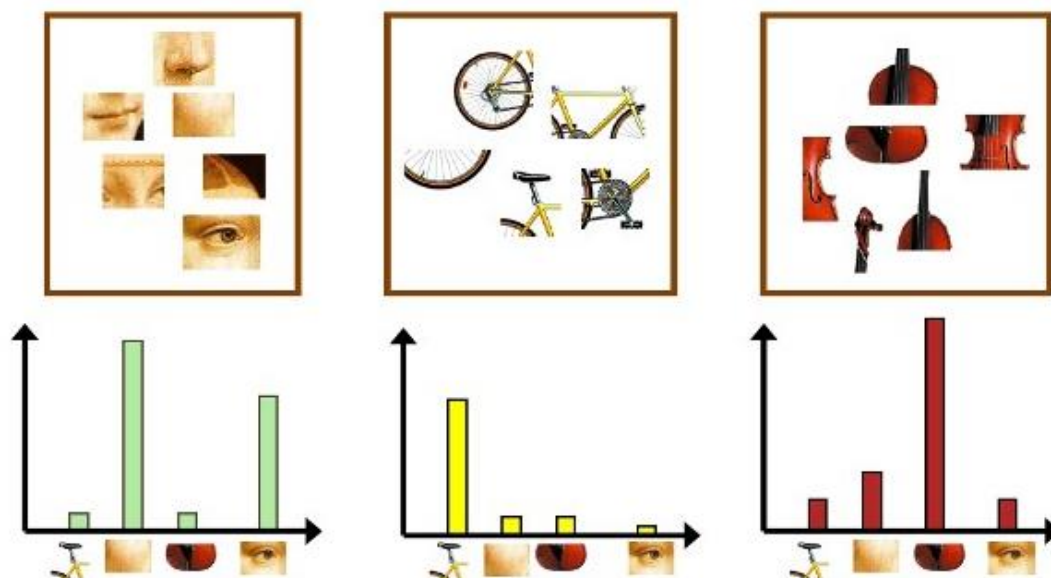


Figure 8 – Bag of Visual Words Histogram Example

2.3.2 Dataset Research

2.3.2.1 JAFFE (The Japanese Female Facial Expression (JAFFE))

The JAFFE dataset consist of labelled images of the head. The dataset consists of 10 people, all of whom are Japanese and female. There is a total of 213 grayscale 8-bit images and the image resolution is 256x256 pixels. The facial expressions are labelled with anger, disgust, fear happy, sad, surprise, and neutral. For a high accuracy model, a total of 213 images is not sufficient and data augmentation techniques will have to be used to generate more images from the existing set to increase the amount of data.

2.3.2.2 FER2013 (Facial Expression Recognition Dataset 2013)

The FER2013 data is in the form of a .CSV file and consists of 28,000 labelled images in the training set, 3,500 images in the development set and 3,500 images in the test set. The images are stored in the file as an array of pixel values. These can be used to reconstruct the image to a .jpg or .png format. The images are labelled with one of seven emotions which are happy, sad, angry, afraid, surprise, disgust and neutral. All images are grayscale and of 48x48 pixel resolution.

2.3.2.3 AffectNet

AffectNet is one of the largest datasets for facial expression recognition and consists of more than a 1 million images with approximately 440,000 of which are manually annotated. The publishers of the dataset are experiencing issues with bandwidth as a result of many downloads meaning the request for the dataset is delayed and a response is yet to be received. The sample images appear to be of the face only and the very large quantity of images would increase the accuracy of the FER model.

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

Figure 9 – AffectNet Dataset Info

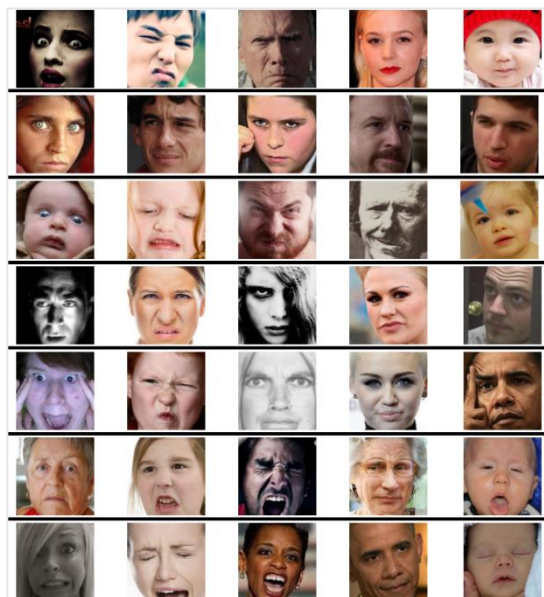


Figure 10 – AffectNet Sample Images

2.3.2.4 FacesDB

The FacesDB dataset consists of 38 subjects who are both male and female and each face expression is associated with a subject such as joy, sadness, fear, anger, disgust, surprise and a neutral face expression. This means that there are only 38 images for each face expression and these images are of the frontal point of view with side views of the face explicitly labelled. Due to the quantity of images the dataset images will have to be augmented to generate more.

2.3.2.5 FEC (Goole Facial Expression)

The FEC (Google Facial Expression Comparison dataset) is in the form of a .CSV file. The dataset consists of face image triplets with each row in the file contains three faces. It is specified which of the two images in the triplet contain the most similar face expression. The dataset was rated and annotated by at least six people. The dataset itself consists of approximately 500,000 triplets and approximately 156,000 face images. The images can involve people doing activities which is unlike some of the other datasets which only include the head. In the CSV file the image, four coordinates are provided that form a bounding rectangle that locates the face.

As the dataset is in the form of a .CSV file the images have be downloaded and cropped using the bounding rectangle coordinates. The process by hand is simply unfeasible and a GitHub repository under the name “FEC Dataset Downloader” was discovered for an application [\[14\]](#) that was said to download all the images from this particular dataset automatically. However, that application used another application under the name “TorCrawler” [\[15\]](#) that was in nature of a web crawler and utilized the TOR browser to rotate the IP address. The TorCrawler functionality was said to be necessary as the dataset image downloads apparently stall after a while if the IP address is not rotated. The TOR Crawler application included instructions that assumed the operating system used was UNIX which meant the instructions could not be followed as the operating system used is Windows 10.

A decision was made to write a program that will attempt to simply pull down the images without the changing of IP addresses which has been successful. However, after further inspection the images may not be entirely suitable to the usability testing scenario as the images of the faces are taken at many different angles which may reduce the accuracy of the model. In a usability testing scenario, the environment is not going to be “natural” in that the scene is not expected to constantly shift and change as it would if the person was in a bustling area. The camera will be in a static position and the angle of the participant in question is unlikely to greatly vary.



Figure 11 – Example of FEC Dataset Image Taken at Angle

2.3.2.6 Emotic (Emotions in Context)

The Emotic dataset is similar in nature to the FEC Google dataset except that it exclusively focuses on images of people in real environments. The images are annotated from a list of 26 discrete categories such as peace, sympathy, anger and surprise to name a few. The annotation are not simply for the face but also include the body. The angle at which the images are taken at and the fact that the focus is not only on the head might pose problems if used to train a facial expression recognition model. This dataset, as the name suggests, focuses on the context which is not a concern in a controlled usability testing environment.



Figure 12 – Emotic Sample Image

2.3.2.7 Conclusion

In conclusion, several facial expression recognition datasets have been researched and analysed. Certain datasets such as the FEC dataset and the Emotic dataset were not applicable in the usability testing environment. Other datasets such as JAFFE, FER2013 and FacesDB datasets were perfect for the training of the model in terms of the types of images. However, the JAFFE dataset and FacesDB dataset will require image augmentation to increase the dataset size. The AffectNet dataset is perfect in terms of both the types of images and the amount of images. However, as a result of these desirable features, the dataset is in high demand and the researchers who have published it are struggling with download bandwidth, meaning I could not acquire the dataset.

2.2. Alternative Existing Solutions to Your Problem

2.2.1 UserZoom Go

UserZoom Go [\[11\]](#) is a website usability testing application in the browser. It has a clean and easy to understand user interface. The researcher can create a “study” which is a usability test. There is an option to create an unmoderated or a moderated study and these studies can be conducted on own users or paid users can be selected based on a desired demographic based on age, gender etc. The application also provides support for both desktop and mobile applications.

The researcher can then create tasks and questions for the user to fill out and complete. These question answers include plain text, a multiple-choice selection and a rating selection. Questions may be asked before, during or after the study. Once the user begins the test, tasks will be displayed in a small window on screen and the user will click the “Complete” button whenever they are done with the task and move on to the next one. The user’s screen, audio and camera may be recorded depending on the usability test specifications. Once the test is over the data will be uploaded automatically and the researcher can view the recordings. The tasks are then reviewed and graded by the researcher with either a “Pass” or a “Fail”.

In conclusion, the website is very well designed and implemented. However, this comes at a starting cost of \$250 a month for a basic plan which allows only 1 researcher seat and 15 studies per year. The displaying of data could be greatly improved upon as there is no charts that intuitively display the data and no averaging of data is done to provide a general overview. Despite these shortcomings the application is visually appealing and provides the features that make it complete in many aspects.

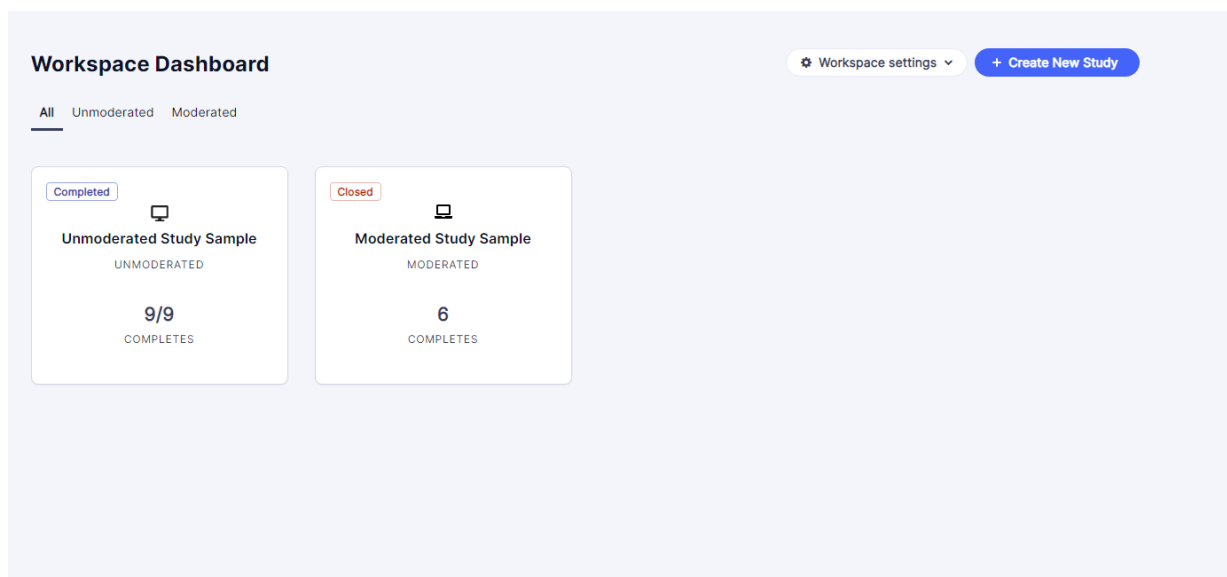


Figure 13 – UserZoom Go Dashboard

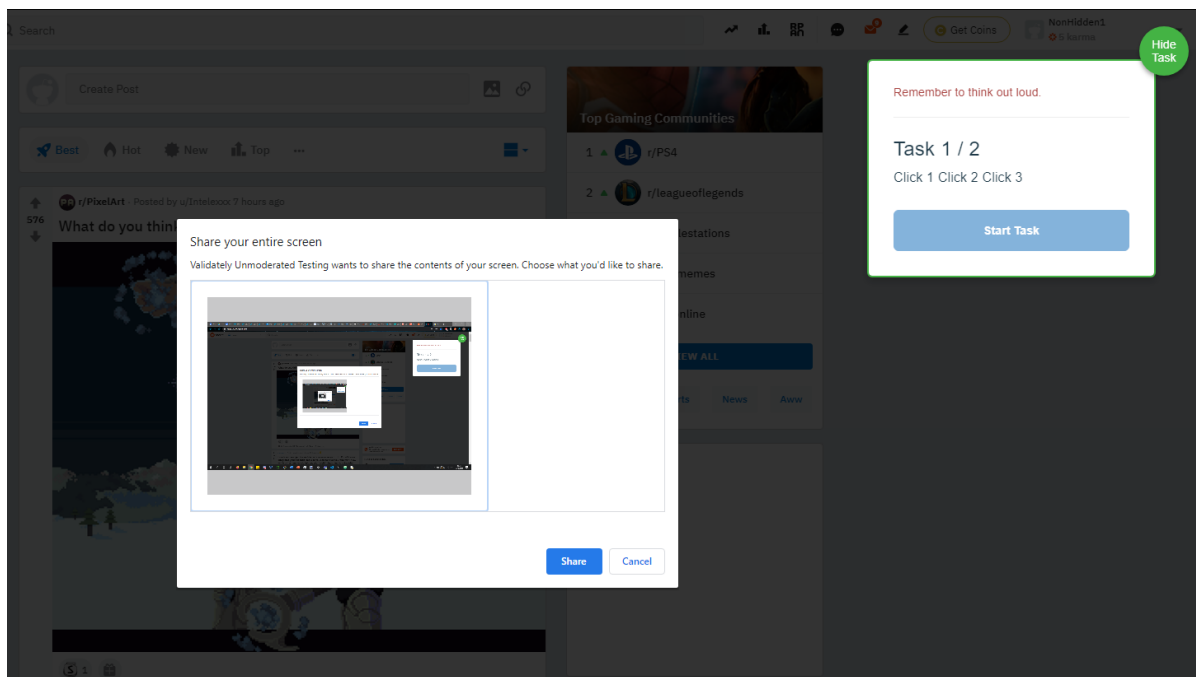


Figure 14 – UserZoom Go Usability Test Start

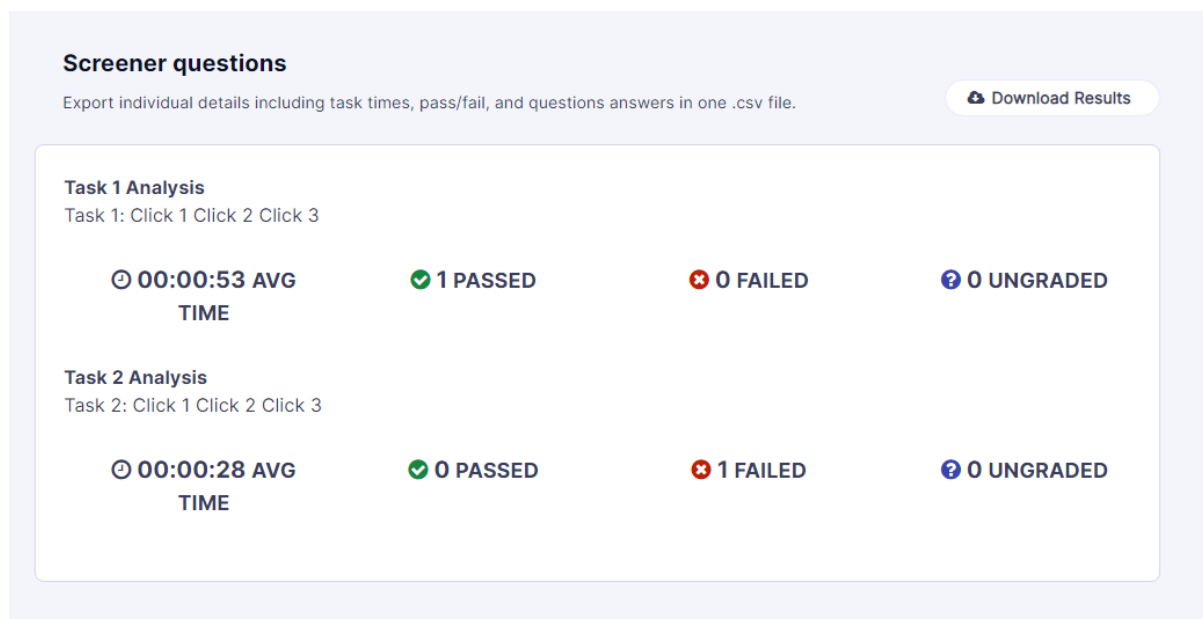


Figure 15 – UserZoom Go Usability Study Results

2.2.2 UserTesting

UserTesting [12] is a website usability testing application in the browser. The application that records the screen and the audio has to be downloaded. The user interface was a somewhat disorientating at first as there was a draft test that was already predefined and the button to create a test had a price tag of \$49 next to the button. The steps to be taken from the dashboard were not obvious as the initial thought process was to create a usability test and launch it. However, the website suggested launching a draft test and creating a new test appeared to cost money.

Creation of a usability test for free was not an option, however for research purposes the test creation steps were taken before the payment. The test options included selecting the audience by demographic such as age, gender, income, country, employee status etc. The participant device selection was between a computer, a tablet or a smartphone. A scenario could be written for the participant before the test begins and pre-test questions could also be defined. The tasks are written in text and a 5 second task can be set up where a user will be shown a screen for 5 seconds and asked to describe what they saw. A verbal response question can be asked and some question options such as a multiple-choice question, a rating scale and a written response require a premium subscription. There is an option to notify team members of the test results via Slack or email.

The website provided a usability test dry run from the participants point of view before a usability test is launched. This provided more insight into usability testing; however, the data was not recorded as it was simply a dry run and seeing how the data is presented after a participant has completed a test was not possible without payment. In conclusion, the usability testing website provided a lot of options, some of which seemed unnecessary. For example, filtering the demographic by “Parental status”. Viewing the results of a test was impossible without payment despite the free trial. Certain parts of the website were disorientating.

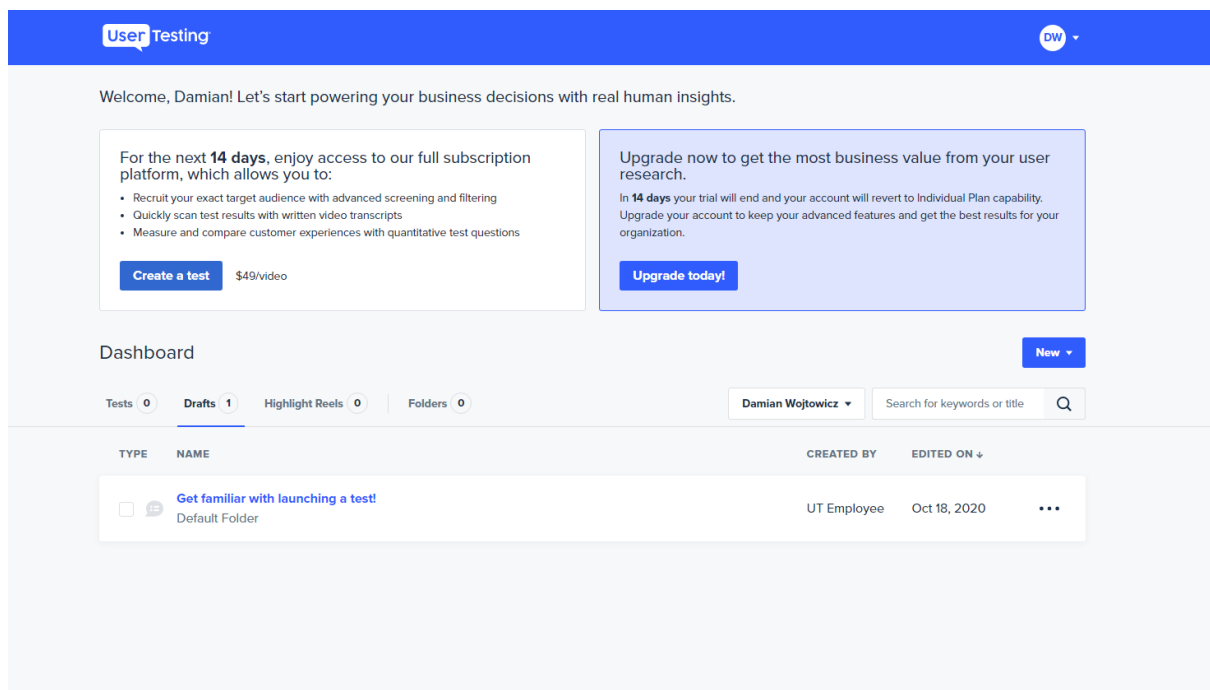


Figure 16 – UserTesting Dashboard

UserTesting

DW

Untitled Audience

General Settings

How many participants do you need?

