



UsabCheck – Usability Testing Suite

Interim Report

DT228
BSc in Computer Science

Damian Wojtowicz
C17413722

Andrea Curley

School of Computer Science
Technological University, Dublin

14/12/2020

Abstract

The return on investment for usability is on average a 135% increase in the desired metrics with only 10% of the budget invested. The design flaws within applications make the applications more difficult to use and these flaws can be discovered through usability testing. The information gathered from a usability test when used to improve the application can improve the user experience leading to greater user productivity, satisfaction and as a result greater profits for the business.

UsabCheck aims to provide a usability testing suite which can be used to conduct usability studies and to investigate the use of facial expression recognition technology to improve software usability testing and user experience testing. The project involves two applications, one of which is used by the participant to take a usability study and another is used by the researcher to configure tests and to display the data and findings.

The UX researchers and developers who wish to test their software can configure usability studies which involve tasks and questions that participants will complete. During the usability test deep learning is used to automatically detect the participant's emotions. The data gathered which includes the recording of the participants screen, timestamps of emotions and the participants input is presented to the researcher for further analysis. The advantage of automatically detecting emotions is that it has the potential to help researchers identify the problem areas more easily and effectively in their software.

Technologies used: Spring, Java, Maven, React, Python, TensorFlow, MySQL

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:

Damian Wojtowicz

Damian Wojtowicz

14/12/2020

Acknowledgements

I would like to thank my project supervisor Dr. Andrea Curley for her guidance, encouragement, and support throughout the project. I like to thank Dr. Damian Gordon for his hard work in coordinating the final year project module. I would also like to thank Dr. John Gilligan and Dr. Susan McKeever for their support in the area of machine learning and artificial intelligence.

Finally, I would like to thank my family and my friends for their support throughout the project.

Table of Contents

1. Introduction.....	8
1.1. Overview.....	8
1.2. Project Description.....	8
1.3. Project Aims and Objectives	9
1.4. Challenges	9
Project Scope.....	10
1.5. Thesis Roadmap.....	10
Chapter 2: Research	10
Chapter 3: UsabCheck Design.....	10
Chapter 4: UsabCheck Development	10
Chapter 5:	10
Testing and Evaluation.....	Error! Bookmark not defined.
Chapter 6: Issues and Future Work	11
2. Research.....	12
2.1. Background Research	12
2.1.1. Emotions in Usability Testing.....	12
2.1.2. Facial Expression Recognition Research	15
2.1.3. Usability Testing Research.....	17
2.1.4. Machine Learning Research.....	17
2.1.5. Dataset Research	22
2.1.6. Ethical Concerns and GDPR	25
2.2. Alternative Existing Solutions to Your Problem.....	26
2.2.1. UserZoom Go	26
2.2.2. UserTesting	28
2.2.3. Loop11	30
2.3. Technologies Research.....	34
2.3.1. Programming Languages	34
2.3.2. Python Libraries	34
2.4. Usability Testing Technologies	36
2.4.1. Usability Testing	36
2.4.2. Video Uploading.....	36
2.4.3. Website Hosting.....	36
2.4.4. Databases	37
2.5. Existing Final Year Projects.....	37
2.5.1. Detecting Bot Twitter Accounts using Machine Learning.....	37
2.6. Conclusions.....	38

3. UsabCheck Design	40
3.1. Introduction	40
3.2. Software & Data Mining Methodologies	40
3.2.1. Scrum Agile	40
3.2.2. CRISP-DM	43
3.3. Overview of System	44
3.4. Web Application	48
3.4.1. Front-End	48
3.4.2. Database	56
3.5. Local App	58
3.6. Facial Expression Recognition & Machine Learning	60
3.6.1. Design With CRISP-DM Methodology	60
3.6.2. Discussion	61
3.7. Conclusions	62
4. UsabCheck Development	63
4.1. Introduction	63
4.2. UsabCheck Web Application	64
4.2.1. Backend Server (Java)	64
4.2.2. Front-End (JavaScript - React)	71
4.3. Local Python Application	98
4.3.1. Introduction	98
4.3.2. Overview	98
4.3.3. User Interface	99
4.4. Machine Learning	110
4.4.1. CSV to Image Converter (CRISP Data Preparation Stage)	110
4.4.2. Model Training (CRISP Modelling Stage)	112
4.5. Conclusions (Machine Learning)	117
5. Testing and Evaluation	118
5.1. Overview	118
5.2. Testing Introduction/Plan	119
5.3. Web Application Testing	120
5.3.1. Unit Testing	120
5.4. Local Python Application Testing	121
5.4.1. Unit Testing	121
5.4.2. Automated Testing	122
5.5. System Integration	123
5.6. Evaluation Introduction/Plan	124

Covid-19 Restrictions	124
Evaluation With Participants	124
FER Model Evaluation Overview	126
Local Application Evaluation Overview	126
Web Application Evaluation Overview	126
5.7. Web Application Evaluation	127
5.7.1. Expert Evaluation	127
5.7.2. Nielsen Heuristics [Source]	128
5.7.3. Usability Testing with Participants	130
5.7.4. NoCoffee (Impaired Vision Simulator)	133
5.8. Local App Evaluation	134
5.8.1. Expert Evaluation	134
5.8.2. Nielsen Heuristics	134
5.8.3. Usability Testing with Participants	136
5.8.4. Emotion Timelines From Blinkee Usability Study	139
5.9. FER Model Evaluation (CRISP Evaluation Stage)	140
5.9.1. Model 1	140
4.4.2. Model 2	148
6. Conclusions and Future Work	151
6.1. Introduction	151
6.2. Project Conclusions	151
6.3. Challenges and Risks	152
6.3.1. Risks	152
6.3.2. Machine Learning Challenges	152
6.3.3. Web Application & Local Application Challenges	153
6.4. System Improvements & Future Work	154
6.4.1. Web Application	154
6.4.2. Local Application	154
6.4.3. Facial Expression Recognition	155
6.4.4. Final Reflections	155
6.4.5. Sprint Chart	156
Bibliography	161
Appendix	168
Appendix 1: Blinkee Usability Study Data	168
Appendix 2: FER Model 1 – Testing & Evaluation With Participants	172

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 [2].

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 (1992) [3] 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 and the cost of further testing would be higher than the returns. Faulkner (2003) [4] also concluded that testers may want to include more than five participants. By randomly selecting 5 participants out of a batch of 100 individuals, it was found that on average these 5-person samples found 85% of the problems. However, the results by in research paper by Bastien *et al* (2003) [5] 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. To create the usability test, the researcher will provide a link to the website they want to conduct the usability test on and provide instructions for the tasks they want the participant of the test/study to complete. They can also create questions they want the user to answer before or after a task.

The *UsabCheck* web application will display statistics on how participants have performed in the tasks and their responses to question. The researcher will have the option to view the screen recording and a camera recording of the participant's face while they are completing the tasks. This includes the audio of the participant speaking. The *UsabCheck* application will display 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 display the tasks and questions for the participant to complete. The local application will also record the screen and monitoring the participants facial expressions. After the usability test is conducted the video will be uploaded to a 3rd party service and a link to that video along with the other data will be sent to the web application that will store it and later display it to the researcher.

1.3. Project Aims and Objectives

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

1. Create a usability testing web application that allows the researchers to create usability study tests and view the results of the usability test.
2. Allow the researchers to view the results of the tests on the website. This includes viewing of the videos, answers to questions and questionnaires, and viewing a journey map with the participants facial expressions.
3. Provide a local (runs on the machine the test participant is using) usability testing application that will record the screen, record the test participants face from the camera, predict the participants facial expression, show the participant the test instructions, ask the participants questions and upload the data to the cloud.
4. For the above aims to be achieved, more specific goals and objectives have to be met. This includes training a machine learning algorithm that will classify facial expressions, implementing a web application and implementing a local application.
5. The machine learning algorithms and datasets will have to be researched, implemented and evaluated.
6. The videos need to be uploaded to a secure storage location and embedded on the website.

1.4. Challenges

1. Determining the best dataset for the face expression recognition which will be a very important factor in the success of the classification.
2. Determining and selecting the best FER model which will have a great effect on the accuracy of face expression classification.
3. Thorough research into the usability testing aspect and how FER can be integrated
4. Selecting the right video uploading method/service.
5. Due to the huge scale of the project, time management and project management is very important and need to be followed. Not exactly a challenge, however it is something that is crucial to the success of the project.
6. Displaying the journey map will be a challenge as at best it will involve heavily modifying an existing implementation and at worst it would be implementing one from scratch.

Project Scope

The scope of this project involves designing and implementing a usability testing 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. The project will involve two applications; a web application and a local application.

1.5. Thesis Roadmap

Chapter 2: Research

This section of the document will discuss the research done in the areas of facial expression recognition, usability testing and machine learning. Machine learning algorithms and datasets for facial expression recognition (FER) will be discussed. After that the existing usability testing application will be reviewed, followed by the programming languages, libraries and technologies. Finally, a previous final year project will be reviewed.

Chapter 3: UsabCheck Design

Firstly, the project consists of two sub projects which is the facial expression recognition model and the usability testing applications. The prototype design chapter will begin with discussing the methodologies used to manage this project. After that the high-level components of the system will be discussed which includes the architecture and a high-level flow chart. Followed by that will be the use case diagrams, screen prototypes and the database ERD. Finally, the facial expression recognition related design will be discussed.

Chapter 4: UsabCheck Development

Chapter 5:

Testing and Evaluation

The different types of testing are discussed, followed by a unit test plan, an integration test plan and a plan to test and evaluate the FER model. Following on the evaluation plan for the system as a whole is discussed.

Chapter 6: Conclusions and Future Work

This chapter will cover the issues encountered in the development process and the potential issues or risks that could occur. A contingency plan is made for the risks should they happen. After that the plans for the future development of the project is described.

2. Research

2.1. Background Research

2.1.1. Emotions in Usability Testing

Landowska (2015) [6] 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 considering 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. For example, boredom can be expressed as subtle sadness which is not the most intuitive representation of the emotional state.

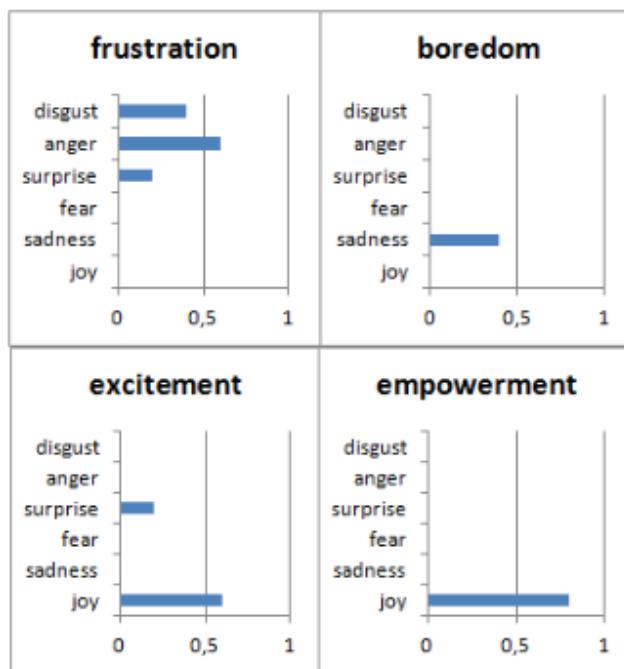


Figure 1 – Ekman's Six Basic Emotion Model

Whissel's wheel emotion model (See Figure 2) has also 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.

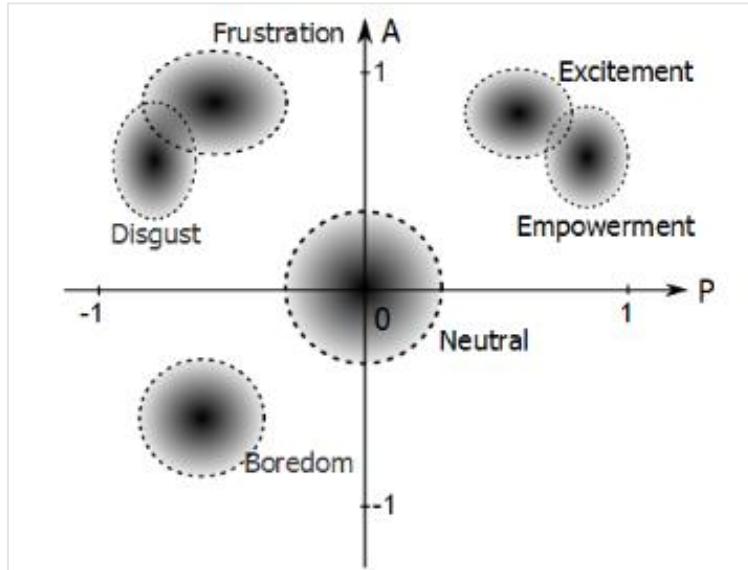


Figure 2 – Whissel's Wheel Emotion Model

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% 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 medium to high 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 could be useful in providing the researchers who will use the *UsabCheck* application with more intuitive data.

Esterwood (2018) [7] 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 displays the user's emotions along with the task they are trying to complete. Figure 3 illustrates the journey map they used in their explanation. 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 indicated by an incline or a decline respectively.

However, this chart does not display exactly what emotion was experienced which makes this approach debatable. Surprise was also 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 can be the result of bad design.

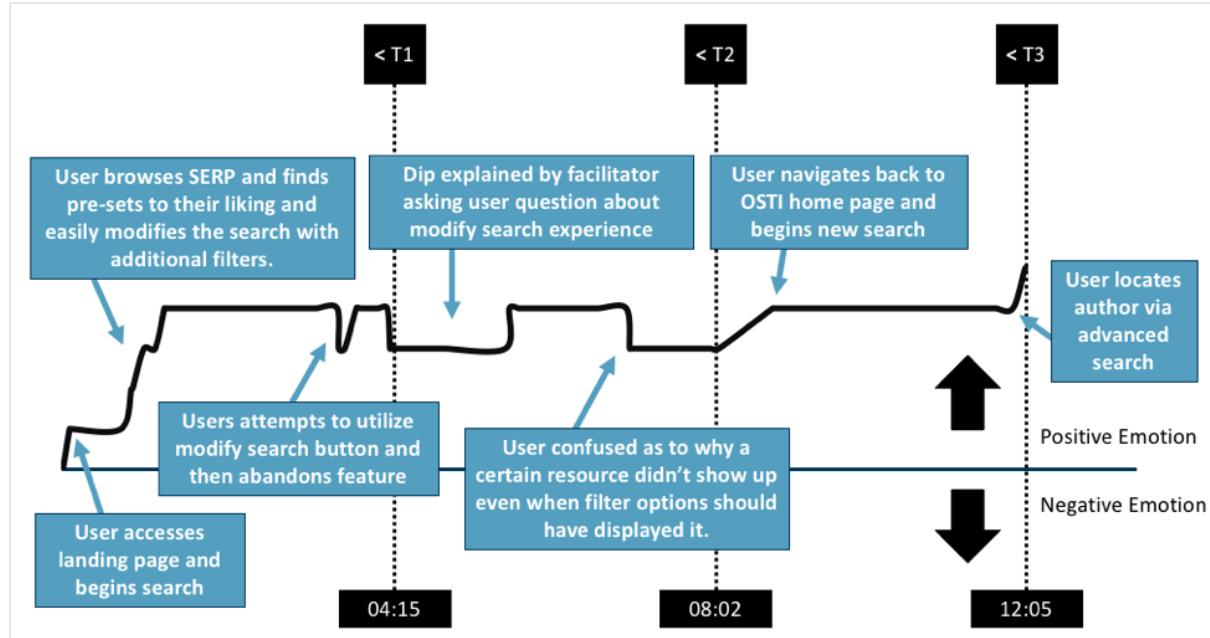


Figure 3 - Example Journey Map

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 camera exclusively for the face. In conclusion this study has provided insight into displaying the emotion data in the context of usability testing that is visually 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 average the emotional states of several participants is very insightful. Users of varying degrees of ability will spend different amount of time on each task and averaging the emotional states to be displayed on a journey map is definitely challenging. 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.

A study by Thomas Schmidt *et al.* (2020) [[Source](#)] “investigated the relationship between emotion recognition software and usability metrics”. They investigated if there is a correlation between data gathered by emotion recognition software and the usability of the software in relation to the SUS (System Usability Scale) score and the task completion rate. The emotion data was obtained from the text, speech, and face. The results show that there was a “weak but significant correlation” between the text which was obtained through the participant’s speech and SUS scores. However, no significant correlation was found between categories of emotion and the completion rate of tasks or the SUS score. Additionally, no significant correlation was found between the valance of emotion and the completion rate of tasks or the SUS score.

2.1.2. Facial Expression Recognition Research

Halder *et al.* (2016) [8] 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 of facial expressions 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 have 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 network is still 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 should that route be taken.

Gaurav (2018) [9] 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, the higher the accuracy the better the model will perform in the application and the less errors it will make. 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 and this architecture will be discussed in detail in another section of the report. A combination of 3 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 confusion matrix in Figure 4 below 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% (11/44) of the time when the actual class was “Disgust”. The model often 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, testing the model, and 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.

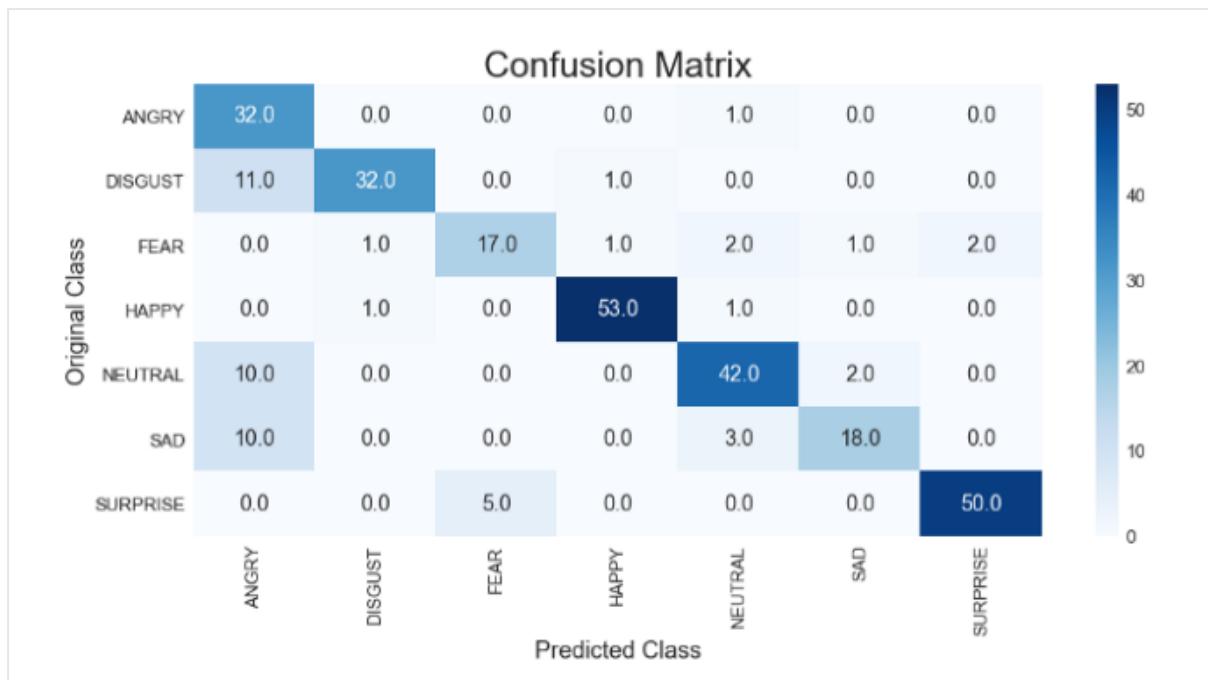


Figure 4 - Confusion Matrix Example

2.1.3. Usability Testing Research

Bastien (2009) [10] reviewed the methods and procedures in usability testing. The usability testing applications the research is based on are in the field of medical and healthcare informatics. *UsabCheck* will be designed to allow usability testing on virtually all desktop applications, meaning this research is relevant. They discussed the previous 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/researchers 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 that 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 of asynchronous testing is that the user's equipment such as the microphone and camera need to be of a certain standard to collect the data accurately. 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, machine learning model 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) [11] explains the confusion matrix and how the values can be used to calculate precision, recall, accuracy, specificity and the AUC-ROC curve. Referring to Figure 5 for the following explanation of the confusion matrix, the cell where the row and column with the same label match is the "True Positive" value. In other words, the value which represents 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 "False Positive" value is when 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 the "Angry" label 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", meaning that the model predicted "Angry" and the true label was "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)	
		Positive (1)	True Positive	False Negative	
		Negative (0)	False Positive	True Negative	

Figure 5 – Confusion Matrix Labels

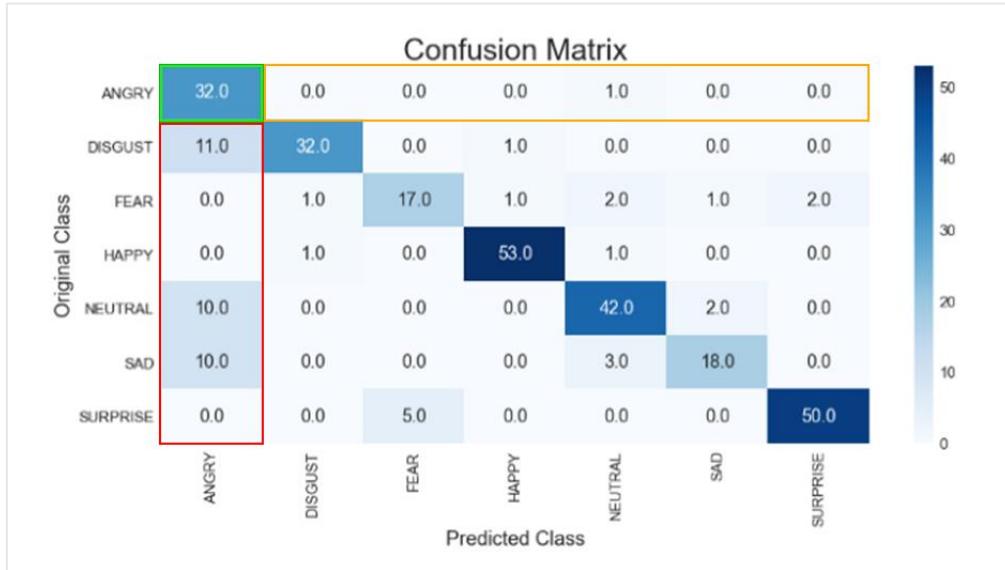


Figure 6 – Confusion Matrix

The article by Koehrsen (2018) [12] 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 in the scenario that the sample size is large and the ratio of the classes is extremely uneven. This is known as the imbalance classification problem. The metrics need to be balanced and tailored to the application. 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.

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

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

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

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

$$\text{Accuracy} = \frac{\text{True}}{\text{Total}}$$

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

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

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 [13] and face recognition [14], 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% [15] 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 are 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 [16]. The fully connected layer is the final layer and uses the results from the previous layer to classify an image [17][18].

Visual Geometry Group (VGG) Model

The VGG-16 Model [19] is a famous CNN model that achieved a 92.7% test accuracy in the ImageNet dataset [20], making it the winner of the ILSVRC (ImageNet Large Scale Visual Recognition Competition) in 2014 [21]. The dataset the model was trained on consisted of 14 million labelled images belonging to 1000 classes [22]. This model can be used to train a facial expression recognition model as shown by Kusuma *et. al*/2020 [23]. 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. The architecture with the layers mentioned is illustrated in Figure 7.

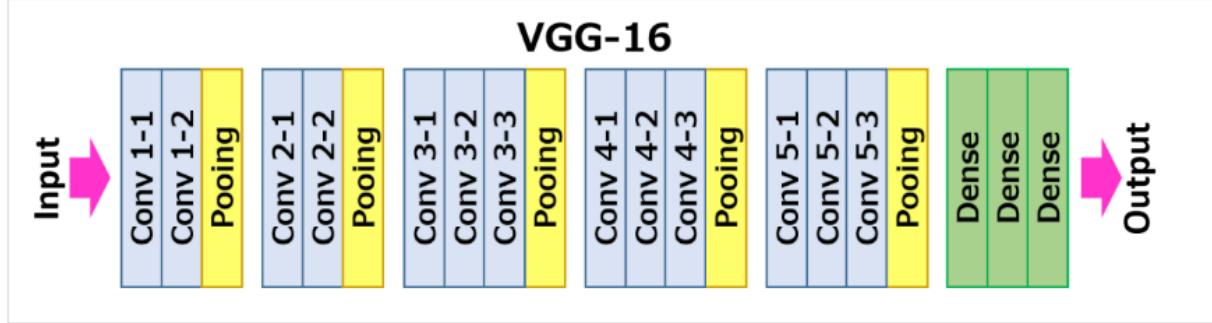


Figure 7 – VGG-16 Architecture

Support Vector Machines (SVM)

Support Vector Machines (SVMs) can be used to solve classification and regression problems. 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 [24].

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 [25][26].

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 of the training images which is later used in the classification of a new mage. An example of this histogram is illustrated in Figure 8. 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 [27][28].

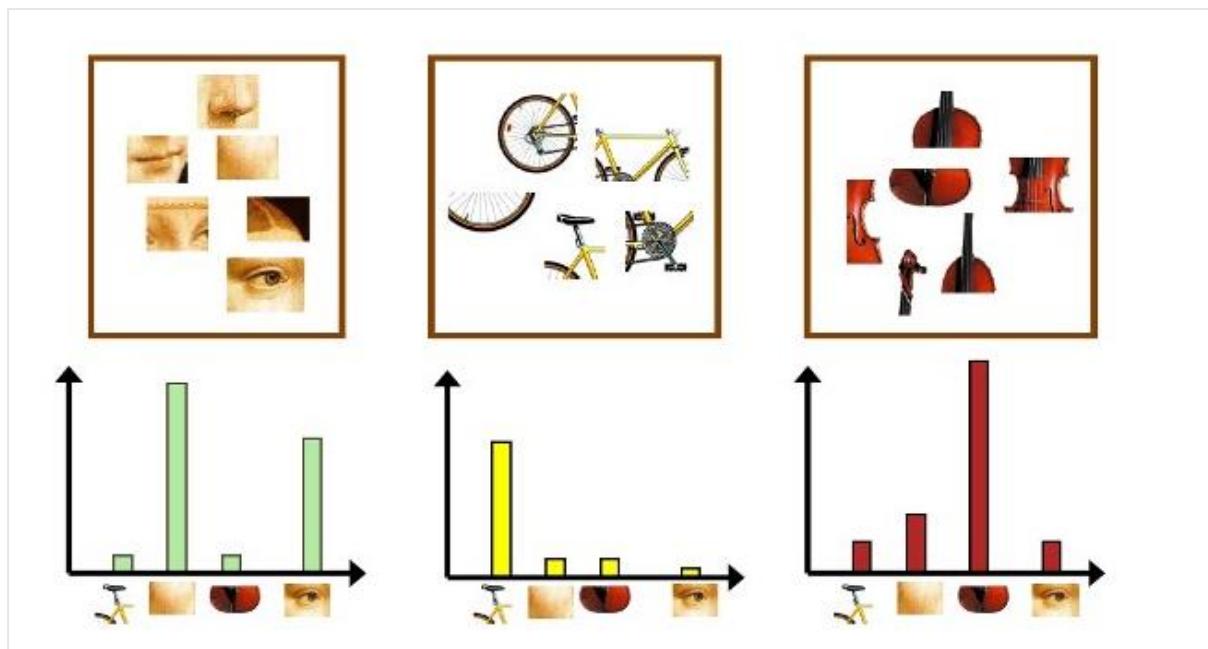


Figure 8 – Bag of Visual Words Histogram Example

Face Detection

Several methods for face detection are discussed in the article by Dwivedi (2018) [29]. The methods for detecting a face can be feature-based, appearance-based, knowledge-based and template matching.

Feature based methods detect the face by extracting features of the face such as the edges and corners in an image. In other words, it looks at the structure of the image and involves training a classifier that can tell the difference between an image that contains a face and one that does not. This method performs very well with an accuracy as high as 94%. An example of this method is the Haar Cascade features algorithm. It is an object detection machine learning algorithm and can be used for face detection with high accuracy. A pre-trained Haar Cascade for frontal face detection can be easily implemented.

An appearance-based method involves training a machine learning algorithm to learn the characteristics and patterns of a face. The data is presented to the algorithm that has no prior knowledge and through statistical analysis it learns what a face looks like.

Knowledge-based methods detect the face in the same way that a person would describe a face. In other words, there is a set of rules that determine if the image is of the face and these rules can include the presence of a mouth, nose, eyes etc. of a certain size and relative position. For example, the mouth is below the eyes. This approach on its own is not sufficient enough to consistently detect faces in images.

Finally, a template matching method uses a pre-defined face template to detect the faces by checking how closely the input image matches the template. Although this method is easy to implement it is not sufficient enough for consistent and accurate face detection. Deformable templates can help with increasing the accuracy of the method.

In conclusion, the method chosen for face detection was the Haar Cascade algorithm as it is very effective and the implementation is simple as the model is already pre-trained.

2.1.5. Dataset Research

JAFFE (The Japanese Female Facial Expression (JAFFE))

The JAFFE dataset [30] consist of labelled images of the frontal face. The facial expressions in the dataset are from 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 on its own and data augmentation techniques will have to be used to generate more images from the existing set to increase the amount of data.

FER2013 (Facial Expression Recognition Dataset 2013)

The FER2013 dataset [31] is in the form of a CSV file and consists of 28,000 labelled images in the training set, 3,500 images in the validation set and 3,500 images in the test set. The images are stored in the file as an array of pixel values which can be used to reconstruct the image to a .jpg or .png format should the need arise. The images are labelled with one of seven emotions which are happy, sad, surprise, angry, afraid, disgust and neutral. All images are grayscale and of 48x48 pixel resolution.

AffectNet

AffectNet [32] 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 mean a higher 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

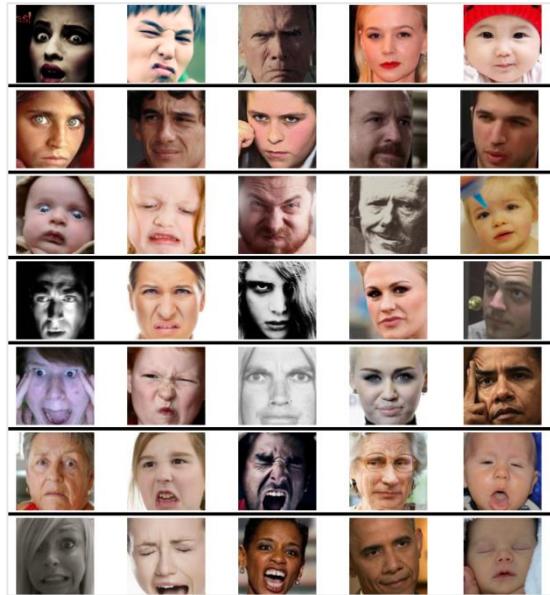


Figure 10 – AffectNet Sample Images

FacesDB

The FacesDB dataset [33] consists of 38 individuals who are both male and female and each individual makes one of each face expression of 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.

FEC (Goole Facial Expression)

The FEC (Google Facial Expression Comparison dataset) [34] is in the form of a .CSV file. The dataset consists of face image triplets with each row in the file containing three faces. It is also specified which of the two images in the triplet contain the most similar face expression. Each image in 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 which allow for the face to be extracted from the image.

As the dataset is in the form of a .CSV file with links to the image, these images that have been 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” [38] was discovered for an application that was said to download all the images from this particular dataset automatically. However, that application used another application under the name “TorCrawler” [39] 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 and performed well. 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 with many objects. The camera will be in a static position and the angle of the participant in question is unlikely to greatly vary. An example of an image is displayed in Figure 11.



Figure 11 – Example of FEC Dataset Image Taken at Angle

Emotic (Emotions in Context)

The Emotic dataset [35] is similar in nature to the FEC Google dataset and 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

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.

2.1.6. Ethical Concerns and GDPR

There have been ethical concerns raised over the use of facial expression recognition and its applications. The ethical concerns do not apply in usability testing in the same way they would generally in other areas such as public safety [36] as an example. This is because when the participant is invited/hired by a company or a developer to participate in a usability study they would have willingly given consent to be part of that study. The facial expression data and any data for that matter is not used for any other purpose than what it is intended for. In a usability study the participant is not and should never be blamed for absolutely anything as the focus is the design of the system and the system. The usability test is designed to test the system and not the participant, meaning the FER results of the usability study will not be used against the participant in any way.

With regards to the datasets, before downloading practically any of the datasets a terms and conditions agreement was signed. This means that the use of the dataset cannot be used for commercial purposes and should be used for educational purposes only amongst other things. The datasets may not be redistributed either which is why it is not uploaded to the UsabCheck GitHub repository.

European Union's General Data Protection Regulation (GDPR) [37] was reviewed to ensure that the UsabCheck application abides by the regulations. This project will not be released for public use and is developed for research purposes only, meaning GDPR will not be as much of a concern. If the application was to be publicly released in a hypothetical scenario a concern could be raised about the uploading of the usability test videos to a 3rd party service, Vimeo.

2.2. Alternative Existing Solutions to Your Problem

2.2.1. UserZoom Go

UserZoom Go [40] 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 illustrates the dashboard, this is the main screen the researcher will see when the login. The view of the participant when they begin the test is displayed in Figure 14. Finally, the results of the usability study are shown in Figure 15.

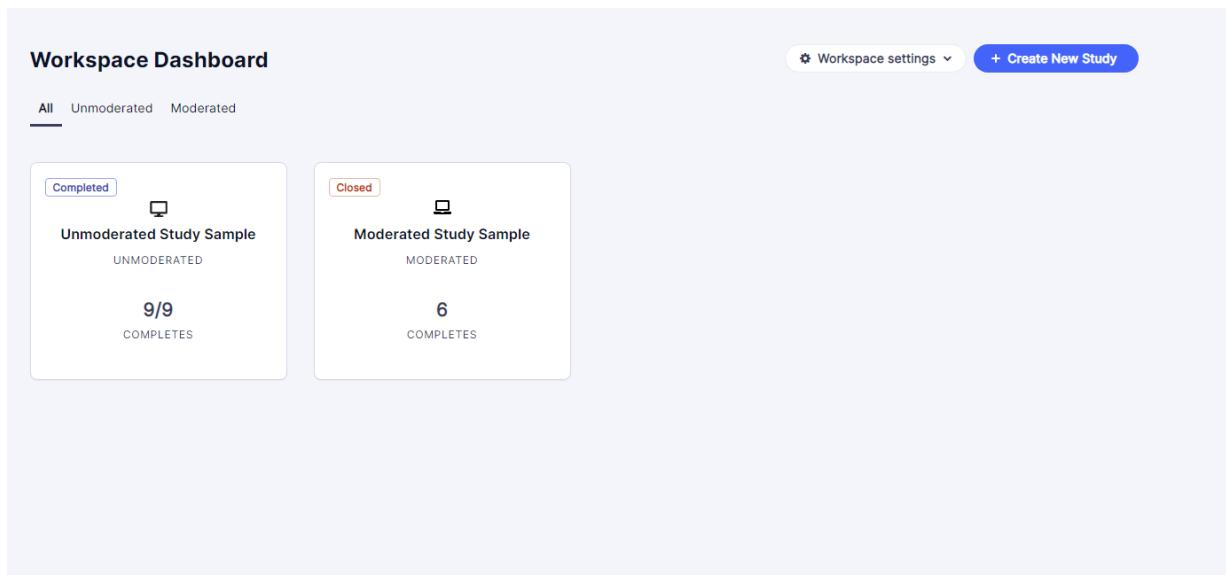


Figure 13 – UserZoom Go Dashboard

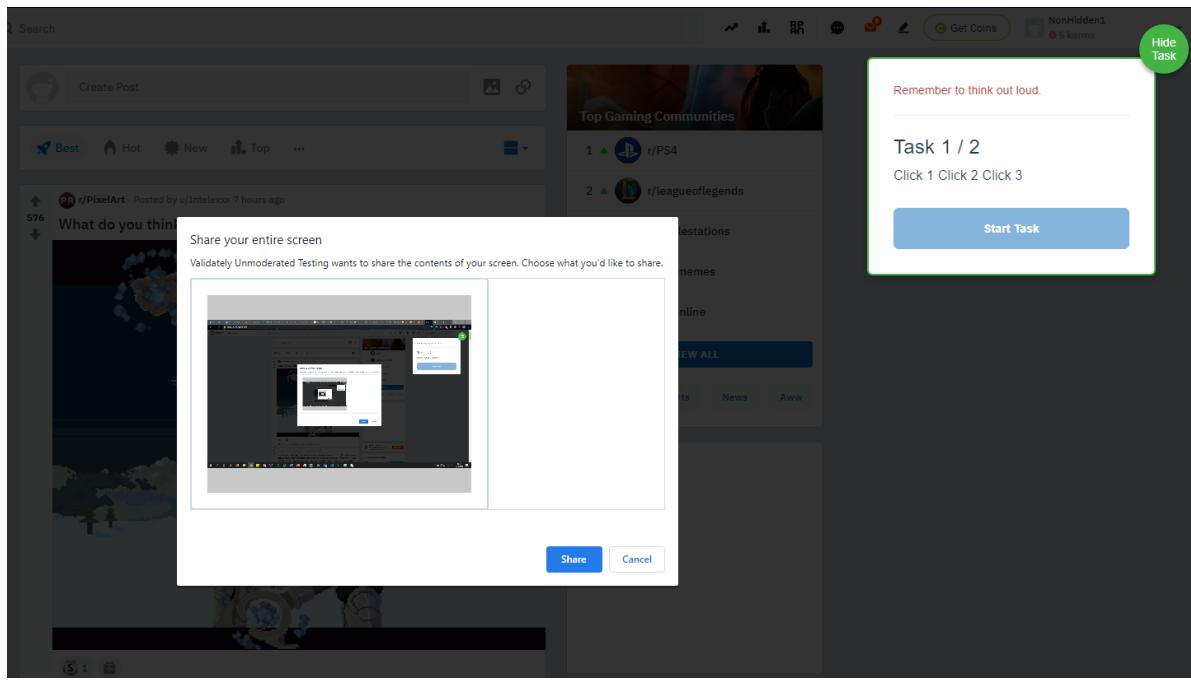


Figure 14 – UserZoom Go Usability Test Start

A screenshot of the UserZoom Go results page. At the top, it says "Screener questions" and "Export individual details including task times, pass/fail, and questions answers in one .csv file." There is a "Download Results" button. Below this, the "Task 1 Analysis" section shows the results for the task "Task 1: Click 1 Click 2 Click 3":

- TIME: 00:00:53 AVG
- PASS: 1 PASSED
- FAIL: 0 FAILED
- UNGRADED: 0 UNGRADED

The "Task 2 Analysis" section shows the results for the task "Task 2: Click 1 Click 2 Click 3":

- TIME: 00:00:28 AVG
- PASS: 0 PASSED
- FAIL: 1 FAILED
- UNGRADED: 0 UNGRADED

Figure 15 – UserZoom Go Usability Study Results

2.2.2. UserTesting

UserTesting [41] 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 it. The steps to be taken from the dashboard (See Figure 16) 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 creation options included selecting the audience (See Figure 17) by demographic such as age, gender, income, country, employee status etc. The participant device selection was between a computer, a tablet and 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. Some question options such as a multiple-choice question, a rating scale and a written response required a premium subscription. The test creation screen is illustrated in Figure 18. 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 which meant that seeing how the data is presented after the usability test has concluded 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 and certain parts of the website were disorientating.

Welcome, Damian! Let's start powering your business decisions with real human insights.

For the next **14 days**, enjoy access to our full subscription platform, which allows you to:

- Recruit your exact target audience with advanced screening and filtering
- Quickly scan test results with written video transcripts
- Measure and compare customer experiences with quantitative test questions

Create a test \$49/video

Upgrade now to get the most business value from your user research.

In **14 days** your trial will end and your account will revert to Individual Plan capability. Upgrade your account to keep your advanced features and get the best results for your organization.

Upgrade today!

Dashboard

Tests 0 Drafts 1 Highlight Reels 0 Folders 0 New

Damian Wojtowicz Search for keywords or title

Type	Name	Created By	Edited On	More
<input type="checkbox"/>	Get familiar with launching a test! Default Folder	UT Employee	Oct 18, 2020	...

Figure 16 – UserTesting Dashboard

User Testing

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
18 to 65+ Remove

Household income (\$)
0K to 150K+ Remove

Gender
 Male Female Remove

Web expertise
 Average web user Advanced web user Remove

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

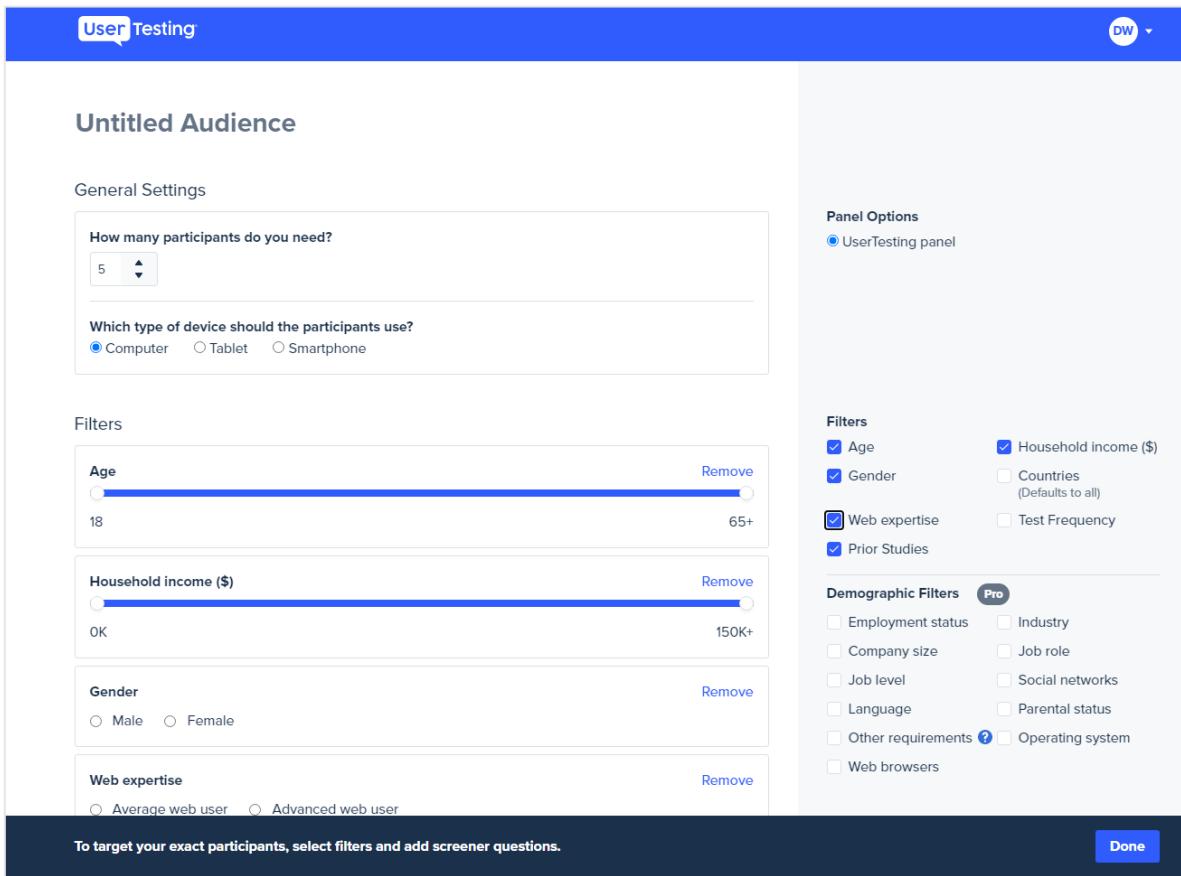


Figure 17 – UserTesting Select Audience

The screenshot shows the 'Test Plan' section of the UserTesting platform. At the top, there's an option to 'Enable Participant View' with a note about recording faces. Below this is a list of six tasks:

- 1 Verbal Response**: A text input field asking participants to explain their initial impressions of the website.
- 2 Task**: A text input field asking participants to describe something they might do on the website.
- 3 Task**: A text input field asking participants to complete a task within two minutes.
- 4 Rating Scale Task**: A text input field asking participants to rate their confidence in completing the task.
- 5 Rating Scale Task**: A text input field asking participants to rate the difficulty of understanding the information.
- 6 Rating Scale Task**: A text input field asking participants to rate the difficulty of understanding the information.

On the right side, there are sections for 'Assets' (URL, Image), 'Tasks and Questions' (Task, Five Second Test, Verbal Response), and another 'Tasks and Questions' section (Multiple Choice, Rating Scale, Written Response). Below these are 'Popular Tasks' and a 'View Examples' link. At the bottom, there's a dark bar with 'Drag and drop tasks, questions, and assets to build your test plan.', 'Preview Test Plan', and 'Done' buttons.

Figure 18 – UserTesting Create Usability Test

2.2.3. Loop11

Loop11 [42] is a website usability testing application in the browser. The researcher can create a new “project” which is a usability test/study. These projects are listed in the main dashboard which is illustrated in Figure 19. Statistics and other information about the overall users’ performance is shown in the project overview along with pie charts that 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. This is illustrated in Figure 20. More statistics about the participants is displayed such as their location, operating system, device and browser which can be seen in Figure 21.

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 in any application. 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 which is illustrated in Figure 23 can be viewed 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 and these are displayed in Figure 22. The questions 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. However, an issue with the website is that the video quality was very low and would not load very fast.

The screenshot shows the Loop11 main dashboard. On the left is a dark sidebar with navigation links: Projects (selected), Themes, FAQ, and Take Tour. The main area has a header with a search bar ('Search Projects'), a 'Create New Project' button, and user profile icons. Below the header is a section titled 'PROJECTS' with tabs: All (1), Closed (1), Launched (0), Draft (0). A single project card is visible for 'Sample Project', which is 'Closed'. It shows metrics: Task Success (47%), Task Fail (42%), Task Abandon (9%), NPS (-68), Participants (21/999), and a 'View Report' button. At the bottom right of the card is a three-dot menu icon.

Figure 19 – Loop11 Main Dashboard

This screenshot shows the Loop11 Project Dashboard for the 'Sample Project'. The top navigation bar includes a search bar, a URL link to 'https://www.tyreagency.com/', and project metadata: UNMODERATED, Modified: 15 Oct 2020, Launched: 18 Jan 2018. Below the navigation are tabs: Dashboard (selected), Tasks, Videos, Questions, Participants, Clickstream & Heatmaps, and a 'View Design' button. The main content area starts with a section titled 'Total tasks & questions' showing success, fail, and abandon rates: Success (47%), Fail (42%), Abandon (9%). It also displays NPS (-68), Lostness (0.39), Avg duration (01:06), and participant counts (27 complete, 21 incomplete). Three pie charts follow, each representing a task: 'Task 1: Clients have worked with' (Success 30%, Fail 48%, Abandon 22%), 'Task 3: How long operating' (Success 78%, Fail 17%, Abandon 4%), and 'Task 5: Apple Pay' (Success 39%, Fail 61%, Abandon 0%). Each chart includes a legend and a table of average metrics: Page views, Time on task, and Lostness.

Task	Success (%)	Fail (%)	Abandon (%)	Page views (avg)	Time on task (avg)	Lostness
Task 1: Clients have worked with	30%	48%	22%	6.4	02:08	0.75
Task 3: How long operating	78%	17%	4%	2.6	00:35	0.25
Task 5: Apple Pay	39%	61%	0%	2.6	00:34	0.17

Figure 20 – Loop11 Project Dashboard

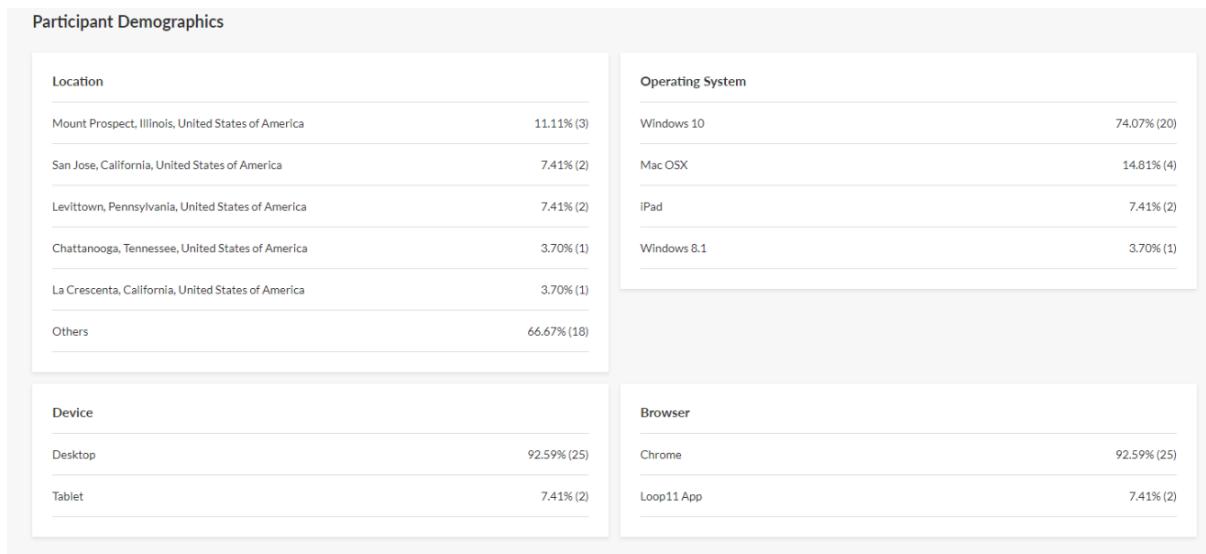


Figure 21 – Loop11 Participant Demographic

Participant Introduction

Task 1
Type: Standard

Do xyz

- Multiple Choice (1 answer)
- Multiple Choice (multiple answers)
- Rating Scale Matrix
- Rating Scale (1 answer)
- Ranking Question
- Open Ended (1 item)
- Open Ended (Multiple items)
- Open Ended (Comments box)
- Net Promoter Score (NPS)
- System Usability Scale (SUS)

+ New Task + New Question

Figure 22 – Loop11 Tasks and Questions



Figure 23 – Loop11 Clickstream

2.3. Technologies Research

2.3.1. Programming Languages

Python

Python [43] 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 Scikit-Learn, 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 [44].

Django [45] 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.

Java

Java [46] is a class-based, object-oriented, platform-independent language. Java can be used to develop the server side of web applications and is the no. 1 choice for developers with over 9 million Java developers worldwide [47]. The popularity of the language and its use in web development makes it an excellent option for developing the *UsabCheck* web application's middle tier. It is a programming language taught in the TU Dublin Computer Science course which makes it familiar and easier to work with.

JavaScript

JavaScript [48] 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 in the middle tier and there is a back-end and front-end framework named Node.js [49] that can be used in the development. There are also front-end frameworks and libraries such as AngularJS [50] and ReactJS [51] in the scenario that the design plan will be to have a Java back-end instead of a JavaScript back-end.

2.3.2. Python Libraries

NumPy [52] 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 [53] 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 in Python. This library can work with NumPy to manipulate the data.

Matplotlib [54] is a library that generates charts and other forms of data visualization in Python. These figures can be interactive and allow the user to zoom in and pan etc. on these figures. This library can be used to display images in Jupyter Notebook and the results of machine learning algorithms can be displayed in various forms . For example, Matplotlib can be used to display a confusion matrix as an example.

Scikit-learn [55] is a machine learning library that includes other libraries such as NumPy, SciPy and Matplotlib to name a few. It is a tool for predictive data analysis and includes implementation of algorithms such as SVM, Nearest Neighbour, Random Forest, K-Means etc.

TensorFlow [56] 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 [57].

Keras API [58] 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 [59] 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 [60] 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 [61] 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 to train a FER model whereby facial features are extracted with OpenCV and used to train a machine learning algorithm. This has hybrid approach has been researched by Halder *et al.* (2016) [8] and discussed in section 0.

2.4. Usability Testing Technologies

2.4.1. Usability Testing

The idea for the usability testing application 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 and this may include a 3rd party service. The website has to be hosted and a database will have to be selected. The technologies for these requirements are listed and described below.

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.

SwiftStack

SwiftStack [62] 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.

StreamingVideoProvider

StreamingVideoProvider (SVP) [63] 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 far more other functionality than is needed for the *UsabCheck* application.

Vimeo

Vimeo [64] 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 and this service is often compared with YouTube. Vimeo has an API that can be used for uploading videos.

YouTube

YouTube is a popular video streaming service where videos can be uploaded and configured in such a way that only people with the link may view the video. YouTube also provides an API [65] that allows users to upload and manage their videos. 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

Tomcat

Apache Tomcat [66] 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 the most widely used Java application server and it is well documented as it has been around for almost 20 years [67].

Google Firebase

Google Firebase [68] 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. Google Firebase is a great option for rapid prototyping and development when building an application from scratch.

2.4.4. Databases

MySQL

MySQL [69] is an open-source relation database management that is well known for its high security, performance and on-demand scalability [70]. A relational database stores the data in tables and each entity is uniquely identifiable. Relational databases must be vertically scaled which often involves purchasing expensive hardware. A relational database is also designed to deal with structured data unlike a non-relational database.

Firestore

Google Firestore is a non-relational database. This means that data is stored in and retrieved from documents and collections unlike a SQL database. This makes it 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

2.5.1. Detecting Bot Twitter Accounts using Machine Learning

Existing projects which involved usability testing have not been found. However, there have been many projects that 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 detect a bot account. There was a total of 4 models for classifying a bot and these were 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, CSS, MySQL database and the Twitter API. Vast amounts of research have been done into each area the application was involved in. 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

This chapter has covered the research in the areas of machine learning, facial expression recognition and usability testing. This chapter has also reviewed the usability testing applications, programming languages, python libraries, video uploading services, website hosting options and databases.

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

Table 1 - Summary of Technologies

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 middle-tier 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.

Table 2 - Datasets Researched

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.
JAFFE	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 be 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 be used. This dataset will be used for comparison with the FER2013 and JAFFE 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.
Haar Cascade	ML Algorithm	The Haar Cascade algorithm is a pretrained face detection algorithm. It will be used in find the bounding box of the face in the images from camera.

3. UsabCheck Design

3.1. Introduction

There are two main sides to this project which are illustrated in Figure 24. The first is the facial expression recognition (FER) machine learning model which involves working with data, training a machine learning model and evaluating it before integrating it into the system. The second side of the project is the usability testing which involves creating two application. These are the local application and the web application. The usability side involves setting up the usability tests, running the usability test, recording the screen and camera, making use of the machine learning model, uploading the videos, uploading the data from the local application to the web server and displaying the data etc. All of this will be explained and discussed in more detail.

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 sub-projects mentioned.

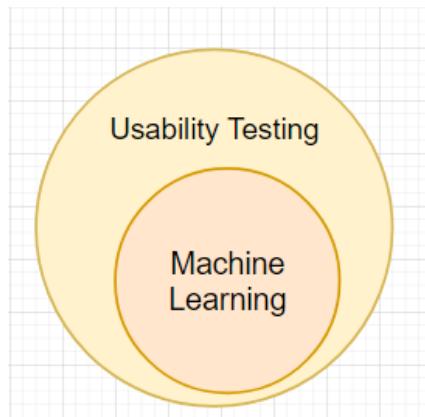


Figure 24 – Usability Testing & Machine Learning Subprojects

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.

3.2.1. Scrum Agile

One of the software methodologies that has been chosen for this project was the Scrum Agile methodology [71]. The Agile methodology is centered around iterative development and allows teams to deliver work with greater predictability which allows for better adaptability to change. The agile software development cycle is illustrated in Figure 25. In Scrum specifically, the development is split into development cycles called Sprints which are no more than four weeks. The project's high-level requirements and goals are defined in the Product Backlog which is a dynamic list of Product Backlog Items (PBIs). This Product Backlog can change depending on the requirements as the project develops. These PBIs 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 has “Increments” which is the usable end-product from a Sprint [72].

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 [73].

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 26 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 Sprint. 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 27 shows the more detailed information of an Epic. For example, the Facial Expression Recognition Epic has 8 issues and 3 of them are already been completed.

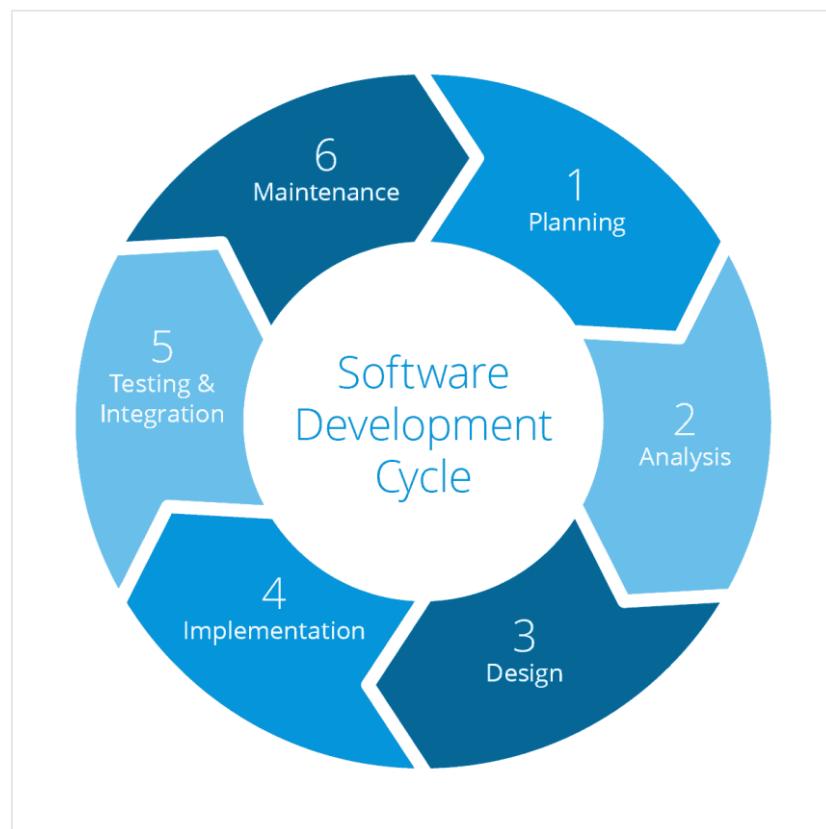


Figure 25 – Scrum Agile Software Development Cycle

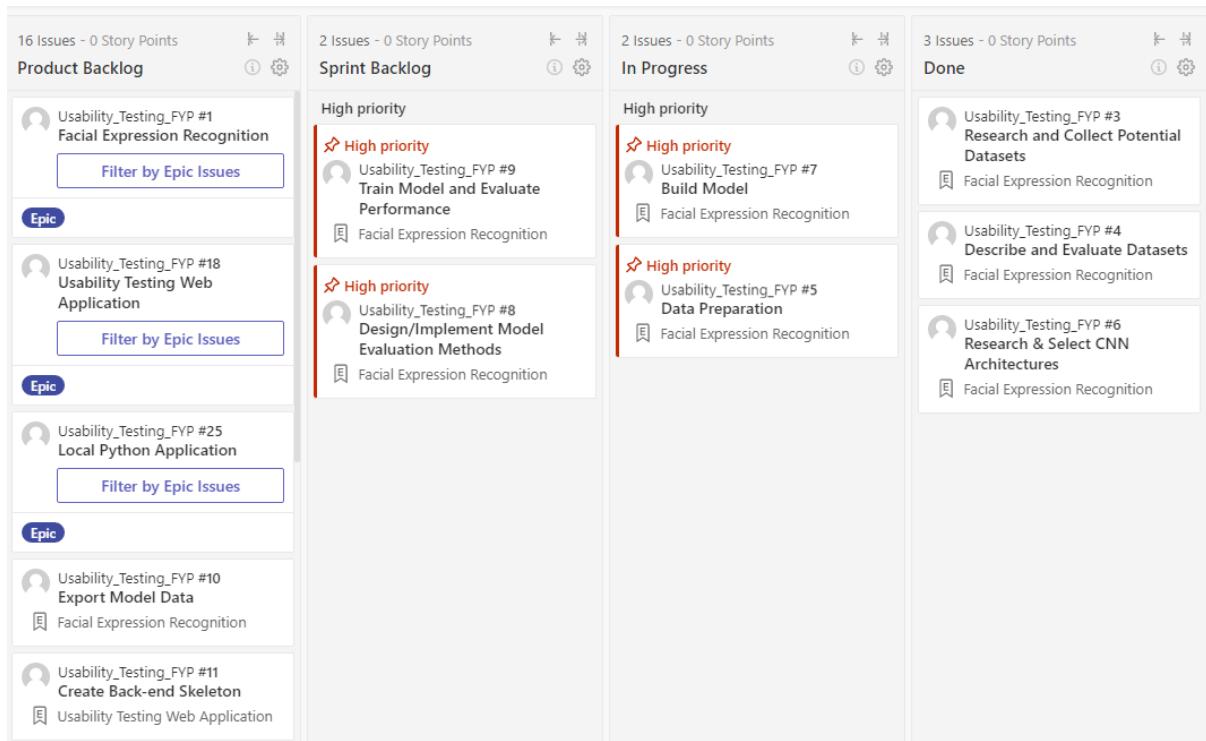


Figure 26 – ZenHub Epic and Sprint Planning (06/11/2020)

This screenshot shows the details for the "Facial Expression Recognition #1" epic:

- Status:** Open (green button) and Epic (purple button).
- Title:** Facial Expression Recognition #1
- Description:** Add a description +
- Dependencies:** Facial Expression Recognition has no dependencies. A "+ add dependency" button is available.
- Progress:** Facial Expression Recognition is an Epic. 3 of 8 Issues completed (progress bar at ~37.5%) and 0 of 0 Epic Points completed.
- Timeline:** Set start and end date button.

Figure 27 – ZenHub Facial Expression Recognition Epic

3.2.2. 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, much like agile it focused on iterative progress. The cyclical iterative process consists of 6 stages which are illustrated in Figure 28 [74][75].

Stage 1: Business Understanding

The goal of this stage is to understand the business objectives and to reveal 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 combing 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 is where a deployment plan, a monitoring plan, maintenance plan is created and the model is deployed.

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.

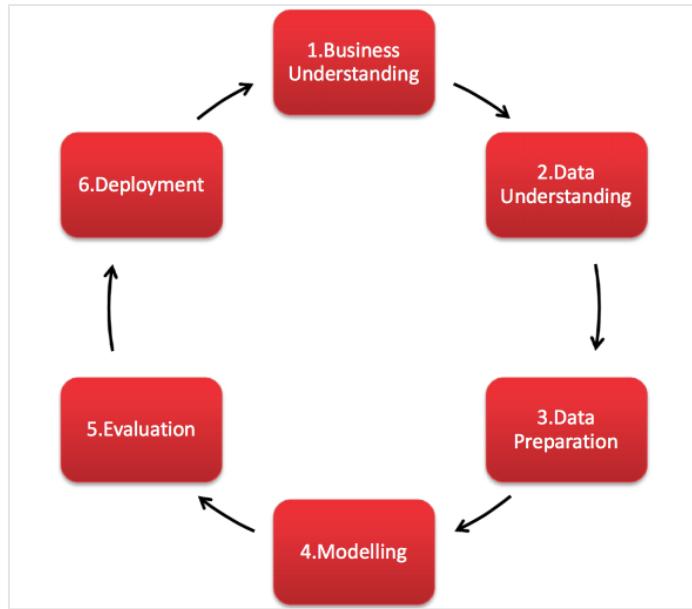


Figure 28 – CRISP-DM Methodology Cycle

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 29 illustrates the high-level steps that are taken in each subproject, showing how each is will be developed. The machine learning subproject involves working with data and training a neural network. The usability subproject involves 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. The assumption for now will be that the researcher invites the participants to their facility to conduct the usability tests. This eliminates potential setup and hardware issues.

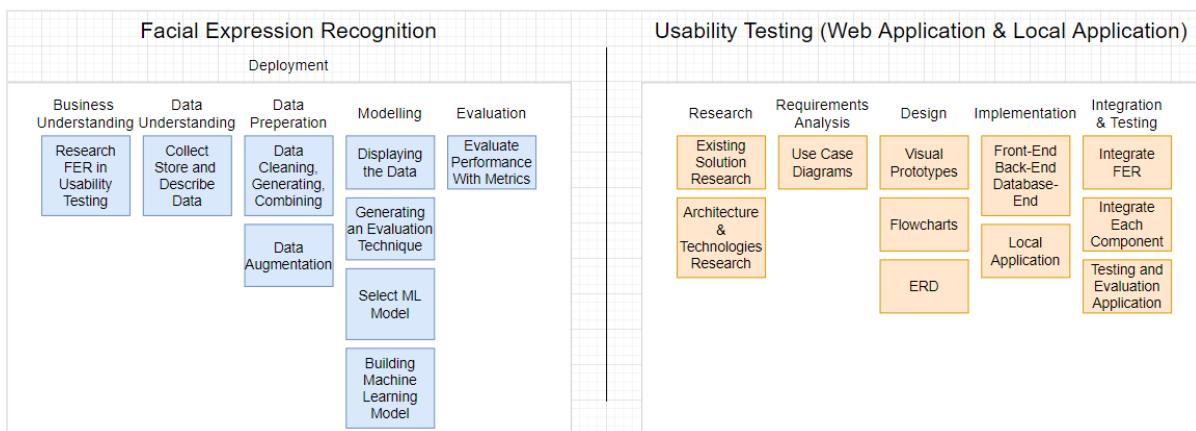


Figure 29 – Subprojects' High-Level Steps

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.

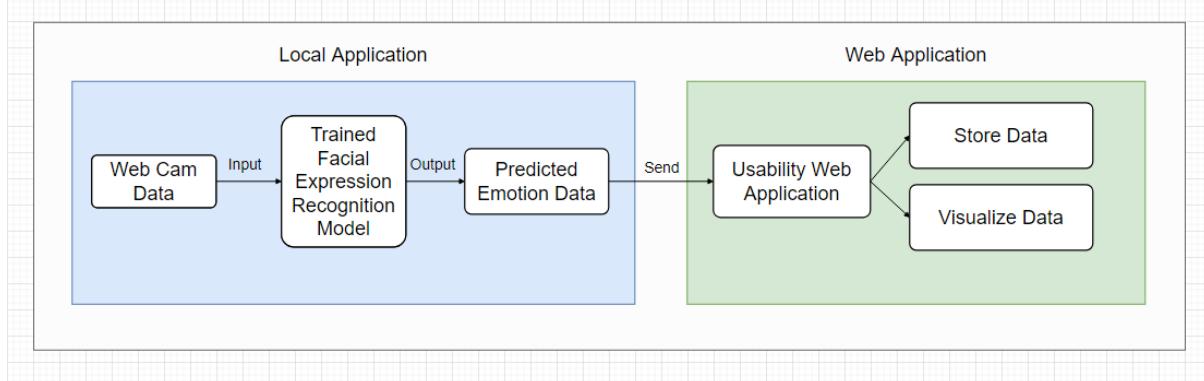


Figure 30 – High Level View of System

A more in-depth view of the architecture of the web application and the local Python application is shown in Figure 31. 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 another 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 research has concluded that this is one of the best and 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. As this solution is also familiar it will lead to less mistakes in development. 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 as this is where the usability test will be ran.