5

Which type of device should the participants use?

☒ Computer
 ☐ Tablet
 ☐ Smartphone

Filters

Age Remove

18 65+

Household income (\$) Remove

0K 150K+

Gender Remove

☐ Male
 ☐ Female

Web expertise Remove

☐ Average web user
 ☐ Advanced web user

Panel Options

☒ UserTesting panel

Filters

☒ Age
 ☒ Household income (\$)

☒ Gender
 ☐ Countries (Defaults to all)

☒ Web expertise
 ☐ Test Frequency

☒ Prior Studies

Demographic Filters Pro

☐ Employment status
 ☐ Industry

☐ Company size
 ☐ Job role

☐ Job level
 ☐ Social networks

☐ Language
 ☐ Parental status

☐ Other requirements ?
 ☐ Operating system

☐ Web browsers

To target your exact participants, select filters and add screener questions.

Done

Figure 17 – UserTesting Select Audience

Test Plan

Enable Participant View

Record participants' faces during the test. This requires participants to use Chrome or a mobile device.

☐

1 Verbal Response Duplicate Delete +

Without leaving the homepage, what are your initial Impressions of the website? Explain your answer.

Characters left: 900

2 Task Duplicate Delete +

Think of something that you might do on this website and describe it out loud. When you've decided, move on to the next task.

3 Task Duplicate Delete +

Take up to 2 minutes to complete the task you just described. Move on to the next task when you're done.

4 Rating Scale Task Duplicate Delete +

Overall this task was...? Explain your answer.

5 Rating Scale Task Duplicate Delete +

How not confident (1) or confident (7) are you that you completed the task successfully? Explain your answer.

6 Rating Scale Task Duplicate Delete +

How difficult (1) or easy (7) was it to understand the information on the

Balanced Comparison ? Pro ☐

Assets

☐ URL
 ☐ Image

Tasks and Questions

☐ Task
 ☐ Five Second Test
 ☐ Verbal Response

Tasks and Questions Pro

☐ Multiple Choice
 ☐ Rating Scale
 ☐ Written Response

Popular Tasks

[View Examples >](#)

Drag and drop tasks, questions, and assets to build your test plan.

Preview Test Plan Done

Figure 18 – UserTesting Create Usability Test

2.2.3 Loop11

Loop11 [13] is a website usability testing application in the browser. The researcher can create a new “project” which is a usability test/study. Statistics and other information about the general performance is given in the project overview and pie charts display the number of participants that have successfully completed the tasks, failed or abandoned the study. Information on the total number of participants and the average duration of the study is displayed. More statistics on the participants is displayed such as their location, operating system, device and browser.

The researcher can view the project in terms of the tasks, the videos, questions and participants. A tab is given for each of the mentioned. This allows for easy navigation which is definitely desirable. The video of the participants completing the tasks and questions can be viewed. All the data on each participant can be displayed. A clickstream & heatmap can be displayed which shows how users progressed through the tasks in terms of the links they clicked on.

The tests can be moderated or unmoderated and there are options to enable or disable screen and webcam recording for the test. Tasks are written in plain text with the option to customize the text with fonts and font sizes. There is a wide selection of questions that can be selected from ask the participant. These include a multiple-choice question with one answer, a rating scale matrix, a ranking question and a multiple choice (multiple answers) question just to name a few. There are options to randomize the order of answers in a given question when applicable e.g. multiple-choice question.

In conclusion, this application has a modern design that is appealing to the eye and easy to use. Navigation is easy and the data is displayed in a presentable manner which includes charts. The clickstream is an interesting idea of tracking how all the users progressed in the task. An issue with the website is that the video quality was very low and would not load very fast.

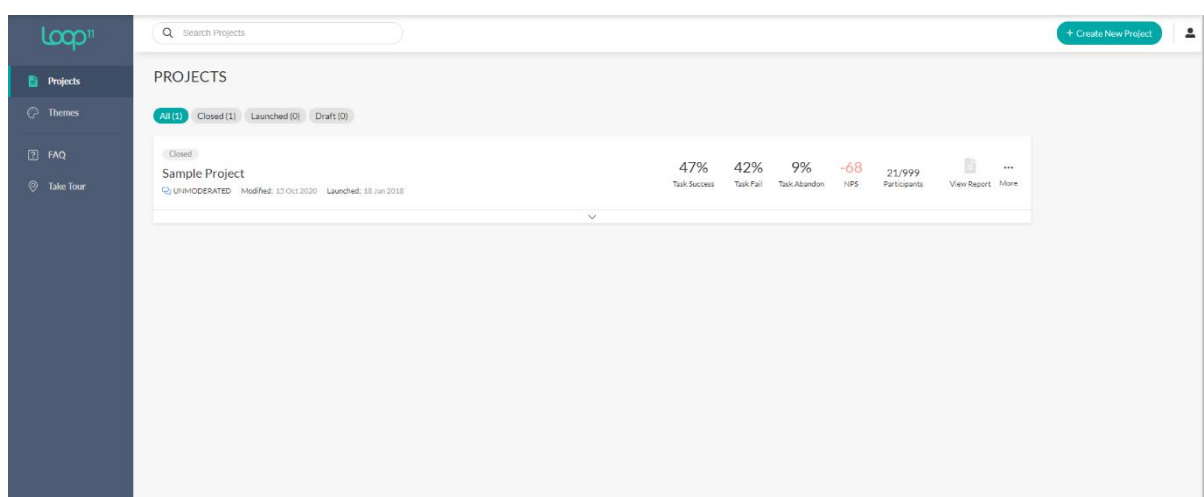


Figure 19 – Loop11 Main Dashboard

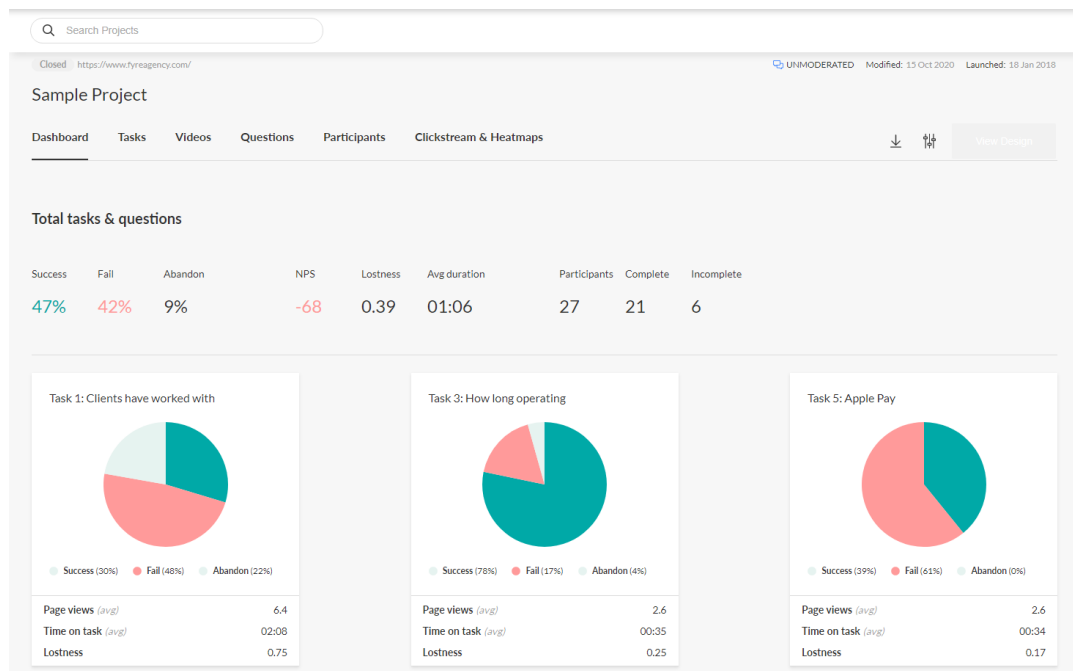


Figure 20 – Loop11 Project Dashboard

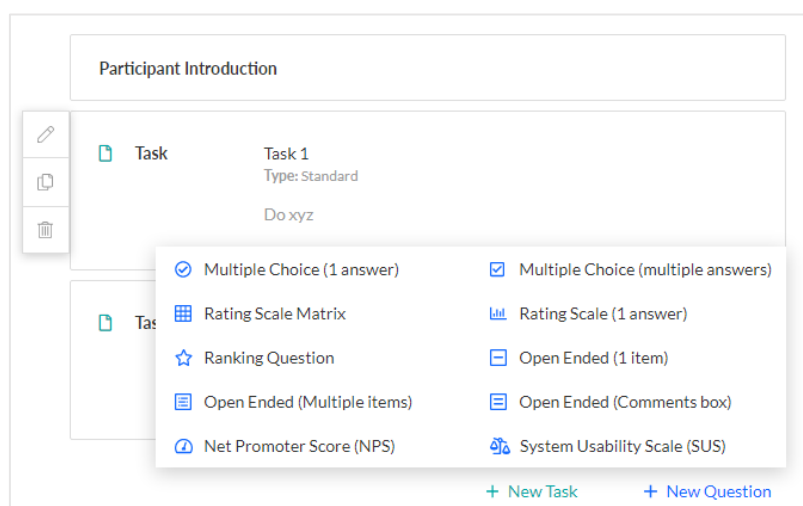


Figure 20 – Loop11 Tasks and Questions

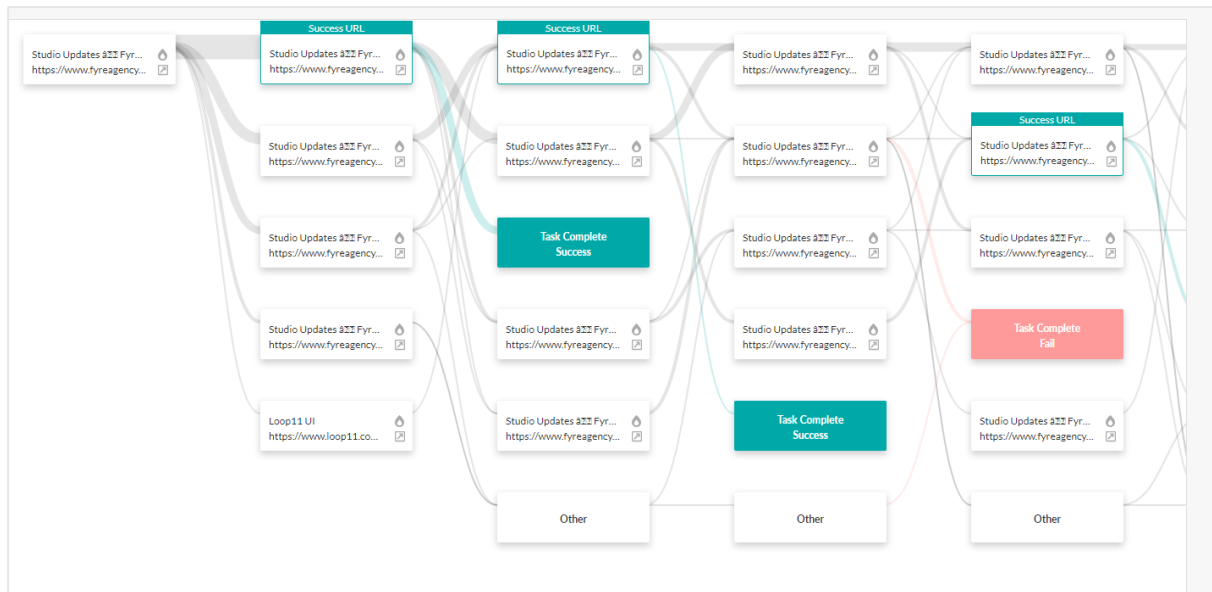


Figure 21 – Loop11 Clickstream

Participant Demographics	
Location	
Mount Prospect, Illinois, United States of America	11.11% (3)
San Jose, California, United States of America	7.41% (2)
Levittown, Pennsylvania, United States of America	7.41% (2)
Chattanooga, Tennessee, United States of America	3.70% (1)
La Crescenta, California, United States of America	3.70% (1)
Others	66.67% (18)
Operating System	
Windows 10	74.07% (20)
Mac OSX	14.81% (4)
iPad	7.41% (2)
Windows 8.1	3.70% (1)
Device	
Desktop	92.59% (25)
Tablet	7.41% (2)
Browser	
Chrome	92.59% (25)
Loop11 App	7.41% (2)

Figure 22 – Loop11 Participant Demographic

2.3. Technologies Research

2.3.3 Programming Languages

2.3.3.1 Python

Python [\[Citation\]](#) is a high-level, interpreted, object-orientated and general-purpose programming language. Python supports a vast amount of libraries which includes machine learning libraries such as TensorFlow, Keras and NumPy, to name a few. The language is free, open source and cross platform. The programming language is popular and finding support online to questions is easy which greatly helps with the development process. The popularity of the language with regards to machine learning and data visualization makes it a very strong contender as a language of choice for machine learning. [\[Citation\]](#)

Django [\[Citation\]](#) is an example of a high-level Python Web framework which allows for web development with Python. Django is integrated with JavaScript and HTML which means JavaScript code and functionality can be used in Django.

2.3.3.2 Java

Java [\[Citation\]](#) is a class-based, object-oriented, platform-independent language. Java can be used to develop the server side of web applications. Java is the no. 1 choice for developers with over 9 million Java developers worldwide [\[Citation\]](#). The popularity of the language and its use in web development makes it an excellent option for developing the UsabCheck web application. It is a programming language taught in the TU Dublin Computer Science course which makes it familiar and easier to work with.

2.3.3.3 JavaScript

JavaScript [\[Citation\]](#) is an interpreted programming language that allows developers to implement complex and dynamic features on web pages. JavaScript compiles the code at run time with the use of a technique called *just-in-time compiling*. JavaScript can run locally on the machine of the user viewing the website and can be used for interaction with the back-end server via a Restful API. It can also run on a server. There is a back-end and front-end framework named Node.js [\[Citation\]](#) and there are front-end frameworks such as AngularJS [\[Citation\]](#) and ReactJS [\[Citation\]](#) if the design plan will be to have a Java back-end as an example.

2.3.4 Python Libraries

NumPy [\[Citation\]](#) is an open source library used for working with numbers, arrays, matrices, linear algebra etc. Images are simply matrices of pixel values and NumPy can be used to alter and manipulate these images.

Pandas (Python Data Analysis) library [\[Citation\]](#) provides functionality that can extract data out of a CSV file as an example and create data frames that are similar in structure to an Excel file. This library can work with NumPy to manipulate the data.

Matplotlib [\[Citation\]](#) is a library that generates charts and other forms of data visualization in Python. These figures can be interactive and allow to user to zoom in and pan etc. This library can be

used to display images in Jupyter Notebook and the results of machine learning training in the forms of a confusion matrix as an example.

Scikit-learn [\[Citation\]](#) is a machine learning library that includes other libraries such as NumPy, SciPy and Matplotlib. It is a tool for predictive data analysis and includes implementation of algorithms such as SVM, Nearest Neighbour, Random Forest, K-Means etc.

TensorFlow [\[Citation\]](#) makes it easy for developers to create and train deep neural network models for desktop, mobile, web etc. This library has many applications which include sound recognition, text-based applications, image recognition and video detection [\[Citation\]](#) .

Keras API [\[Citation\]](#) is fully integrated with TensorFlow and serves as a wrapper for TensorFlow and Theano, providing a simple and intuitive interface for the machine learning libraries.

FastAI [\[Citation\]](#) is another deep learning library that provides its users with high-level components and it is GPU optimised just as the other libraries mentioned. An article [\[Citation\]](#) detailed the first impressions of the FastAI library and impression was that it is a good library to use however it comes with some slight learning curve drawbacks. The wiser decision would be to use TensorFlow and Keras to train the deep learning model as the two libraries are more popular and there is more online support as a result.

OpenCV [\[Citation\]](#) is a computer vision, machine learning and image processing library. It can be used to manipulate images which includes resizing, rotating, detecting edges, converting the image to different colour spaces etc. This library can be used in a hybrid approach whereby features are extracted with OpenCV and used to train a machine learning algorithm.

2.4. Usability Testing Technologies

2.4.1 Usability Testing

As previously explained the idea for the usability testing is to create a web application that will store the data and allow researchers to configure the usability tests. For this goal to be achieved video uploading and streaming has to be implemented. The website has to be hosted and a database will have to be selected.

2.4.2 Video Uploading

Several video streaming services have been researched to find the most convenient method of uploading and streaming the videos reliably.

2.4.2.1 SwiftStack

SwiftStack [\[Citation\]](#) is a data storage and management platform for data-intensive applications. It is easily scalable and supports HTTP Range requests which means that pseudo-streaming is supported. Pseudo-streaming involves downloading the file and playing that file as it is being downloaded. This is how YouTube streams their videos.

2.4.2.2 StreamingVideoProvider

StreamingVideoProvider (SVP) [\[Citation\]](#) provides video streaming and other services. The videos uploaded can be embedded in the *UsabCheck* application. SVP provides video tutorials for how to use their services and a developer's API for cURL, Java, PHP, Ruby, Python and JavaScript. The videos can be managed and statistics can be provided. The service provides more functionality than is needed for the *UsabCheck* application. However, this is not a problem.

2.4.2.3 Vimeo

Vimeo [\[Citation\]](#) is an online video streaming site that allows users to upload their videos and these videos can be shared and embedded on websites. Videos can also be uploaded to Vimeo much the same as YouTube.

2.4.2.4 YouTube

YouTube is a video streaming service and videos can be uploaded and the videos can be configured in such a way that only people with the link may view the video. YouTube also provides an API that allows users to upload the video. However, YouTube sets a restriction on videos uploaded via an API and only a few videos can be uploaded each day. To increase the amount of videos that can be uploaded with the API a quota has to be requested.

2.4.3 Website Hosting

2.4.3.1 Tomcat

Apache Tomcat [\[Citation\]](#) is a lightweight application server designed to execute Java servlets and render web pages that use Java Server page coding. It provides a relatively quick load and redeploy times. Apache Tomcat is most widely used Java application server and it is well documented as it has been around for almost 20 years [\[Citation\]](#).

2.4.3.2 Google Firebase

Google Firebase [\[Citation\]](#) is a Backend-as-a-Service application development software that enables developers to develop iOS, Android and Web apps. It is the server, the API and the datastore all in one. However, the database, the Firebase Realtime Database and Firestore are non-relational databases. This means that storing and retrieving data is stored in documents and collections unlike a SQL database. This makes is very fast and efficient, however, not all applications are suited to this type of storage and this will be further discussed in the design section.