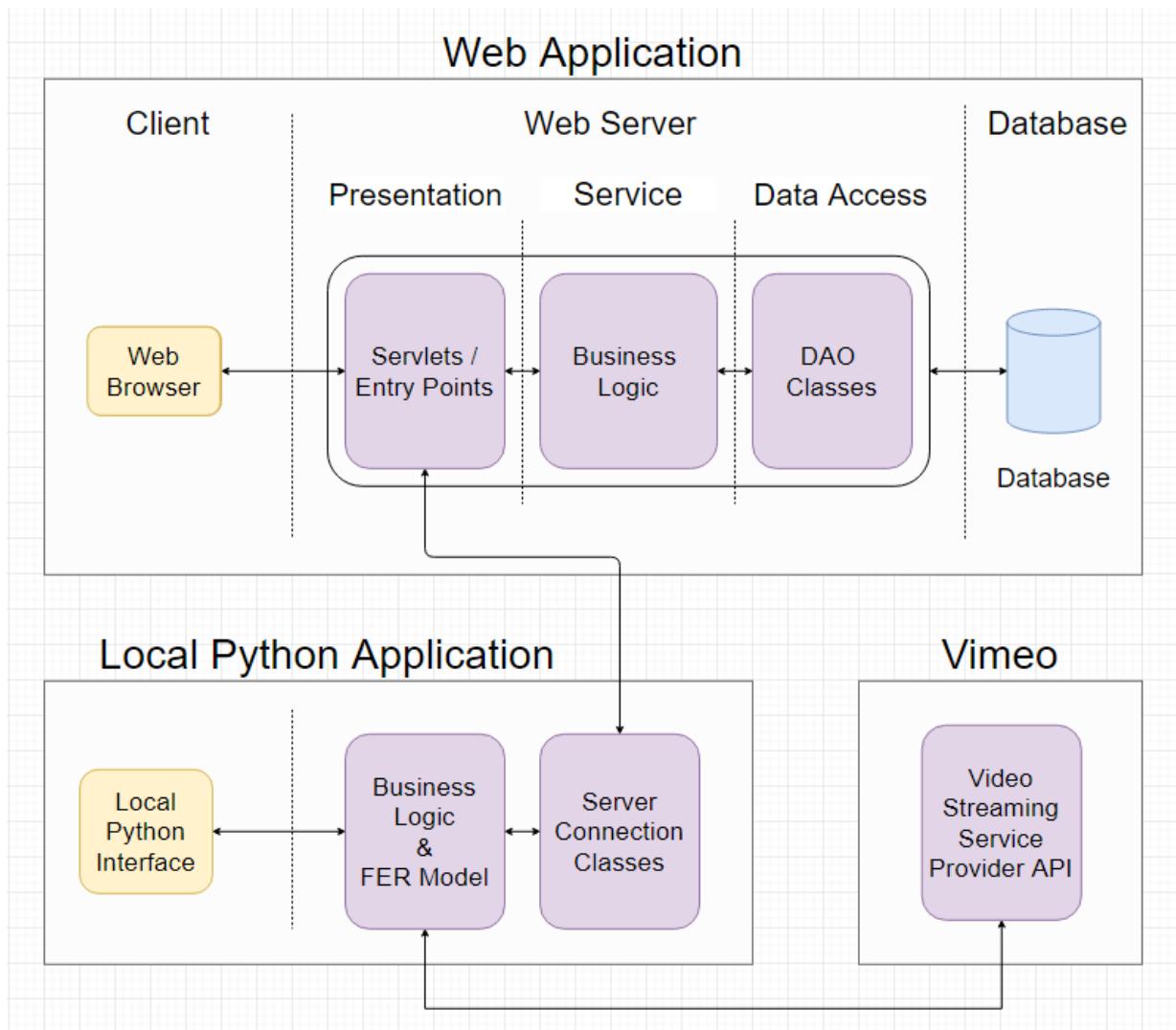


Figure 31 – Architecture of Applications

The high level operations of the system which includes the creation of a usability test and the usability test being conducted with a participant are illustrated in Figure 32. 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 screen prototypes of the user interface.

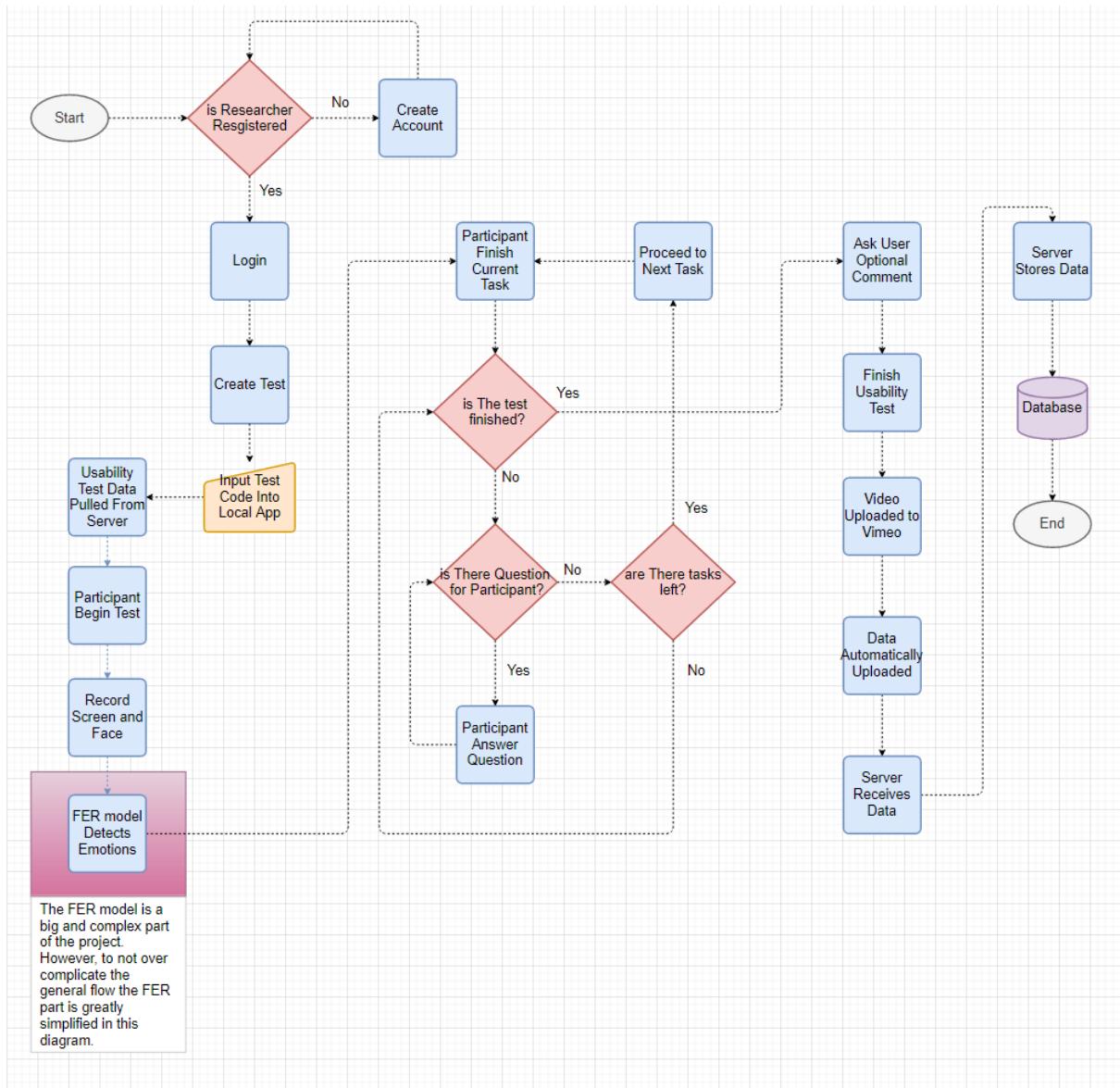


Figure 32 – High Level Flowchart of System

3.4. Web Application

3.4.1. Front-End Use Case Diagrams

The first iteration of the developer/researcher (“developer” and “researcher” are used interchangeably as it could be either or both) use case diagram is illustrated below in Figure 33. There is register and login functionality so that each 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. To explain what is meant by the average success rate the following example will be used. A test is conducted twice, the test has 5 tasks, the 1st participant 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 researcher can also watch the video of the screen recording and the camera recording. The researcher can view what emotions the participant showed on a timeline.

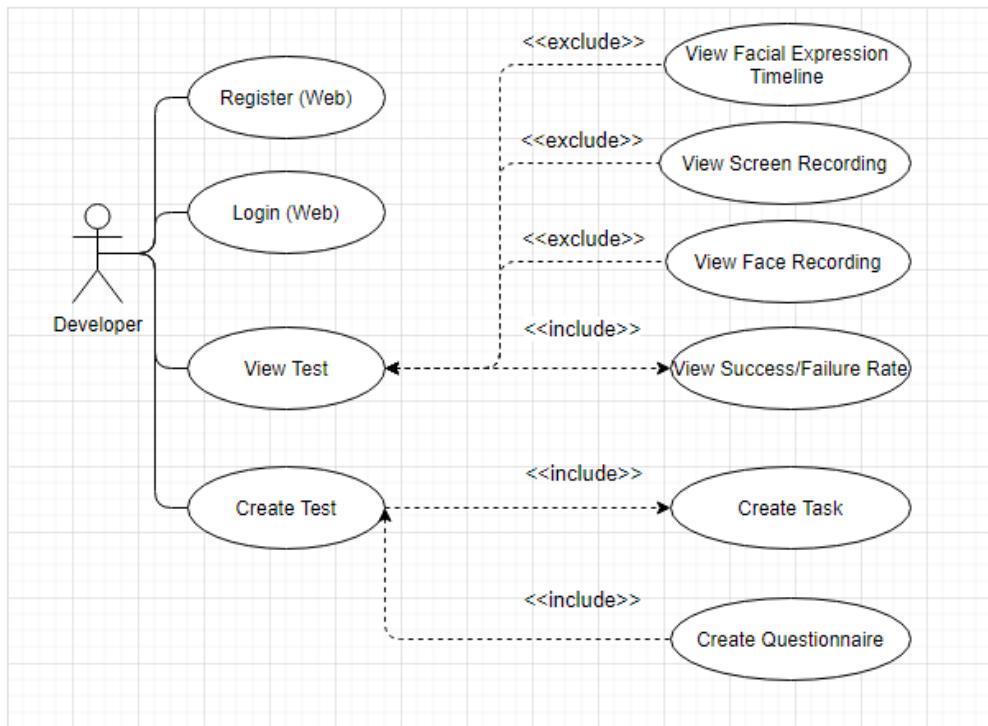


Figure 33 – Researcher Setting Up Usability Test Use Case

The next iteration of the researcher use case diagram is displayed below in Figure 34. Additional features were added to the the use case. On top of 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 they 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. with a journey map, which was been discussed in section 2.1.1

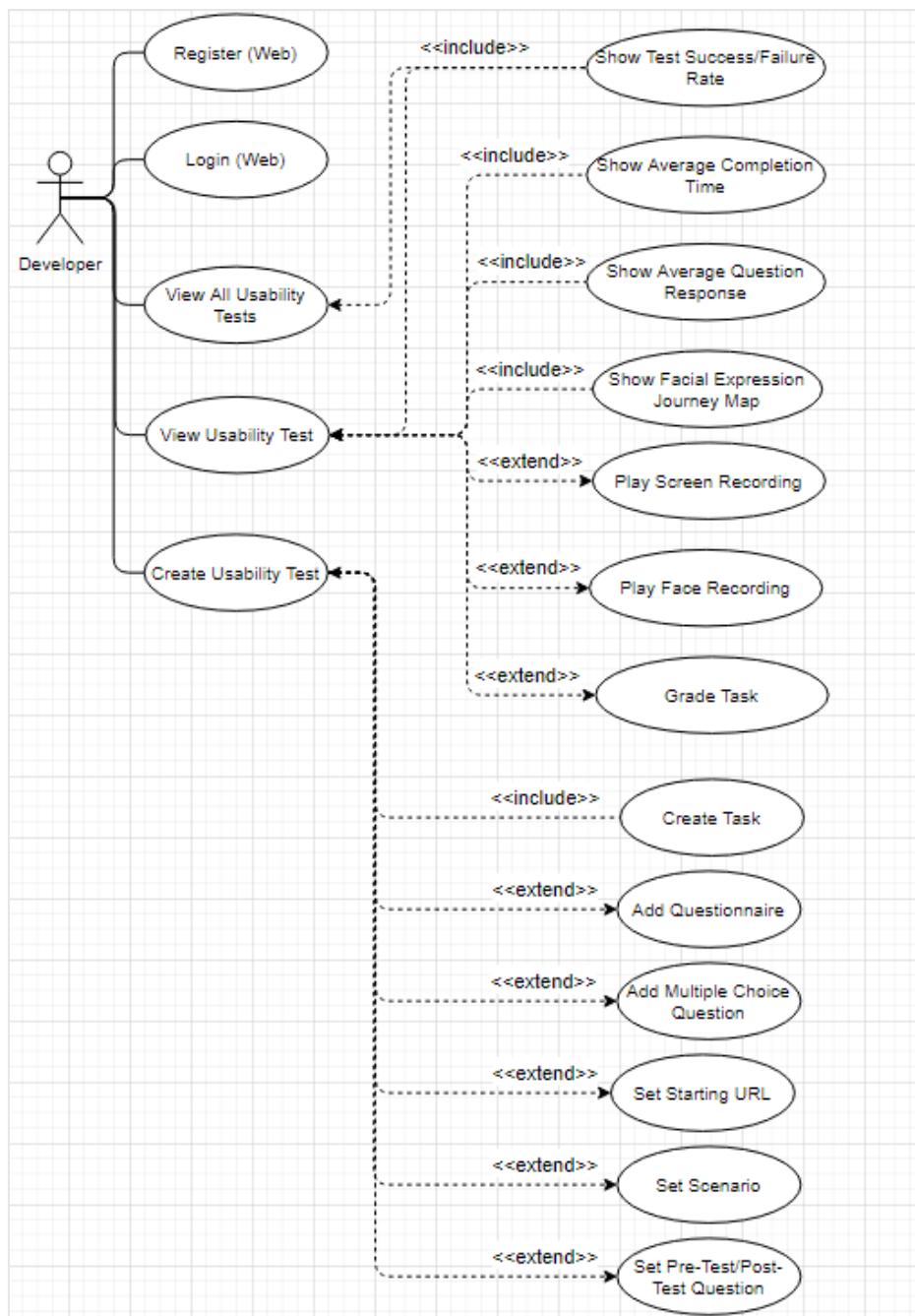


Figure 34 – Researcher Setting Up Usability Test Use Case (Second Iteration)

The use case diagram for the participant who is going to be using the local python application is illustrated below in Figure 35. First the test details are obtained from the web server and then using this data the local application will execute the correct instructions at the right time. For example, 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.

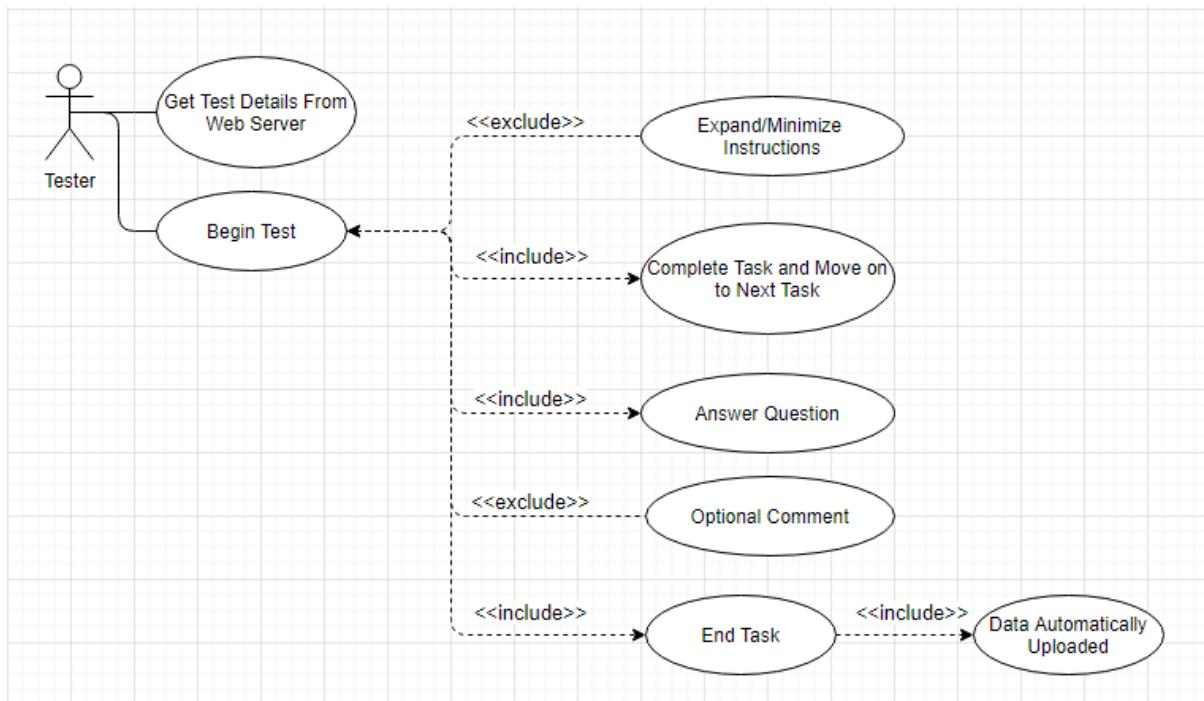


Figure 35 – Participant Use Case Diagram

Screen Prototypes

The “Dashboard” of the web application is the main screen the researcher will see after login and is illustrated below in Figure 36. The prototypes are designed with JustinMind. This screen would correspond to the “View All Usability Tests” use case. On this screen the researcher can create new projects. A new project can be created for each application to be tested and each project will have multiple usability tests. These usability tests will 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%).

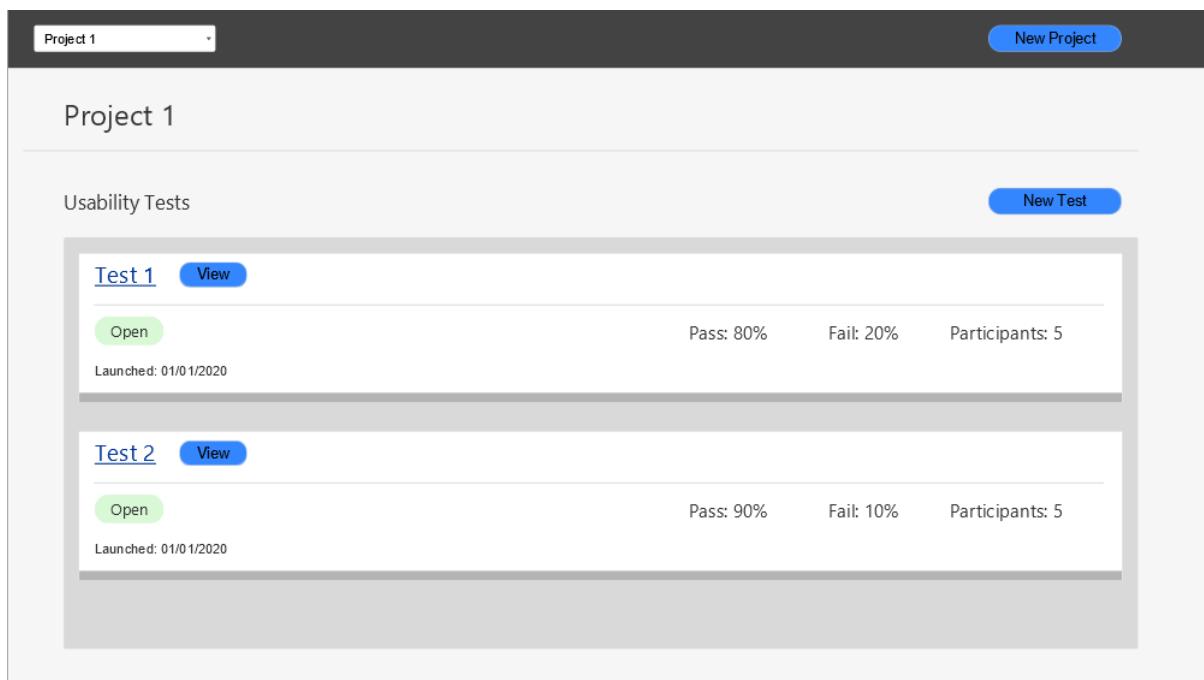


Figure 36 – Web Application Dashboard

The usability test creation screen which corresponds to the “Create Usability Test” use case is illustrated below in Figure 37. The researcher will name their test and insert questions or tasks. The options for answering questions is plain text and a multiple choice selection where the user is asked to pick one answer. Although not yet shown on the prototype, the instruction and question “boxes” can be moved up and down, meaning their order can be changed after they were added.

The screenshot shows the 'Create Test' interface. At the top left is a back arrow and the title 'Create Test'. Below it is a 'Test Details' section with a 'Test Name' input field containing 'Test Name'. Under 'Pre-Test Questions', there are two buttons: 'Text' (selected) and 'Multiple Choice'. Below these are 'Tasks' (+) and 'Question' (+) buttons. The 'Instruction' section contains two stacked input fields, both with red 'X' clear buttons. A blue 'Add Step' button is located between them. The 'Multiple Choice Question' section has one input field with a red 'X' clear button and a blue 'Add Answer' button. A green-outlined box encloses the 'Instruction' and 'Multiple Choice Question' sections. At the bottom is a large green 'Create Test' button.

Figure 37 – Web Application Usability Test Create

The test results screen that the researcher is going to see after the usability test is conducted is illustrated below in Figure 38. This screen prototype corresponds to the "View Usability Test" use case. The usability test name in this sample scenario 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 section. Below that, the average data on the individual tasks is displayed. 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 facial expressions is obtained and plotted on the bar chart. The idea is that a face expression that occurs 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 participant's 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 38 – Web Application View Usability Results (Overview)

The page where the researcher can view the recordings and grade the usability test is illustrated below in Figure 39. 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 also contain 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 within the video progress bar. The video markers are going to be implemented with Videojs-markers [76]. 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 appear to add unnecessary depth making the chart less clear.

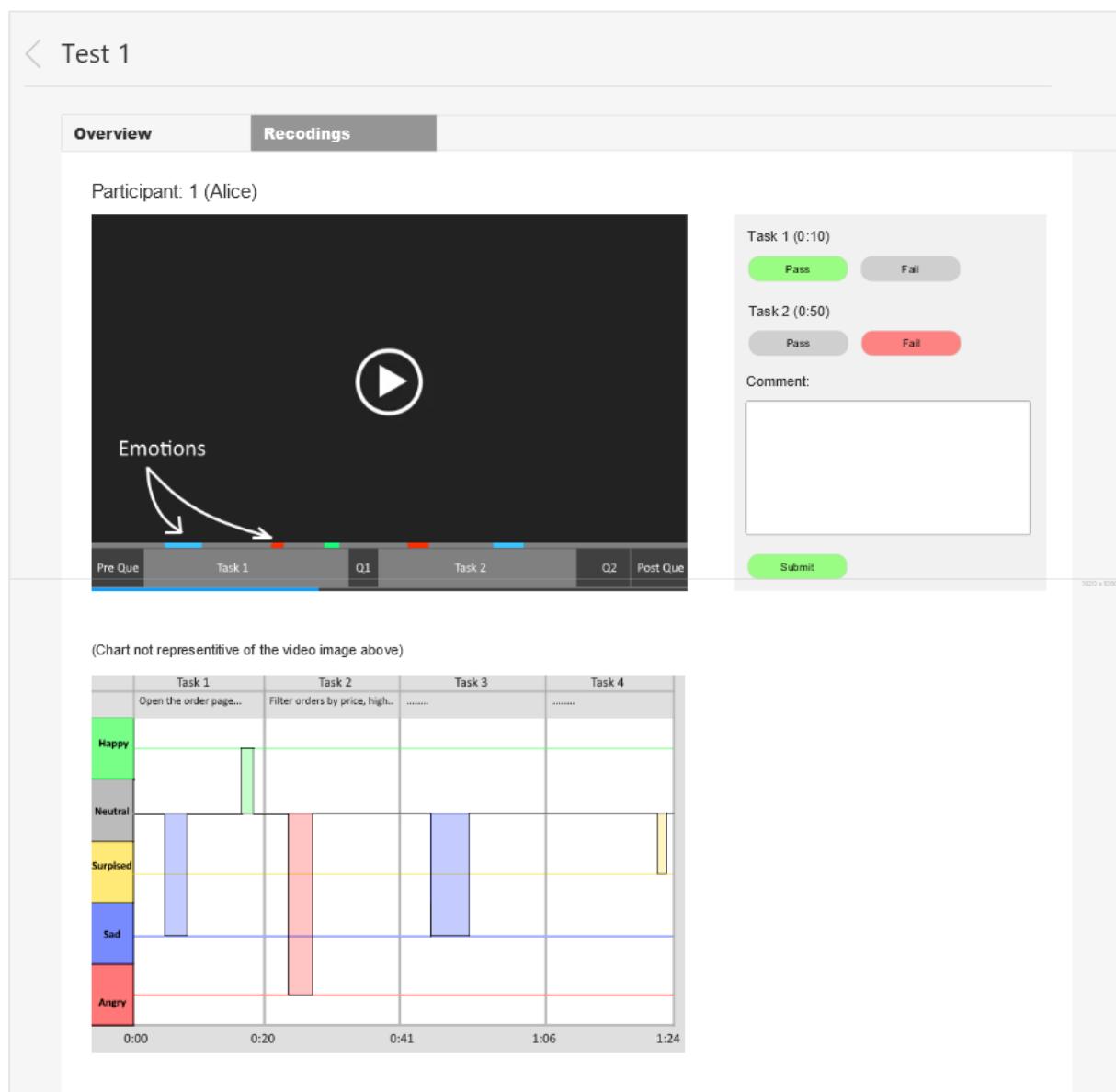


Figure 39 – Web Application View Recordings

The second design of the journey map after the evaluation is illustrated in Figure 40. 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.

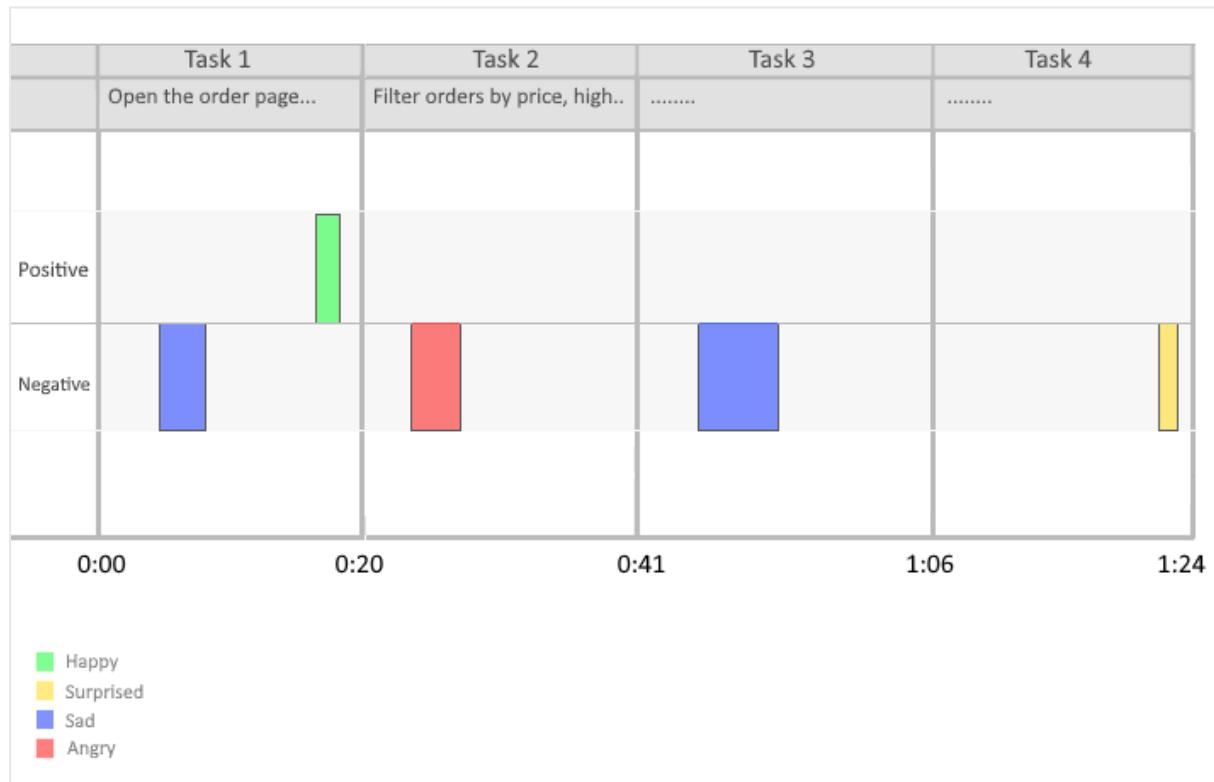


Figure 40 – Journey Map v2

3.4.2. 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.

The design of the database is shown in Figure 41. Each researcher can have many projects, and these projects can have many tests. Each test must have tasks and may have questions. The “TestInstance” is a usability test that has been conducted with a participant. The “Test” is what the researcher has created and contains the data about how the usability test is to be conducted. At the moment the emotions captured are stored in the “VideoTimeStamps” table and this data can be used to populate the journey map and the emotion markers on the video. The question and answer tables are generic and will store JSON objects. This will allow for scalability of the questions and answers, however, storing the data in such a format does not provide the structural benefits of a relational database and will need to be managed carefully. Another option is to use inheritance and create a table for each question and answer.

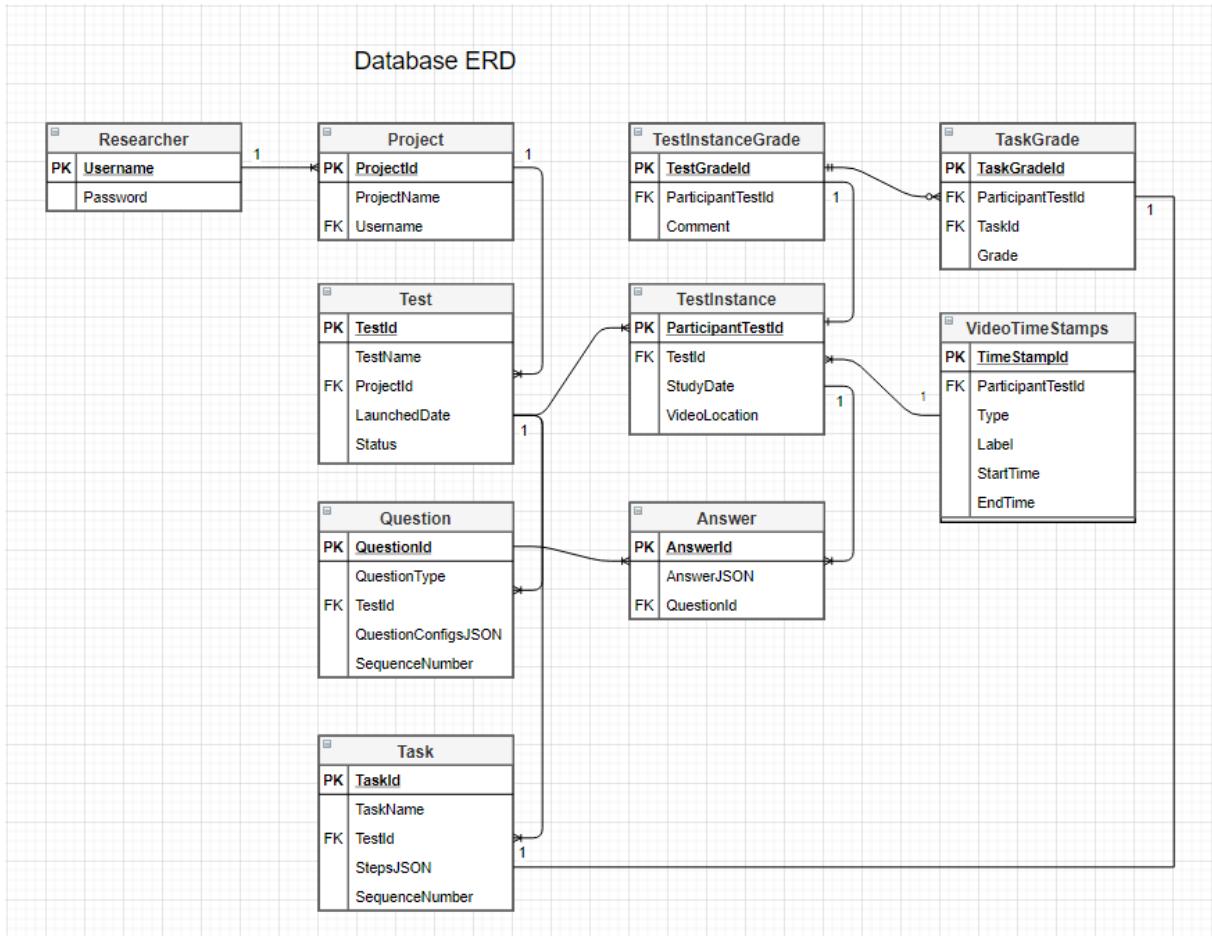


Figure 41 – Web Application ERD v1

The ERD below in Figure 42 - Web Application ERD (Final in Production) is the final design that is used in the deployed and complete application.

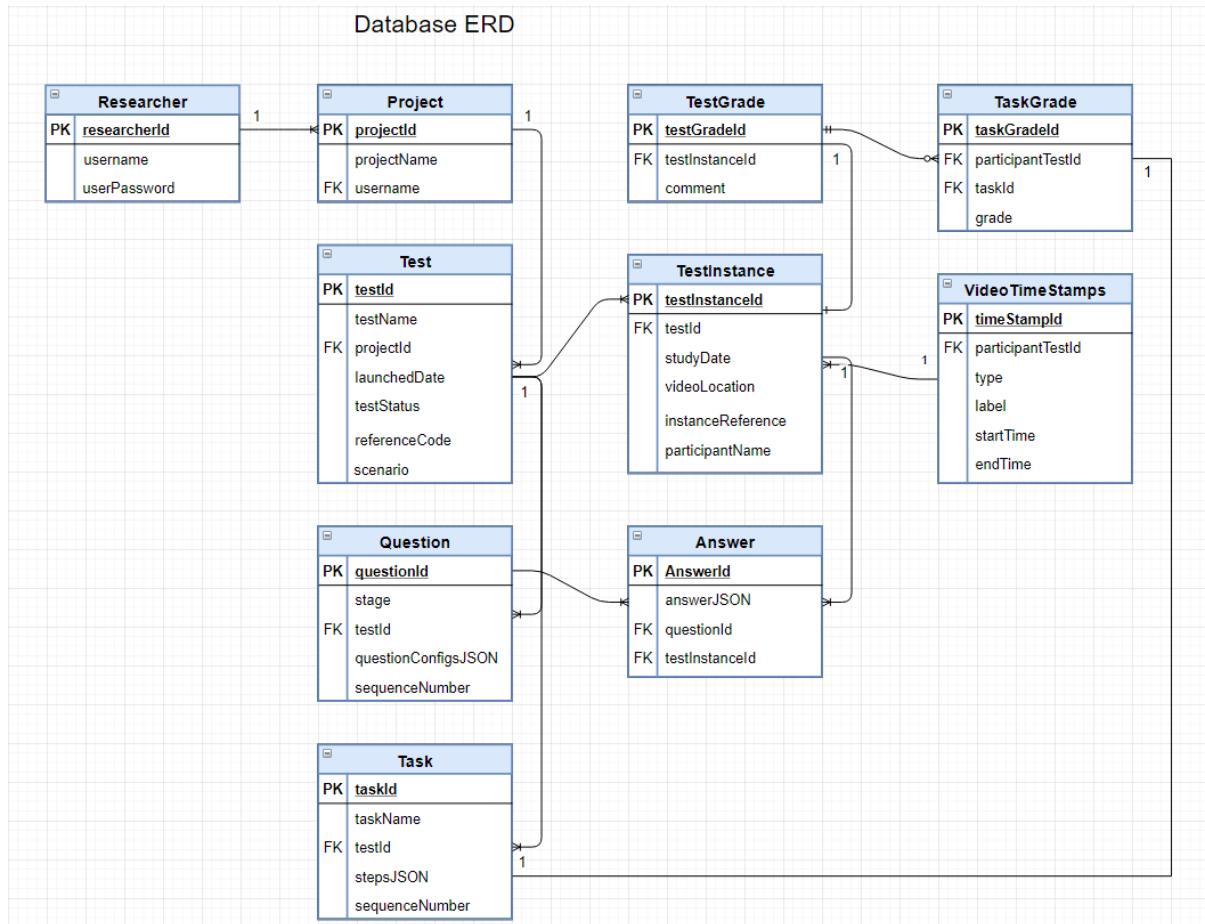


Figure 42 - Web Application ERD (Final in Production)

3.5. Local App

The local python application that will run on the machine the participant is going to be using is illustrated in Figure 43. 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 corresponds to the “Get Details from Web Server” use case in Figure 35. Once the details are verified as correct the participant will enter their name and click “Begin” to begin the test.

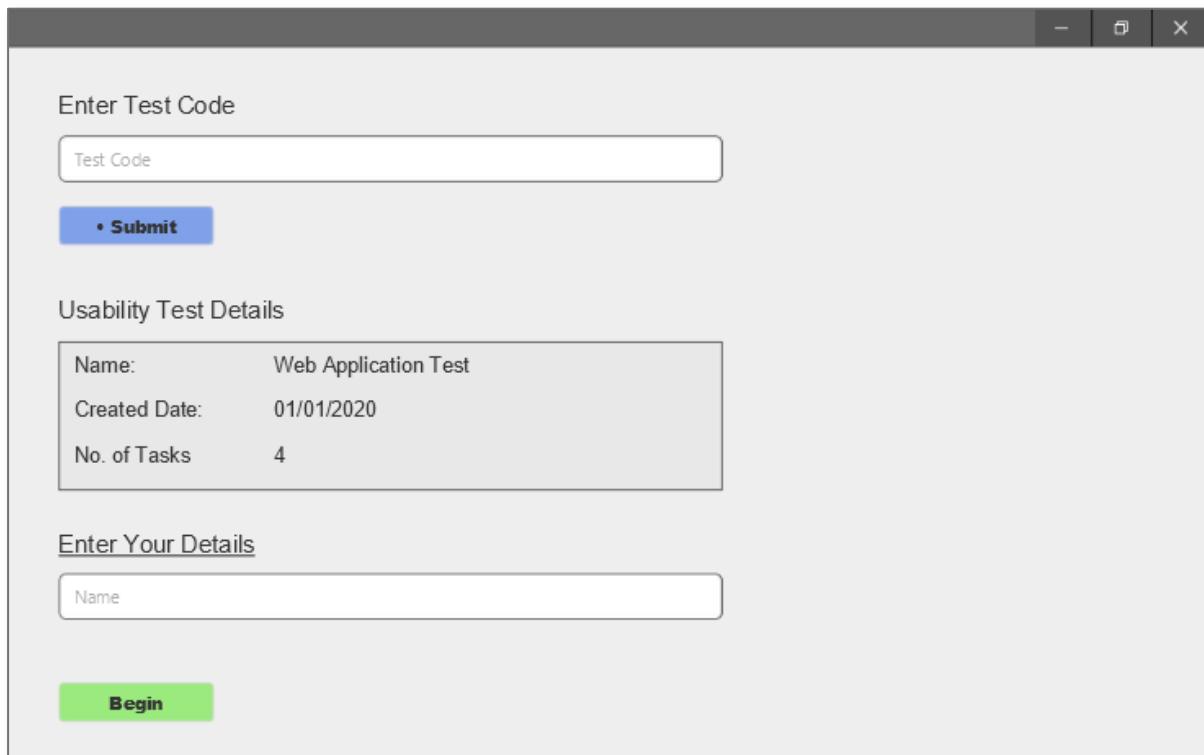


Figure 43 – Local App First Screen

Once the test begins the participant will see a similar screen to what is illustrated in Figure 44. 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 which is used 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.

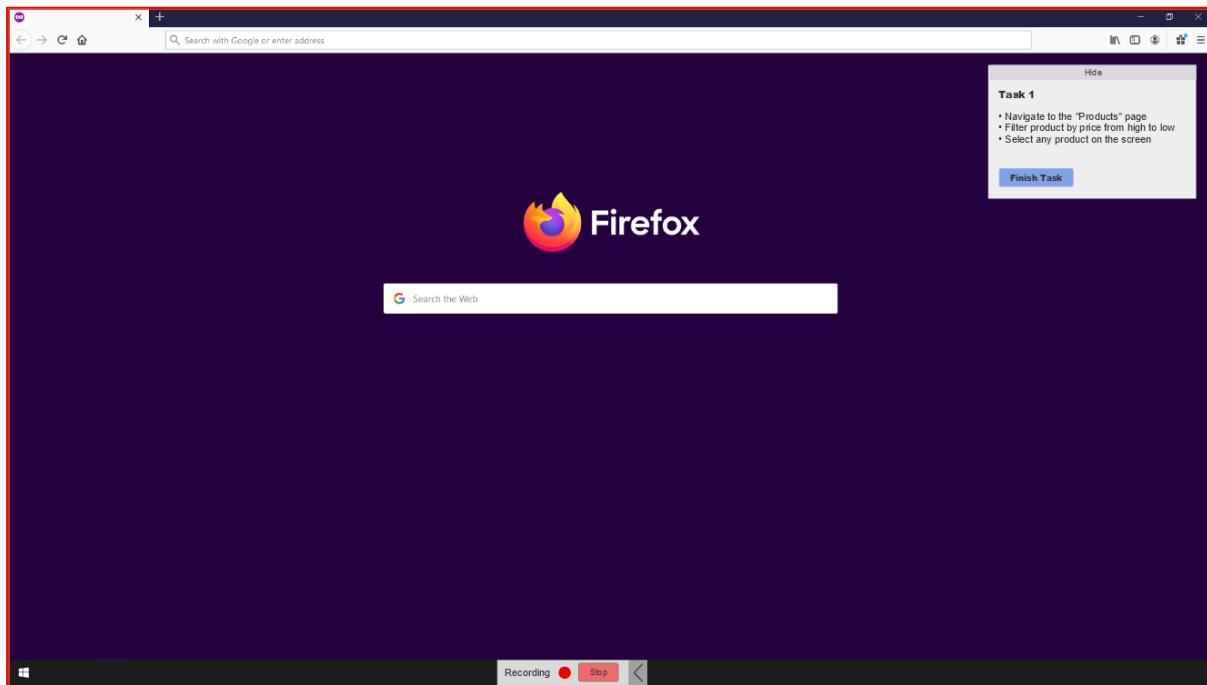


Figure 44 – Local App Usability Test Started

3.6. Facial Expression Recognition & Machine Learning

3.6.1. Design With CRISP-DM Methodology

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) [6] which was discussed in section 2.1.1 there was mention of combining the basic six 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 dataset research section 2.1.5 the reasons for why some 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 combing 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. Instead of creating the images from pixel values, 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 folders, “Training”, “Validation” and “Testing”. The folders within each of the three folders mentioned will follow the same structure with a folder for each emotion and this is illustrated in Figure 45. 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 which 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 45 – 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, sheer, zoom, shift and flip the image horizontally by a random value.

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 where the design aspect is applicable. The charts for modelling the data (e.g. Journey map) have already been discussed in the research section 2.1.1 and in the design section 0. The evaluation techniques for the model were also previously discussed in the research section 2.1.4, and to recap 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) which is SoftMax activation followed by Cross-Entropy loss. The CNN architecture of choice is the VGG-16 architecture described in section 0. The next two stages in the CRISP-DM methodology are evaluation and deployment. Both of these stages will be discussed in a later sections of the report.

3.6.2. 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.

3.7. Conclusions

The stages of the project and the tasks that need to be completed (development) are broken up into 3 stages and illustrated below in Figure 46. These requirements are broken up into sprints and shown at the very end of the report in – Sprint Chart Overview (All Stages).

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 46 – Project Tasks & Requirements

4. UsabCheck Development

4.1. Introduction

Software Used in Development

To backup and manage this project a GitHub repository was created. Git is a version control tracking system that 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 and the development of the local application with Python. The FER model and other machine learning related functionality was developed in Jupyter Notebook in Anaconda.

Anaconda allows for the creation of a development environment which contains packages that are separated from the packages installed on the host machine. 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 and charts can be displayed. This makes it very suitable for data science and machine learning.

Development Sections

The sections of the development chapter are shown in Figure 47. The number next to the name is the order in which they are discussed.

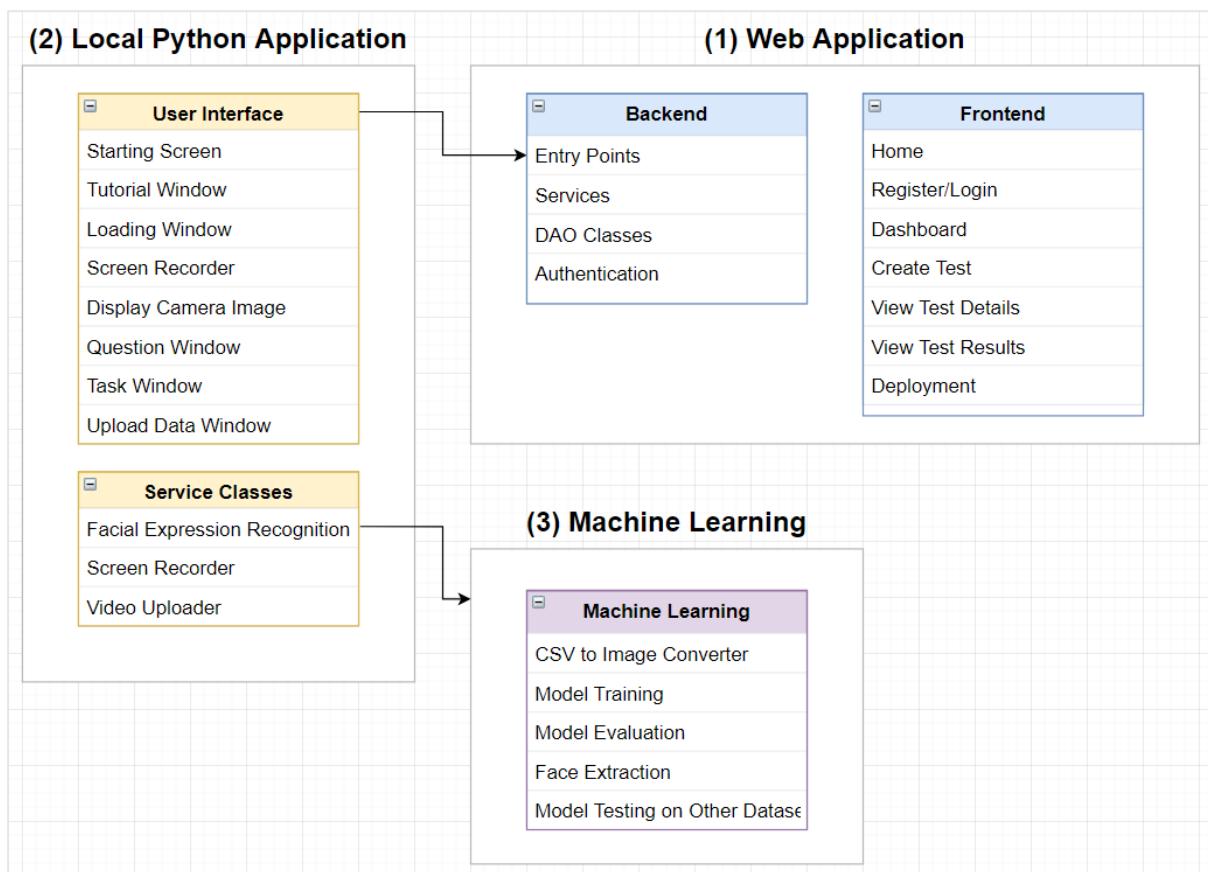


Figure 47 - Development Sections

4.2. UsabCheck Web Application

4.2.1. Backend Server (Java)

Overview

The Java backend server is used to store and retrieve data. It is accessed by the React frontend of the web application and the local python application. The hierarchy of are the Java classes and packages that are in the web application which currently deployed on a server is shown below in Figure 48. The packages and classes are explained in greater detail in the following sections.

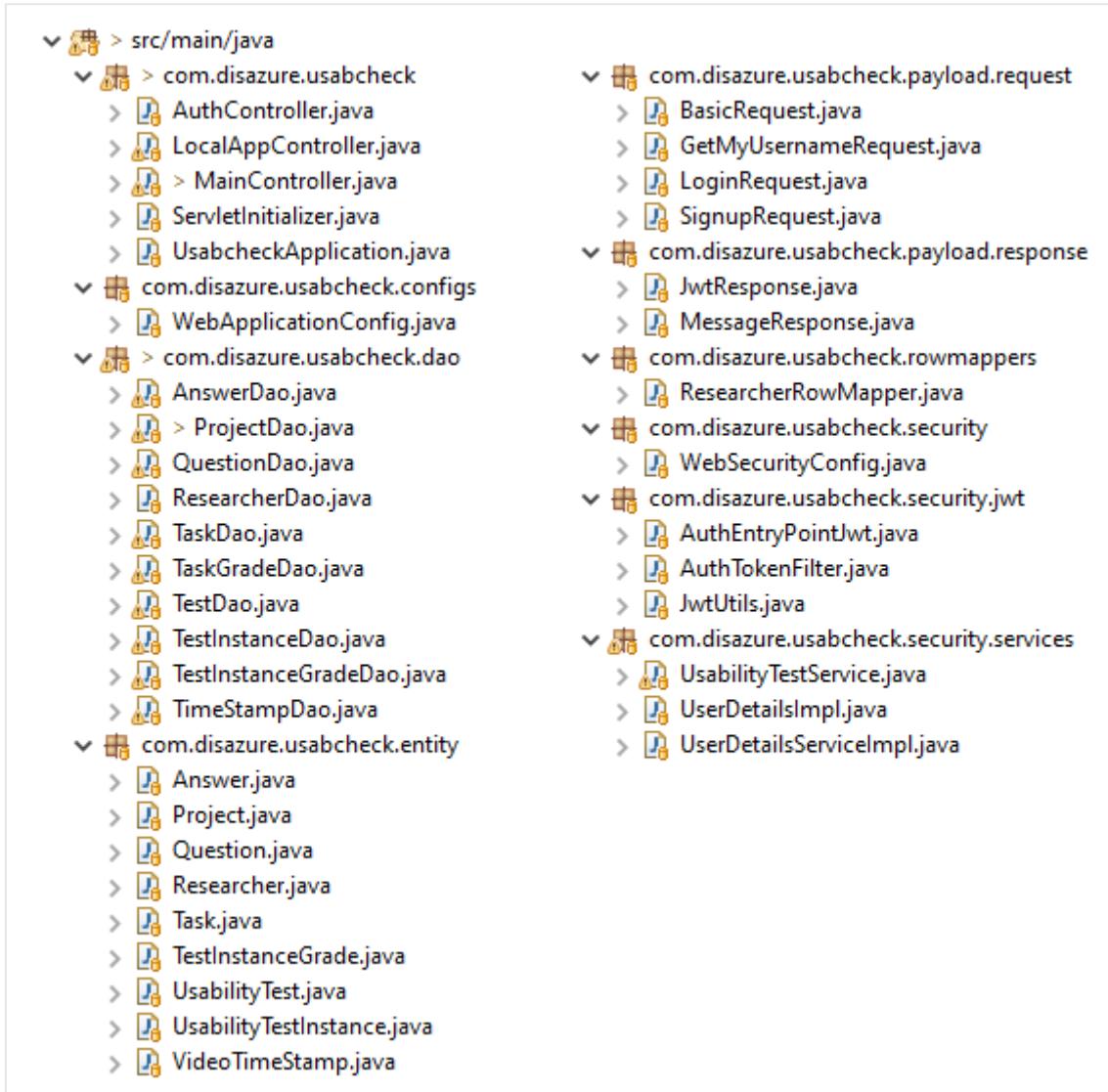


Figure 48 – Web App Backend Java Classes

The relationships between all of the classes in each package is shown below in Figure 49.

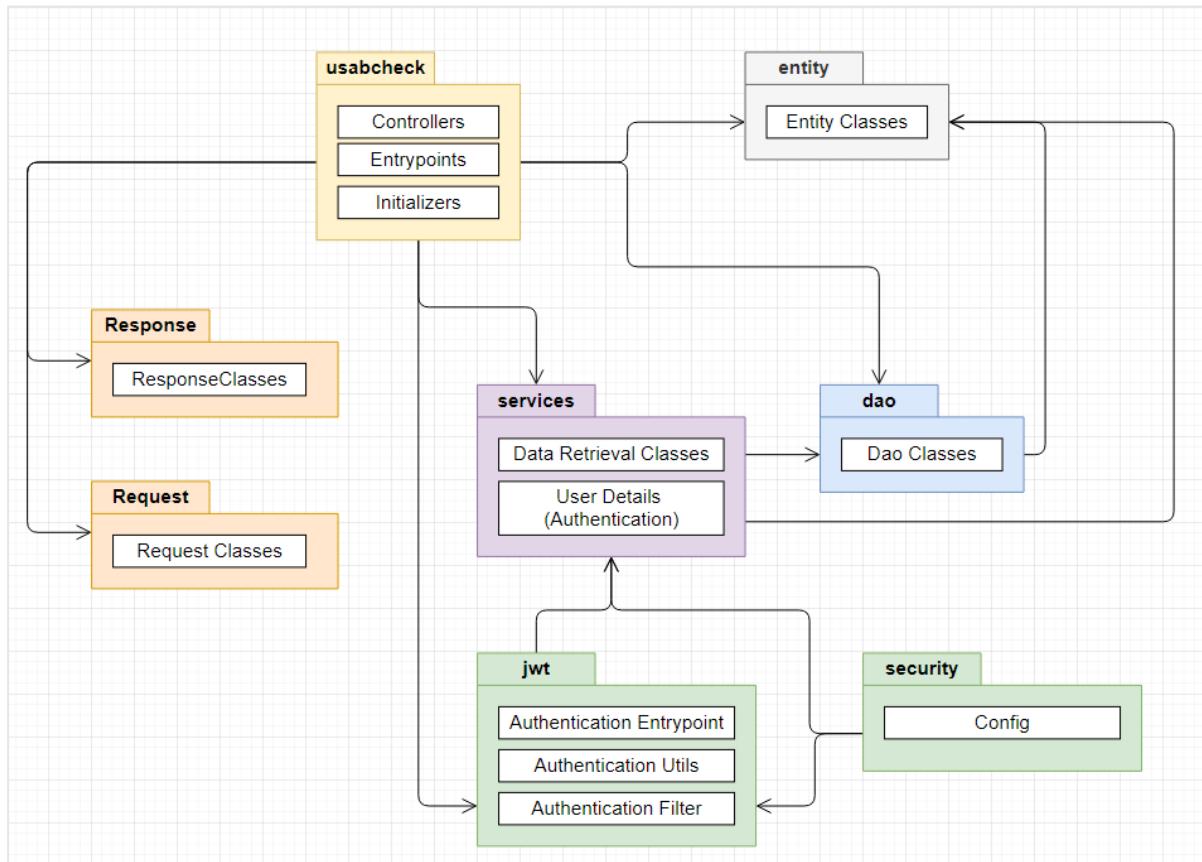


Figure 49 – Relationship Between Packages

Entry points/Controllers

The *MainController*, and *LocalAppController* are the two classes which contain entry points used by the web application's frontend and local app respectively. These classes provide APIs that allow for the storing and retrieval of data. The controller classes make use of the DAO and Service classes to retrieve and/or process the data.

To access the *MainController*, the user must be authenticated as the functionality requires a username for verification purposes. The *LocalAppController* has no such requirements as the local python app has no login. The user verification process is explained in the next sections.

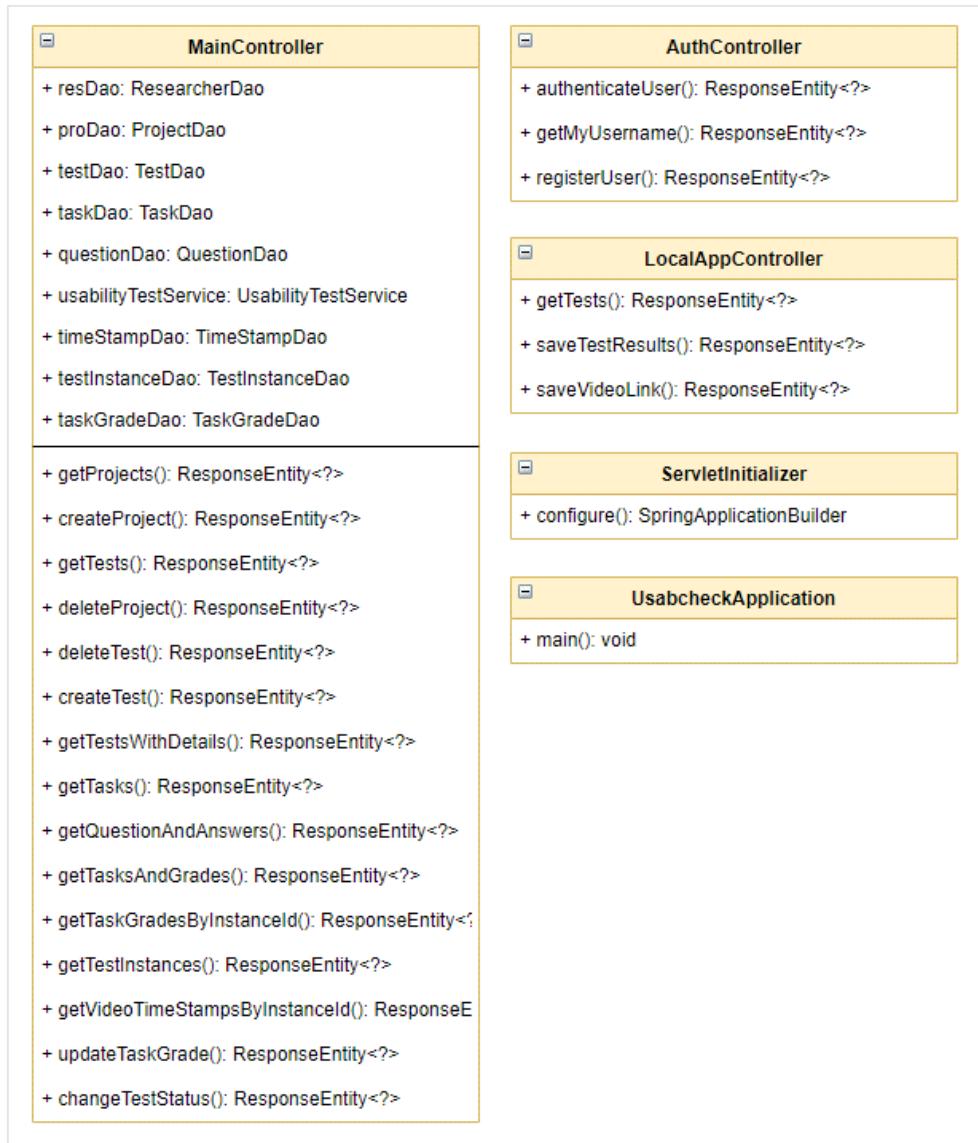


Figure 50 - Controller Classes

Services

The *UsabilityTestService* class is used by the *MainController* to retrieve more complex data from the database. Initially the *MainController* accessed the DAO directly as the queries were simple. However, as the development progressed there was a need of retrieving data from multiple tables (e.g. *getTasksAndGrades()*) and manipulating/combing that data. To manage these more complex requests the *UsabilityTestService* was created.

The *UserDetailsImpl* and *UserDetailsServiceImpl* are used for authentication/authorization purposes. The *Security* classes use this service to check the username and password and the Controller classes use this service to get the username of the user currently logged in.

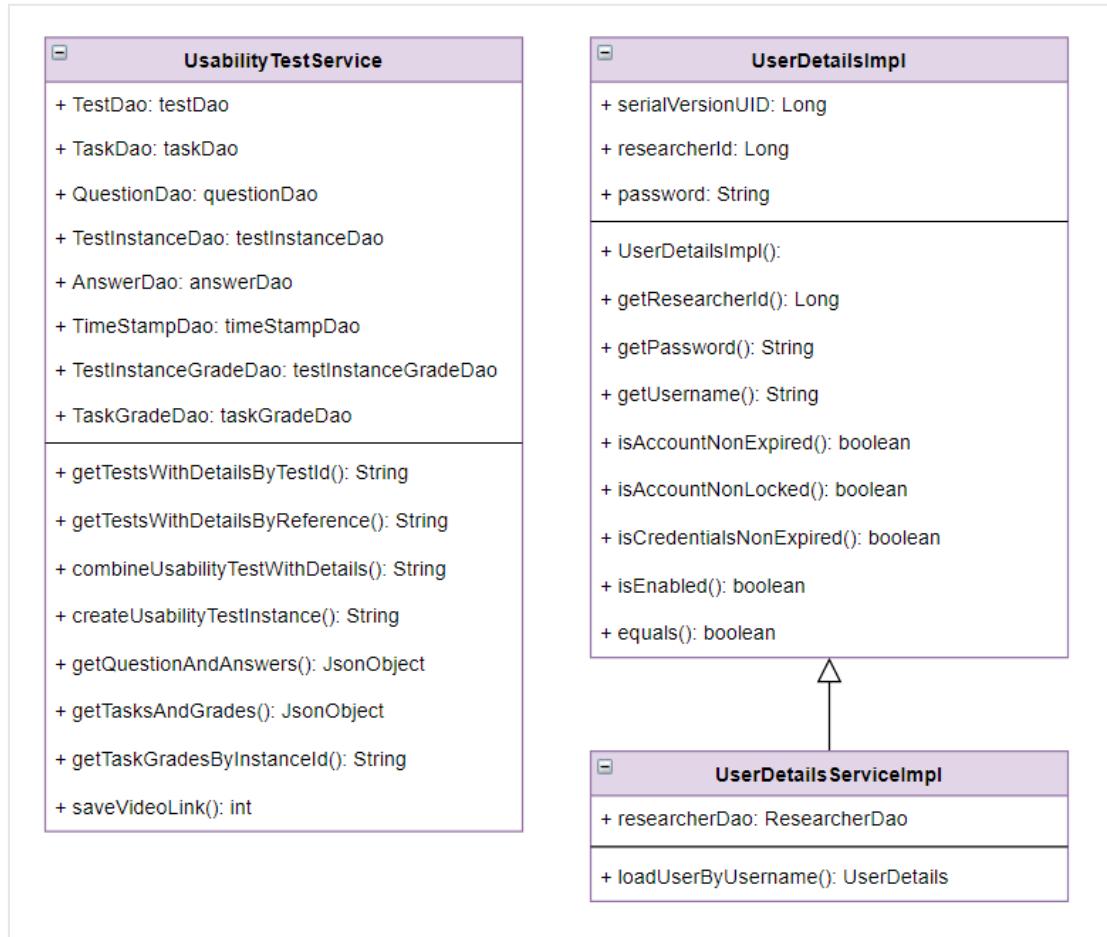


Figure 51 - Service Classes

Data Access Objects (DAO) Classes

The DAO classes are used to access a table in the database. There is a DAO class for each table in the database and JDBC was used to access the data. There was a choice of using JDBC or the ORM Hibernate framework and the decision was to use JDBC. There are many benefits to using the Hibernate framework such as inbuilt lazy loading, caching, and connection management which need to be manually implemented if JDBC is used.

However, the disadvantages according to [\[Source\]](#) is that it can be slower in some cases due to runtime based mapping, it is more complex with joins and more importantly has a “steep learning curve”. Being new to many aspects of this project and having never used Hibernate before it seemed unwise to add unnecessary risk on top of the existing risks. When making the decision there was a high degree of certainty in being able to achieve the desired functionality with JDBC and plain SQL at the cost of the benefits that Hibernate provides.

The DAO classes and their attribute and methods are displayed below in Figure 52.

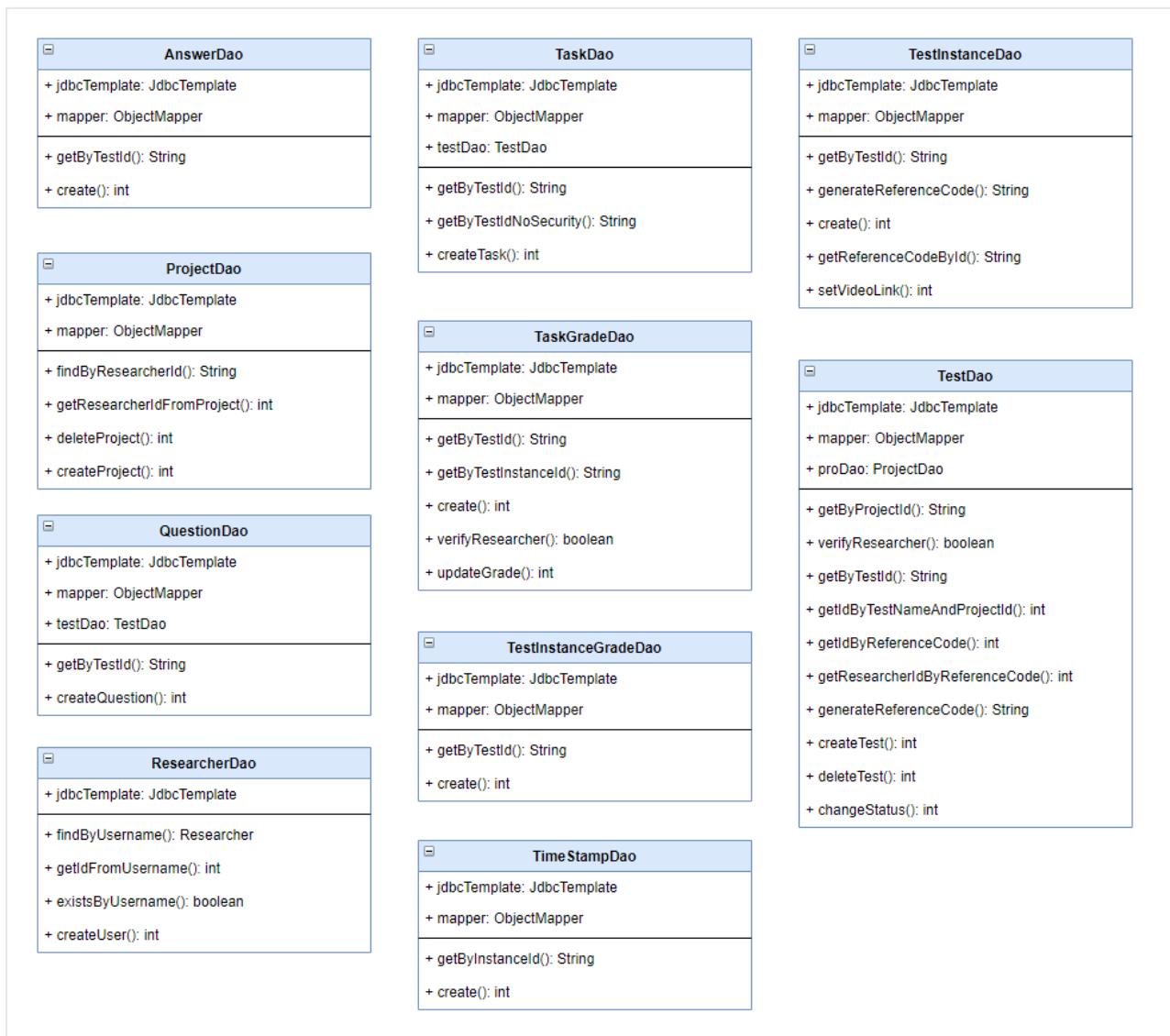


Figure 52 – DAO Classes

To use the vast majority of the features of the web application the user is authenticated and only then can they access the *MainController* entry points and make requests. To prevent the user from accessing data they do not have rights to the username is retrieved from the JWT token that the user sends and the *researcherId* is obtained from their username. That identifier is sent to the DAOs whenever making any request. The database query verifies the *researcherId* to ensure that the user can only access the data that is related to them.

Taking the *deleteTest* method as an example the parameters passed are the *testId*, *testName* and *researcherId* which can be seen in Figure 53. The usability test is deleted only if the usability test belongs to the user that is currently authenticated. Additionally, to help prevent SQL injection, prepared statements were used and the place where the parameters go are indicated by the "?" sign.

```
public int deleteTest(int testId, String testName, int researcherId) {
    // Delete ensures that the researcherId matches. Any user can send a delete request but that doesn't
    // mean they should be allowed to delete something.
    String sql = ""
        + "DELETE test "
        + "FROM test "
        + "JOIN project using(projectId) "
        + "JOIN researcher using(researcherId) "
        + "WHERE researcherId = ? AND testId = ? AND testName = ?";

    return jdbcTemplate.update(sql, researcherId, testId, testName);
}
```

Figure 53 – SQL Query in DAO

When making a new entry in the database the Entity classes are passed to the DAO. Each entity corresponds to a table and contains all the attributes that the table does. The user sends data to the entry point which packages the attributes into an entity object and sends it to a DAO. When retrieving data JSON objects are used. The data is first retrieved in the form of a Map object which is converted to a JSON String and that can be directly sent back to the client requesting the data.

In hindsight there are minor improvements that could have been made to the naming convention of the DAO methods. For example, “*deleteTest*” should be simply “*delete*”.

Authentication

JWT (JSON Web Token) was used to authenticate the user. A tutorial by Bezkoder [\[Source\]](#) was followed to implement the user authentication. Unnecessary parts such as user roles were removed from the tutorial code when implementing the authentication and additional configuration was done to meet the requirements of the project. This includes the access configuration for each of the entry points.

To briefly explain how JWT authentication works, after the user is registered they login and send their username and password to the server via a secure POST request. The server verifies the details and returns a JWT token. This token can then be stored in user LocalStorage or in HTML5 Web Storage Cookie with httpOnly flag. Whenever the user makes a request to the server the token is sent, it is validated and used to process the request. However, storing the JWT token in LocalStorage exposes it to the vulnerability of a XSS attacks as any script that runs on the web page can access the token and if a malicious script is ran then the user credentials can be stolen [\[Source\]](#).

The Security classes are shown below in Figure 54.