2.5. Existing Final Year Projects

Existing projects which involved usability testing have not been found. However, there have been many projects that have used machine learning. One such example is a project by Brendan Tierney, titled “Detecting Bot Twitter Accounts using Machine Learning”. The purpose of the project was to use machine learning to evaluate whether or not a Twitter account is a bot. The complexity of the project stemmed from the machine learning aspect which involved the use of multiple machine learning models as a way to more accurately a bot account. There was a model for classifying based on the user data, the tweets of the user, the sentiment and timing. Another form of complexity came from testing various machine learning algorithms to see which one performs best. The project also faced challenges such as the accuracy of the dataset and the speed of the application.

The project technologies included the Django framework, JavaScript, Bootstrap, HTML, and CSS. MySQL database and the Twitter API. Vast amounts of research have been done into each area. The challenges and problems were identified, and a solution was proposed. The project implements features that the existing applications lack which was detecting the followers of an account. The weakness of the project was that the follows of a user were not processed and identified fast enough in some cases because of the constraint imposed by the Twitter API.

2.6. Conclusions

The table below displays the technologies chosen and the reason for the decision.

Technology Chosen	Version	Summary and Application of Technology
Python	3.7.3	Python will be used in the local application usability testing application and for training machine learning model. This is because Python offers a wide range of libraries for machine learning.
Java	11.0.3	Java will be used in the web application's back-end. It is one of the most popular languages for a web server backend.
JavaScript	ES6	JavaScript will be used for the web application's front-end. It is compatible with the Java backend and very useful in front-end development.
ReactJS	17.0.1	ReactJS is a JavaScript library for the front-end of the web application. It has a large community and as a result there is online support and the UI tools are developed by the community.
Maven	3.6.3	Maven will be used in the management of the Java back-end. It is a very useful tool for managing the libraries and dependencies and compiles the project into a single file that can be deployed.
Spring	5.3.1	Spring is a framework for dependency injection and will be used in the Java back-end of the web application. It is used to build decoupled systems.
NumPy	1.17.3	A python library for working with numbers and will be used in building the FER model.
Pandas	1.1.2	A python library for extracting data out of spreadsheets. Can be used in working with the dataset.
Matplotlib	3.1.1	A Python library for plotting charts. Used in the evaluation of the FER model.
Scikit-learn	0.23.2	A machine learning Python library. It is a combination of other libraries and can be used in evaluating the FER model etc.
TensorFlow	2.3.0	A high-level machine learning Python library. Will be used via the Keras API.
Keras	2.4.3	The Keras API is what will be mainly used to build the FER model. It provides an easy and intuitive interface to the TensorFlow library.
OpenCV	4.4.0.44	A Python library with functionality to manipulate images. It will be used when working with the FER datasets.
MySQL	8.0	A relation database management system that the back-end web application will connect to. It is chosen over a non-relation database such as Firestore because the database follows a hierarchical structure.

Tomcat	10.0	A tomcat server will be used to host the Java back-end application. Apache Tomcat provides a relatively quick load and redeploy.
Vimeo	_____	Vimeo is a video uploading platform much like YouTube. It will be used for uploading the usability testing recordings. It was chosen over the other options as it has a simple API and the service is offered at an acceptable price.

The table below outlines the decisions made with regards to the machine learning aspect of the project.

Name	Type	Summary and Application
FER2013	Dataset	It is a large dataset which will increase the accuracy of the machine learning model. All the images are of the front of the face making it very suitable.
JAFPE	Dataset	This dataset is of the front of the face. However, the size of the dataset is not large enough and data augmentation techniques will used. This dataset will be used for comparison with the FER2013 and FacesDB datasets.
FacesDB	Dataset	This dataset is of the front of the face. Once again, the size of the dataset is not large enough and data augmentation techniques will used. This dataset will be used for comparison with the FER2013 and JAFPE datasets.
VGG16	ML Architecture	An award-winning CNN model used in classifying images. It is an excellent model for FER. Another option was ResNet-50 which is an almost equally great model.

3. Prototype Design

3.1 Introduction

There are two main aspects to this project. One aspect is the facial expression recognition (FER) machine learning model which involves processing of data, training a model etc. and the other aspect is the usability testing which processes the output data of the FER model. The software methodology will be discussed first, followed by an overview of the system and then a section will be allocated for each of the aspects mentioned.

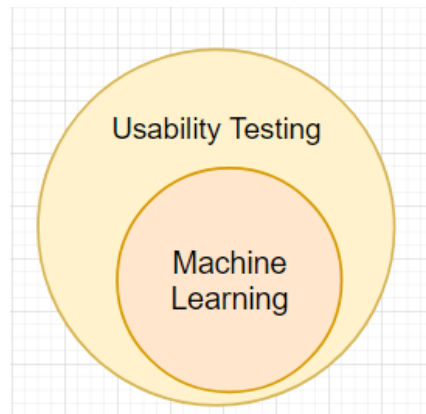


Figure 23 – Usability Testing & Machine Learning Subprojects

3.2 Software & Data Mining Methodologies

Due to the nature of the project two methodologies were used to manage this project. The first one is the Scrum Agile methodology and it is used to manage the more traditional software side of the project which is the web application and the local python application. The CRISP-DM methodology will be used to manage the data and machine side of the project that involves building a FER model. The CRISP-DM methodology is integrated into the Agile Scrum methodology and this will all be explained in this section of the report.

Scrum Agile

The software methodology that has been chosen for this project was the Scrum Agile methodology [\[Citation\]](#). The Agile methodology is centered around iterative development and allows teams to deliver work with greater predictability and allows for better adaptability to change. In Scrum specifically, the development is split into development cycles called Sprints which are at least no more than four weeks. The project's high-level requirements and goals are defined in the Product Backlog is a dynamic list of Product Backlog Items (BPIs). This Product Backlog can change depending on the requirements as the project develops. These BPIs are used as input to the Sprint Backlog, which is a list of tasks, user stories, bug fixes etc. which need to be completed by the end of the sprint. At the end of each sprint the team reviews and evaluates the work. The Scrum methodology also involves "Increments" which is the usable end-product from a Sprint [\[Citation\]](#).

Although this methodology is most commonly applied in a scenario with a team, the methodology itself provides a very useful process for managing a project with only a single individual. The product quality is improved as a result of breaking down the project requirements into manageable pieces

and the focus is on the user with the project being able to adapt to the changing requirements. Prioritizing and reviewing the tasks increases the productivity of the developer as they can easily track the development and readjust if necessary [Citation]. ZenHub is an Agile project management tool that is natively integrated into GitHub and it will be used to manage the Sprints, BPIs, Epics and sprints etc. Figure 3.2.2 below illustrates what the management of Epics and Sprints looks like. All the requirements/issues are in the “Product Backlog” and gradually moved to the “Sprint Backlog” with each coming Spring. Requirements that have been completed are moved to the “Done” section. Once they are reviewed and quality assessed they are officially finished and the issue is closed. Figure 3.2.3 shows a more detailed information on an Epic. For example, the Facial Expression Recognition Epic has 8 issues and 3 of them have already been complete.

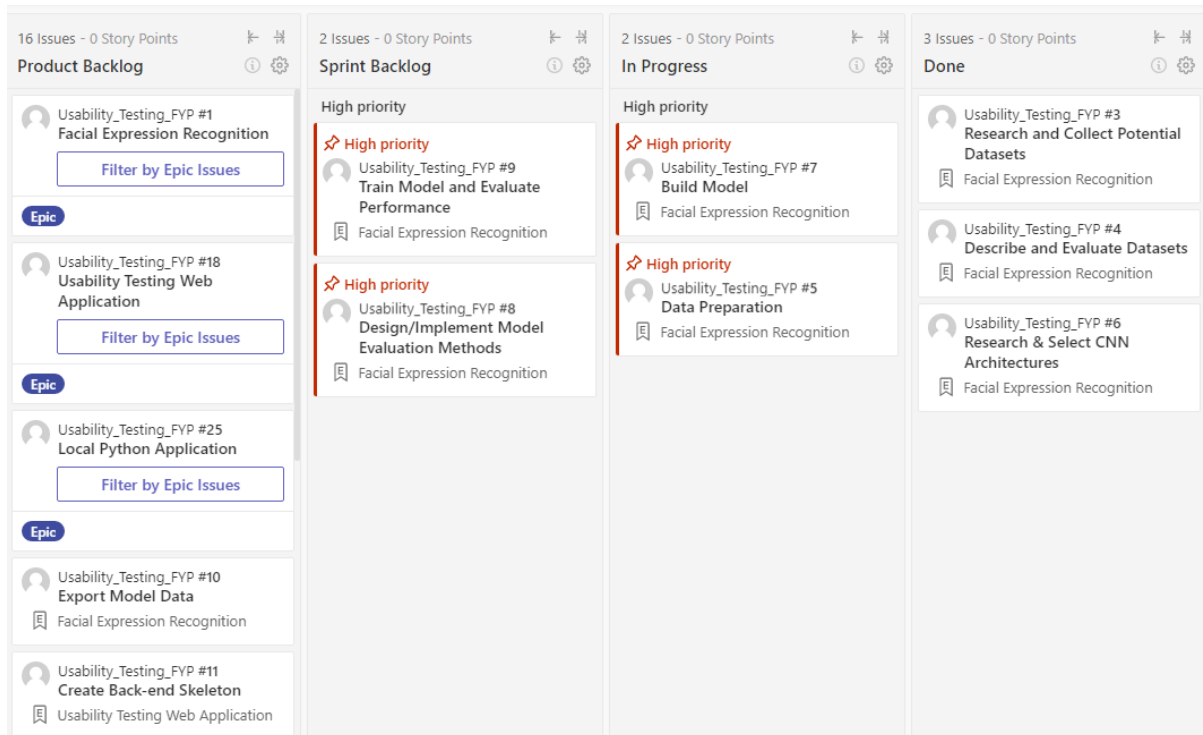


Figure 24 – ZenHub Epic and Sprint Planning

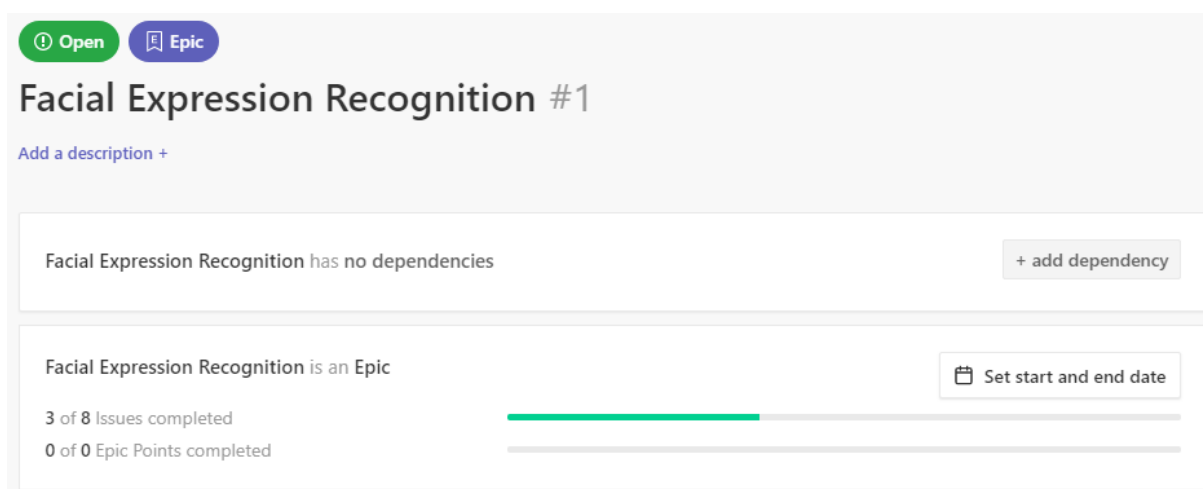


Figure 25 – ZenHub Facial Expression Recognition Epic

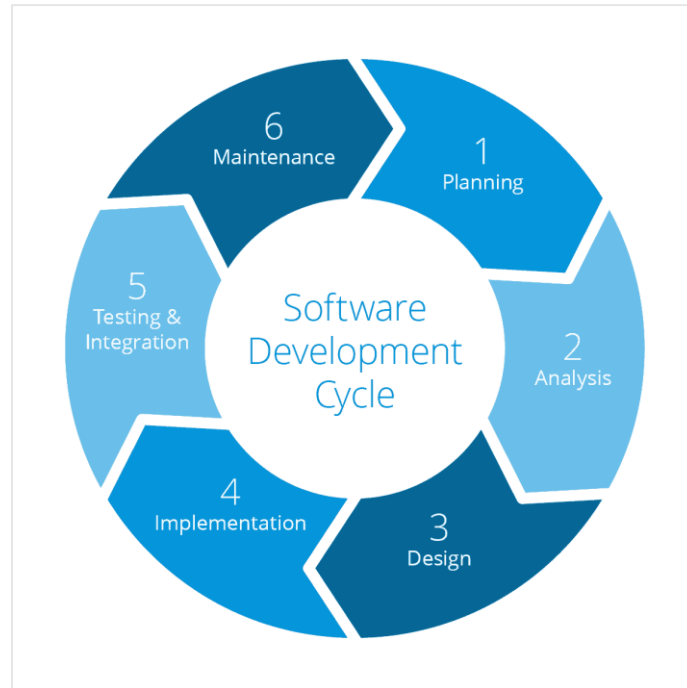


Figure 26 – Scrum Agile Software Development Cycle

CRISP-DM

The CRISP-DM (cross-industry process for data mining) methodology was used in the design and development of the facial expression recognition model. This methodology is used in the data mining process and much like agile it focuses on iterative progress. The cyclical iterative process consists of 6 stages.

Stage 1: Business Understanding

Understand the business objectives and the goal of this stage is to unveil factors that could affect the outcome of the project. This involves looking at the resources, constraints, risks and requirements, and then forming a plan.

Stage 2: Data Understanding

This stage involves creating an initial data collection report that lists the data sources that have been acquired and where they have been obtained from. The second stage also involves the creation of a data description report that outlines the format, quantity and other relevant features of the data.

Stage 3: Data Preparation

This stage involves analysis, selecting, cleaning, constructing and combining the data.

Stage 4: Modelling

Involves selecting the modelling technique, generating a test design for how the model will be evaluated, building the model, and finally, assessing the model.

Stage 5: Evaluation

This stage involves reviewing the models and determining the future steps that will need to be taken to ensure the requirements are satisfied.

Stage 6: Deployment

The final stage where a deployment plan, a monitoring plan and maintenance plan is created

[Citation] [Citation].

This methodology is needed as data science is different in nature compared to software development in that significant time is spent on experimentation. CRISP-DM methodology is integrated into Scrum Agile by creating backlog items that are data science specific.

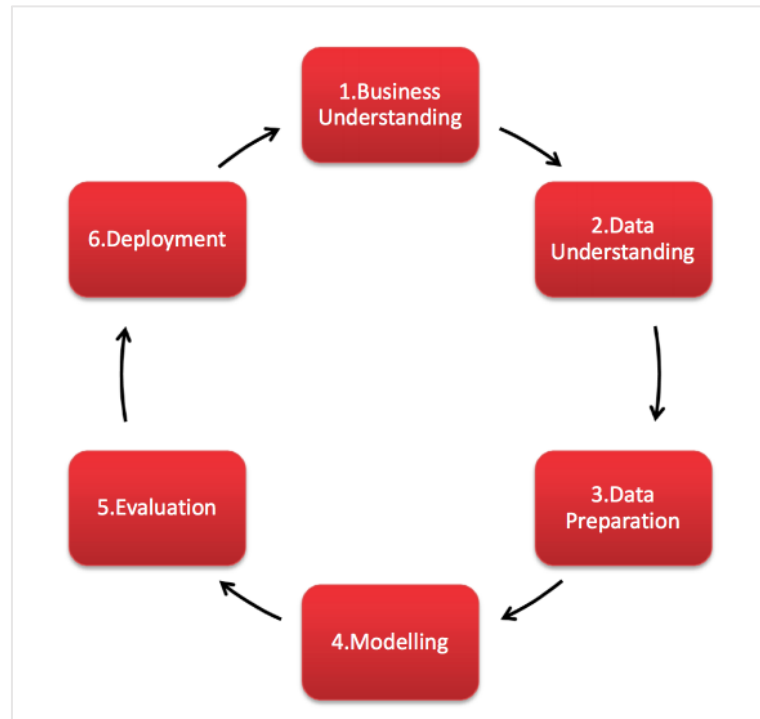


Figure 27 – CRISP-DM Methodology Cycle

3.3 Overview of System

The project is divided into two significant subprojects. These two subprojects are the facial expression recognition machine learning model and the usability testing applications. Each have their own set of requirements and can be thought of as separate projects that are later integrated. Figure 3.3.1 illustrates the high-level steps that are taken in each subproject, showing how different each is. The machine learning subproject involves working with data and training a neural network. The usability subproject is involving a web application to configure the usability tests and displaying the results/data. The local application runs the machine learning model and runs the usability test.

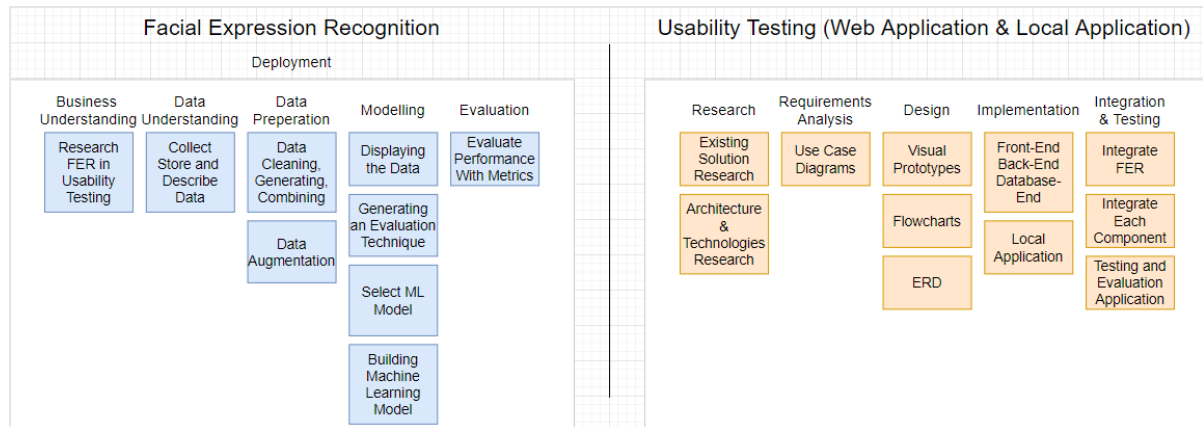


Figure 28 – Subprojects' High-Level Steps

This project involves two applications. One application runs locally on the participants computer where the usability test will be conducted. Another application is the web application which has a front-end, back-end and database. There is another very important part to this project which is the facial expression recognition model.

Below is a very high-level view of the system that shows the data from the webcam being fed into the machine learning model that makes a prediction of the facial expression. The data is processed in the local application before it is uploaded to the web application where it is then stored and visualised for the researchers. Other data is also sent; however, the diagram is designed to specifically show how the machine learning aspects fits into the system.