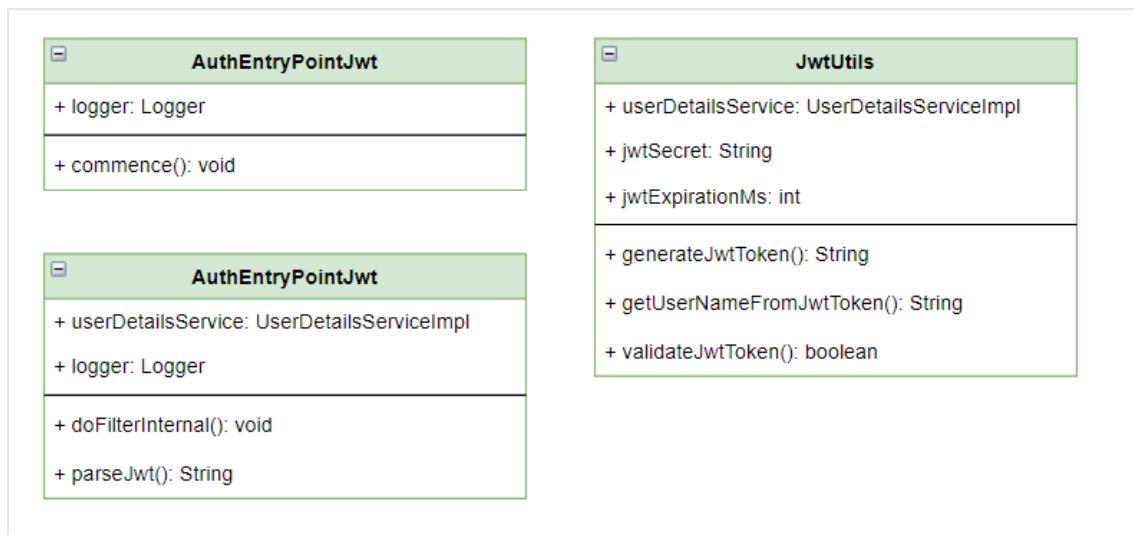


Figure 54 – Security Classes

When storing the username and password in the database the password is encrypted using Spring Framework's PasswordEncoder. The Researcher table is shown below in Figure 55.

	researcherId	username	userPassword
	4	testuser123	\$2a\$10\$wpXySfMF58Jro4e3Zc0WH..dLYrG2WX/HeJi3fbjGH.ZiGiq0zH2L.
	5	testuser1234	\$2a\$10\$skouiReUlkk7KQsjnfsEVe.dGIA2dUJ1l9P/kWM75nz535BzuWEcm
	6	SampleUser1	\$2a\$10\$N9HIgayeEEKeBQIkU.0uHOI5gWFSQc8rsg4Z5HFbVNIAAsouX8a
	11	SampleUser2	\$2a\$10\$Dm8Ty0R40MC4GOVTBmj6i.X9nULzF.N4/I9NRqtJLB57KzxMyfPv.
	12	SampleUser3	\$2a\$10\$5307Eb0k/WPNv3Tr2tcTlOpEmd9jbkJgBiuGjd8bEymVlOhZBdIH2
	13	SampleUser4	\$2a\$10\$5k5G1PswiLIV21.1Pwq5J1ObOzlg3Nfx9AXE2kuv7IUUybJ5T5A3X6

Figure 55 – Encrypted User Password

4.2.2. Front-End (JavaScript - React)

Overview

The classes that make up the frontend of the web application are shown below in Figure 56. React creates a single page application whereby the CSS and styles are shared between all of the pages. For instance, App.js contains the navigation bar and as a result all of the other pages will have that navigation bar. This has also resulted in problems/conflicts between styles imported from bootstrap and required that only styles needed were selected and imported.

The various classes and components will be explained in more detail with visual examples in the following sections.

src	
components	
dropdownGenerator.component.js	
dynamicList.component.js	
modalButton.component.js	
modalContainer.component.js	
multiplechoiceQuestion.component.js	
tabs.component.js	
taskCreate.component.js	
taskGrading.component.js	
testContainer.component.js	
textQuestionCreate.component.js	
videoBars.component.js	
forms	
createProjectForm.js	
deleteProjectForm.js	
deleteTestForm.js	
infoForms.js	
modal	
infoModalUtilities.js	
modalTemplate.js	
services	
auth-header.js	
auth.service.js	
server.service.js	
store.js	
styles	
# form.css	
tabs	
JS testResultOverview.component.js	
JS testResultRecordings.component.js	
utilities	
JS utils.js	
views	
JS createTest.component.js	
JS dashboard.component.js	
JS home.component.js	
JS login.component.js	
JS register.component.js	
JS viewTestDetails.component.js	
JS viewTestResults.component.js	
# App.css	
JS App.js	
JS App.test.js	
# bootstrap.min.css	
# index.css	
JS index.js	
img logo.svg	
JS reportWebVitals.js	
JS setupTests.js	

Figure 56 - Web App Front-end Classes

The interaction between all of the folders can be seen in Figure 57. The “views” folder contains the pages such as the home page, dashboard, login and register etc. The components folder contains the various reusable components that are used by the views and other components. The views folder makes use of services such as the authentication service and a server access service to communicate with the backend server and make request.

The “tabs” and “modal” folders both contain components; however, these components were separated into their own folders so they can be more easily distinguished. Forms (forms folder) are used to display content within modals, and utilities (utils folder) include functionality such as notification popups. There are instances where the components invoke methods on the parent, hence the bi-directional relationship. An example of this is when a child component is updated which requires the parent view/component to be updated as well.

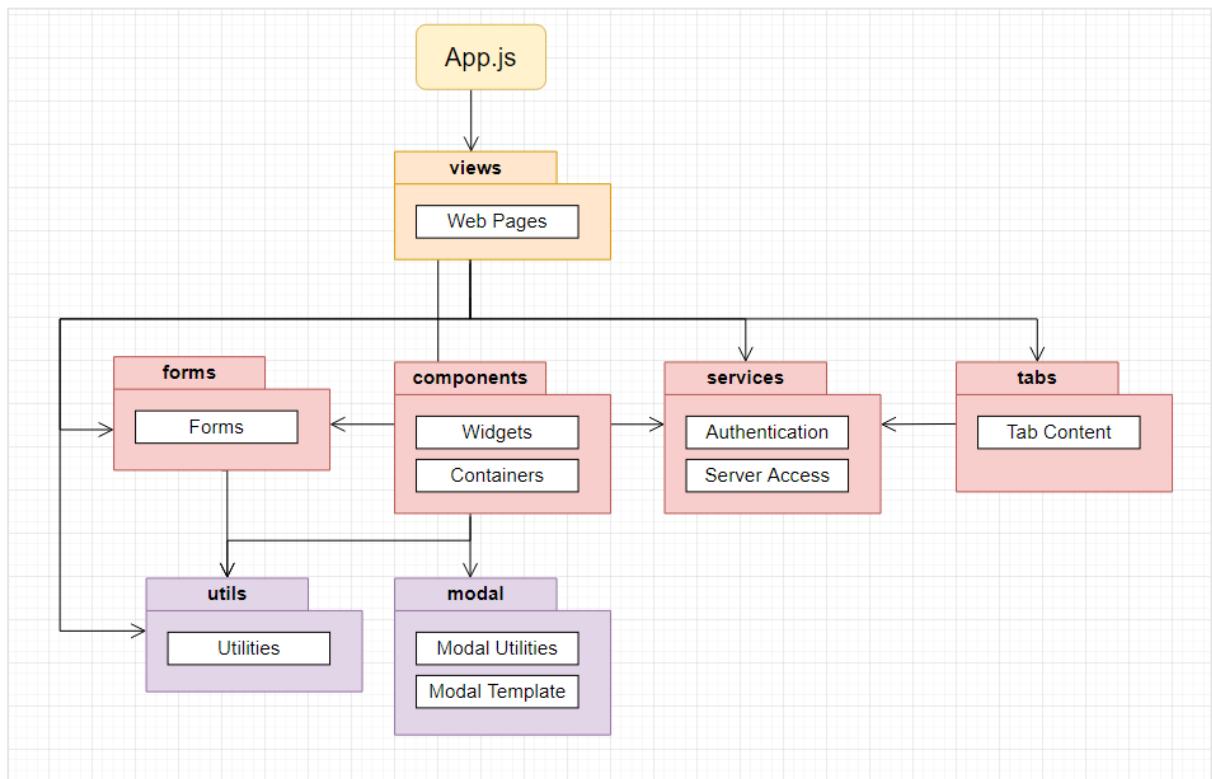


Figure 57 - Web App Class Front-end Relationships

The relationships between pages is shown in displayed in Figure 58. It should be noted that the “Overview” and “Recordings” pages are not pages that have their own URL but are instead tabs within the “View Test Results” page.

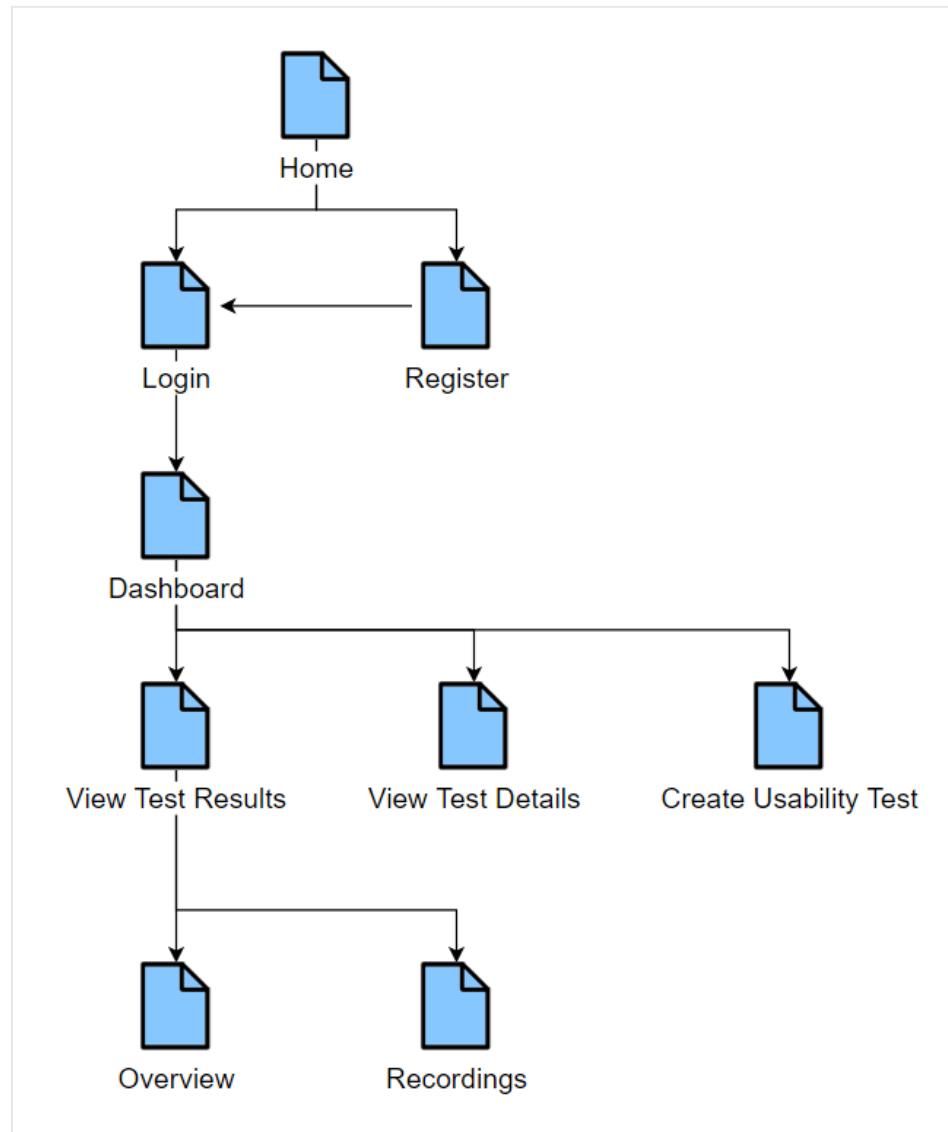


Figure 58 - Webapp Page Relationship Diagram

Home Page

The home page contains some general information about the application as shown in Figure 59. From there the user can navigate to the sign up and login page.

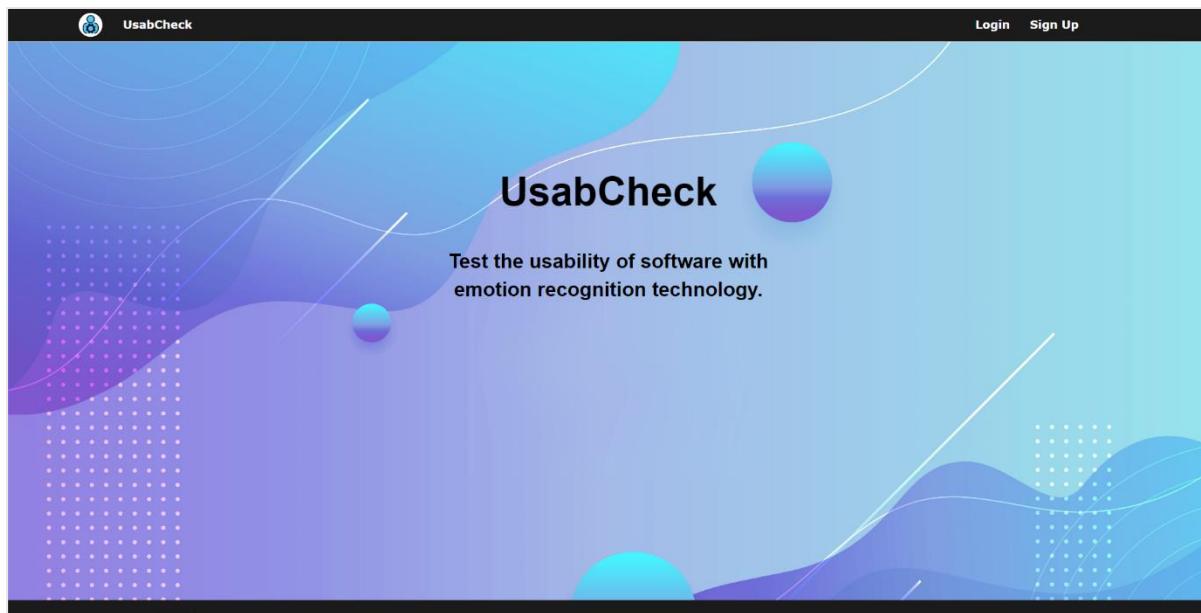


Figure 59 - Home Page

Register/Login

The login and register pages are separate pages and the UI is shown in Figure 60. For simplicity there is no email address and no password changing as the complexity of the project does not stem from this part.

Two side-by-side UI mockups. The left one is titled "Login" and the right one is titled "Register". Both pages have a light gray header bar. Below the title on each page is a "Username" field represented by a white input box with a thin gray border. Underneath each field is a "Password" field, also represented by a white input box with a thin gray border. At the bottom of each page is a blue rounded rectangular button labeled "Login" on the left and "Sign Up" on the right.

Figure 60 – Login and Register Pages

Dashboard

When the user first logs in the dashboard page will indicate that they need to create a project as shown in Figure 61.

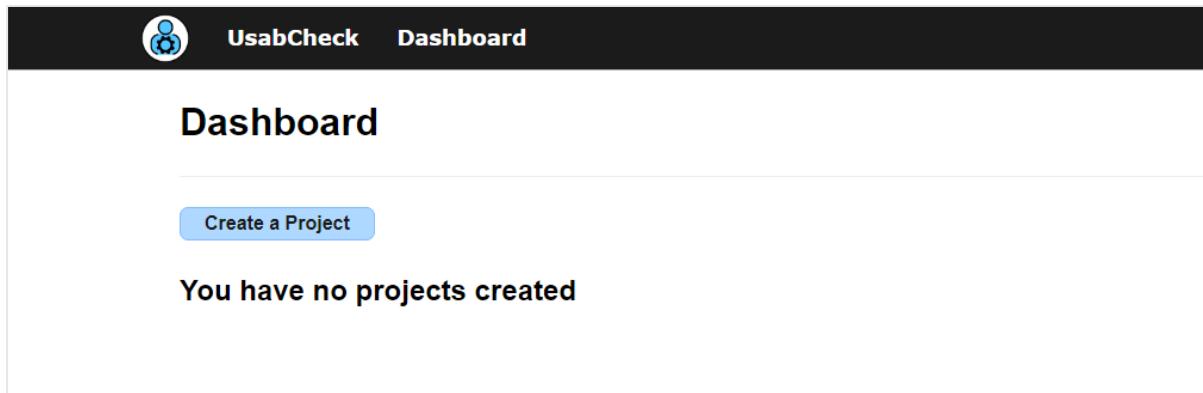


Figure 61 - Dashboard (No Project)

When they click the “Create a Project” button a modal will show up and they will be asked to enter a project name as shown in Figure 62. The starter code for the modal was obtained from a tutorial by Krissanawat Kaewsanmuang [\[Source\]](#). The code was modified to suit the needs of the project by adding additional parameters and entirely restyles to fit the theme of the application.

The components/classes involved are *modalButton.component.js* (the button that is clicked to open the modal), *modalTemplate.js* (the modal box), *modalContainer.component.js* (wraps the button and template into a single component), and *createProjectForm.js* (the content within the modal).



Figure 62 - Dashboard (Project Create)

Once a project is created additional options show up on screen as shown in Figure 63. This is the “Choose Project” dropdown and “Delete a Project” modal pop up. To create a dropdown the bootstrap dropdown was used but was heavily modified and built upon. Firstly the style of the button was changed entirely and the drop down arrow icon had to be added to indicate that it is a dropdown. A new component named *dropdownGenerator.component.js* was implemented that utilized the dropdown from bootstrap. The dropdown generator would take the data, populate the dropdown and managed the events and state of the component. This includes the keeping track of the currently selected item and handling the *onItemSelect* event. Although bootstrap was used in the creation of the dropdown there was still a lot of work that had to be done integrate the dropdown and ensure it looks appropriate.

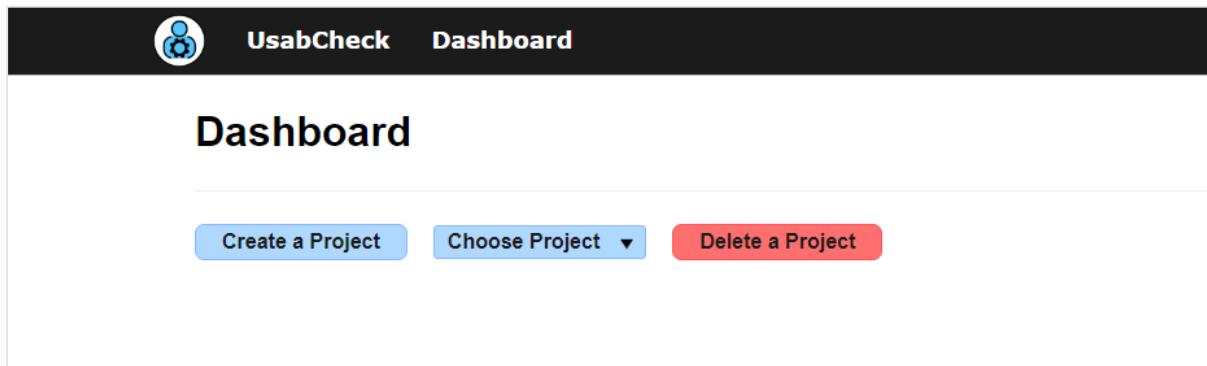


Figure 63 - Dashboard (Project Created)

When a project is selected, if there is no usability studies/tests created then the user/researcher will be asked to create a usability test. This can be seen below in Figure 64.

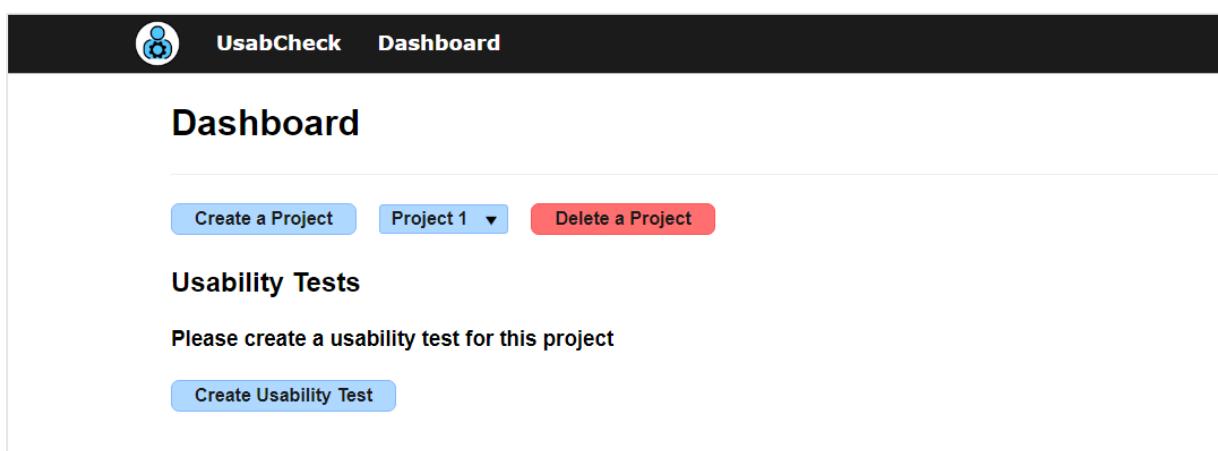


Figure 64 - Dashboard (Project Selected & No Usability Tests)

When the researcher clicks on the “Delete a Project” button a popup will show on screen notifying them about the consequences of deleting a project. All the tests and data relating to that project will be deleted permanently. They will select the project they want to delete from a dropdown and click on the “Delete Project” button. A similar popup is shown when the researcher would like to delete a usability test.

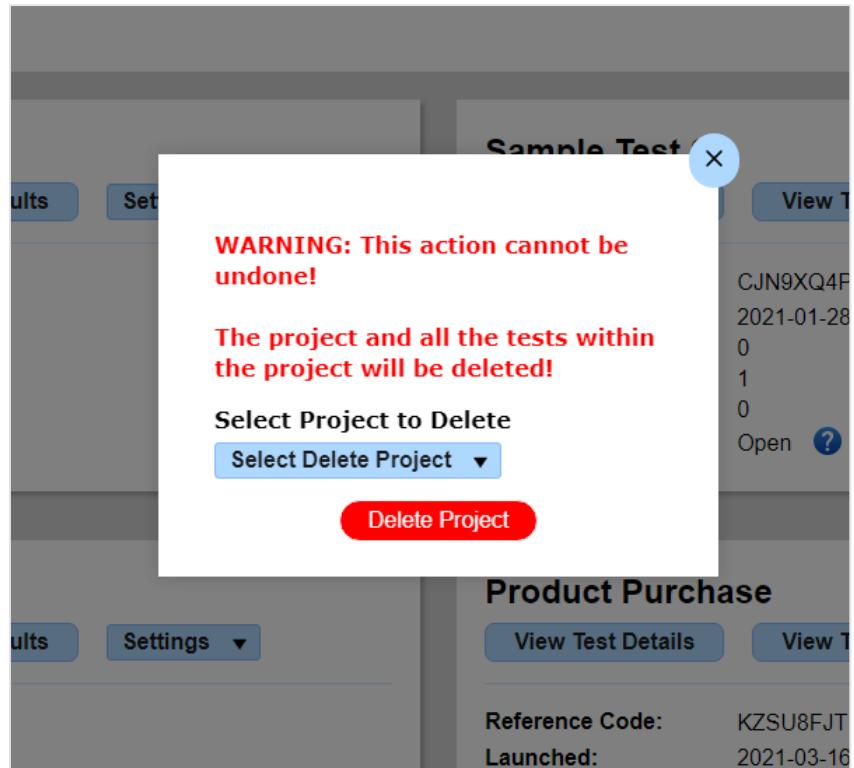


Figure 65 - Project Delete Modal

When a researcher has tests created and they select a project, the usability tests for that project will show up under the heading “Usability Tests” as can be seen in Figure 66. The details about each test is displayed such as the number of tasks and questions within the test, and the status of the test. The reusable component box used to display the data is named *testContainer.component.js*.

A reference code/number is generated when a usability test is created. This code is entered into the local python application which the participant will use to take the test. The purpose of using a randomly generate number is that only the researcher will know the random reference code and this will help ensure that only individuals with the code can take the test. An idea for future work could be to implement a password for security reasons to ensure that unauthorized individuals, even with the reference code cannot take the usability test.

The researcher can view the details of the test which includes the sequence of tasks and questions by clicking the “View Test Details” button. They can also view the test results which will display the data from the usability studies. Under the “Settings” dropdown, the test can be deleted or closed. A closed test prevents participants from taking a test, meaning no additional data can be added to the results. The usability test can be reopened or closed at any time.

The screenshot shows the UsabCheck dashboard interface. At the top, there is a navigation bar with a user icon, the text "UsabCheck", and a "Dashboard" button. Below the navigation bar, the word "Dashboard" is prominently displayed. A horizontal menu bar contains four buttons: "Create a Project" (blue), "Project1" (with a dropdown arrow) (light blue), "Delete a Project" (red), and "Create Usability Test" (blue). The main content area is titled "Usability Tests". It displays two entries, each in a separate card:

- Sample Test 5**
 - View Test Details
 - View Test Results
 - Settings ▾

Reference Code: H8VH1NLA ?
Launched: 2021-01-27
No. of Tasks: 3
No. of Questions: 4
No. of Participants: 35
Status: Open ?
- Sample Test 7**
 - View Test Details
 - View Test Results
 - Settings ▾

Reference Code: H9ON26UF ?
Launched: 2021-01-28
No. of Tasks: 1
No. of Questions: 1
No. of Participants: 0
Status: Open ?

On the right side of the dashboard, there are vertical panels for "Sample" and "Product" sections, which are partially visible and show placeholder data.

Figure 66 - Dashboard (Project Selected & There are Usability Tests)

Create Test

When the user clicks on the “Create a Usability Test” button in the dashboard they will be redirected to the “Create Usability Test” page which is displayed in Figure 67. The initial screen will show the currently selected project and the user will need to enter details such as the name of the test. The (?) icon will open a modal popup with information that will explain to the user what each field is and an example of the input they can enter. The “scenario” of a test will be displayed to the participant in the local python application before they begin the usability study. The scenario is used to give the participant any information they should know before the usability study begins.

The screenshot shows the 'Create Usability Test' interface. At the top, it says 'Create Usability Test' with a help icon. Below that, there's a 'Project' dropdown set to 'Project1'. A 'Test Name' field has a help icon and contains the placeholder 'Test Name'. Under 'Scenario', there's a large text area with a help icon. In the 'Pre-test Questions' section, there are two buttons: '+ Text Question' and '+ Multiple Choice Question'. In the 'Usability Test' section, there are three buttons: '+ Task', '+ Question (Text)', and '+ Question (Multiple Choice)'. At the bottom, there's a prominent green 'Create Test' button.

Figure 67 - Create Test (Initial Screen)

Examples of the information popups when the (?) button is pressed are shown below in Figure 68. These give information about what a usability test is and an example of what the input could be. The classes involved are *infoForms.js* (contains the content for each popup), modal classes mentioned previously (creates the modal), and *infoModalUtilities.js* (creates the (?) button and wraps all of the functionality into one component).

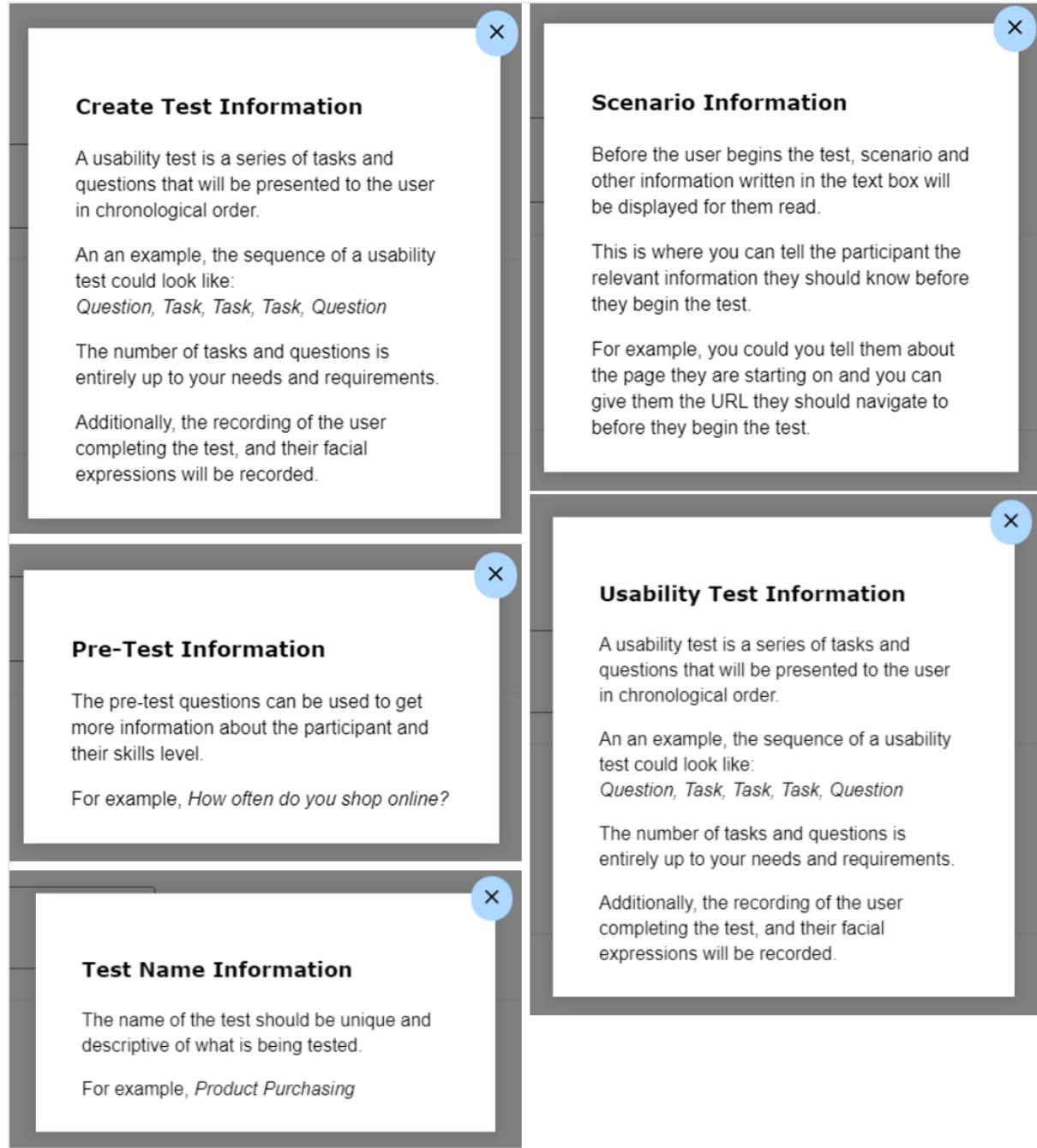


Figure 68 - Information Popups

The choice of questions within the usability study are “Text Answer” and “Multiple-Choice” questions. These can be seen in Figure 69. These components were created entirely from scratch with the exception of the dropdown component discussed previously. For the text answer question the researchers enters a question in plain text and the participant will answer in plain text. Under the “Options” dropdown the question box can be deleted and this prevents the user from accidentally clicking “Delete” as they now need to take two steps to delete the box. Using the arrows on the right the user can rearrange the order of the boxes. The component for the text answer box is *textQuestionCreate.component.js*.

The multiple-choice question box allows the researcher to enter the question and an arbitrary number of options. These options can be dynamically added and removed as needed. The component for the box is *multipleChoiceQuestion.component.js* and the dynamic list of answer choices is *dynamicList.component.js* which was also implemented from scratch.

The screenshot displays a user interface for creating pre-test questions. At the top, there is a header titled "Pre-test Questions" with a help icon. Below the header, there are two main sections:

- Question (Text Answer):** This section contains a "Question" input field and a "Options" dropdown menu with a help icon. To the right of the input field are two small arrow icons for rearranging the order of the boxes.
- Question (Multiple Choice):** This section contains a "Question" input field and a "Options" dropdown menu with a help icon. To the right of the input field are two small arrow icons for rearranging the order of the boxes.

Below these sections, there is a "Answer Choices" heading followed by three "Answer Choice" input fields, each with a red circular "X" button to its right. At the bottom of this section is a blue "Add Answer" button.

At the very bottom of the interface are two buttons: "+ Text Question" and "+ Multiple Choice Question".

Figure 69 - Create Test (Pre-Test Questions)

The body of the usability study consists of tasks and questions for the participant to complete as can be seen in Figure 70. This stage comes after the pre-test questions. The question components are the same as previously discussed and these components can be arranged in any combination and sequence to fit the requirements of the usability study. The task box component is *taskCreate.component.js* and is similar to the multiple-choice question in that it contains a dynamic list. The researcher can enter an arbitrary number of instructions for the participant to complete.

Usability Test ?

Task Task Name Options ?

Enter Task Instruction ×

Add Instruction

Question (Text Answer) Options ?

Question

Question (Multiple Choice) Options ?

Question

Answer Choices

Answer Choice ×

Add Answer

+ Task + Question (Text) + Question (Multiple Choice)

Figure 70 - Create Test (Body of Usability Study)

View Test Details

The researcher can view the usability test/study that they have created and the details of that test. The colour scheme from the create usability test is kept for consistency. The test details of a sample usability study on the view test details page can be seen in Figure 71.

View Usability Test

Usability Test Details

Test Name: Product Purchase
Launched Date: 2021-03-16
Status: Open

Scenario/Information

In this test you are going to purchase a product.

Pre-test Questions

Question (Multiple Choice)

Question:
How adept are you with online shopping?

Choices:

- Not at all
- Somewhat
- Very adept

Usability Test

Task

Task Name:
Item Selection

Instruction:

1. Search for "phones" in the search bar
2. Filter the items in order of High to Low
3. Click on the first item
4. Add item to basket

Question (Text Answer)

Question:
Do you have any feedback for us?

Figure 71 - View Test Details

View Test Results (Overview)

When the researcher clicks on the “View Test Results” button in the dashboard page they will be redirected to the view test results page which contains the “Overview” and “Recordings” tabs. The page will first open the “Overview” tab which will be discussed in this section.

During the development when it came to the implementation of the Overview and Recording tabs it was discovered that a new design was needed. This was for several reasons. Firstly, some parts of the original design no longer made as much sense such as viewing the task charts by selecting the task from a dropdown. With the dropdown approach only one task chart could be viewed at a time. It was much better to simply display all of the tasks charts at once making it easier to compare performances of the tasks. Secondly, the original design did not account for displaying of text answers and that needed to be designed. Finally, some aspects of the original design such as an “Emotion bar chart” was made redundant with the new widget created in the “Recordings” tab which will be discussed later. In short, the addition and removal of aspects in the original design meant that a new design had to be created. The new design is shown below in Figure 72 and Figure 73.

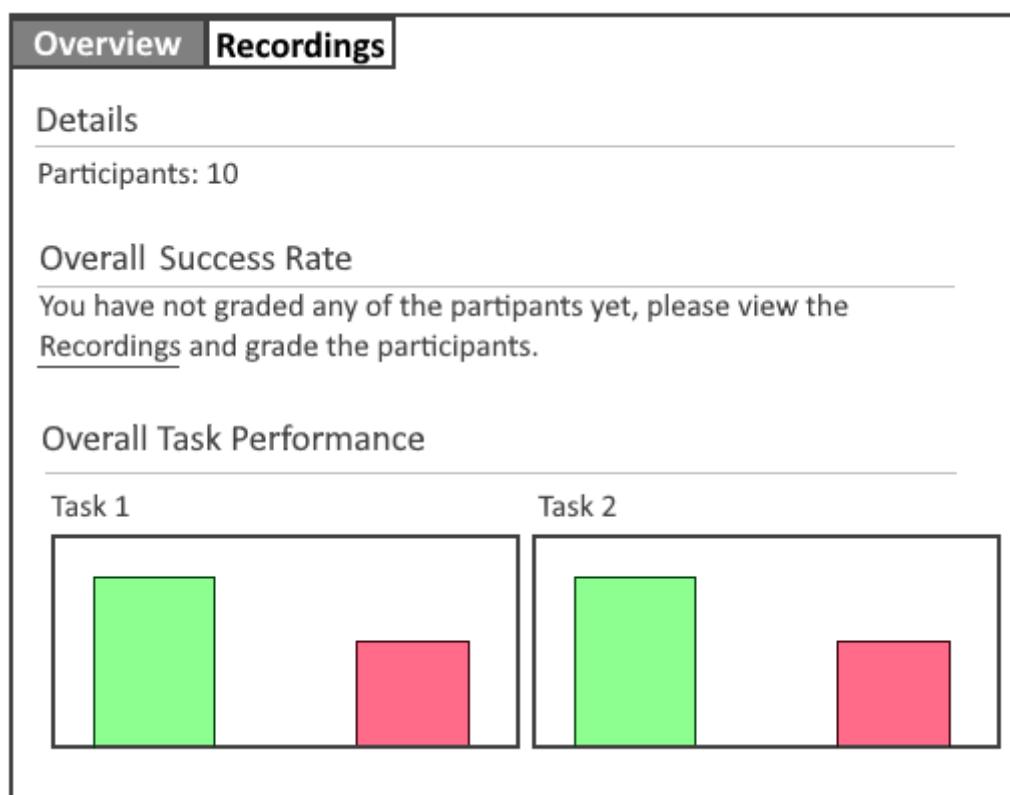
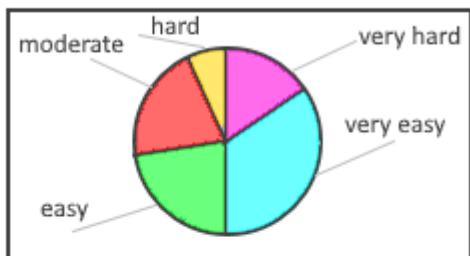


Figure 72 - Overview Tab New Design (Part 1)

Overall Question Answers

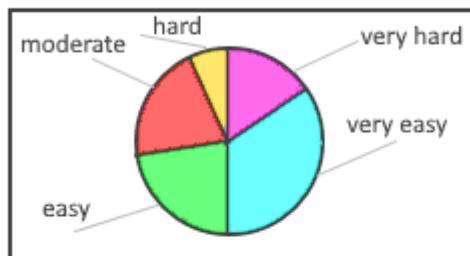
Multiple Choice Question:

How difficult did you find the task?



Multiple Choice Question:

How difficult did you find the task?



Text Question Answers

Question: *What is your name?*

Bob

Alice

John

Peter

Figure 73 - Overview Tab New Design (Part 2)

The design shown in Figure 72 and Figure 73 is further improved upon after expert evaluations and research. For example, Gestalt Laws of Perceptual Organization [\[Source\]](#) discusses how individuals perceive objects. This includes perceptually filling in missing information and grouping objects. The principle of enclosure was applied whereby the background colour of each group of content was shaded a light grey colour to allow for users to easily identify where one group of content begins and where it ends. The principle of proximity was applied throughout the entire application, this includes leaving a space between groups of content to separate them.

The implementation of the design will be discussed next. When the researcher views the results of a test when no participant has yet taken the usability test, a message will explain that there is no data to display. This can be seen in Figure 74.

Test Results: Product Purchase

Overview Recordings

Details

No. of Participants: 0
No. of Tasks: 1
No. of Questions: 2

There is no data to display as no participants have taken the test.

Figure 74 - View Test Results (No Data)

Otherwise, the researcher will be presented with charts and data. Firstly, there is an “Overall Task Success Rate” that displays the task grades of all of the participants in a single chart. For example, if there are 31 participants and there are 3 tasks then there are 93 total tasks attempted by all participants. The tasks will be graded manually by the researcher in the “Recordings” tab. Below in Figure 75 the number of tasks passed is 24, number of tasks failed is 7 and 62 tasks are yet to be graded. The charting library used is *ApexCharts*. The JavaScript class for this page is *testResultsOverview.component.js*.

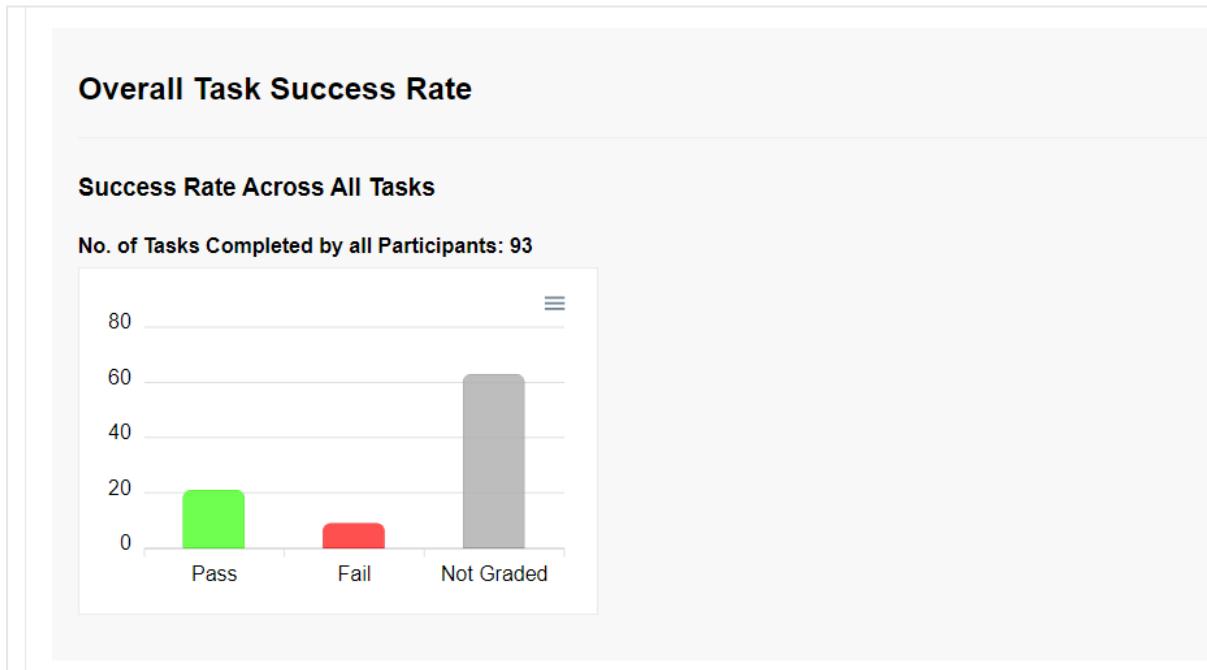


Figure 75 - View Test Results Overview (Overall Task Success)

Although a charting library was used there was still a significant amount of configuration and processing of data that needed to be done to display the charts. For example, the data needed to be retrieved and parsed and documentation was read to ensure that the charts display perfectly. One of the challenges encountered was with pie charts which would not scale correctly by default as the number of items within the chart would change the size of the pie. This was due to the legend of the chart increasing in size and shrinking the circle instead of the chart increasing in size while keeping the size of the pie consistent. After reading the documentation and a lot of trial and error it displayed correctly.

To give the researcher a clearer idea of how participants performed for each task of the usability study there is a section named “Individual Task Performance” as shown in Figure 76. The number in square brackets before the name of the task is the “sequence number”, it represents the order of the task or question in the test. Number 3 means it is the 3rd task or question that was presented to the participant. The researcher can hover over the bars in the chart to get the exact number the bar represents.

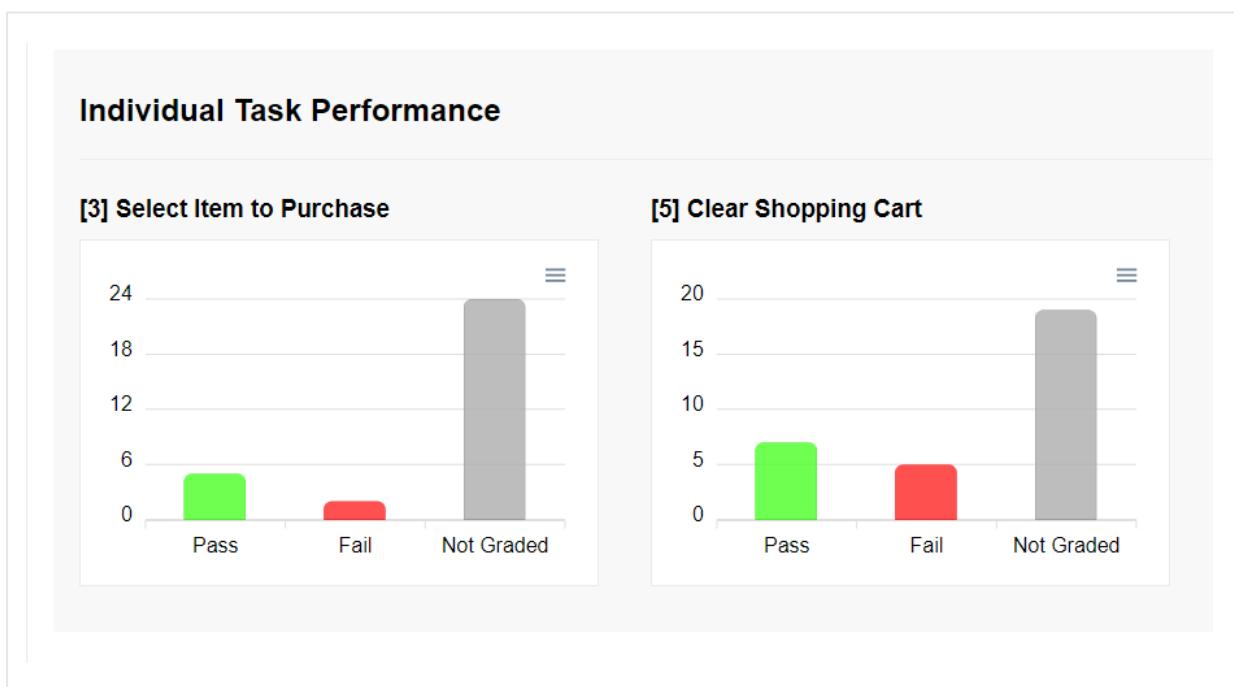


Figure 76 - View Test Results Overview (Individual Task Performance)

The answers to the multiple-choice questions is displayed as a pie chart as shown in Figure 77. The researcher can hover over the segments in the charts to get the exact value. For example, 38.7% is 12 out of 31.

Multiple-Choice Question Answers

[4] Question: *How would you rate the difficulty of that task?*

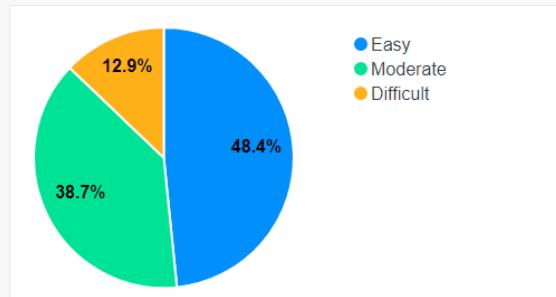


Figure 77 - View Test Results Overview (Multiple-Choice Question Answers)

The text answers to open-ended questions cannot be displayed on a pie or bar chart as each reply is expected to be different. Instead the answers are displayed on a list as shown in Figure 78. The researcher can scroll and read the reply each participant has given.

Text Question Answers

[3] Question: *When searching for the Patrick's day items where did you search first?*

Answers:

Quick Recommendations on home page

Logo on website of St.Patrick's day

Below the banner of the webpage

[4] Question: *Have you encountered any problems during the task?*

Answers:

No, but it was really slow and i wasnt sure if things were working.

When typing into the search bar it didnt come up as suggested when I wrote in short

The text box of the application was in front of the "items per page" selection option

Figure 78 - View Test Results (Answers to Text Questions)

View Test Results (Recordings)

In the “Recordings” tab the researcher can select a test instance from a dropdown and watch the video of the screen recording for each participant. As planned the videos are uploaded to Vimeo by the local Python application after the participant finishes the test. The emotion timestamps and all of the other data is uploaded the backend server where it is stored. The frontend retrieves that data and displays it.

In the design section there were ideas for how to display the emotions that the user shows on a video timeline. The initial idea was to display the video markers with a library named videojs-markers, however after that library did not display the video markers and after trying several other libraries with no success a new design was created. The new design is shown in Figure 79 which closely resembles the final result.

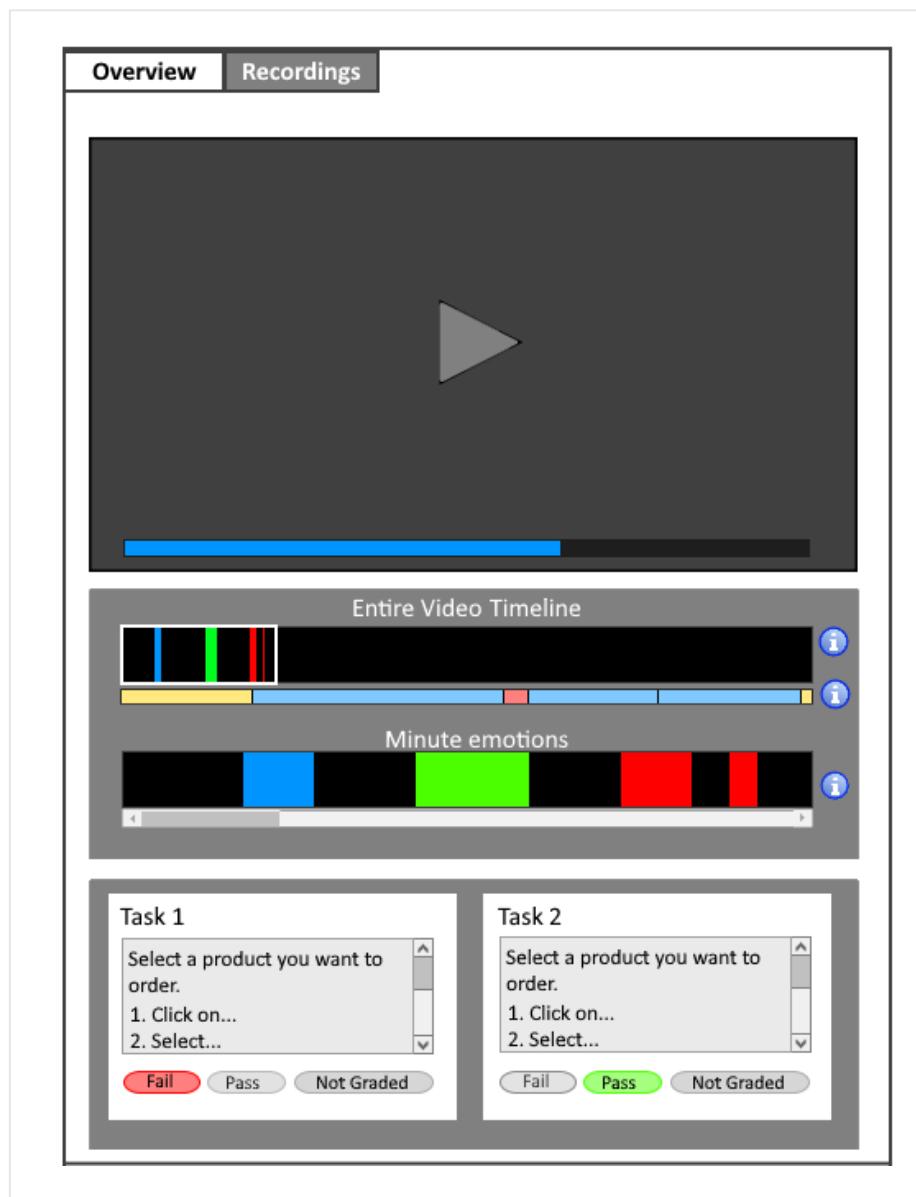


Figure 79 - Recordings Tab New Design

The recording of the video, the emotion timestamps and task timestamps can be seen in Figure 80. The JavaScript class which contains the tab content is *testResultsRecordings.component.js*.

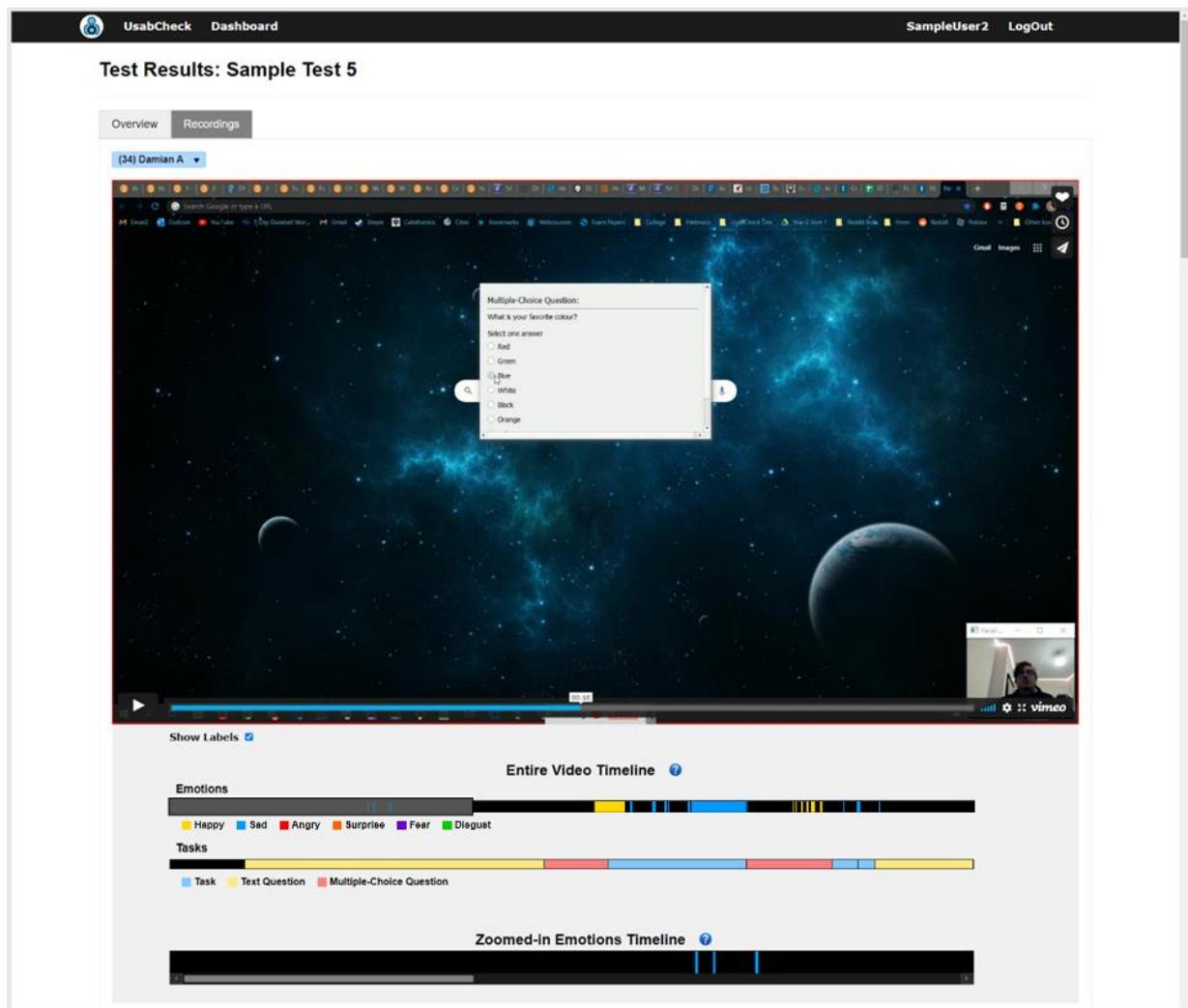


Figure 80 - View Test Results Recordings (View Video)

The video fullscreen functionality had to be changed with the help of the Vimeo API. Normally, when the researcher goes into fullscreen they are unable to look at the video timelines located below the video which displays when emotions occur. Changes were made to allow the researcher to scroll down while also having the video in a sort of fullscreen.

When the user clicks on the fullscreen button in the bottom right corner of the video the application detects that the user entered fullscreen and using the Vimeo API, the fullscreen mode is immediately exited. The application then performs calculations to stretch the video as much as possible to either the width or the height of the screen, whichever is the longest. The end result is that the user can view the video in fullscreen (as much as possible) while still having the ability to scroll down. Black borders were added to the sides of the video. To exit fullscreen the user clicks the same button they did to open fullscreen which is located in the bottom right corner of the video.

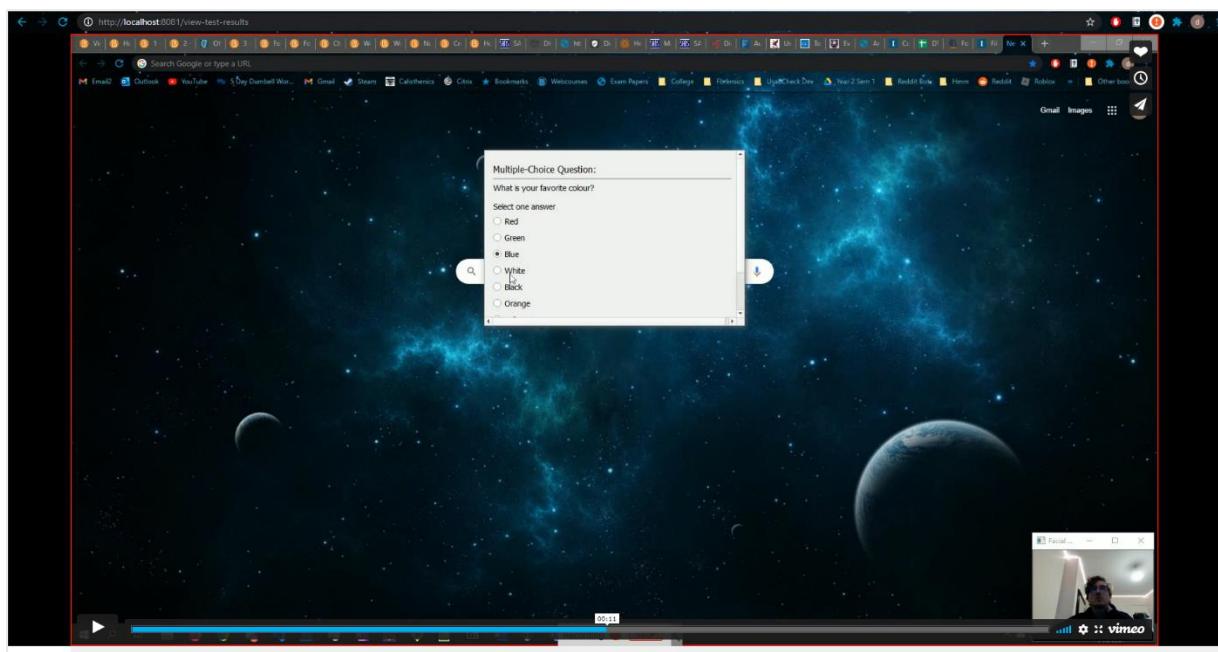


Figure 81 - Video Test Results Recordings (Video Full-Screen)

The displaying of the emotion data and the timestamps of the participant progressing through the usability study is shown in Figure 82. The design in Figure 82 is created from the ground up and has been incrementally improved upon after multiple expert evaluations, discussions, and research.

There are 3 bars/timelines, the first two are the “entire video timeline” bars and show emotions across the entire video. The progress bar in the video is in line with the bars underneath it. The first bar shows the emotions of the participant and the second bar shows the tasks and questions that the participant completes. This shows the researcher exactly when a task begins and ends and what emotions occur within that task. The colours/timestamps within the bar can be clicked to jump to the position in the video and this was implemented with the help of the Vimeo API.

The 3rd bar was created because videos can be long and clicking on a pixel-wide timestamp can be difficult, if not impossible for individuals with disabilities. The 3rd bar is a zoomed in section of the 1st bar and allows the researcher to scroll along the “entire emotions timeline” and take a closer look. The researcher can click on the zoomed in emotions bar to jump to the time in the video just as with the other two bars. As the researcher is scrolling on the zoomed bar the grey slider window shown on 1st bar will move to indicate the position of the zoomed in bar.

This design replaces a few of the widgets/data display options that was planned in the design section. Firstly, there are the video markers, this design displays the location of emotions across multiple bars and is far better than what an existing library could provide. Secondly, the journey map is no longer needed as this effectively serves as a journey map that also has the ability to interact with the video. Thirdly, there was an idea to count the number of emotions and their length and create a sort of point system that can then be used to represent the emotions on a bar chart. That idea didn’t quite make sense to begin with and this is a far better and clearer alternative. The component in the code is named *videoBars.component.js*.

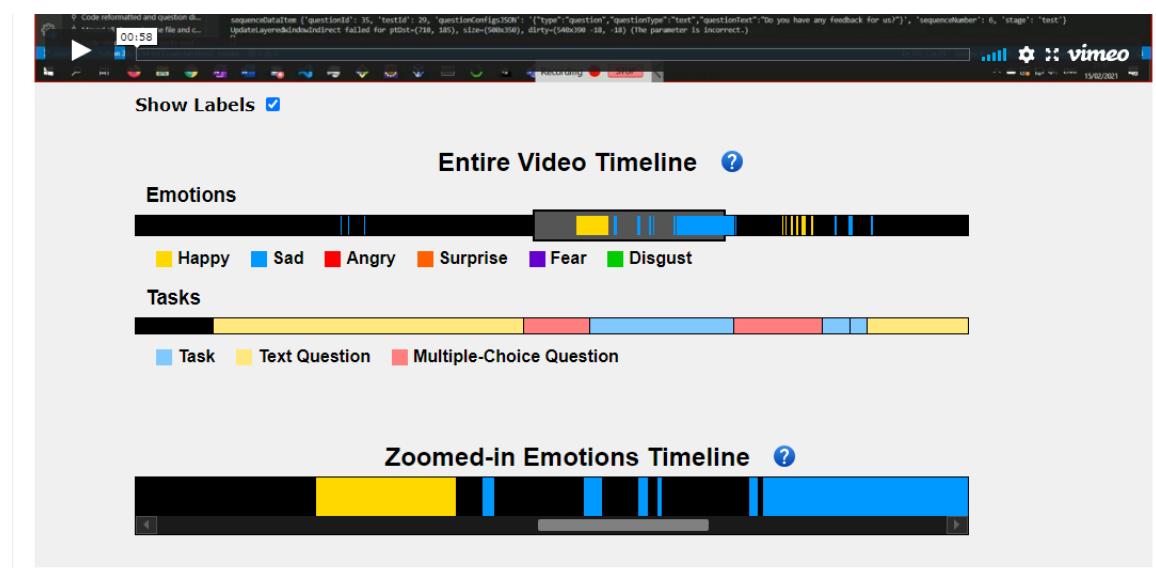


Figure 82 - Video Timeline Bars

For experienced user there is the options to hide the labels to reduce the amount of space the chart takes up and as a result reduce the amount of scrolling the researcher will need to do. The timeline charts with hidden labels are shown in Figure 83. It should also be noted that the video bars scale automatically with the size of the screen and are always inline with the progress bar in the video.

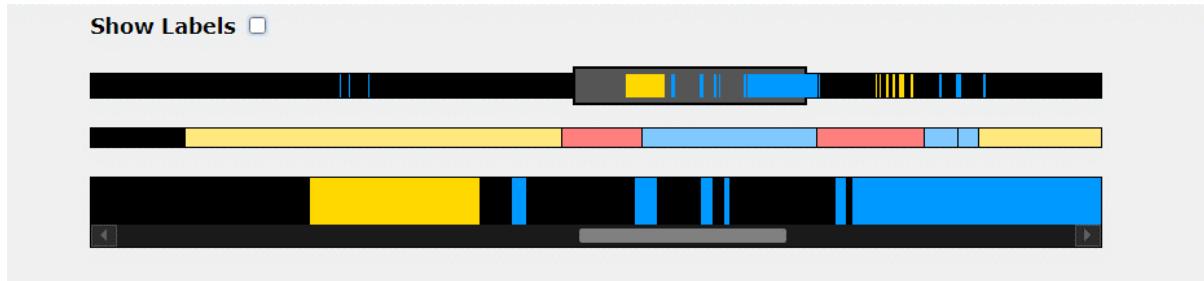


Figure 83 - Video Timeline Bars (No Labels)

To discuss the progression and development an earlier iteration did not have the labels nor any information that could tell the user the bars and colours represent and the functionality is available. One of the challenges were that the labels it would take up too much space. This was solved by making a checkbox to show and hide the labels. Another piece of functionality that was added across the entire application was to implement the blue (?) information buttons that the user can click on to get more information if they need it. There was also application of Gestalt's laws to make the design more intuitive. There is a margin between the first two bars and the zoomed in bar to group the first two bars. The video bars are also designed to be dynamic and will shrink or expand with the size of the page to always be in line with the bar in the video.

To better understand how people associate emotions and colours a dissertation by Gohar Kadar, Navit 2008 [\[Source\]](#) was reviewed. Below in Figure 84 is a histogram showing the percentage of individuals who have agreed that a colour is appropriate for an emotion. It can be seen that red is very closely associated with anger and yellow with happiness.

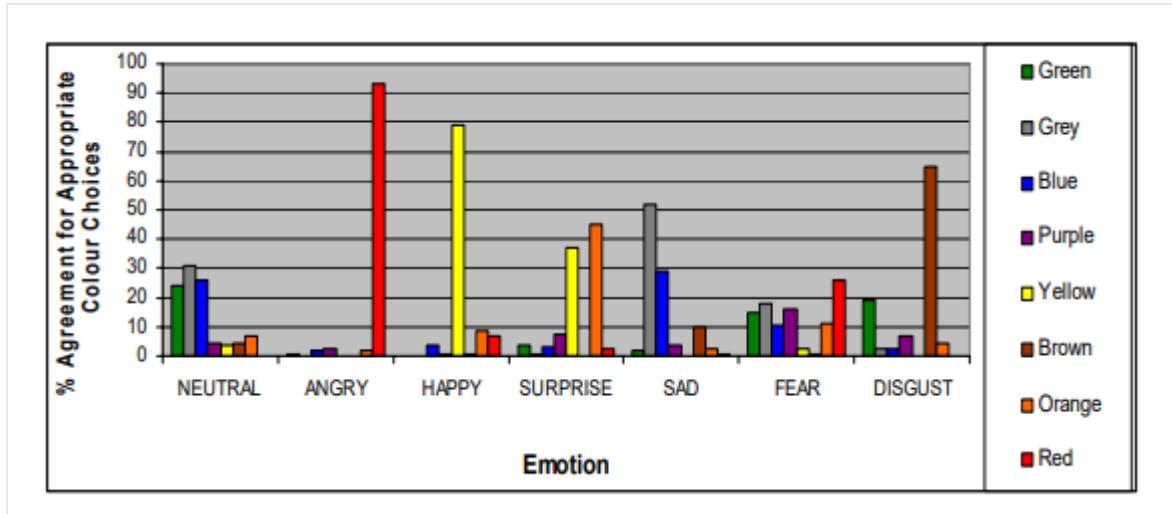


Figure 84 - Percentage agreement of Appropriate colour choices for each of the emotional categories

When developing the web application and before conducting any research the colours were assigned as shown in Figure 85. After researching colours and emotions the final colour association used is displayed in Figure 86.



Figure 85 – Colours Personal First Guess



Figure 86 - Colours After Research

The task grading is displayed in Figure 87. The task name and the instructions for the task are shown and the researcher will change the state from “Not Graded” to either a “Pass” or “Fail” depending on whether or not the participant has completed the task correctly, if at all. When the researcher opens the “Overview” tab again the bar charts will automatically update to the new values.

The screenshot shows a "Task Grading" section with three separate boxes, each representing a different task:

- Task: Select Item to Purchase**: Contains instructions "Do X.....", "Do Y.....", and "Select Z.....". Below the instructions are three buttons: "Fail" (gray), "Pass" (green, highlighted), and "Not Graded".
- Task: Clear Shopping Cart**: Contains instructions "Do A.....", "Do B.....", and "Select C.....". Below the instructions are three buttons: "Fail" (red), "Pass" (gray), and "Not Graded".
- Task: Purchase Item**: Contains instructions "Do E.....", "Do F.....", and "Do G.....". Below the instructions are three buttons: "Fail" (gray), "Pass" (gray), and "Not Graded" (blue, highlighted).

Figure 87 - Test Results Recording Tab (Task Grading)

Other Components/Details

Notification updates are shown on screen whenever the user performs some action such as deleted/created a test or project. If the update succeeded then a notification like the one shown in Figure 88 will be displayed. If the update failed then a red notification will tell the user the action they tried to perform has failed as shown in Figure 89. A library was used to display these.