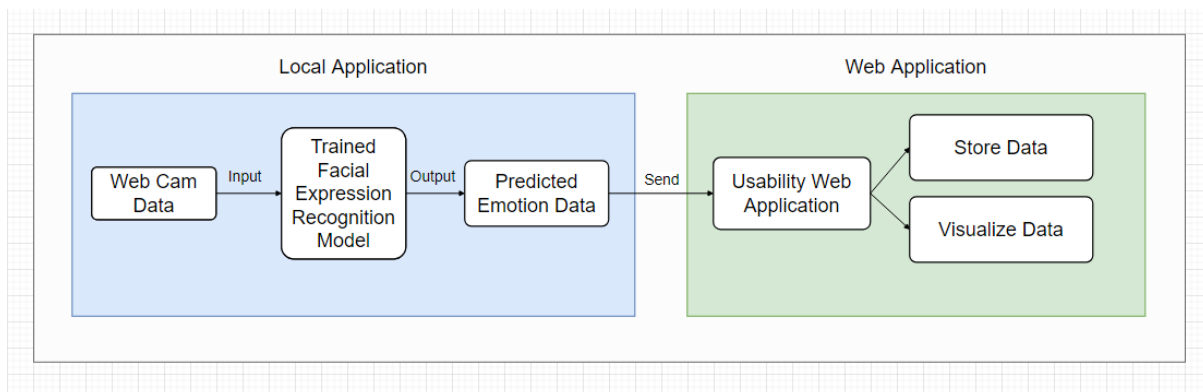


Figure 29 – High Level View of System

Figure 3.3.1 illustrates a more in-depth view of the architecture of the web application and the local Python application. The web application is a 3-tier architecture with a client, server and database. The front-end of the web application will be programmed in JavaScript and the ReactJS library is going to be used to aid development. The other option researched was the Angular Framework for JavaScript and both Angular and React are great options for front-end web development. Neither of the two have been worked with before and React was chosen as it is said to have an easier learning curve and is the more popular option with a bigger community. From a more objective point of view the UI tools are developed by the community and there is a wide range of options of libraries that provide charting UI which is very necessary to display the data in this project.

The back-end of the website will be programmed in Java with the Spring framework, managed with Apache Maven and hosted on an Apache Tomcat server. The reason for this decision was mainly personal as it is something familiar and will lead to less mistakes in development. Additionally, the research has also concluded that this is one most popular options for setting up a back-end for a web application in Java. The reason Java was chosen over Python and Django is that some sources claim that Django has a steep learning curve and it would not be wise to attempt learn an entirely new framework given the time constraints. It would simply be an unnecessary risk to the project given that there is familiarity with a perfectly suitable solution in Java. For the database tier MySQL was chosen.

The local python application connects to the back-end via the REST API and the same goes for the front-end of the web application. The data is processed in the service tier and passed to or received from the DAO classes that connect to the database. The FER predictions by the model are made locally on the participants computer.

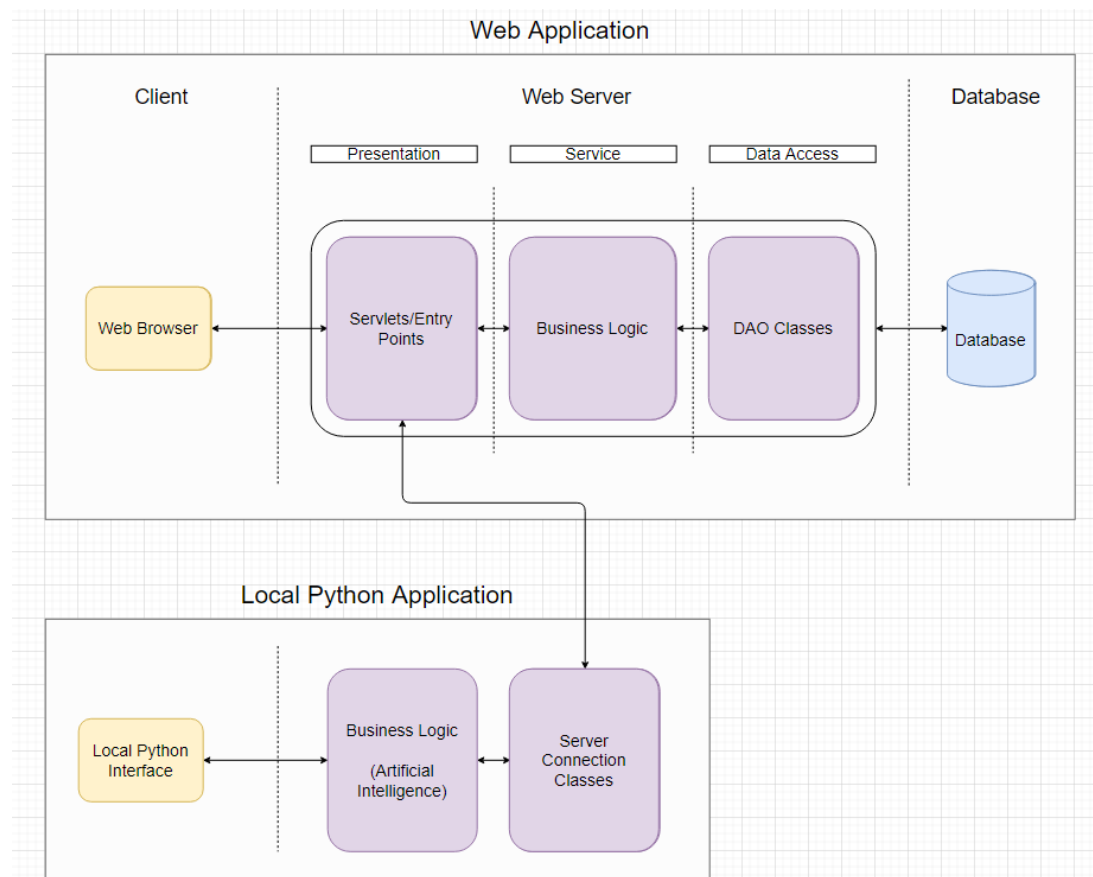


Figure 30 – Architecture of Applications

Figure 3.3.3 illustrates the high level operations of the system which includes the creation of a usability test and the usability test being conducted with a participant. This diagram does not go into low level detail when it is not necessary and is mainly used to show how the components fit together and how the program flows to help increase the understanding of the system. This diagram also does not show the researcher viewing the results of the usability test as this will be explained in detail with the help of visual prototypes of the user interface.

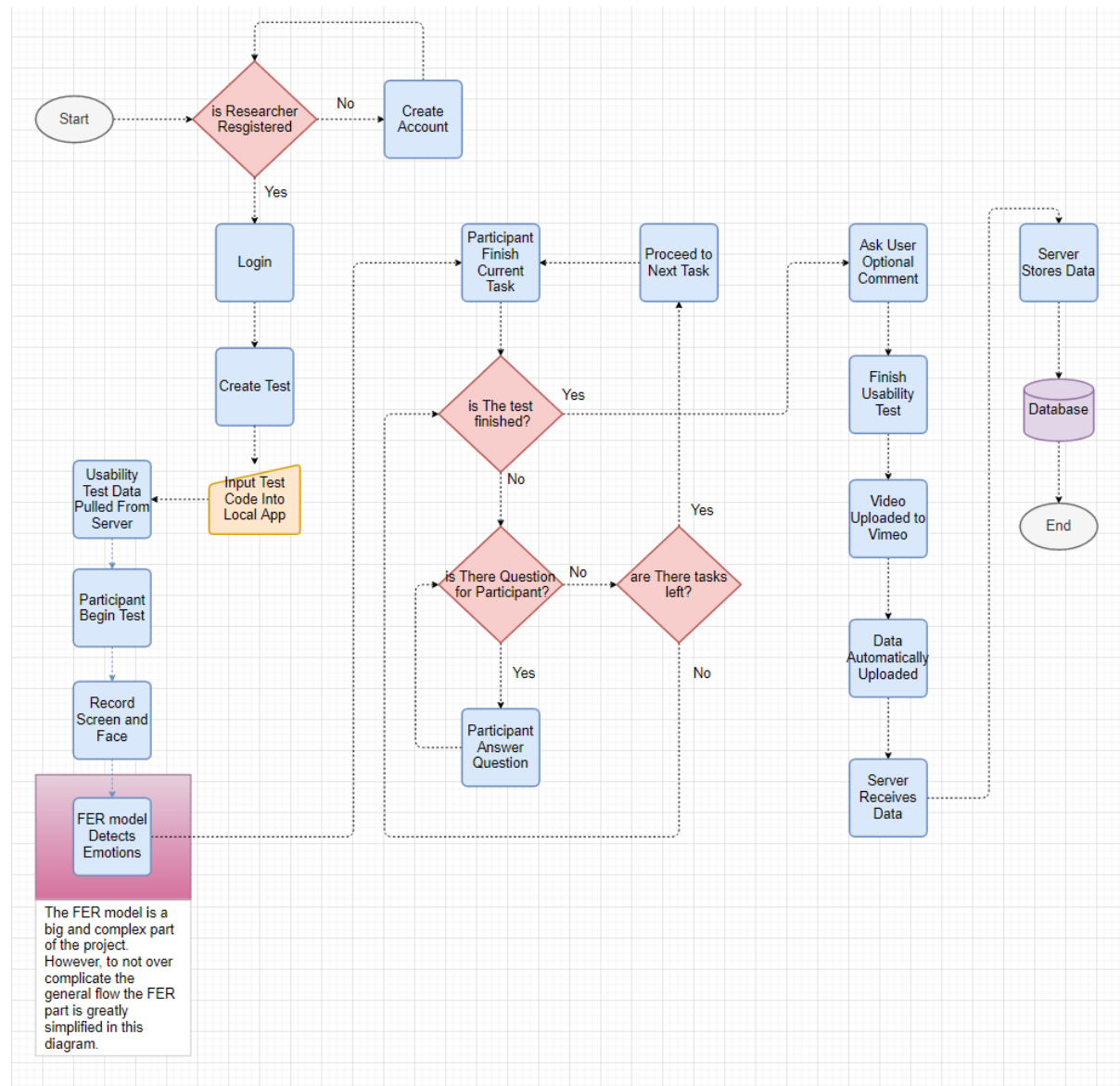


Figure 31 – High Level Flowchart of System

3.4. Web Application

3.4.1 Front-End

Use Case Diagrams

The first iteration of the developer/researcher use case diagram is illustrated below in Figure 32. There is register and login functionality so that each developer/researcher has their own separate account. The researcher can create usability tests which include creating tasks and questionnaires for the participant to complete. After the usability test is conducted the researcher can view the average success and failure rate of the tasks within the tests. For example, the test is conducted twice, the test has 5 tasks, the 1st participants failed one task and the 2nd participant failed two tasks then the success rate would be 7/10 as three tasks were failed amongst the two participants out of a total of 10 tasks.

The developer/researcher can watch the video of the screen recording and the camera recording. The researcher can view what emotions occurred when on a timeline.

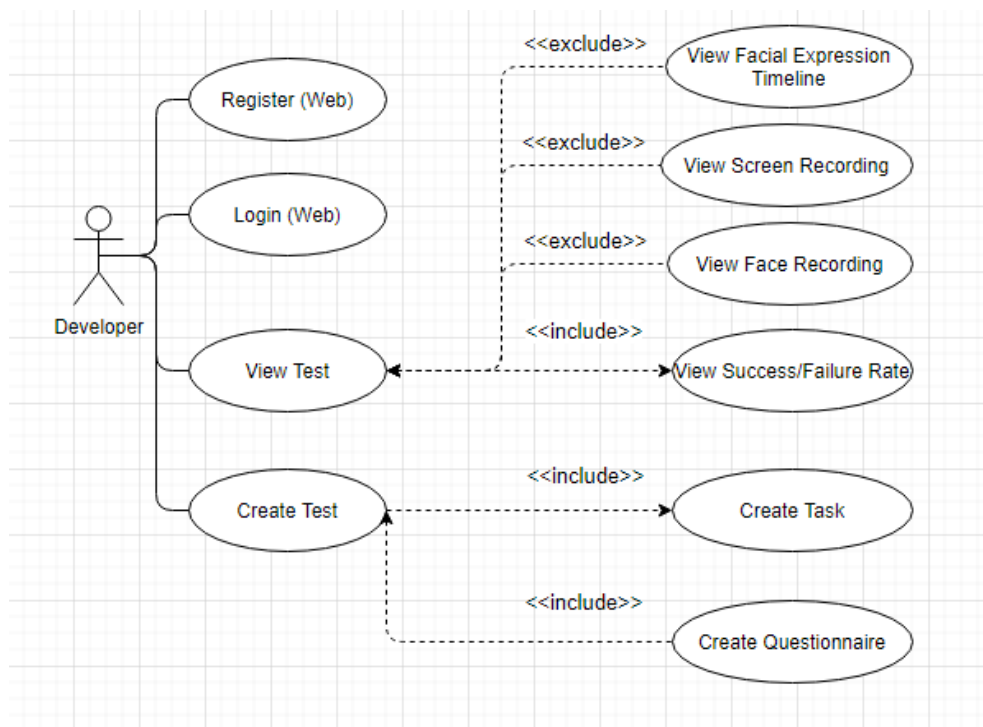


Figure 32 – Researcher Setting Up Usability Test Use Case

The upgraded use case diagram for the developer is displayed below in Figure 33. Additional features were added to the create test use case. Aside from the task creation and the questionnaire creation there are more options such as a multiple choice question which can come in the form of an open-ended multiple choice question and a rating question. Questions may be asked before or after the test and a scenario can be created for user before the begin the test. The starting URL can be set with Selenium so that the user does not have to copy and paste the link into the browser manually. When it comes to viewing the test results, the metrics and charts to be displayed are more defined in this use case diagram. The use case is also more specific on how the emotion data will be displayed i.e. a journey map, which is going to be explained in the later section of the report.

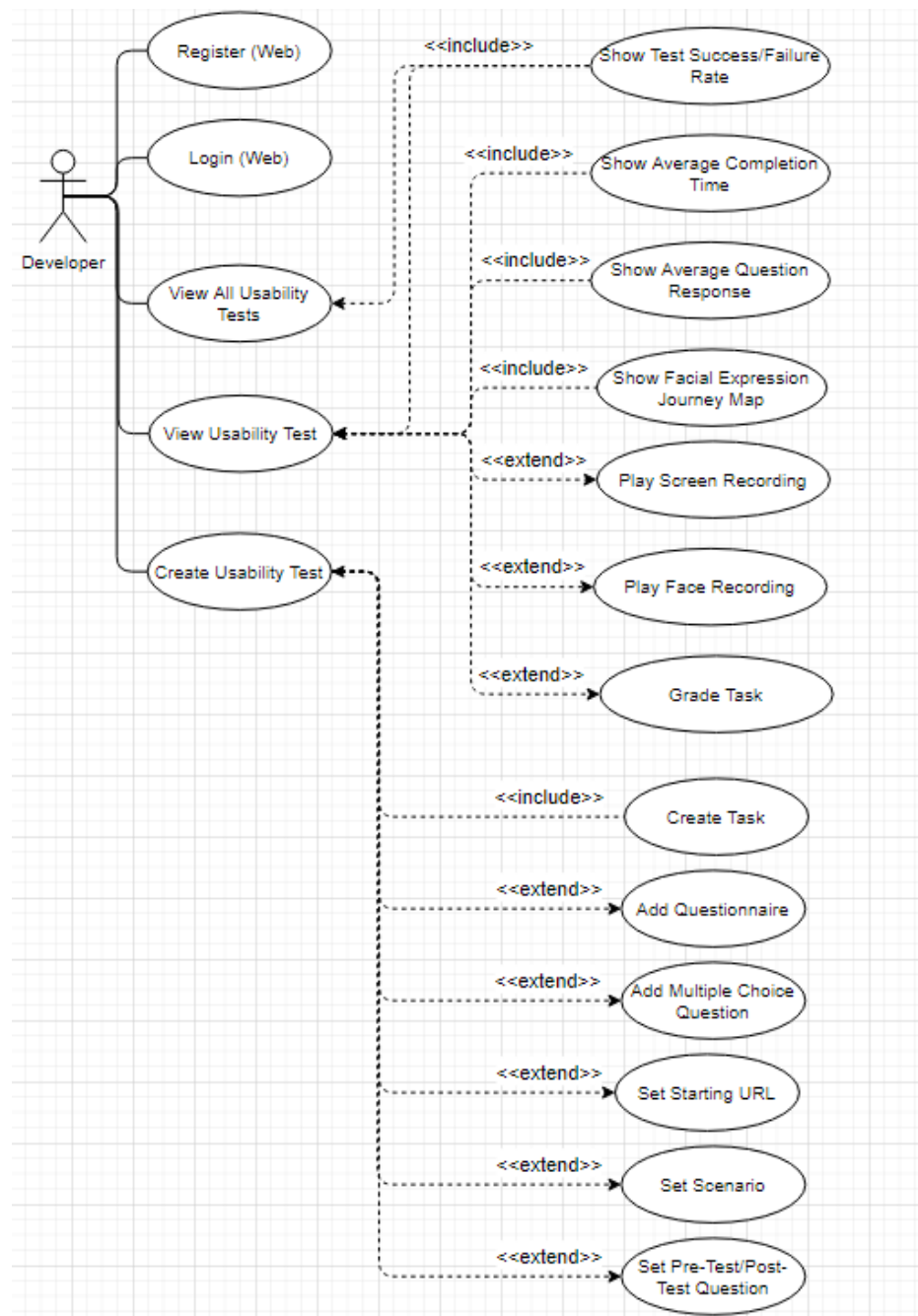


Figure 33 – Researcher Setting Up Usability Test Use Case (Second Iteration)

The use case diagram for the participant who is going to be using the local python application is illustrated below in Figure 34. First the test details are obtained from the web server. Using this data the local application will execute the correct instructions at the right time such as displaying the tasks and asking the participant questions etc. The test will begin and the user can minimize/maximize the instructions window that will be on the screen at all times. Once they are finished with the task they will press a button on the instruction window to proceed to the next task. The user will have to answer all the questions they are asked and will have the option to write an optional comment if they would like to add anything after the test is finished. Once the test is finished the data is uploaded automatically to the web server.

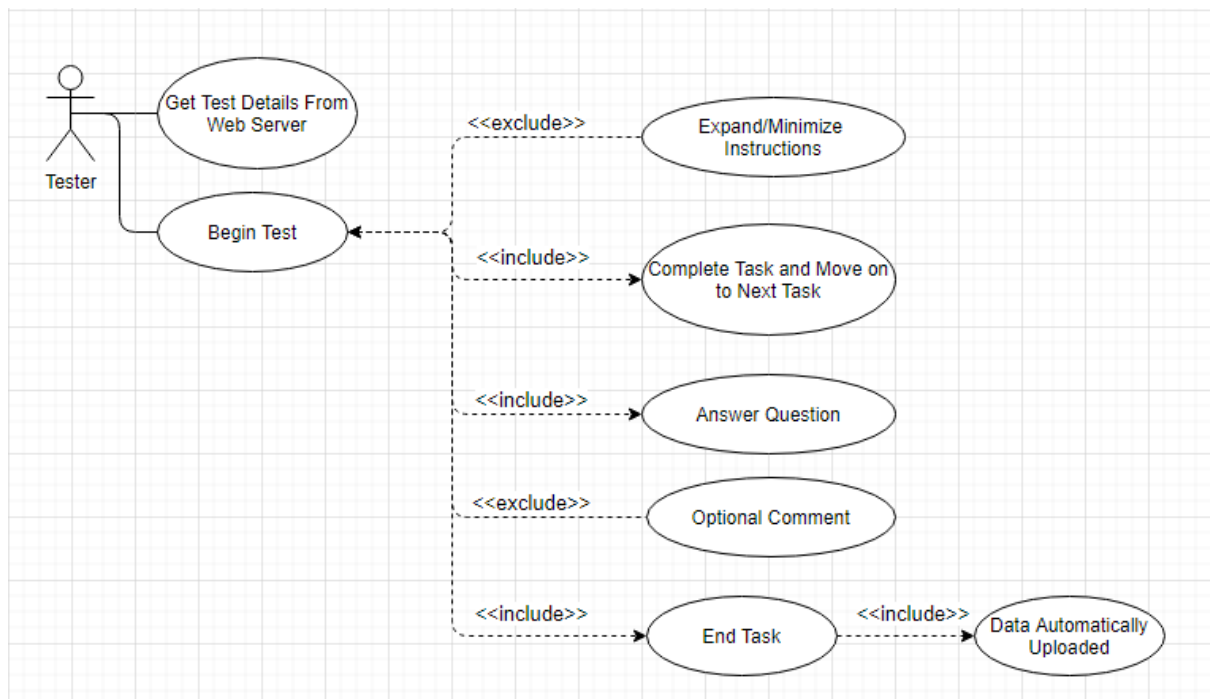


Figure 34 – Participant Use Case Diagram

Screen Prototypes

The “Dashboard” of the web application is the main screen the researcher will see after login and is illustrated below in Figure 35. The prototypes are designed with JustinMind. This is where the researcher can create new projects. This would relate to the “View All Usability Tests” use case. A “New Project” can be created for a new application to be tested and each project will have multiple usability tests. These usability tests can test various functionality of the program. In the example below we can see the status of the usability test (Open), the date the test was created (01/01/2020), the number of participants (5), the number of tasks passed (80%) and the number of tasks failed (20%).

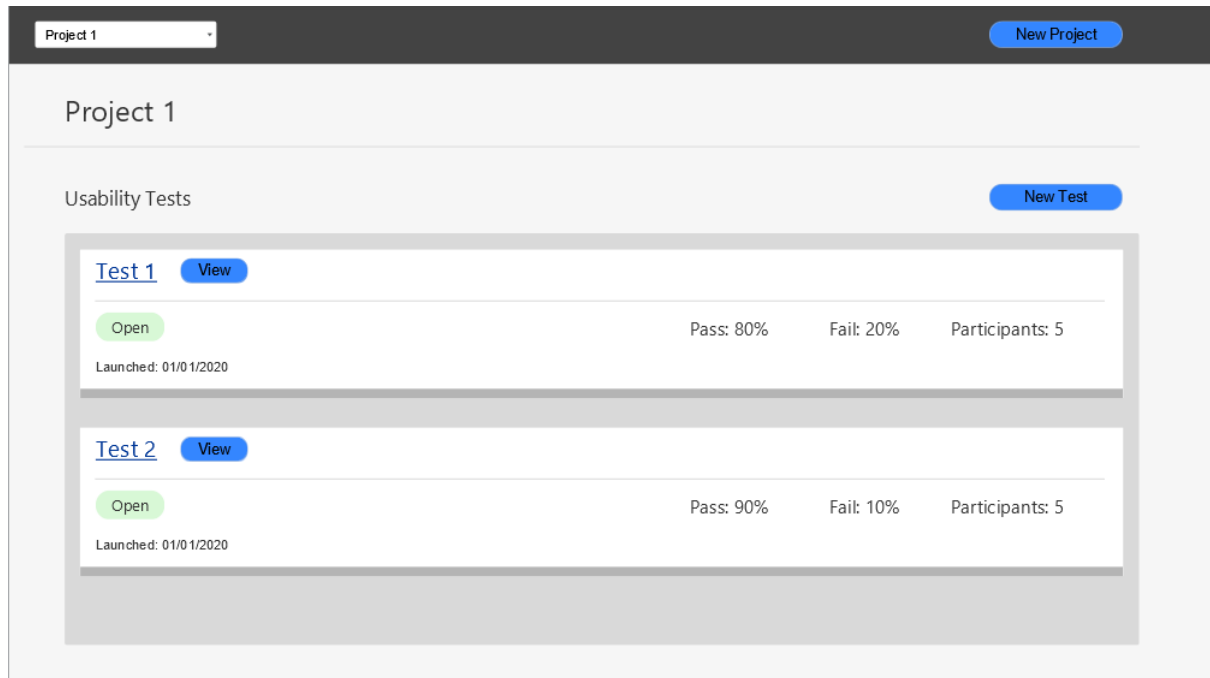


Figure 35 – Web Application Dashboard

The usability test creation screen which corresponds to the “Create Usability Test” use case is illustrated below in Figure 36. The researcher will name their test and insert questions or tasks. The question can be answered in plain text. There is a multiple choice question where the user is asked to pick one answer. There is a multiple choice questions with multiple answer.

<

Create Test

Test Details

Test Name

Pre-Test Questions

Text

Multiple Choice

Tasks

+

 Question

+

Instruction

Enter Instruction

X

Enter Instruction

X

Add Step

Multiple Choice Question

Enter Question

Enter Answer Choice

X

Add Answer

Instruction

Enter Instruction

X

Enter Instruction

X

Add Step

Create Test

Figure 36 – Web Application Usability Test Create

The test results screen that the research is going to is illustrated below in Figure 37. This is the “View Usability Test” use case. The usability test name in this case is “Test 1” and this test has multiple tasks which include “Task 1”. The total success and failure rate across all the tasks is given in the overview. Below that the researcher can choose to view the data on an individual task. The first chart on the top left shows the success and failure rate. The chart on the right shows the emotions of all the participants during that task. This chart excludes the “Neutral” emotion as it does not provide any information. The frequency of the other facial expressions is obtained and plotted on the bar chart. The idea is that a face expression that occur for any length of time below 1 second are assigned a value of 1. If the face expression persists for longer than one second then the value will be incremented every other half second. In other words, if a face expression persists for 2 seconds it will be worth 3 points. This type of point system is used to capture facial expressions that persist on the participants face for an extended period of time and more significance is assigned these facial expressions. This timing will have to be experimented with to gain an optimal result.

Another type of chart displayed is a pie chart and these can be used to display the answers to questions by all the participants. A possible question that a researcher might want to ask could be how difficult the participant found the task the rating could be from 1 to 5.

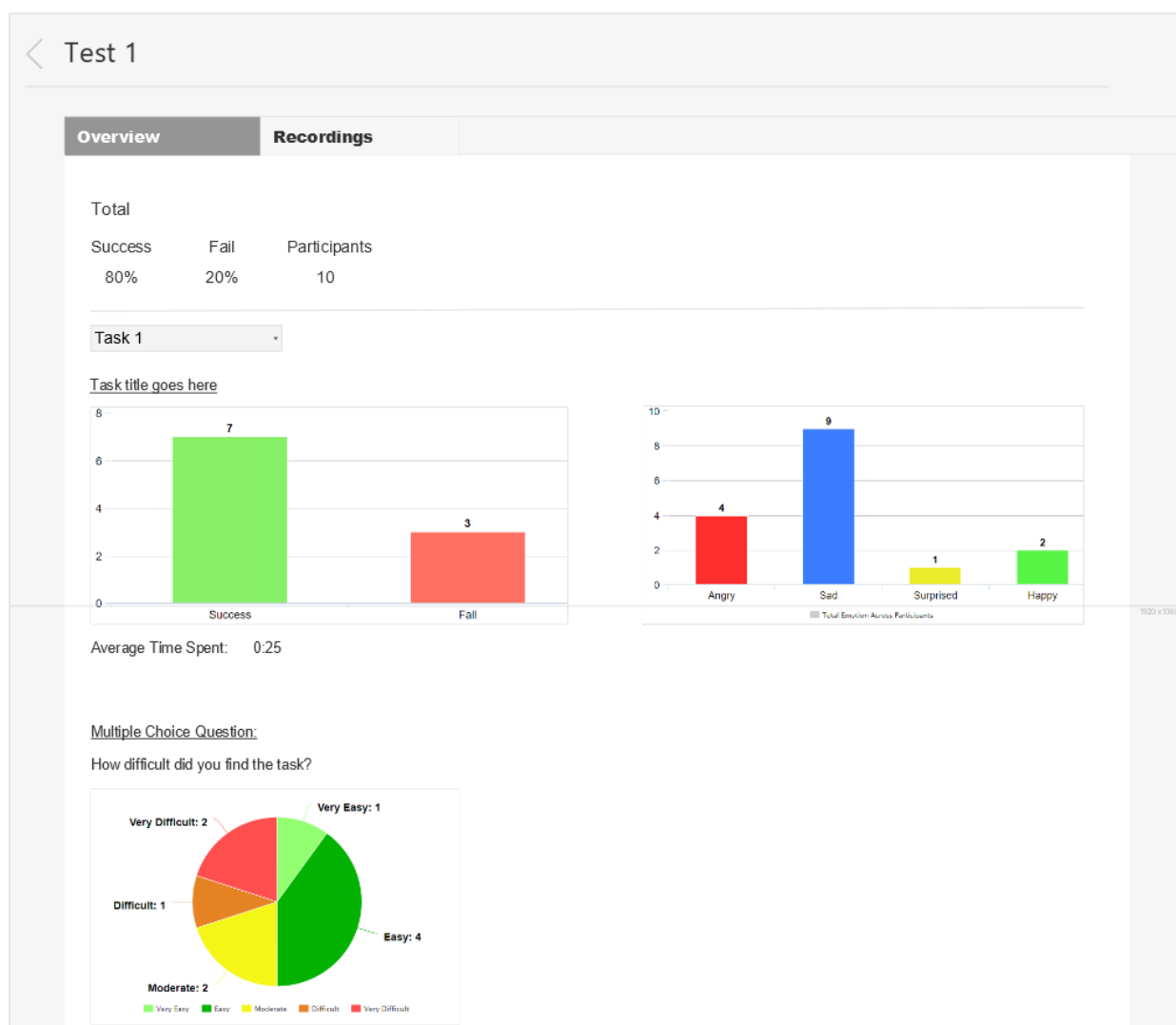


Figure 37 – Web Application View Usability Test Results

The page where the researcher can view the recordings and grade the usability test is illustrated below in Figure 38. The tasks will be graded with either a “Pass” or a “Fail” depending on whether or not the user has successfully completed the task. The video will be of the user’s screen as they complete the questions and tasks and will include the user’s camera feed. This will allow the research to not only see what the user is doing but also their face expression. The emotions are displayed in their respective colour below the video. The video markers are going to be implemented with Videojs-markers [Citation]. This will help the researcher(s) find the points of interest in the video faster and this will also show them an overview of the video in terms of the emotions. Below the video is the first implementation of the journey map. The chart was evaluated by Andrea Curley, the project supervisor, and one of the possible issues identified was the visual effect the length of the bars might have. The length of the bars are simply the result of the vertical structuring of the emotions with “Happy” being at the top and “Angry” being at the bottom. The length of the bars seem to add unnecessary depth and as a result makes the data the chart is displaying less clear.

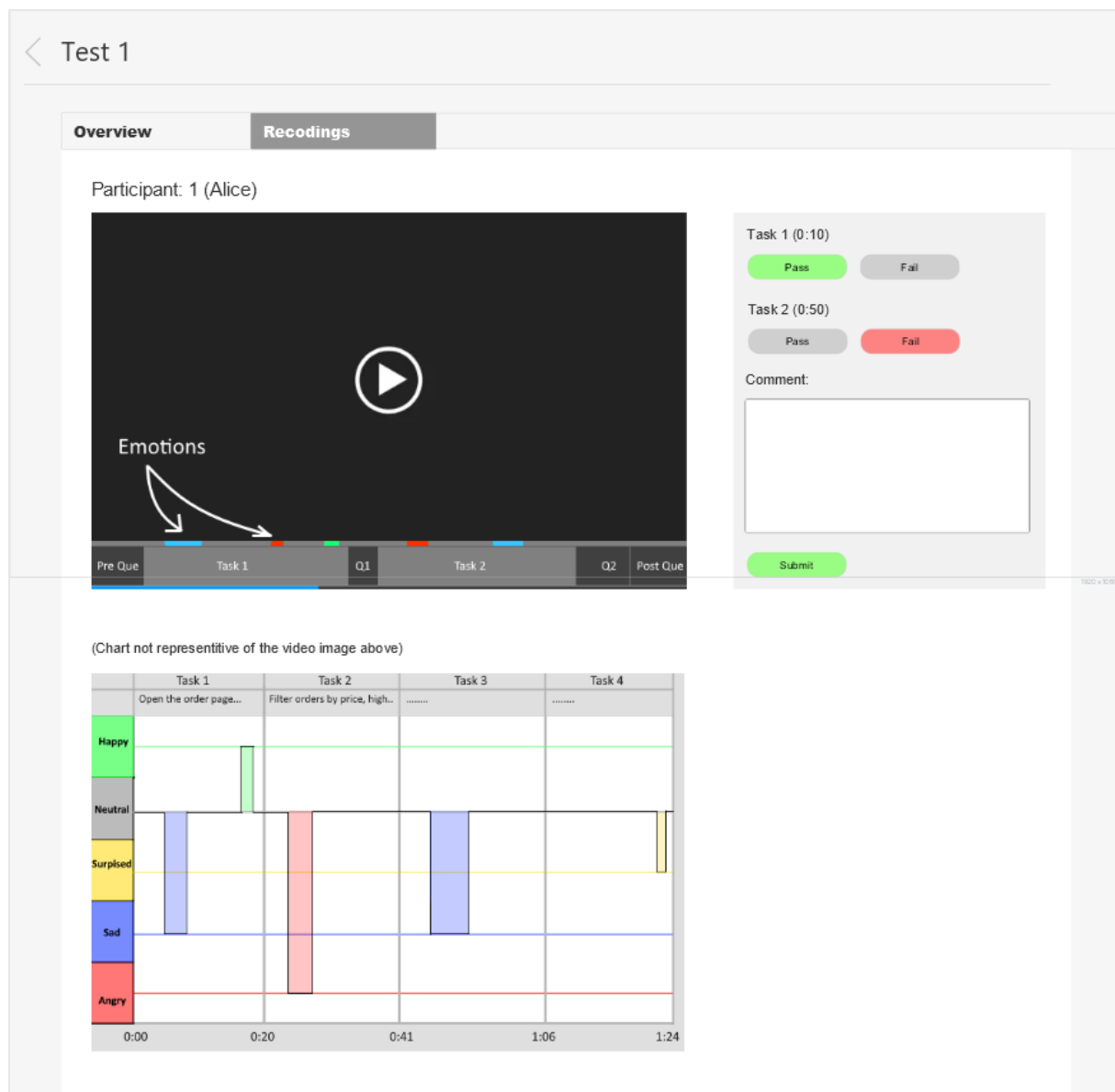


Figure 38 – Web Application View Recordings

The second design of the journey map after the evaluation is illustrated in Figure 39. In this version the emotions are split into positive and negative with each bar now being the same length. “Surprise” is labelled as a negative emotion because usable functionality should not be surprising to the user. If the participant is surprised by anything this should not be brushed off as a positive thing and should be investigated.

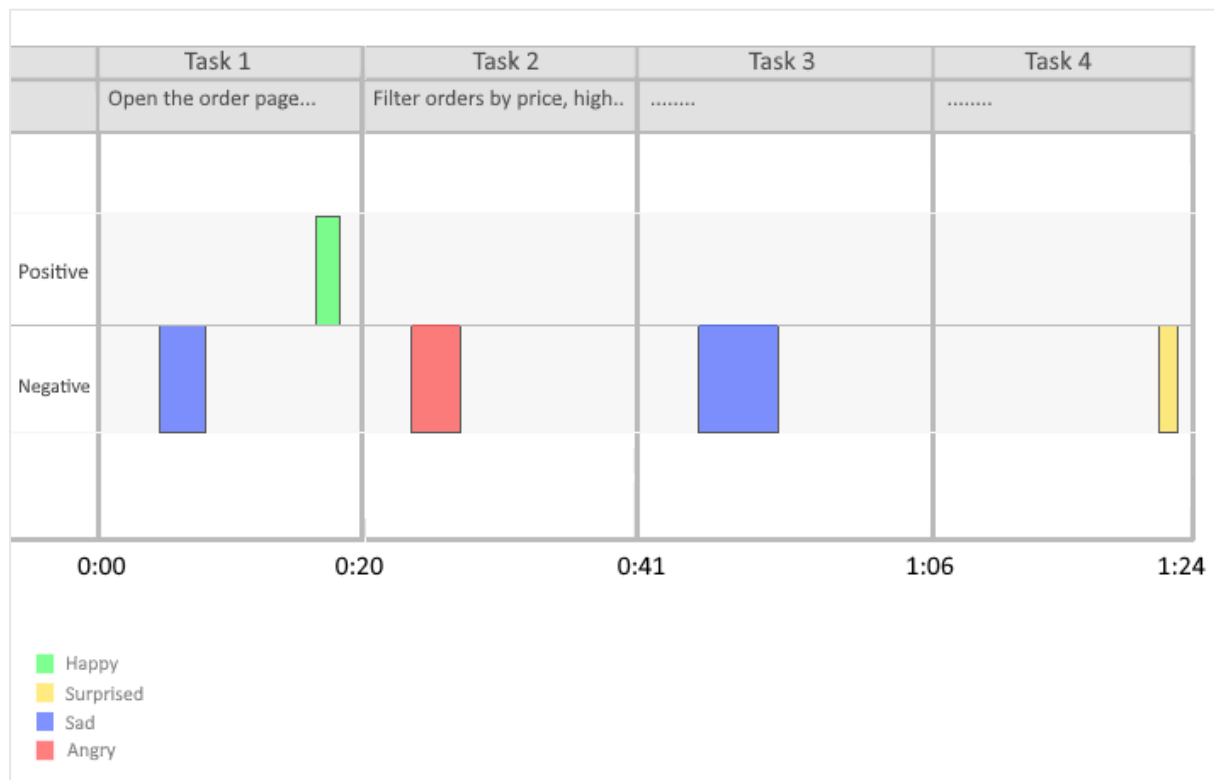


Figure 39 – Journey Map v2

3.4.2 Back-End

3.4.3 Database

For the database tier, the RDBMS chosen was MySQL. The reason for this choice over Firestore was because the data follows a hierarchical structure and it makes more sense to use a relation database as opposed to a non-relation database like MongoDB and Firestore.

Figure 3.4.3.1 illustrates the design of the database. Each researcher can have many projects, and these projects can have many tests. Each test must have tasks and may have questions. In the first iteration the ParticipantTest table referred to an instance of usability test. In other words, this table is used to store information about a completed usability test. The name “ParticipantTest” did not make complete sense and this was changed to “TestInstance” in the next iteration.