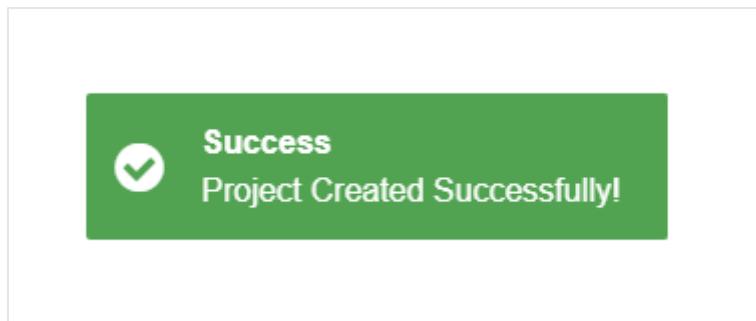


Figure 88 - Notification Updates (Successful)

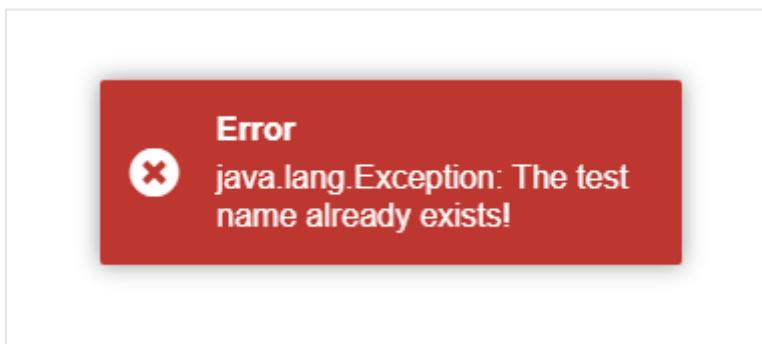


Figure 89 - Notification Updates (Failed)

If the user attempts to submit the create test form without filling in all of the fields they will be notified as shown in Figure 90. This was achieved with a html attribute.

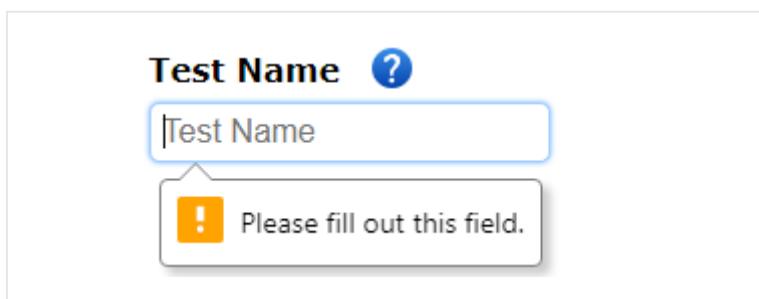


Figure 90 - Form Validation

Web App Deployment

The web application was first built and compiled with the help of the frontend-maven-plugin [\[Source\]](#) that runs the build of the frontend React application and with the help of the maven-antrun-plugin [\[Source\]](#) the compiled resources are moved to the maven project directory. Once the frontend is compiled, both the frontend and backend are packaged into the form of a JAR file.

The web application was deployed to Heroku App as it is a free service and makes redeployment easy. The GitHub repository to the UsabCheck project was linked to Heroku which allows Heroku to compile and deploy the application directly from the repository. There were a few challenges with the deployment of the UsabCheck web application, one of which is deploying from a sub-directory within the GitHub repository. A Heroku build pack [\[Source\]](#) had to be used to allow for deploying from a sub-directory. The web application was deployed to <https://usabcheck.herokuapp.com/>.

Database hosting was also required as the development database was localhost. CloudClusters.io provided a free trial for a MySQL database which has worked well and a decision was made to use their services for database hosting.

4.3. Local Python Application

4.3.1. Introduction

The local python application is the application that the participant is using to complete the usability study. This application runs locally on the machine the participant is using. The features of the application include displaying tasks, displaying questions, screen recording, facial expression recognition, uploading data to the backend web application, and uploading video to Vimeo. The library used to generate the graphical user interface is PyQt5.

4.3.2. Overview

The classes in the local application are shown below in Figure 91. The main program, *Main.py* is responsible for spawning of the windows and components, retrieving data, and uploading data.

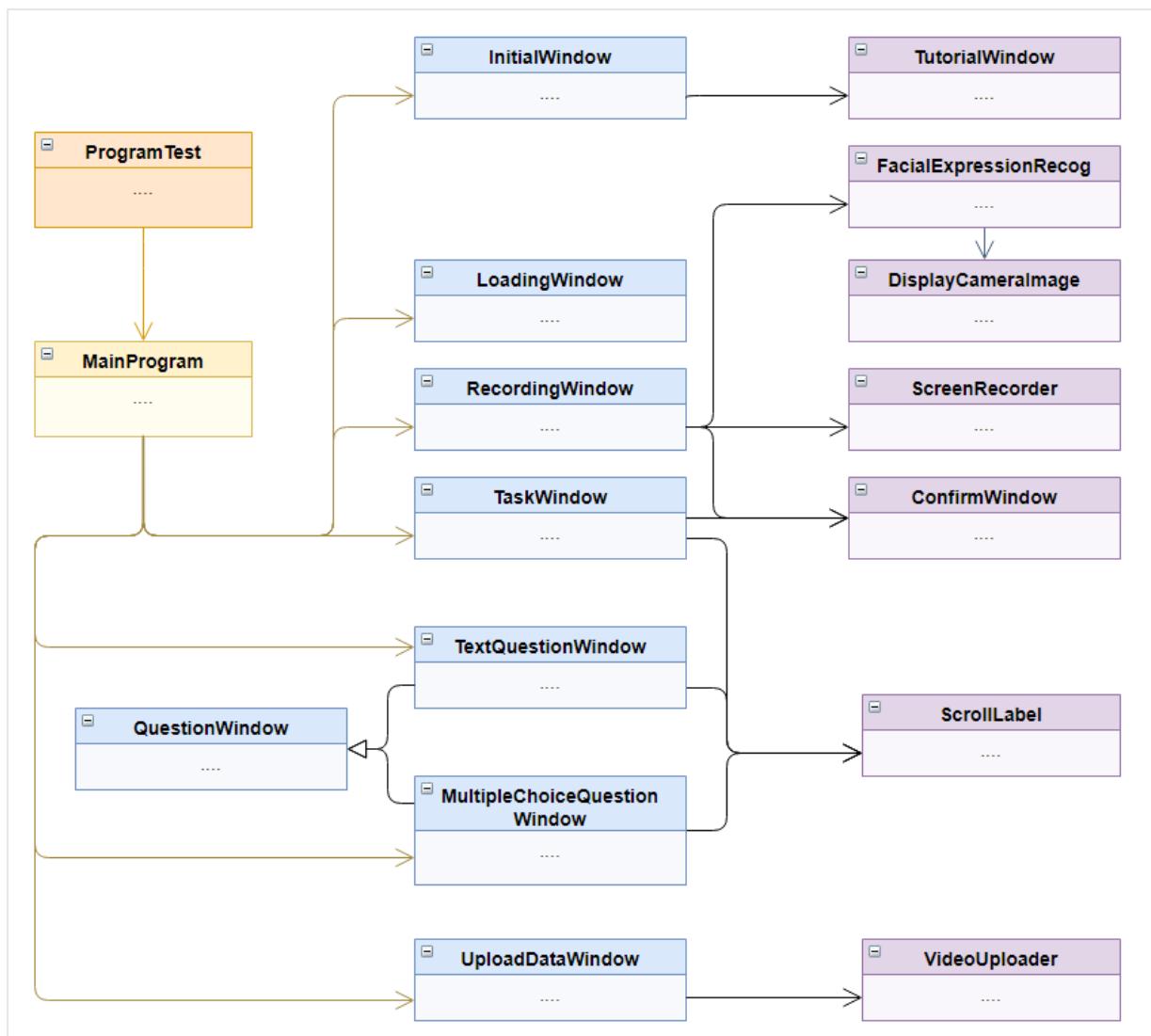


Figure 91 - Local Application Classes

4.3.3. User Interface

Starting Screen

As mentioned in the previous sections, when a usability test is created a reference number is generated. This reference number will be used to retrieve the data from the backend web application. The initial screen is shown in Figure 92. After the data has been retrieved from the server it is displayed as shown in Figure 93. The test details shown include the test name, created date, the number of tasks and questions, and the scenario are displayed. The details will be verified by the research who is in the same room as the participant. The scenario information is displayed for the researcher and participant and includes any information they should know before the test begins. The class is *InitialWindow.py*.

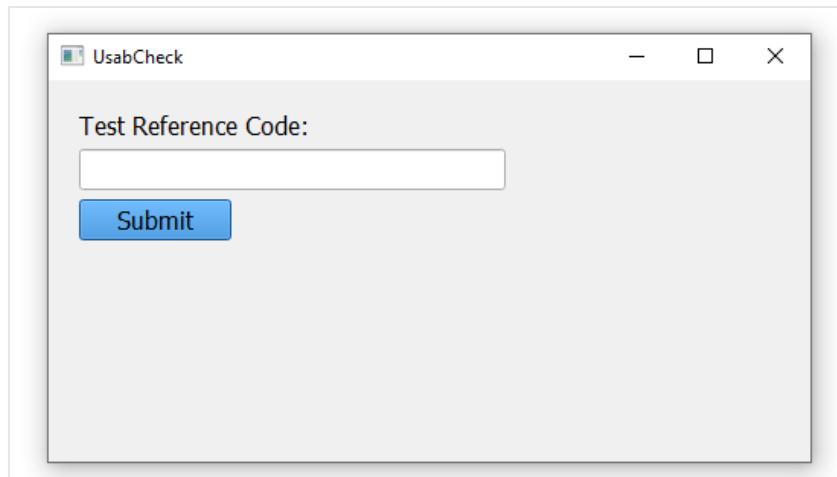


Figure 92 - Initial Screen

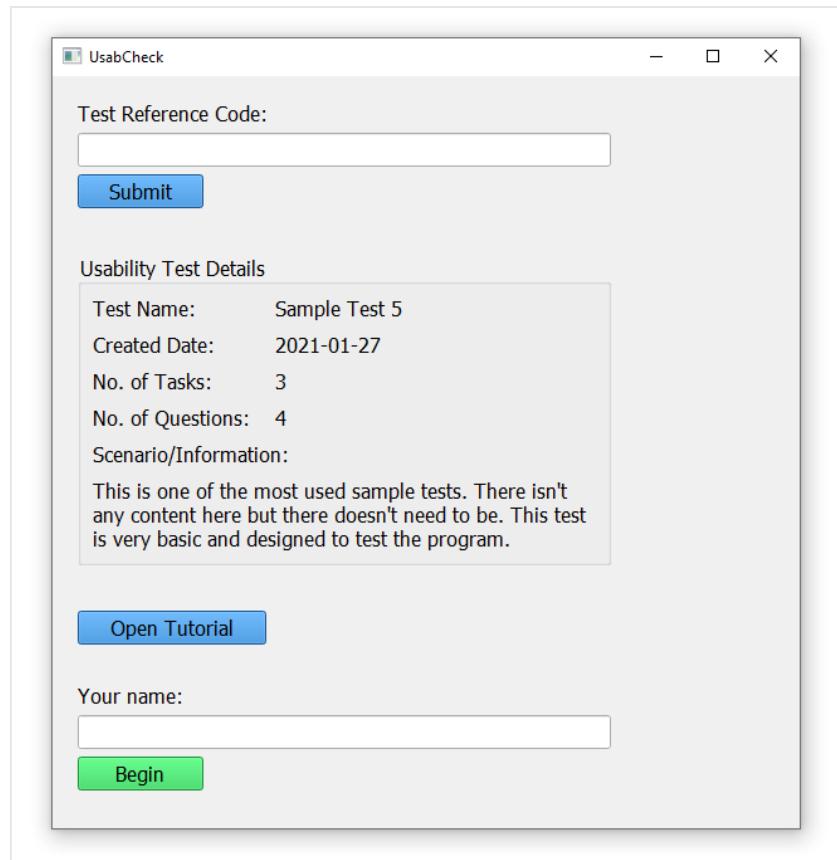


Figure 93 - Initial Screen (After Data Retrieval)

Tutorial Window

The user can click the “Open Tutorial” button shown in Figure 93 which will open a new window that displays a GIF on a loop of the various features of the UsabCheck local application. This tutorial is used to highlight that a window, such as the recording slider, despite not having a title bar can be moved by clicking anywhere on the window and panning. An example is shown in Figure 94 and features a GIF of a mouse cursor clicking on the slider and moving it left to right. It also shows the recording slider being minimized by clicking the grey button on the right of the slider. The participant can move to the next tutorial by clicking the “Next” button.

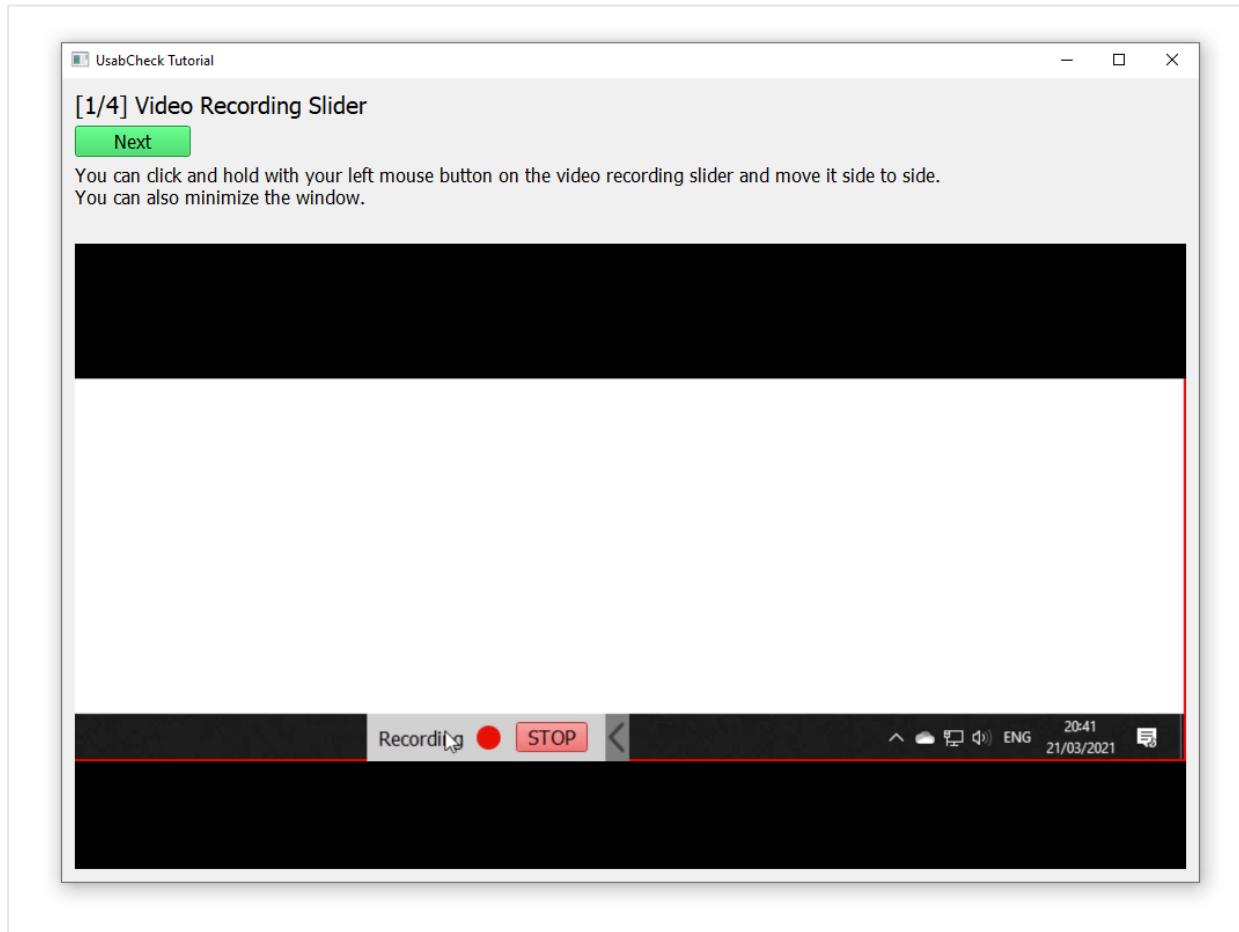


Figure 94 - Tutorial Window (Video Recording Slider)

Loading Screen

After the “Begin” button is clicked the application is going to connect to the camera and start up the FER model. This takes a few seconds and to be as transparent as possible a loading screen is displayed. First the camera is loaded and then it is configured (referring to the loading of the machine learning model). This is displayed in Figure 95 and Figure 96. The loading of the camera and the FER model is done on separate threads to speed up the process. The window shown is *LoadingWindow.py*.

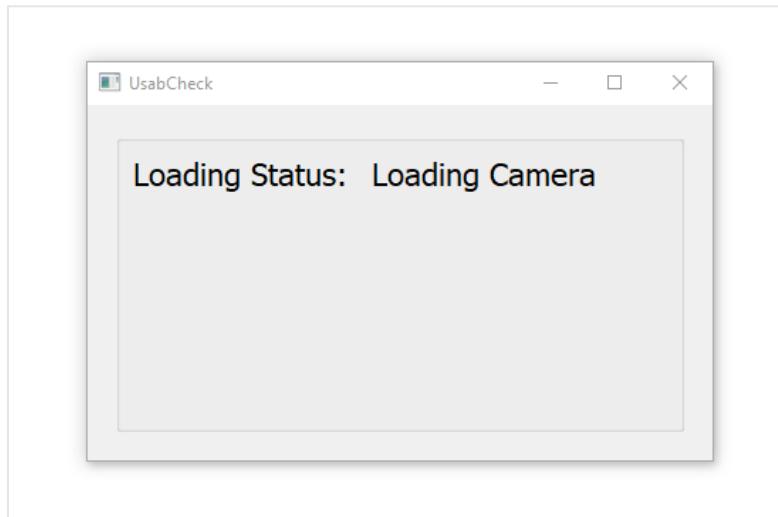


Figure 95 - Loading Screen (Loading Camera)

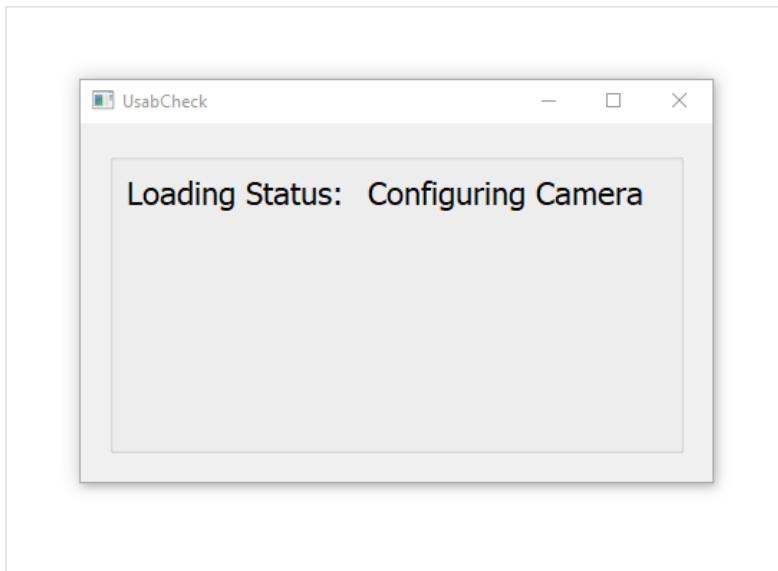


Figure 96 - Loading Screen Configuring Camera

Screen Recorder

The screen recorder GUI is shown in Figure 97. The content of screen has been replaced with white to highlight the recording UI. The screen is surrounded by a red border to indicate that screen is currently being recorded and there is a small window on the bottom that also shows that the screen is being recorded. The usability test can be stopped at any time by clicking the "STOP" button. The name of the window is *RecordingWindow.py* and it utilizes the *ScreenRecorder.py* and *FacialExpressionRecog.py* classes for the recording and facial expression functionality respectively.

The screen recording functionality had to be created from almost the bottom up, this is because across all of the online implementations for a Python screen recorder not a single one has synced the video frames to real time. The problem was that if for example, the video is configured to 30 frames per second, however, 20 frames are captured in a second then the video will be sped up as the number of frames determines the length of the video. This was a serious problem for two reasons. Firstly, the time an emotion was captured had to be perfectly in sync with the video. Secondly, the length of the video had to be consistent with real time as to show the researcher how long a participant took.

The solution implemented was to keep track of the elapsed time since the recording started and ensure that the number of frames was consistent with the time of the video. This means either duplicating or not adding frames as needed. Another challenge with the video recording was that the cursor would not display. None of the screenshot Python libraries that were tried were able to capture the cursor. Online solutions suggested to add the cursor into the image via image processing techniques. The implementation of the solution was to obtain the position of the user's cursor and use image processing techniques with the OpenCV library to add the cursor to the frame. That works well in most cases, however, the visual changes in the cursor such as the shape cannot be captured in this way.

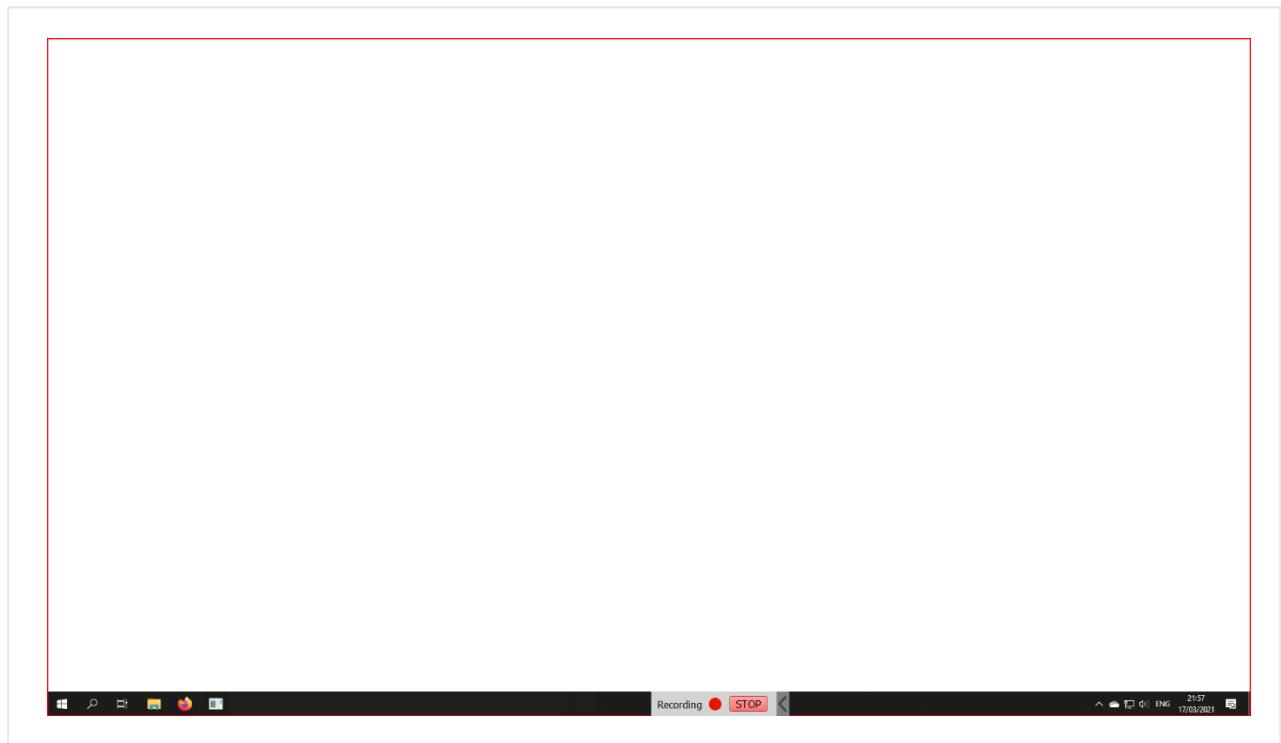


Figure 97 - Screen Recorder (Full Desktop View)

A close-up of the screen recorder GUI is shown in Figure 98. If the user clicks on the “STOP” button a pop up will ask them if they are sure they want to exit the study as shown in Figure 99.



Figure 98 - Screen Recorder (Components)

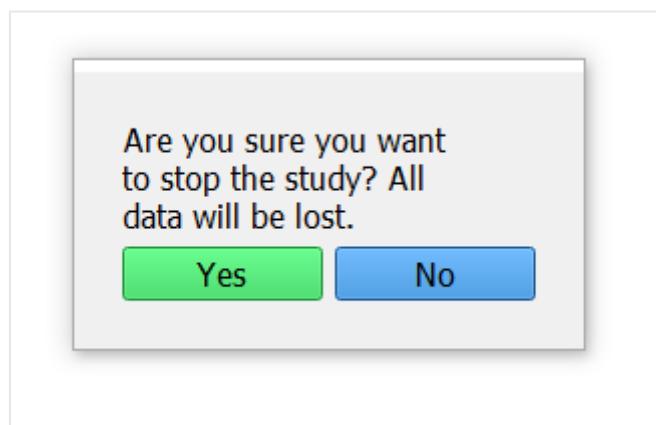


Figure 99 - Confirm Popup

Facial Expression Recognition

For reference, the development of the machine learning models is discussed in 4.4 Machine Learning. The implementation and integration of the FER model in the local app involves first loading in the model, then capturing the frame from the camera, detecting the face using Haar cascade, cropping the face from the image using the coordinates obtained from the face detection algorithm, passing the cropped face to the model, obtaining the prediction, and finally storing the data in a variable.

An example of the data which was recorded is shown in Figure 100. The startTime and endTime is obtained directly from the screen recorder as to ensure that it is consistent with the video. Error checking has been implemented to cater for the camera not detecting a face and camera loading. The class responsible for the functionality is *FacialExpressionRecog.py*.

```
{"startTime": "9.577802419662476", "label": "Sad", "endTime": 10.011746644973755},  
{"startTime": "10.011746644973755", "label": "Neutral", "endTime": 10.145078659057617},  
{"startTime": "10.145078659057617", "label": "Sad", "endTime": 10.213141918182373},  
{"startTime": "10.213141918182373", "label": "Neutral", "endTime": 10.445076704025269},  
{"startTime": "10.445076704025269", "label": "Happy", "endTime": 11.809728860855103},  
{"startTime": "11.809728860855103", "label": "Neutral", "endTime": 11.876791954040527},  
{"startTime": "11.876791954040527", "label": "Sad", "endTime": 13.146242141723633},  
{"startTime": "13.146242141723633", "label": "Neutral", "endTime": 13.213302373886108},  
{"startTime": "13.213302373886108", "label": "Sad", "endTime": 16.711094617843628},  
{"startTime": "16.711094617843628", "label": "Angry", "endTime": 16.843215703964233},  
{"startTime": "16.843215703964233", "label": "Sad", "endTime": 18.577550172805786},  
{"startTime": "18.577550172805786", "label": "Happy", "endTime": 19.51138186454773},  
{"startTime": "19.51138186454773", "label": "Sad", "endTime": 20.37821340560913},  
{"startTime": "20.37821340560913", "label": "Neutral", "endTime": 20.445361375808716},  
{"startTime": "20.445361375808716", "label": "Sad", "endTime": 20.52943754196167},  
{"startTime": "20.52943754196167", "label": "Neutral", "endTime": 20.976831197738647},  
{"startTime": "20.976831197738647", "label": "Sad", "endTime": 21.150986194610596}
```

Figure 100 - FER Data (Local App)

A class within the *FacialExpressionRecog.py* file named *DisplayCameraImage* is responsible for displaying the camera feed in a small window that spawns on the bottom left corner of the screen. This window is captured in the screen recording and the researcher who is watching the video of the usability test can see the facial expressions for themselves. The window is shown in Figure 101.

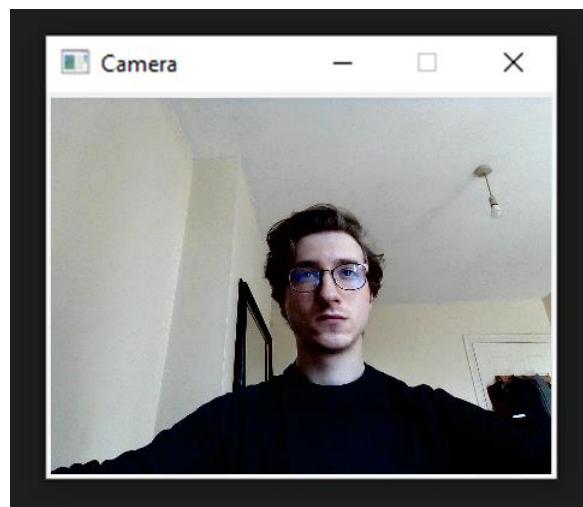


Figure 101 - Camera Feed Window

Question Window

One type of question that can be asked is multiple-choice questions which is shown in Figure 102 and Figure 103. Both the text answer question (*TextQuestionWindow.py*) and the multiple-choice question (*MultipleChoiceQuestionWindow.py*) inherit from a class named *QuestionWindow.py*. The features of a question window is that the user can hold click anywhere on the window and then drag to move the window wherever they wish. In the multiple-choice question, once the participant selects an answer the submit button will appear. The progress is always displayed on tasks and questions to show the user how many more “tasks” they need to complete (a question is technically a task a user needs to complete).

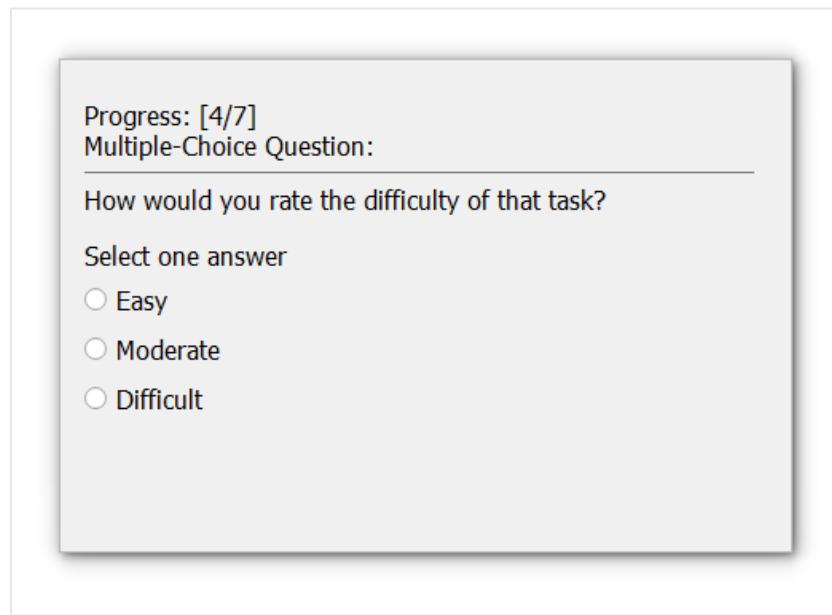


Figure 102 - Multiple-Choice Question Window

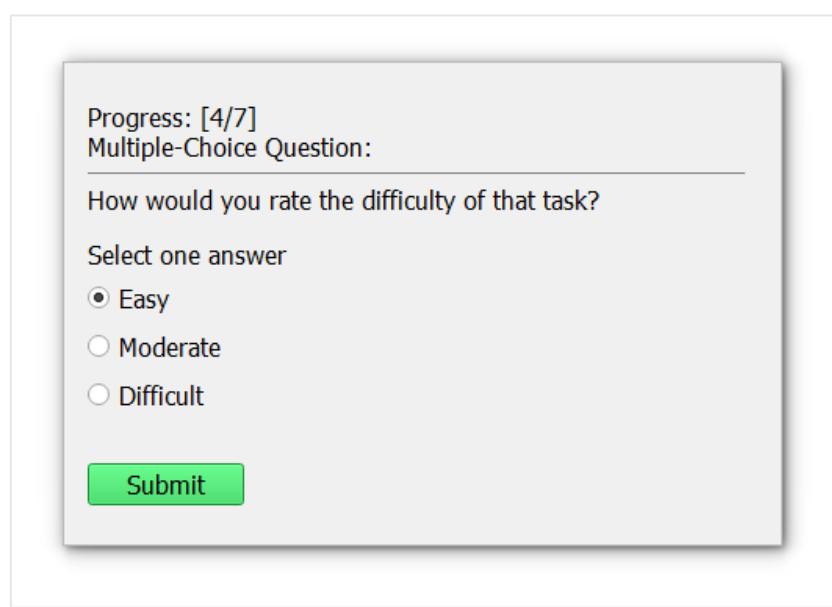


Figure 103 - Multiple Choice Question Window (After Selection)

The text answer question is displayed in Figure 104. The user enters any text into the window as their answer and this question type is intended for open-ended questions.

A screenshot of a computer window titled "Text Question Window". The window displays the following information:

- Progress: [7/7]
- Question: Do you have any feedback for us?
- Your answer: The features of the program were easy to use. I don't have any suggestions for improvements. I thought the product purchase was especially well designed.
- Submit button

Figure 104 - Text Question Window

Task Window

The task window gives the user the instructions they need to carry out. The window by default appears in the top left corner of the screen. As the user completes the steps in the task they will scroll down. At the bottom, beneath all of the steps in the task is a “Finish Task” button that the user can click when they are finished with the task or have decided to give up if they cannot complete it.

Once the button is clicked another window will appear over the existing window to confirm that the user wants to proceed to the next task in case they have misclicked. If the participant finds that the window takes up too much space they may wish to “Hide” it and the content will be hidden. What is left behind is simply a button. As with the question windows, the user may hold click anywhere on the task window a drag to window to the position they like. The class for this window is *TaskWindow.py* and it utilizes the *ScrollLabel.py* class for the scrolling functionality.