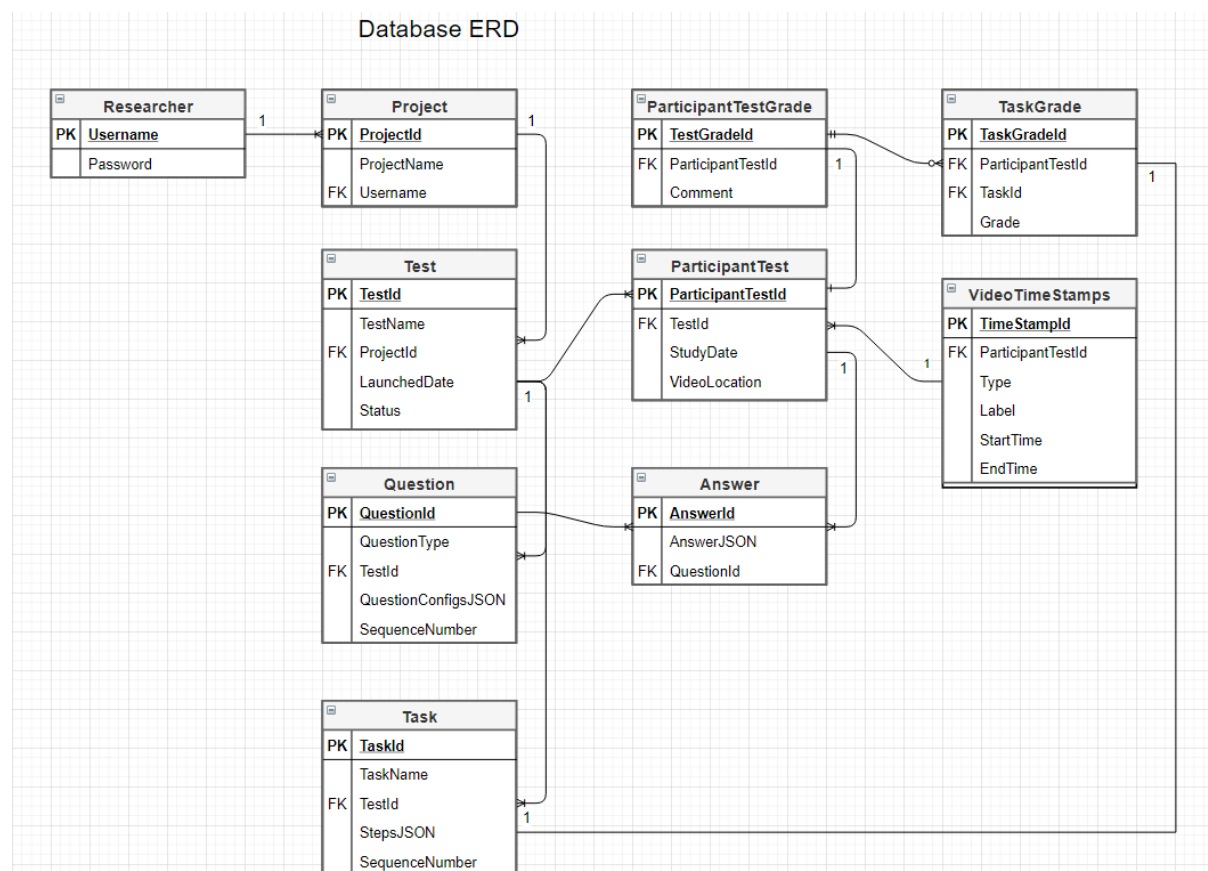
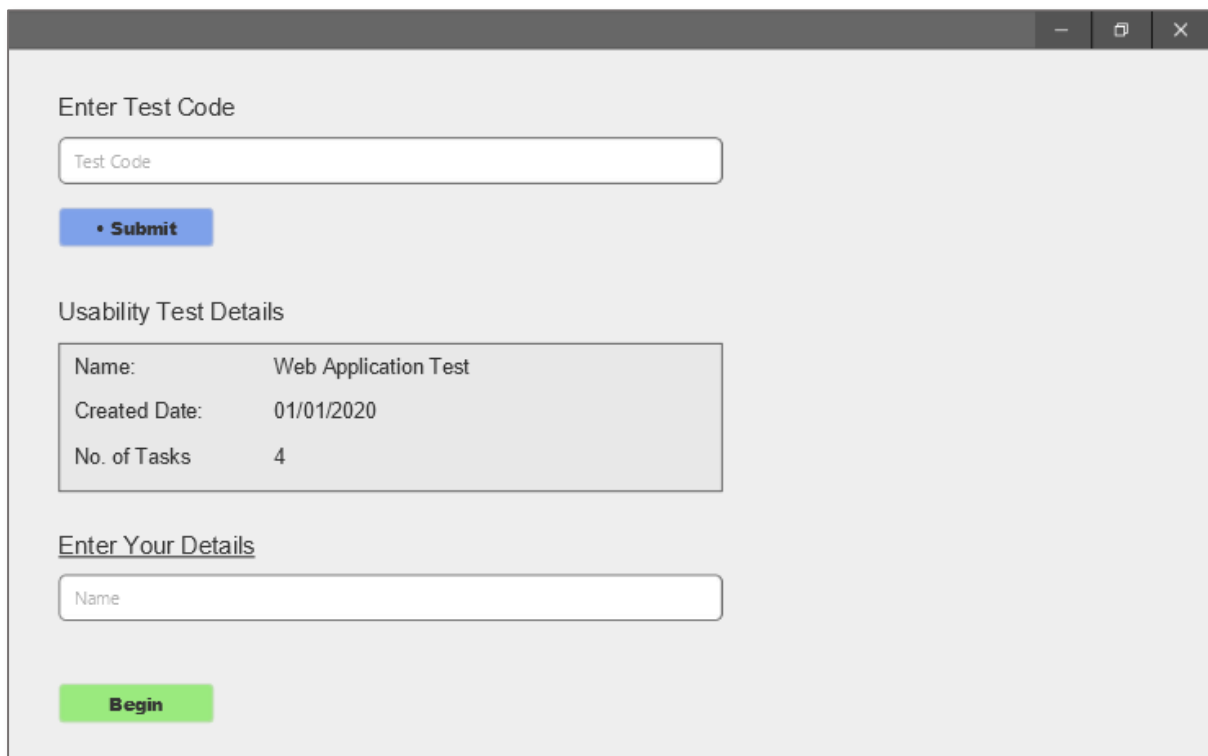


Figure 40 – Web Application ERD

3.5 Local App

The local python application that will run on the participants machine is illustrated in Figure 41. The participant or the researcher (if they are in the same room) will enter the test code into the dedicated text box and click “Submit”. The details for the usability test will be pulled from the web server back-end and displayed on screen. That was the “Get Details from Web Server” use case. Once the details are verified as correct the participant will enter their name and click “Begin” to begin the text.



The screenshot shows a web application window with a light gray background. At the top, there is a title bar with standard window controls. The main content area is divided into three sections. The first section, titled "Enter Test Code", contains a text input field with the placeholder text "Test Code" and a blue button labeled "Submit". The second section, titled "Usability Test Details", contains a table with three rows of information. The third section, titled "Enter Your Details", contains a text input field with the placeholder text "Name" and a green button labeled "Begin".

Name:	Web Application Test
Created Date:	01/01/2020
No. of Tasks	4

Figure 41 – Local App First Screen

Once the test begins the participant will see a similar screen to what is illustrated in Figure 42. In the top right corner are the instructions for the task that the participant needs to complete. Around the edge of the screen is a red border. This is to visually indicate that the session is being recorded. At the bottom center is a small box that says “Recording” to definitely let the participant know they are being recorded. They may press the “Stop” button to stop the test prematurely. Once the participant is finished with their task they are to click the “Finish Task” button to proceed to the next task.

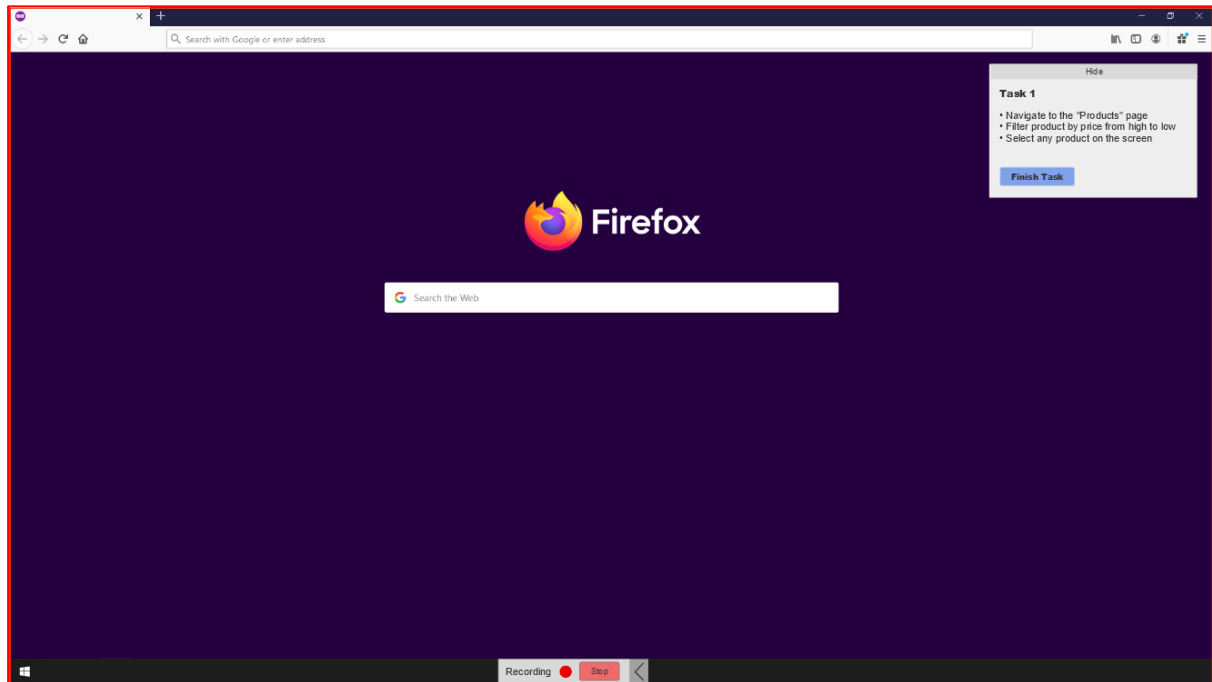


Figure 42 – Local App Usability Test Started

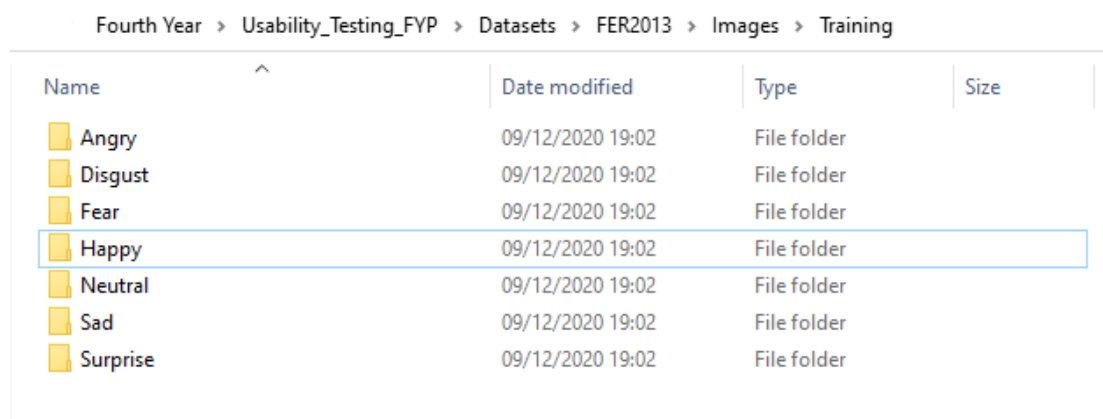
3.5. Facial Expression Recognition & Machine Learning

CRISP-DM Methodology in Prototype Development

This section will discuss the design decisions made with regards to facial expression recognition and machine learning. Following the CRISP-DM methodology the first stage is “Business Understanding”. This involved research into the field and the questions such as “How to present the emotion data?”, “What emotions are necessary in the field of usability testing?”. In the research paper by Landowska (2015) [4] there was mention of combining the basic emotions to create more complex emotions. The risk associated with combining emotions is the prerequisite that it has been correctly detected and secondly that it will be interpreted correctly by the researcher. Emotions such as empowerment might make more sense than simply “Happiness”, however, it can be misleading and more difficult to detect as the participant could simply find some part of the system amusing. Boredom is an emotion that consists of “sadness”. Once again, this can be misleading as someone frowning could mean much more than boredom. Hence a decision was made to simply display the basic emotions.

The second stage of the CRISP-DM methodology is “Data Understanding”. The data has been collected, stored and described. Practically every dataset researched was different in some way and the data format especially needs to be made consistent so that the input into the machine learning model is the same. In the research section the reasons for why certain datasets were not chosen for training the model were already discussed. In the research section the datasets were also described and discussed in detail.

The third stage is “Data Preparation” and involves analysing, cleaning, constructing and combining the data. The FER2013 dataset is in the form of a .CSV file with a column dedicated to the pixel values of the grayscale image. The other datasets that are planned to be used are the JAFFE and FacesDB datasets, both of which are in the form of .tiff and .tif images respectively. A decision was made to convert the pixel values in the FER2013 dataset to actual images that can be stored on the computer. This will be accomplished with the help of the NumPy, Pandas, and OpenCV libraries. The other option was to simply write a method for taking the data out of the CSV file but not storing it and instead feeding it directly into the machine learning algorithm for training. However, for the sake of consistency this was not the route taken. There will be three directories, “Training”, “Validation” and “Testing”. The folders within each of the directories will follow the same structure with a folder for each emotion and this is illustrated in Figure 3.5.1. The both Keras and scikit-learn Python libraries can be used to shuffle and split the data if testing and validation images have not been provided. This is the case with the JAFFE and FacesDB datasets.



Fourth Year > Usability_Testing_FYP > Datasets > FER2013 > Images > Training			
Name	Date modified	Type	Size
Angry	09/12/2020 19:02	File folder	
Disgust	09/12/2020 19:02	File folder	
Fear	09/12/2020 19:02	File folder	
Happy	09/12/2020 19:02	File folder	
Neutral	09/12/2020 19:02	File folder	
Sad	09/12/2020 19:02	File folder	
Surprise	09/12/2020 19:02	File folder	

Figure 43 – Dataset Folder Structure (Training)

As mentioned in the research section data augmentation techniques will be used to generate more data from the existing dataset. This can be accomplished with the Keras ImageDataGenerator function which will rescale, rotate, shear, zoom, shift and flip the image horizontally.

The fourth stage is “Modelling” and involves selecting the modelling technique, generating a test design for how the model will be evaluated, building the model, and finally, assessing the model. There is quite a lot involved in this stage and the focus in this section of the report will be wherever the prototype aspect is applicable. The charts for modelling the data (e.g. Journey map) have already been discussed so the focus will be on the evaluation techniques for the model and as discussed in the research section the metrics used will be the confusion matrix, accuracy, recall and precision. The intended loss function that will be used is Categorical Cross-Entropy loss (also known as Softmax Loss). It is a Softmax activation plus a Cross-Entropy loss. The CNN architecture of choice is the VGG-16 architecture. The next two stages in the CRISP-DM methodology are evaluation and deployment. Both of these stages will be discussed in later sections of the report.

Discussion

As mentioned previously the data from the camera will be fed into the FER model. The output data at the very end will be a dictionary of timestamps (HH:MM:SS:MS) with alternating facial expressions. For example, the sample output will look like:

```
{  
    "00:00:00:00": "Neutral",  
    "00:00:05:00": "Happy",  
    "00:00:06:10": "Neutral",  
}
```

In the presence of no emotion in the scenario that the model did not detect anything the “None” value will be stored. The timestamp will be to a tenth of a second in accuracy to ensure the facial expressions are accurate.

3.6 Conclusions

The stages of the project and the tasks that need to be completed (development) are broke up into 3 stages and illustrated below in Figure 44. These tasks are planned as sprints and illustrated in **Error! Reference source not found.** For the sake of readability, the sprints plan is further split up into two images that are just enlarged versions of the original. These are displayed in **Error! Reference source not found.** and **Error! Reference source not found.**

Name	Description	Stage
FER Dataset Selection and Preparation	Choose the datasets that will be used and do all the preparations such as data augmentation, resizing, separating etc.	Stage 1
FER Model Selection and Training	Choose the architecture/model that will be trained on the datasets.	Stage 1
FER Model Evaluation Metrics	Implement charts and other forms of visualization used in the evaluation of the model (e.g. Confusion matrix).	Stage 1
FER Model Output Data Configuration	Make the model output the FER data in a certain format which can then be sent to the web server for storage etc.	Stage 1
Create Basic Back-end and Basic Front-end	A basic front-end and back-end needs to be implemented as a foundation for implementing the other features.	Stage 2
Database Setup	Create the database and the DAO classes in the web server that will be used to store the data.	Stage 2
Researcher Login	Implement the registration and login in the web server. The website is for the researchers to configure the tests etc.	Stage 2
Basic Usability Test Configuration	Implement basic usability test creation in the web server allowing other features to be implemented.	Stage 2
Local Python Application	The local python application will receive the test details and the basic usability test can be conducted.	Stage 2
Upload FER Data to Web Server and Store Data	Upload the data from the FER model after the usability test to the web server where it is then stored.	Stage 2
Screen Recording for Local Application	Record the participant's screen and save the recording to the file.	Stage 3
Upload Video	Upload both the screen recording and the camera recording to the 3rd party service.	Stage 3
More Usability Test Configuration Options	Implement the other features of the usability testing such as multiple choice questions, rating questions etc.	Stage 3
Display Usability Test Results & Data	Implement the charts used to display the data (e.g. Journey map, pie chart, bar charts etc.)	Stage 3
Embed Usability Testing Video on Website	Embed the video in the website, allowing the researcher to view it.	Stage 3

Figure 44 – Project Tasks & Requirements

4. Prototype Development

4.1. Introduction

This section will cover the machine learning development and software used in the development process. To backup and manage this project a GitHub repository was created. Git is a version control tracking system and tracks changes of files. Eclipse IDE for Enterprise Java Developers is used for creating the web application's backend and Visual Studio Code is used for the frontend development with JavaScript. The FER model and other machine learning related functionality will be created in Jupyter Notebook in Anaconda. With Anaconda an environment is created which is a directory where all the packages are stored. This includes Python packages and these packages, even if present on the machine need to be installed in the environment. This is useful as it allows the developer to have multiple versions of Python or Python libraries on the same machine. Jupyter Notebook a very useful development tool that serves as not only an IDE but also a presentation tool where images can be displayed. This makes it very suitable for data science and machine learning.

4.2 Machine Learning

4.2.1 CSV to Image Converter (CRISP Data Preparation Stage)

Figure 4.2.1.1 illustrates the FER2013 dataset in its original format, a CSV file. The *pixels* column consists of 2304 (48 x 48) pixel values which are converted into an image. Figure 4.2.1.2 illustrates a sample image post conversion. Each image has a label which corresponds to one of the 7 emotions (Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral) with a value ranging from 0 to 6.

	A	B	C	D
1	emotion	pixels	Usage	
2	0	70 80 82 72 58 58 60 63 54 58 60 4	Training	
3	0	151 150 147 155 148 133 111 140	Training	
4	2	231 212 156 164 174 138 161 173	Training	
5	4	24 32 36 30 32 23 19 20 30 41 21 2	Training	
6	6	4 0 0 0 0 0 0 0 0 3 15 23 28 48	Training	
7	2	55 55 55 55 55 54 60 68 54 85 151	Training	
8	4	20 17 19 21 25 38 42 42 46 54 56 6	Training	
9	3	77 78 79 79 78 75 60 55 47 48 58 7	Training	
10	3	85 84 90 121 101 102 133 153 153	Training	
11	2	255 254 255 254 254 179 122 107	Training	
12	0	30 24 21 23 25 25 49 67 84 103 12	Training	
13	6	39 75 78 58 58 45 49 48 103 156 8	Training	
14	6	219 213 206 202 209 217 216 215	Training	
15	6	148 144 130 129 119 122 129 131	Training	
16	3	4 2 13 41 56 62 67 87 95 62 65 70 6	Training	
17	5	107 107 109 109 109 109 110 101	Training	
18	3	14 14 18 28 27 22 21 30 42 61 77 8	Training	
19	2	255 255 255 255 255 255 255 255	Training	
20	6	134 124 167 180 197 194 203 210	Training	
21	4	219 192 179 148 208 254 192 98 1	Training	
22	4	1 1 1 1 1 1 1 1 1 1 2 2 2 2 7 12 2	Training	

Figure 46 – FER2013 Dataset CSV File



Figure 45 – FER2013 Pixel Values as Image

Below is the code for opening the CSV file, obtaining the relevant information and saving the image as a .jpg file. A dictionary is created that contains the label number and the label name. This dictionary is used to ensure that the image is stored in the correct folder.

The libraries such as Pandas, NumPy, OpenCV and Matplotlib are imported as these will be used in this script (See Figure 4.2.1).

```
import pandas as pd
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

Figure 47 – Imports

A dictionary is created to map the label number to the corresponding name. The path variables are initialized and the CSV file is opened/read with Pandas (See Figure 4.2.2).

```
# Maps the emotion label to the name so that it can be saved into the correct directory
emotionsDict = {
    0: "Angry",
    1: "Disgust",
    2: "Fear",
    3: "Happy",
    4: "Sad",
    5: "Surprise",
    6: "Neutral"
}

datasetFolder = "C:/Users/New User/Desktop/Fourth Year/Usability_Testing_FYP/Datasets/FER2013"
datasetCSV = datasetFolder + "/fer2013.csv"
imageFolder = datasetFolder + "/Images"

dataFrame = pd.read_csv(datasetCSV)
```

Figure 48 – Path Variables and Opening File

Each row in the CSV file is iterated over and the values in each column is extracted. The pixel values are then stored in a 48x48 matrix and this matrix is saved as an image to the correct folder (See Figure 4.2.3).

```
imageFormed = None
for index, row in dataFrame.iterrows():
    emotionNum = row['emotion']
    imagePixels = row['pixels']
    usage = row['Usage']

    width, height = 48, 48
    imageFormed = np.fromstring(imagePixels, dtype=int, sep=" ").reshape((height, width))

    outFilePath = imageFolder + "/" + usage + "/" + emotionsDict[emotionNum] + "/" + str(index) + ".jpg"
    cv2.imwrite(outFilePath, imageFormed)
```

Figure 49 - CSV to Image Converting and Exporting

The final end results is images in their respective folders as shown below. All the images labelled “Happy” are in the “Happy Folder”.

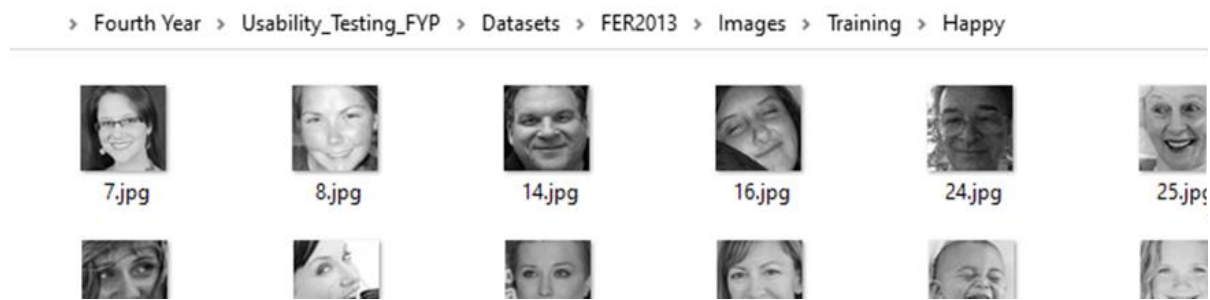


Figure 50 – Converted Images

4.2.2 Model Training (CRISP Modelling Stage)

Model 1

The first attempt at training a CNN began with training a modified VGG16 architecture from a GitHub repository found online. The dataset used was FER2013 with a total of 35,866 images split into training, validation and testing in the ratio of 80:10:10. The validation accuracy on the FER2013 dataset was an acceptable 57.7% with the state of the art for a VGG model for that dataset being 72.4%. The model was trained for 25 epochs. However, the confusion matrix for this model would not display properly. Unable to find the root cause and unsure if the issue was with the model or input to the confusion matrix another model was used to get more information that could help with the problem. Later in the evaluation data from the camera was fed to the model and successfully predicted the emotion given the accuracy indicating that the problem lies in the input to the confusion matrix.

Model 2

In the second model the CNN architecture was different (See Figure 4.2.2.1) and unlike the other model it did not involve data augmentation/generation. The dataset used and the ratio split is the same as with the previous model. The model has achieved a 56% validation accuracy after 30 epochs which is less accurate than the other model by a significant amount. However, this time the confusion matrix successfully displayed the data. This was confirmed by manually adding up the values and obtaining the accuracy from the matrix and comparing it with the expected accuracy.

The architecture of the model below is similar to the VGG architecture and was mainly used for experimentation purposes. The Architecture diagram is illustrated in Figure 4.2.2.1.

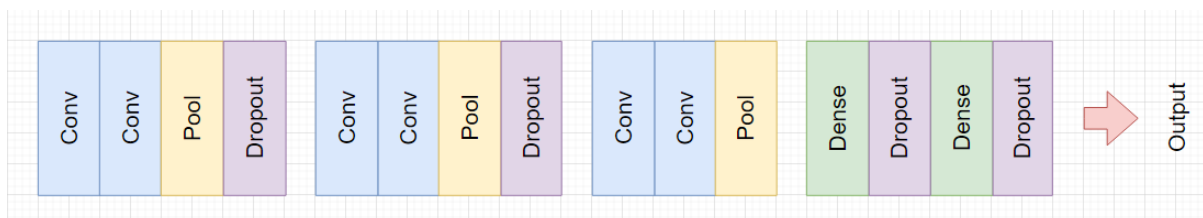


Figure 51 Second Model Architecture

```
Epoch 25/30
898/898 [=====] - 11s 12ms/step - loss: 1.1026 - accuracy: 0.5763 - val_loss: 1.2094 - val_accuracy: 0.5503
Epoch 26/30
898/898 [=====] - 11s 12ms/step - loss: 1.0948 - accuracy: 0.5845 - val_loss: 1.1906 - val_accuracy: 0.5534
Epoch 27/30
898/898 [=====] - 11s 12ms/step - loss: 1.0943 - accuracy: 0.5822 - val_loss: 1.2040 - val_accuracy: 0.5478
Epoch 28/30
898/898 [=====] - 11s 12ms/step - loss: 1.0988 - accuracy: 0.5813 - val_loss: 1.1979 - val_accuracy: 0.5539
Epoch 29/30
898/898 [=====] - 11s 12ms/step - loss: 1.0792 - accuracy: 0.5889 - val_loss: 1.2048 - val_accuracy: 0.5534
Epoch 30/30
898/898 [=====] - 11s 12ms/step - loss: 1.0663 - accuracy: 0.5928 - val_loss: 1.1776 - val_accuracy: 0.5603
```

Figure 52 – Second Model Training

To explain the code, first the paths to the train, validation and testing data is initialized. The training data will be used in the training of the model, the validation data will be used to check how the model performs on unseen data and finally the testing data, much like the validation data will be used to see how the model performs on another set of unseen data.

```

trainDataDir = 'C:/Users/New User/Desktop/Fourth Year/Usability_Testing_FYP/Datasets/FER2013/Images/Training'
validationDataDir = 'C:/Users/New User/Desktop/Fourth Year/Usability_Testing_FYP/Datasets/FER2013/Images/PublicTest'
testDataDir = 'C:/Users/New User/Desktop/Fourth Year/Usability_Testing_FYP/Datasets/FER2013/Images/PrivateTest'

```

Figure 53 – Path Declaration

The dataset images are read from the folders and stored in their respective arrays (X_train, X_valid). Another array is for the labels (train_y, valid_y) and the index in the arrays is used to map the image to the label.

```

dataLabels = {
    "Angry": 0, "Disgust": 1, "Fear": 2, "Happy": 3, "Sad": 4, "Surprise": 5, "Neutral": 6
}

def loadDataset(datasetDirectory):
    global dataLabels
    data = []
    labels = []

    emotionDirList = os.listdir(datasetDirectory)
    for folder in emotionDirList:
        # Folder exists but has not been selected so we ignore it
        if folder not in dataLabels.keys():
            continue

        label = dataLabels[folder]

        # Iterate over the images
        imgList = os.listdir(datasetDirectory + "/" + folder)
        for imgName in imgList:
            img = cv2.imread(datasetDirectory + '/' + folder + '/' + imgName)
            img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            data.append(np.array(img.ravel(), 'float32'))
            labels.append(label)

    return data, labels

X_train, train_y = loadDataset(trainDataDir)
X_valid, valid_y = loadDataset(validationDataDir)

X_train = np.array(X_train, 'float32')
train_y = np.array(train_y, 'float32')
X_valid = np.array(X_valid, 'float32')
valid_y = np.array(valid_y, 'float32')

print(train_y.shape, valid_y.shape, X_train.shape, X_valid.shape)
not_categorical_valid_y = valid_y.copy()

```

Figure 54 – Loading Dataset from File

An arbitrary number of features and batch size is selected. The number of labels corresponds to the number of facial expressions and the number of epochs. It has been observed that past 25 epochs the accuracy on the validation dataset does not increase by much if at all.

```
num_features = 64
num_labels = 7
batch_size = 32
epochs = 30
width, height = 48, 48

train_y = np_utils.to_categorical(train_y, num_classes=num_labels)
valid_y = np_utils.to_categorical(valid_y, num_classes=num_labels)

# Data is normalized between 0 and 1
# Train
X_train -= np.mean(X_train, axis=0)
X_train /= np.std(X_train, axis=0)
# Validation
X_valid -= np.mean(X_valid, axis=0)
X_valid /= np.std(X_valid, axis=0)
# Reshape
X_train = X_train.reshape(X_train.shape[0], 48, 48, 1)
X_valid = X_valid.reshape(X_valid.shape[0], 48, 48, 1)
```

Figure 55 – Configurable Variables and Data Normalization

The code for the architecture illustrated in Figure 51 is displayed below in Figure 56.

```
# 1st convolution layer
model = Sequential()

model.add(Conv2D(64, kernel_size=(3, 3), activation='relu', input_shape=(X_train.shape[1:])))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.5))

# 2nd convolution layer
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.5))

# 3rd convolution layer
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model.add(Flatten())

# Fully connected neural networks
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(num_labels, activation='softmax'))
```

Figure 56 – CNN Architecture Code

4.2.3 Model Evaluation & Discussion (CRISP Evaluation Stage)

Model 1

The model has been trained to recognise 7 different emotions, just as the second model. Below Figure 57 illustrates the accuracy increase in the last few epochs with the final accuracy on the validation set of 57%. The accuracy increase and loss decrease is illustrated in Figure 58. Oddly the validation set accuracy was always higher than the training set accuracy. After further research this could be caused by the Dropout layer when training. Another possibility is the training and validation ratio.

```
Epoch 00022: val_loss improved from 1.10744 to 1.09371, saving model to Emotion_little_vgg2.h5
755/755 [=====] - 18s 24ms/step - loss: 1.3033 - accuracy: 0.5104 - val_loss: 1.0937 - val_accuracy: 0.5894
Epoch 23/25
755/755 [=====] - ETA: 0s - loss: 1.3076 - accuracy: 0.5096
Epoch 00023: val_loss did not improve from 1.09371
755/755 [=====] - 18s 24ms/step - loss: 1.3076 - accuracy: 0.5096 - val_loss: 1.1065 - val_accuracy: 0.5672
Epoch 24/25
754/755 [=====>.] - ETA: 0s - loss: 1.2822 - accuracy: 0.5241
Epoch 00024: val_loss improved from 1.09371 to 1.08735, saving model to Emotion_little_vgg2.h5
755/755 [=====] - 19s 25ms/step - loss: 1.2824 - accuracy: 0.5239 - val_loss: 1.0874 - val_accuracy: 0.5843
Epoch 25/25
755/755 [=====] - ETA: 0s - loss: 1.2906 - accuracy: 0.5197
Epoch 00025: val_loss did not improve from 1.08735
755/755 [=====] - 19s 25ms/step - loss: 1.2906 - accuracy: 0.5197 - val_loss: 1.0895 - val_accuracy: 0.5773
```

Figure 57 – First Model Training

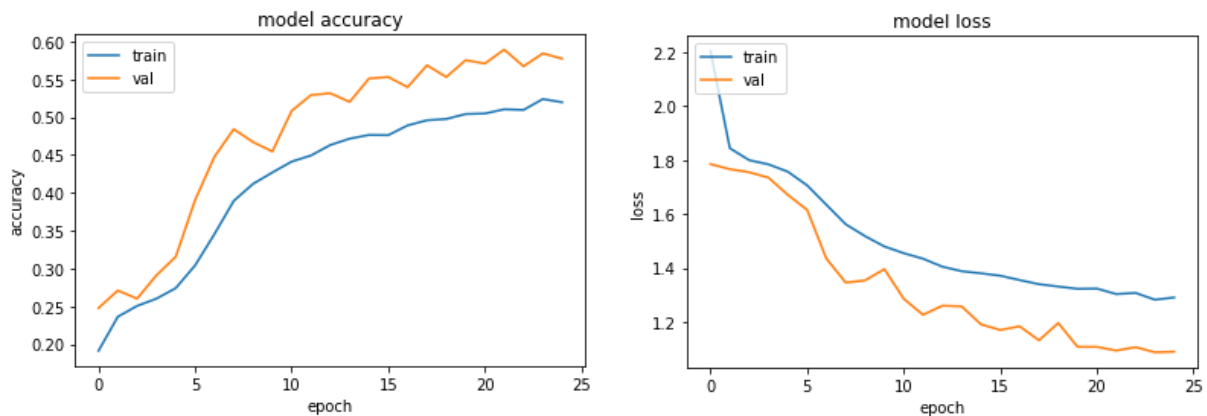


Figure 58 – First Model Accuracy and Loss

The model is finally put to practice in a real-world testing environment. This is illustrated in Figure 59. The emotions of fear and disgust were very difficult to detect and would most of the time fall under the “Sad” or “Angry” category. Generally, the model has performed very well in detecting the 5 basic emotions below. The “Happy” facial expression is illustrated twice to highlight that both an open mouth smile and a subtle smile are both captured. This is important as in a usability testing environment the facial expressions are expected to be subtle at times. The camera location has also been experimented with and the camera position at the top of the screen has yielded the best results. In the research paper by Landowska (2015) the camera was at the bottom of the screen which did not prove as effective.

To detect the face Haar Cascade classifier was used. The face detected is shown with a blue bounding box. The face is then extracted and processed before being fed to the model for emotion classification.

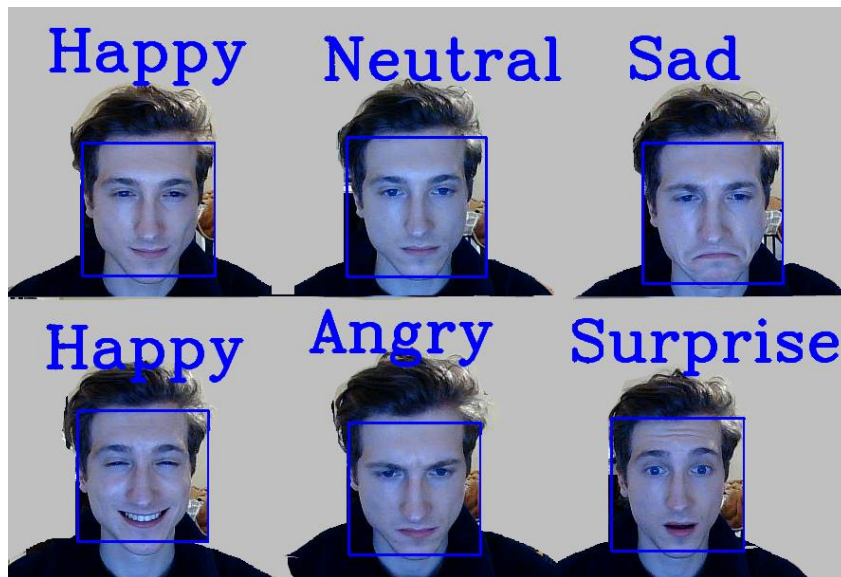


Figure 59 – First Model Camera Feed Predictions

Model 2

The second model, as mentioned before has achieved an accuracy of 56%. The model's precision in detecting a happy facial expression is 76% with a recall of 79% which is much higher than the other emotions. The precision of "Disgust" is 73%, however the recall was at a lowest 24% and the sample size was much smaller than the other emotions. The model could be possibly be improved by removing the "Fear" facial expression as it is not exactly applicable in the usability testing scenario and only increases the complexity. The "Disgust" emotions is incorrectly predicted as angry and sad in a lot of cases. From the matrix we can also see that the model has predicted sad when the true label was natural and vice versa.

The data from the camera was fed to the model as previously done with Model 1. The predictions appeared to be not very accurate at all and could definitely not be used in a production environment. This was because the emotion "Happy" was detected even when the face was neutral, sad, angry etc. The model has an extreme bias towards this facial expression. In conclusion, although the accuracy of the second is somewhat low it provides some insight into how emotions are predicted by looking at the confusion matrix and seeing which types of emotions are likely to overlap. This experiment also serves as a foundation for evaluating the future models.

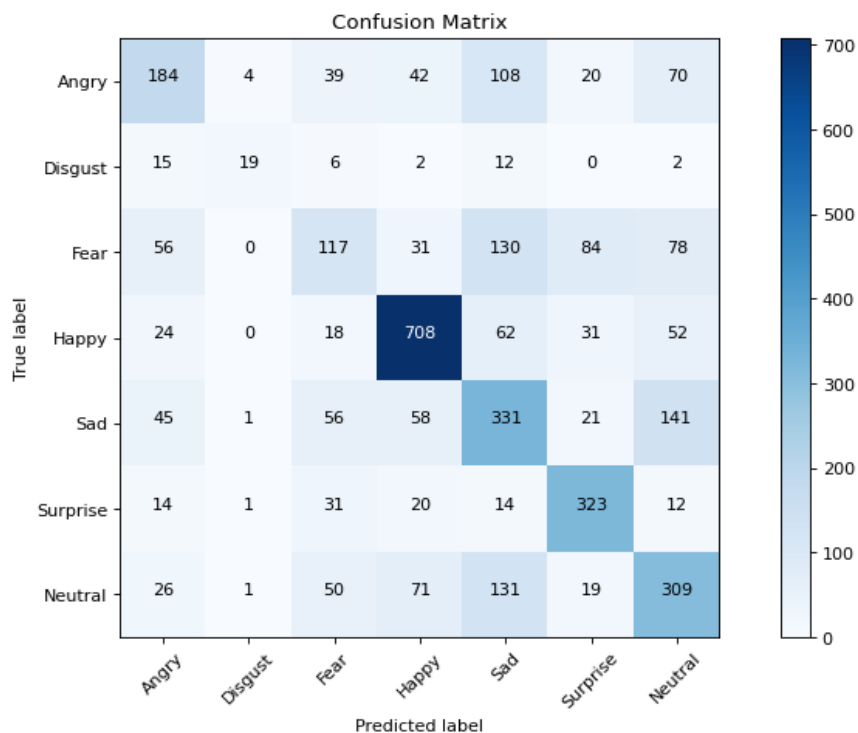


Figure 60 – Confusion Matrix of Second Model

		precision	recall	f1-score	support
Angry	0	0.51	0.39	0.44	467
Disgust	1	0.73	0.34	0.46	56
Fear	2	0.37	0.24	0.29	496
Happy	3	0.76	0.79	0.78	895
Sad	4	0.42	0.51	0.46	653
Surprise	5	0.65	0.78	0.71	415
Neutral	6	0.47	0.51	0.49	607
accuracy				0.55	3589
macro avg		0.56	0.51	0.52	3589
weighted avg		0.55	0.55	0.55	3589

Figure 61 – Metrics for the Second Model

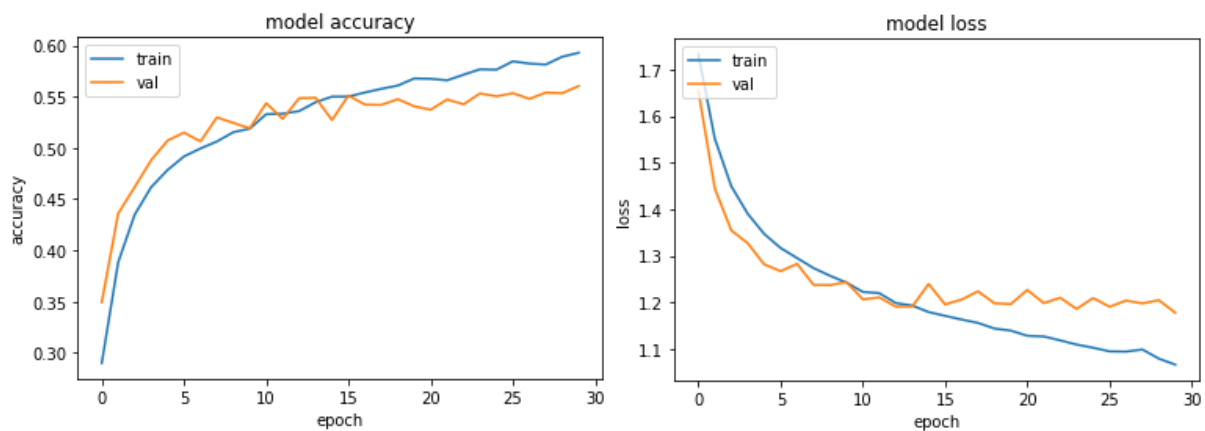


Figure 62 – Accuracy and Loss Charts for the Second Model

4.6. Conclusions

In conclusion two models were trained and evaluated. The first model is more complex and predicts with relatively high accuracy on both the validation data and the real-world camera feed data. However due to technical issues the confusion matrix and other in-depth metrics could not be displayed. The second model was implemented differently and detected the facial expression with a lower accuracy. However, with the second model the displaying of the confusion matrix and other metrics was successful which allowed for more in-depth evaluation. The plan going forward is to display the metrics for the first model and train the model on the JAFFE and FacesDB datasets. Data augmentation techniques can be employed in the second model to try and increase the accuracy.

5. Testing and Evaluation

5.1. Introduction

This project will be continuously tested throughout the development cycle. The CRISP and Agile methodologies are being followed and both have testing in their cycle. With Agile testing will occur at each sprint and the implemented functionality for that sprint will be tested. Each unit will be tested to ensure it is working as intended and provides the required functionality. In the CRISP-DM methodology evaluation is one of the stages and it involves evaluating the model with the designed testing plan. Throughout the development of the project changes are consistently committed to GitHub to ensure that changes can be rolled back to the previous working version in the scenario that major errors occur.

Black-box testing involves testing the functionality/system with no prior knowledge of the internal workings of the system which includes the design/architecture etc. This testing can be done via the user interface and a user provides some type of input whether it be a mouse click or keypresses and then observes the output of the system. This type of testing can be used to find bugs and errors in the user interface, error handling and functionality errors etc [\[Citation\]](#).

White-box testing involves looking at the internal structure of the system and more specifically at the code. The person looking at the code has prior knowledge of the system. White-box testing can be automated with methods and classes that input many possible input types into the unit in an attempt to discover errors and ensure that the correct input gives the correct output. Automated testing is very useful as the system changes and new functionality can break existing functionality that worked previously. It saves the developer/testers from having to manually check to make sure that the functionality is still working which is time consuming [\[Citation\]](#).

Grey-box testing can be thought of as a combination of white-box testing and black-box testing as the tester has at least some prior knowledge of the internal working of the system. The tester does not necessarily have access to the code, however with the knowledge they possess they are able to conduct more complex tests [\[Citation\]](#).

The following sections will discuss the plan for unit testing, integration testing, system evaluation. The machine learning and facial expression recognition specific evaluation will also be discussed.

5.2. Plan for Testing

5.2.1 Unit Testing

Unit testing will be used to test the functionality of each component that is part of the system. Unit tests can be black box or white box. Black-box testing is done using the user interface and white-box testing involves testing the internal workings of a unit. These will help find the bugs in the system and ensure that the functionality is working.

Test Description	Expected Outcome	Pass/Fail
Researcher register	User will be redirected to login screen and a pop up will notify them that the register was successful	
Researcher login with correct details	User will be redirected to main screen (Dashboard)	
Researcher logout	User will be redirected to login screen	

New project button is pressed	A pop-up form will appear on screen with input fields	
New test button is pressed	The "Create Test" page will open	
Add instruction in create test	A box will appear on screen with an "Enter Instruction" field that needs to be filled out	
Add step to instruction	Another text field will appear in the instruction box	
Add text question in create test	A box will appear on screen with a "Enter Question" field	
Add multiple choice question in create test	A box will appear on screen with a "Enter Question" and "Enter Answer Choice" field	
Add/Remove instruction step	The additional instruction steps that have been previously added will be removed	
Create Usability Test	The researcher clicks on "Create Test" button and the new test is stored in the database	
Click "View" test button in dashboard	The user will be redirected to the "View Test" screen	
Bar chart displaying data	The proportions of the chart are correct and correspond to the values	
Pie Chart displaying data	The proportions of the chart are correct and correspond to the values	
Task select in "Overview"	A drop-down menu will pop up with the tasks that can be selected by user	
Play video	Video will play	
Grading tasks	Clicking "Pass" or "Fail" will change the colour of the button to green or red respectively	
Display journey map	Journey map displays correctly and reflects the data	
Screen is recording	A red border will contour the border of the screen the screen will be captured	
Screen recording can be stopped	User clicks on "Stop" button and the recording will cease	
Move on to next task	The user clicks on the "Finish Task" button and moves on to the next stage of the usability test.	
Video saved locally	The video will be saved to a local directory under the correct name	
Video upload to Vimeo	A link is returned by the Vimeo API and can be played	
Task instruction is displaying to participant	Task instructions for the participant display on screen	
SQL injection working	Data is correctly stored in the database using the SQL query. Data retrieved from database with SQL query.	
Web server REST Entry points working	Data can be passed to or retrieved from the web server via POST and GET requests.	

Facial expressions detected correctly with real users	The user will make various facial expressions in front of the camera and the prediction will display on screen.	
---	---	--

5.2.2 Integration Testing

Integration testing involves connecting the units/components and testing them as a group. This type of testing is conducted to find flaws in the interaction between components.

Modules	Expected Result of Interaction
Local Application, Web Application Entry point	The local application will connect to the entry point of the web application and retrieve test data.
Local Application, FER Model	The local application will obtain the data from the camera and feed it into the model to make a prediction. The local application will then process that data.
Front-end, Middle-tier (Back-end)	The front-end of the web application will connect to the back-end server and retrieve/send data.
Middle-tier (Back-end), Database	With the use of the DAO classes the back-end server will connect to the database.
Local Application, Vimeo	The video post usability test will be uploaded to Vimeo with the use of the Vimeo API and a video link will be obtained.

5.2.3 FER Model Testing

Individuals who have no prior knowledge of the system will come in and test the FER model as a single component. The setup will be simple and will involve a window displaying the input from the camera and the prediction of the facial expression. The user will be asked to make various facial expressions and the accuracy of the model will be recorded. The camera position and the angle of the participant's face will be tested for later evaluation. At the end the users will answer the multiple-choice questions shown below.

1. How well has the model predicted your facial expressions, generally speaking?
2. Which expression do you feel was best predicted?
3. Which expression do you feel was worst predicted?

5.3. Plan for Evaluation

The system will be evaluated in two ways. Firstly, the FER model will be evaluated on the metrics such as accuracy, precision, recall, F1 score. These can be displayed using a confusion matrix and line charts. The FER model has been trained on the training set and tested on the validation and test sets to ensure that the evaluation is accurate as the model has not seen that set before. These metrics will be compared to the state-of-the-art metrics for that particular dataset and model.

Finally, the model will be fed data from the camera to truly see how it performs in a real-world environment. The plan is to have a batch of 20 individuals come in and test the system. The model will be evaluated with the data collected from these participants to determine the optimal angle of the camera, the angle of the participant's face in relation to the camera and the detection accuracy of the model.

To evaluate the usability testing application as a whole there will be two types of participants. The first are the researcher/developer type individuals that have some understanding of usability testing. This is important as individuals who have no understanding of usability testing and the field will simply not understand the purpose of the application. Setting up a usability test and viewing the data is unlike other applications and websites designed for the general population such as shopping websites that the user already understands the purpose of.

To the UsabCheck application, a usability test created with the UsabCheck application will be used to determine the usability of the application itself. The researcher type participants will be asked to follow the instructions on screen telling them to create their own usability test etc. This will test both the local application that runs the usability test as well as the web application used to create tests.

The second batch of participants that have no knowledge of the field will be brought in to test only the local application. A usability test for an arbitrary website will be created and the participants will follow the instructions. The facial expression recognition data and other data will be obtained. The participants will be asked questions about their emotions to determine if the data reflects how they felt. Other questions about the usability of the local application will be asked. These questions will include questions based on Nielsen's 10 Heuristics [\[Citation\]](#).

5.4. Conclusions

In conclusion in this section has covered the various types of testing which include unit testing, integration testing, white-box testing, black-box testing and grey-box testing. The unit and integration tests have been defined and the means by which the FER model will be tested and evaluated has been explained. System evaluation will be conducted with two types of users who will test the system. Some of the testing will be conducted using the UsabCheck application as that is the purpose of the application.

6. Issues and Future Work

6.1. Introduction

This section will cover the issues encountered and how these have been or will be overcome. The potential risks of the project will be discussed and the contingency plans in the scenario that the risk occurs. After that, the plans and future work will be discussed.

6.2. Issues and Risks

One of the issues encountered was the displaying of a confusion matrix for the first model. The values did not appear to match the accuracy of the model which performed very well when the images were fed from the camera. This will need to be investigated further, but other than that the model appears to work fine otherwise. Another issue encountered was with the second model. Although the accuracy was not much worse than the first model, the performance on the images from the camera was not great and as a result it cannot be used in a production environment. An attempt could be made to try and increase the accuracy of the model with data augmentation, however, there may not be a need for this as the first model is adequate.

The issues encountered at the beginning were to do with setting up the Anaconda environment and TensorFlow. Firstly, installing python libraries with “conda” instead of “pip” caused great problems and took a few days to figure out. The Conda installer aims to do more than pip in terms of installing the packages. Pip only installs Python packages and Conda installs packages that contain software in other languages. Another issue encountered was with installing the TensorFlow and Keras libraries. There were issues with running the libraries on the GPU and additional code had to be added when importing the libraries. The whole process was time consuming and difficult to set up.

Although not a blocker type issue, the AffectNet dataset could not be downloaded as the researchers who published it encountered bandwidth problems due to the demand of the dataset. The dataset would be very useful in this project and a request has been sent, however, a response is yet to be received.

The potential risks and possible issues will be discussed next. There are risks involved with uploading the video's that will be embedded on the website. Should there be a problem with the Vimeo API and uploading of the videos, the videos can be uploaded to the web server directly and stored there. The reason Vimeo is used is because the server generally doesn't have as much storage space. In the worst-case scenario, the videos can be manually uploaded to YouTube and embedded in the web application that way. The reason they will not be uploaded to YouTube automatically is because of the severe API restrictions.

Another potential risk is the implementation of the journey map. The journey maps implemented in JavaScript that have been researched are not exactly the same as what is designed. Their design will have to be modified and there are some potential risks associated with that. In the worst-case scenario, an existing design cannot be modified and it will have to be implemented from scratch. This would be very problematic as implementing a chart from scratch is very time consuming which could get in the way of other things. If the implementation will appear to take too long then the chart might need to be taken out entirely as to not hinder the development of the project. This is not a problem however as the emotion markers on the video will display where each emotion occurs. The journey map would be nice to implement as an additional way to view the data and is therefore low priority as it is not crucial.

6.3. Plans and Future Work

Front-end

Although the prototypes have been created for the pages the development of the web application has yet to begin. Since the front-end is dependent on the middle-tier for the register, login and other dynamic functionality, the development of the front-end will not begin until the entry points in the middle tier are set up and the middle tier connects to the database.

The pages that will be implemented will be the Register, Login, Dashboard, Create Test, View Test (Overview) and View Test (Recordings). As mentioned previously the ReactJS library will be used to aid development of the front-end. The front-end will connect to the back-end via REST entry points. There is a lot involved with the front-end as the test creation questions and tasks can be dynamically added, removed and moved up or down. The viewing of test results will include bar charts and pie charts, a journey map, a video player with emotion markers, and grading of the tasks. The data will need to be retrieved, processed and sent to the back-end.

Middle-tier

This is the Java web server that will receive requests from the front-end and the local application. As such it will be tasked with processing these requests and access the database via the DAO classes. This will include the registering the user and verifying their login credentials, creating and deleting projects, creating and deleting tests. The test data must be retrievable by the local application so that it can perform the required operations and the data from the usability test after it is conducted must be stored. This may be time consuming as there is no access to existing projects in Java meaning code cannot be copied and pasted to set up the basic architecture which can then be customized to suit the project's requirements. This would include the entry points, a DAO class, servlets, data parsing, JSON object access, dependencies etc. Although these components are very familiar from working in industry for several months, they have yet to be put into practice on a personal project made from scratch.

Database

As mentioned previously it is a MySQL database and the ERD has been created and is ready to be implemented. [\(More on this\)](#)

Local Application

Facial Expression Recognition

6.3.1. Sprint Chart

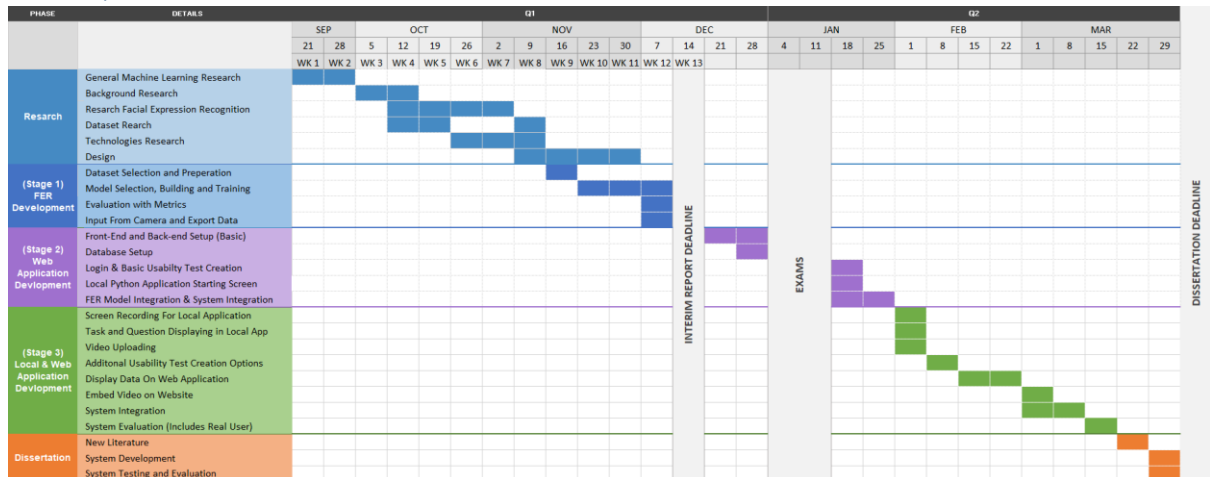


Figure 63 – Sprint Chart Overview (All Stages)

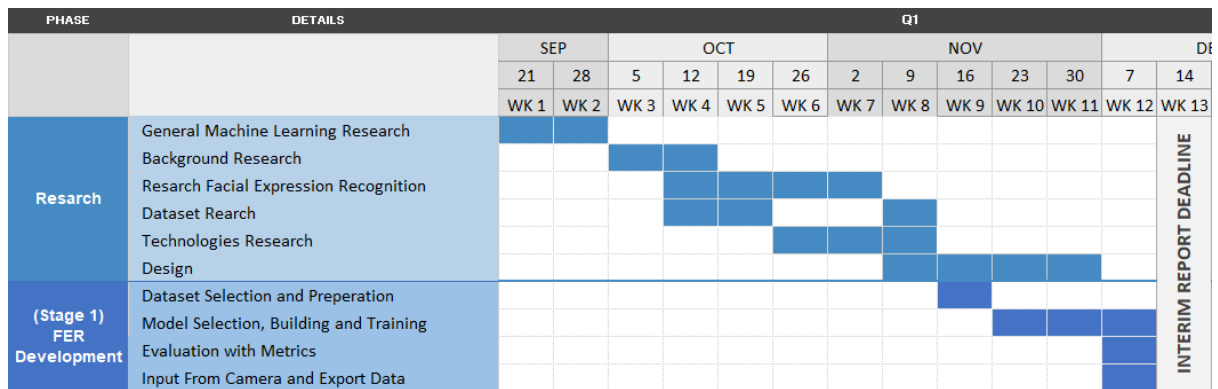


Figure 64 – Sprint Chart First Two Stages (Zoomed in for Readability)

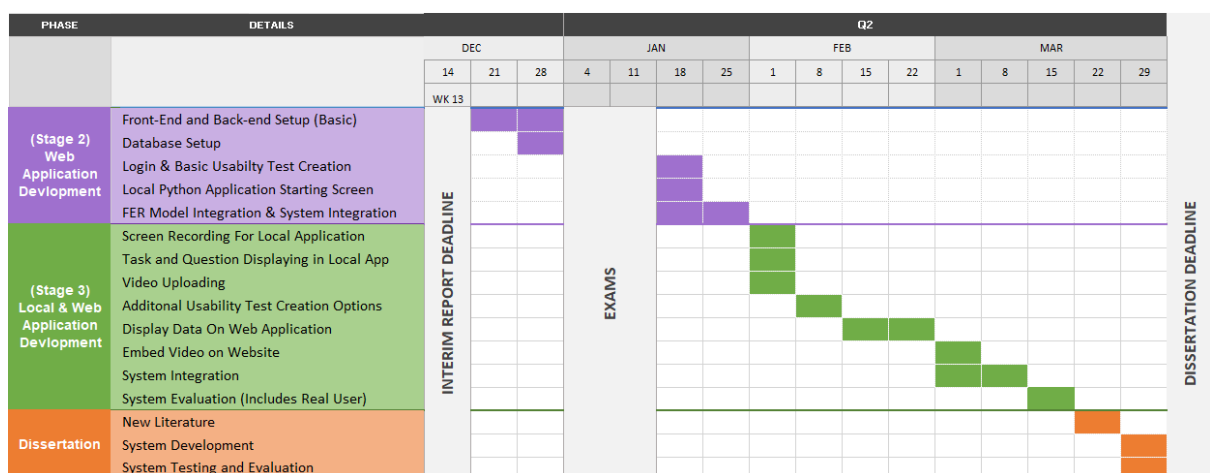


Figure 65 – Sprint Chart Last Three Stages (Zoomed in for Readability)

Bibliography