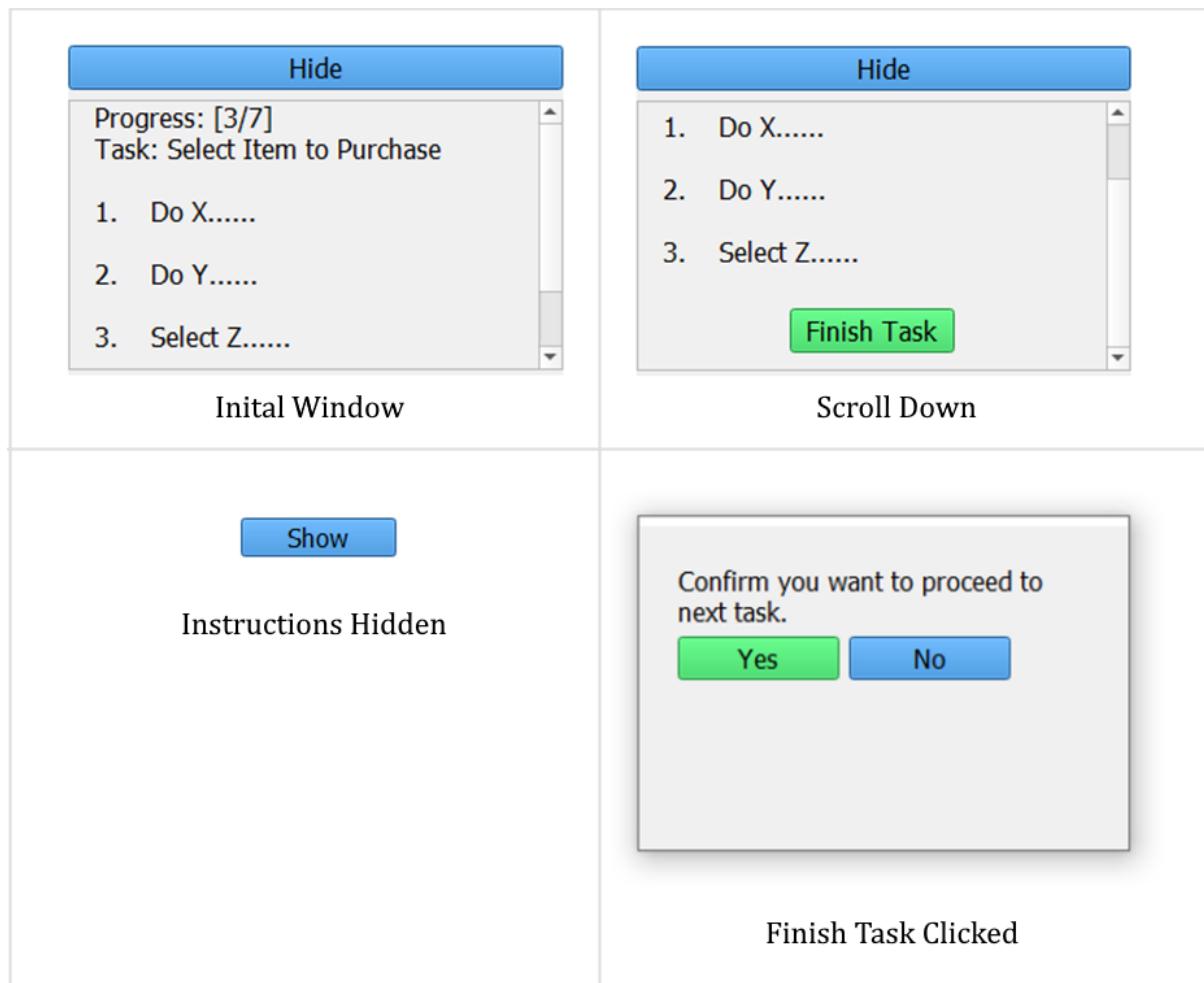


Figure 105 - Task Window States

Video Uploading

Once the last task/question is completed by the user the usability test finishes and the user is presented with the data upload screen as shown in Figure 106. The data is first uploaded to the backend server and an instance reference code is obtained. The video is then uploaded to Vimeo and once the upload is complete the local application, sends the instance reference code and the video link to the backend to update it. The idea is that only the client that uploaded the data to the backend knows their instance reference code which can be used to update the video link. To make this process even more secure a password can be implemented.

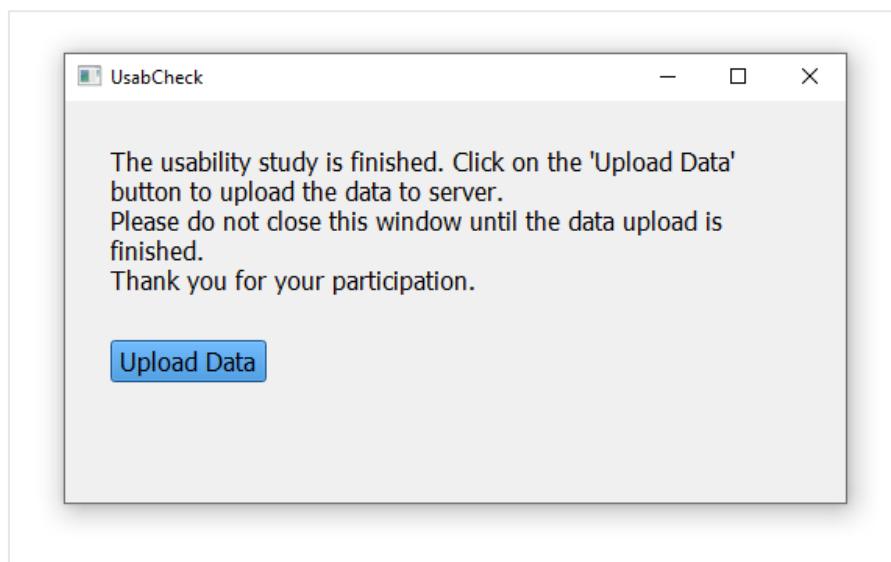


Figure 106 - Data and Video Uploading (Initial Screen)

As the data is uploading a message is displayed for the user and the “Data Upload” button is greyed out. This can be seen in Figure 107. Once the video has finished uploading the message changes and notifies the user that the window can now be closed as shown in Figure 108.

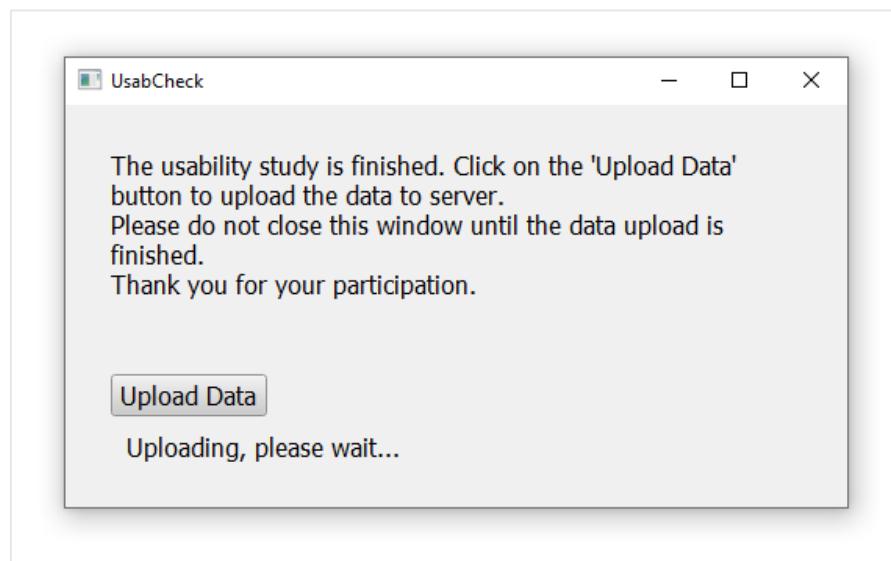


Figure 107 - Video Uploading (Uploading Stage)

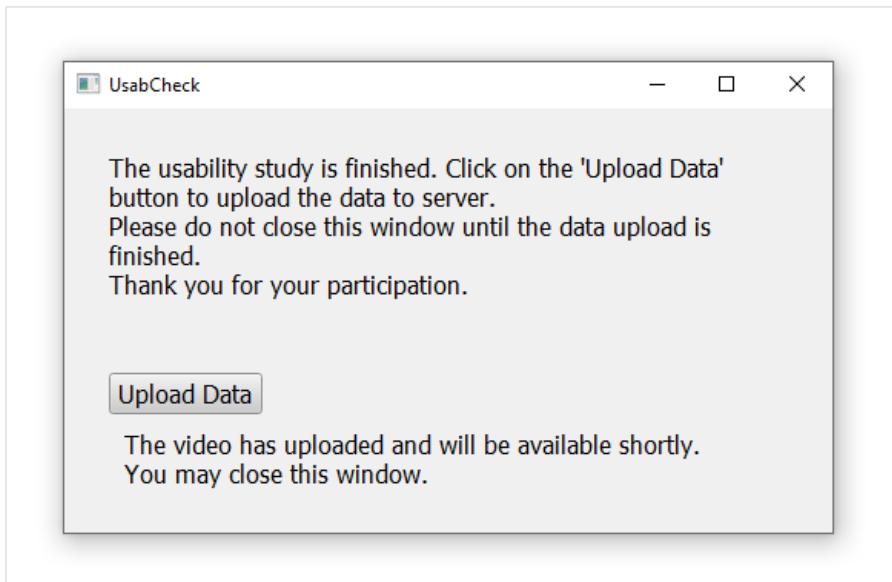


Figure 108 - Video Uploading (Finished Stage)

Local App Deployment

The UsabCheck local application was compiled with the PyInstaller library. There were some challenges with the deployment as initially the entire application was compiled into a single .exe file and that resulted in several problems. The first problem was accessing the assets that the project needs such as machine learning models, images, videos, gifs, etc. Another problem was the boot up time. The application took ~40 seconds to start up as it was unpacking all of the files. The solution was to compile the application as a directory. The .exe file was still created; however, it would be within a directory with all of the necessary files. That reduced the bootup time to only 4 seconds which was acceptable. Compiling the application is important for several reasons, one of the most important ones is that the libraries and other environmental components are included. Otherwise the user would need to install the libraries and/or set up an environment of sorts.

4.4. Machine Learning

4.4.1. CSV to Image Converter (CRISP Data Preparation Stage)

The FER2013 dataset in its original format is a CSV file shown in Figure 110. The *pixels* column consists of 2304 (48 x 48) pixel values which are converted into an image. Figure 109 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. For input consistency reasons a CSV to image converter was implemented to convert the pixel values to a grayscale image.

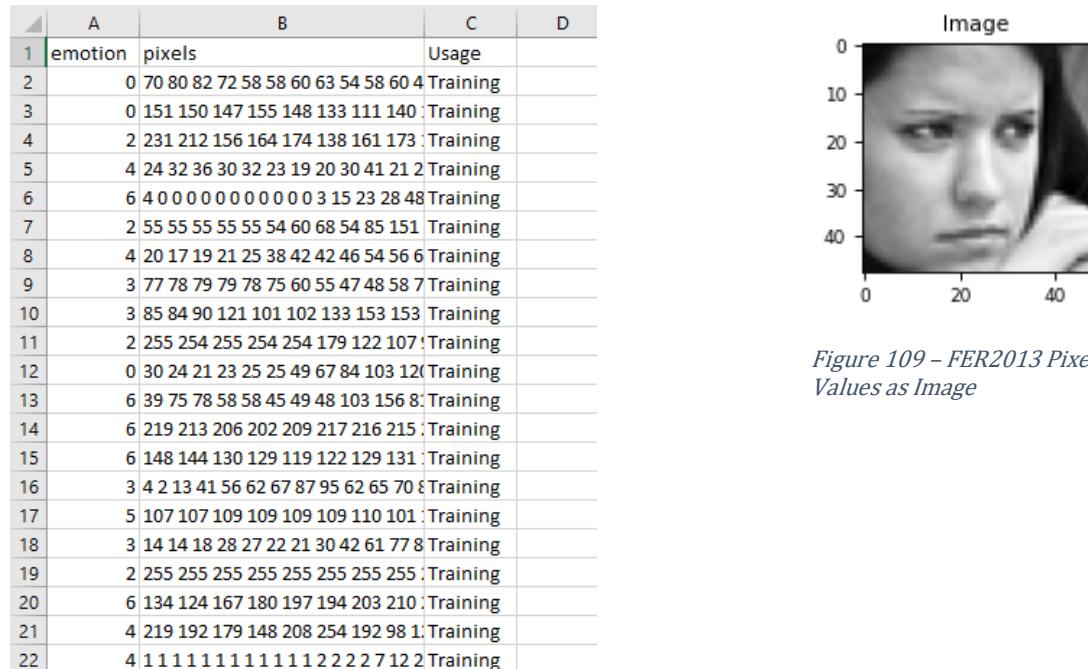


Figure 110 - FER2013 Dataset CSV File

Below is the code for opening the CSV file, obtaining the relevant information, and saving the image as a .jpg file. First, the libraries such as Pandas, NumPy, OpenCV and Matplotlib are imported as these will be used in this script and the import code is shown in Figure 111.

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

Figure 111 - Imports

A dictionary variable 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 path variables are initialized and the CSV file is opened/read with Pandas. The code is shown below in Figure 112.

```
# 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 112 – 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. This can be seen in the code in Figure 113.

```
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 113 - CSV to Image Converting and Exporting

The final end results is images stored in their respective folders as shown in Figure 114 below. For example, All the images labelled “Happy” are in the “Happy Folder”.



Figure 114 – Converted Images

4.4.2. Model Training (CRISP Modelling Stage)

Before implementing the chosen model (VGG-16 architecture) and training the model, more research was conducted into the implementation of CNNs with Keras and TensorFlow, and the implementation of the VGG-16 architecture. In the research chapter the VGG-16 FER model implementation by Gaurav (2018) [9] was reviewed. However, when attempting to train with their model issues were encountered and other implementations were researched. Two GitHub repositories were found and both architectures were trained and evaluated. The input data used was the processed FER2013 dataset that was discussed in previous section 4.4.1.

Model 1

The first model is a modified VGG-16 architecture (Figure 7 – VGG-16 Architecture) implemented by Ashish Kushwaha [77]. The dataset used to train this model was the FER2013 dataset (processed in previous section) 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% [78]. The model was trained for 25 epochs.

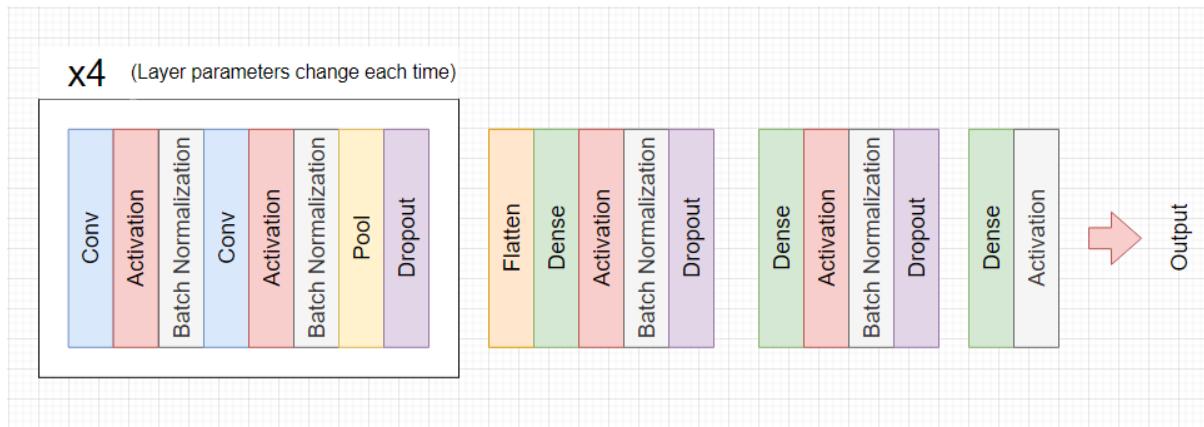


Figure 115 - Model 1 Diagram

The model has been trained to recognise 7 different emotions. Below Figure 116 illustrates the accuracy increase in the last few epochs with the final accuracy on the validation set being 57%. The accuracy increase and loss decrease is illustrated in Figure 117. 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 could be the training and validation ratio imbalance.

```

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 116 – First Model Training

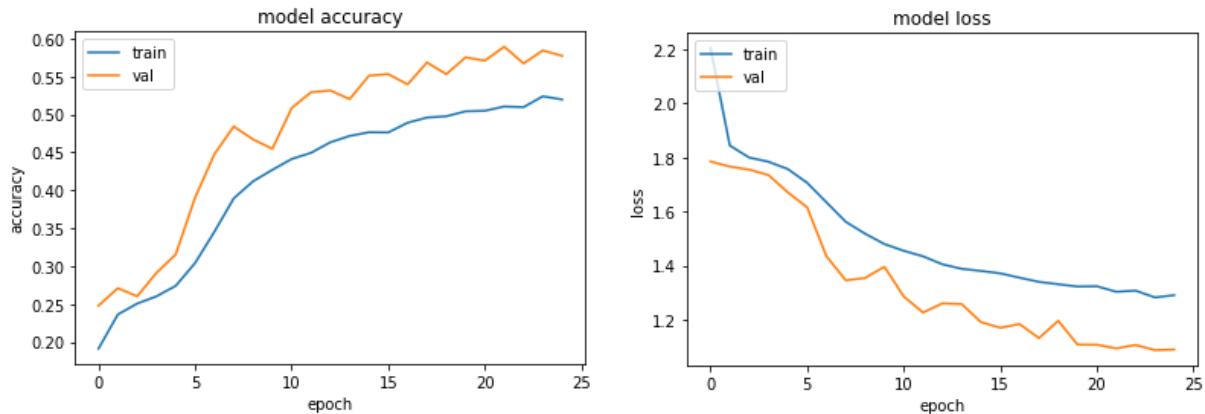


Figure 117 – First Model Accuracy and Loss

However, initially the confusion matrix for this model would not display properly. Being unsure if the issue was with the model or input to the confusion matrix another model was trained to get more information that could help with the problem.

Model 2

In the second model was implemented by Neha Yadav [79] the CNN architecture was different to the first model and is illustrated in Figure 118. 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 almost the same as the other model. The accuracy can be seen in Figure 119. 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 118. The accuracy increase as the model trains is displayed in Figure 120.

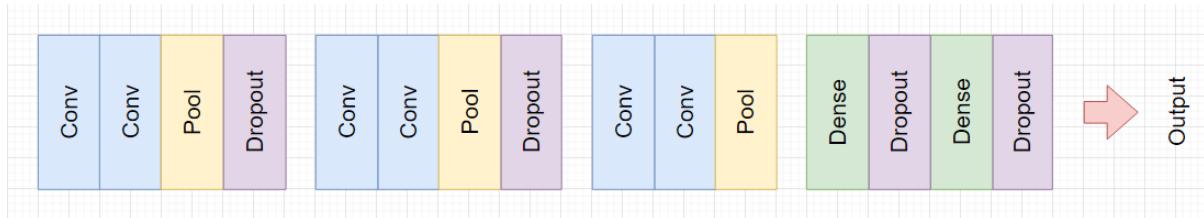


Figure 118 Second Model Architecture

```

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 119 – Second Model Training

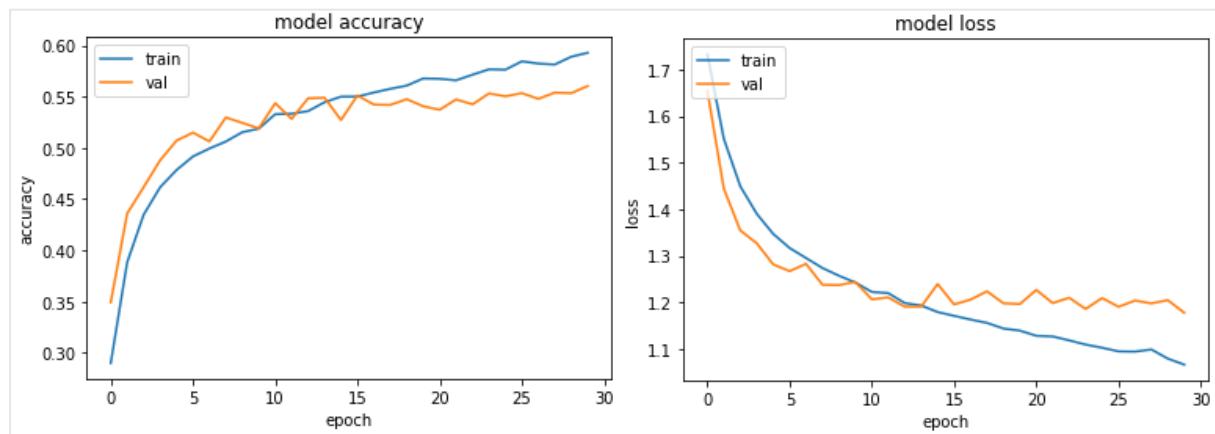


Figure 120 – Training Accuracy and Loss Charts for the Second Model

To explain the code, first the paths to the train, validation and testing data is initialized below in Figure 121. 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 121 – Path Declaration

A *loadDataset* method has been implemented to read the images from the directory. 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. The code for this can be seen in Figure 122.

```
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
        imageList = os.listdir(datasetDirectory + "/" + folder)
        for imgName in imageList:
            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 122 – Loading Dataset from File

An arbitrary number of features and batch size is chosen in Figure 123. 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. The code in Figure 123 has been written by Neha Yadav.

```

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 123 – Configurable Variables and Data Normalization

The code for the architecture illustrated in Figure 118 is displayed below in Figure 124. The code for the model has been taken directly from written by Neha Yadav's GitHub repository.

```

# 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 124 – CNN Architecture Code

4.5. Conclusions

In conclusion, the development chapter consisted the development of the web application, local application, and the FER model. The aims and objectives of the project were achieved. The web application allows researchers to create usability studies and view the results of the studies in the forms of widgets, charts, and video recordings. The emotion and task timestamps are successfully displayed under the video and allow for the researcher to analyse the usability study in terms of the user's actions, performance, and emotions. The local application has successfully integrated the FER model, records the screen, displays tasks, and questions for the participant, and uploads the data and video screen recording. The machine learning development section involved data preparation, selecting two VGG-16 variation deep learning models, and training the models. At first glance it would appear that both models have the same accuracy.

5. Testing and Evaluation

5.1. Overview

This chapter is split into testing and evaluation. The testing chapter mainly involves unit testing of the web application and local application. The integration of components is also tested in the testing section. The FER model testing is under the evaluation section as that grouping of information made more sense. The sub-sections of the testing and evaluation sections can be seen in Figure 125.

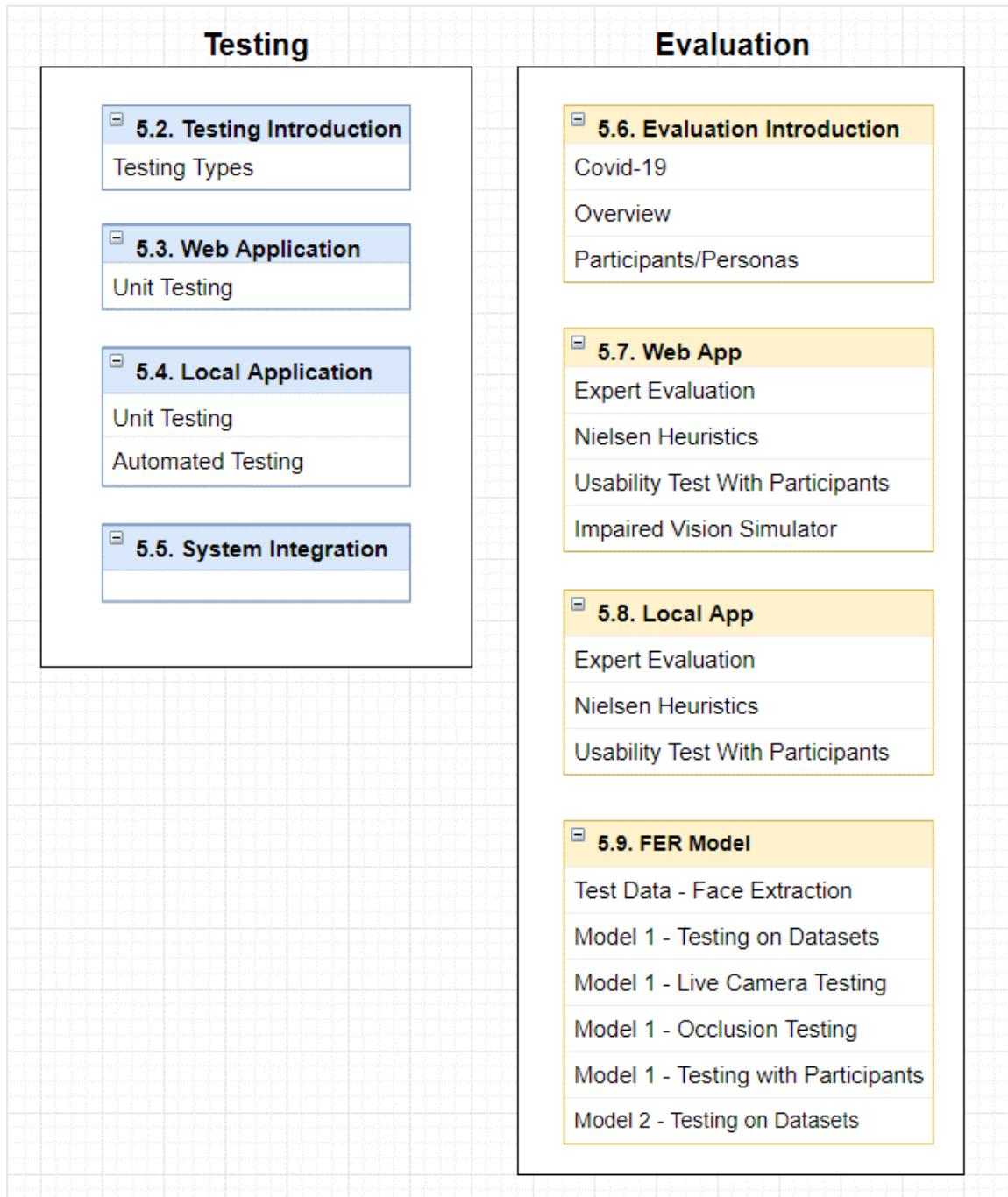


Figure 125 - Testing and Evaluation Overview

5.2. Testing Introduction/Plan

This project was continuously tested throughout the development cycle. The CRISP and Agile methodologies were followed and both have testing in their cycles. With Agile, testing occurred at each sprint and the implemented functionality for that sprint was tested. Each unit was 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/evaluation 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, functionality errors etc. [80]. This type of testing was applied during the usability studies as participants with no knowledge of the internal workings used the application.

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 [81]. This type of testing was applied by the author throughout the development of this project.

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 [82].

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

5.3. Web Application Testing

5.3.1. 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.

Table 3 - Web Application Unit Tests

Test Description	Expected Outcome	Pass/Fail
Researcher register	The account will be created and the user will be notified	PASS
Researcher login with correct details	User will be redirected to main screen (Dashboard)	PASS
Researcher logout	User will be redirected to login screen	PASS
“Create a Project” button is pressed	A pop-up form will appear on screen with input fields	PASS
“Create Usability Test” button is pressed	The “Create Test” page will open	PASS
Add Task in create test	A box will appear on screen with an “Enter Instruction” field that needs to be filled out	PASS
Add step to instruction	Another text field will appear in the instruction box	PASS
Add text question in create test	A box will appear on screen with an “Enter Question” field	PASS
Add multiple choice question in create test	A box will appear on screen with a “Enter Question” and “Enter Answer Choice” field	PASS
Add/Remove instruction step	The additional instruction step that have been previously added will be removed	PASS
Create Usability Test	The researcher clicks on “Create Test” button and the new test is stored in the database. User is notified of successful creation.	PASS
Click “View Test Details” button in dashboard	The user will be redirected to the “View Test Details” screen	PASS
Bar chart displaying data	The proportions of the chart are correct and correspond to the values	PASS
Pie Chart displaying data	The proportions of the chart are correct and correspond to the values	PASS
Play video	Video will play	PASS
Grading tasks	Clicking “Pass” or “Fail” will change the colour of the button to green or red respectively	PASS

Display Emotion video bars/timelines	The emotions bar display the data correctly. Emotion timestamps are in line with video progress bar and appear in the right places.	PASS
SQL injection working	Data is correctly stored in the database using the SQL query. Data retrieved from database with SQL query.	PASS
Web server REST Entry points working	Data can be passed to or retrieved from the web server via POST and GET requests.	PASS

5.4. Local Python Application Testing

5.4.1. Unit Testing

Table 4 - Local Application Unit Tests

Test Description	Expected Outcome	Pass/Fail
Screen is recording	A red border will contour the border of the screen and the screen will be captured	PASS
Stop Recording button pressed	User clicks on “Stop” button and a confirmation window will appear.	PASS
Recording Stop confirmed	User clicks confirm the recording stops. Clicking cancel simply hides the window.	PASS
User clicks “Finish Task” button	The user clicks on the “Finish Task” button and a confirmation window appears.	PASS
User confirms “Finish Task”	Clicking “Yes” moves on to the next stage of the usability test.	PASS
Video saved locally	The video will be saved to a local directory under the correct name	PASS
Video upload to Vimeo	A link is returned by the Vimeo API and can be played	PASS
Task instruction is displaying to participant	Task instructions for the participant display on screen	PASS

5.4.2. Automated Testing

[\[Source\]](#) [\[Source\]](#)

Automated tests have been created to test the local python application. The library used to create the UI is called PyQt5 and it includes a testing feature which allows for automated clicking on the GUI components such as buttons. The QTest clicking simulation is shown in Figure 126. To run the tests the Python unittest library was used and the file in which the tests reside in is named *ProgramTest.py*. The test in Figure 126 enters the usability study's reference code, the data is pulled and processed and then verified. The test in Figure 127 tests the multiple-choice question window. Sample data is passed, processed and an option is selected.

```
def test_submitRef(self):
    # Input reference code
    QTest.keyClicks(self.program mainWindow.refCodeInput, "OBVIVVGV")

    # Press submit
    submitBtn = self.program mainWindow.submitBtn
    QTest.mouseClick(submitBtn, Qt.LeftButton)

    # Verify that the retrieved and processed data is correct
    self.assertEqual(self.program mainWindow.testName, "UsabCheck Web App")
    self.assertEqual(self.program mainWindow.createdDate, "2021-03-26")
    self.assertEqual(self.program mainWindow.noOfTasks, 10)
    self.assertEqual(self.program mainWindow.noOfQuestions, 14)
```

Figure 126 - Local App Automated Unit Testing (Part 1)

```
def test_multipleChoiceQuestion(self):
    # Initialize sample data
    sequenceDataItemStr = r"""{"questionId":103,"testId":123,"questionConfigsJSON":"{\\\"type\\\":[\"q
    sequenceDataItem = json.loads(sequenceDataItemStr)
    totalNumberOfItems = 5

    # Create and show the window
    window = MultipleChoiceQuestionWindow(None, totalNumberOfItems, sequenceDataItem)
    window.show()

    # Check that the data has been processed and displayed correctly
    self.assertEqual(window.radioButtonLayout.count(), 3)
    self.assertEqual(window.radioButtonLayout.itemAt(0).widget().text(), "Easy")
    self.assertEqual(window.radioButtonLayout.itemAt(1).widget().text(), "Moderate")
    self.assertEqual(window.radioButtonLayout.itemAt(2).widget().text(), "Difficult")
    self.assertEqual(window.progress, "Progress: [4/5]\\n")

    # Select an option and check to make sure that the option is selected
    window.radioButtonLayout.itemAt(1).widget().click()
    self.assertEqual(window.choiceSelected, "Moderate")
```

Figure 127 - Local App Automated Unit Testing (Part 2)

5.5. System Integration

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.

Table 5 - System Integration

Modules	Expected Result of Interaction	Pass/Fail
Local Application, Web Application Entry point	The local application will connect to the entry point of the web application and retrieve test data.	PASS
Local Application, FER Model	The local application will obtain the data from the camera and feed it into the model to make a prediction.	PASS
Front-end, Middle-tier (Back-end)	The front-end of the web application will connect to the back-end server and retrieve/send data.	PASS
Middle-tier (Back-end), Database	With the use of the DAO classes the back-end server will connect to the database.	PASS
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.	PASS

5.6. Evaluation Introduction/Plan

Covid-19 Restrictions

The initial idea was to have a batch of at least 5 individuals test the system. However, due to Covid-19 travel restrictions and some individuals not having a camera which is needed to test the model that number has been reduced to 4 individuals to test the machine learning model. The number of participants to test the local app has been reduced to 3 and the number of participants to test the web app has been reduced to 2.

Evaluation With Participants

The evaluation procedure with participants is shown below in Figure 128. First the FER model was evaluated on its own. Next the local app which has the FER model integrated into it was evaluated by testing a poorly designed online shopping website. Finally, the web application was evaluated. With this procedure the participants were introduced to the various functionality of UsabCheck a step at time. When evaluating the local app they were already familiar with the machine learning aspect of it and when evaluating the web application they were already familiar with how a usability test is conducted as they have participated in one.

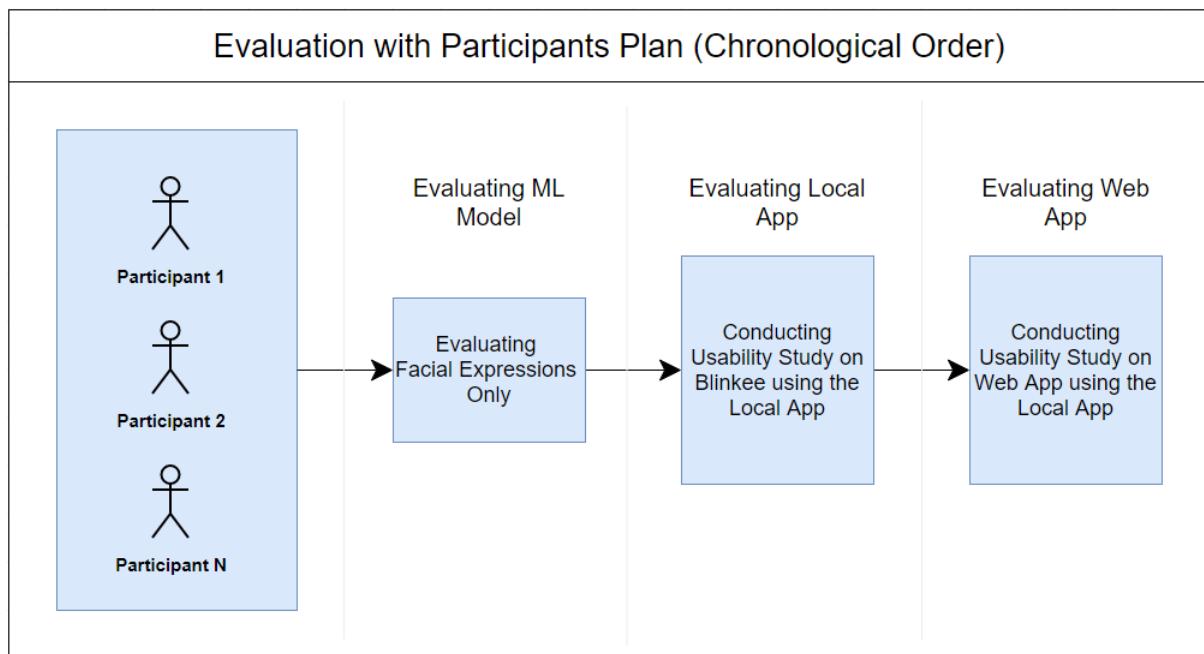


Figure 128 - Evaluation Procedure in Chronological Order

Two personas (excluding expert evaluator) were involved in testing and evaluating the UsabCheck application. Firstly, there are student developer type individuals who are tech savvy and understand software engineering and software development. This persona is shown in Figure 129. The other persona is a student who is familiar with using software for their personal day to day needs, however, their background is not that of developer and have close to no understanding of software development.

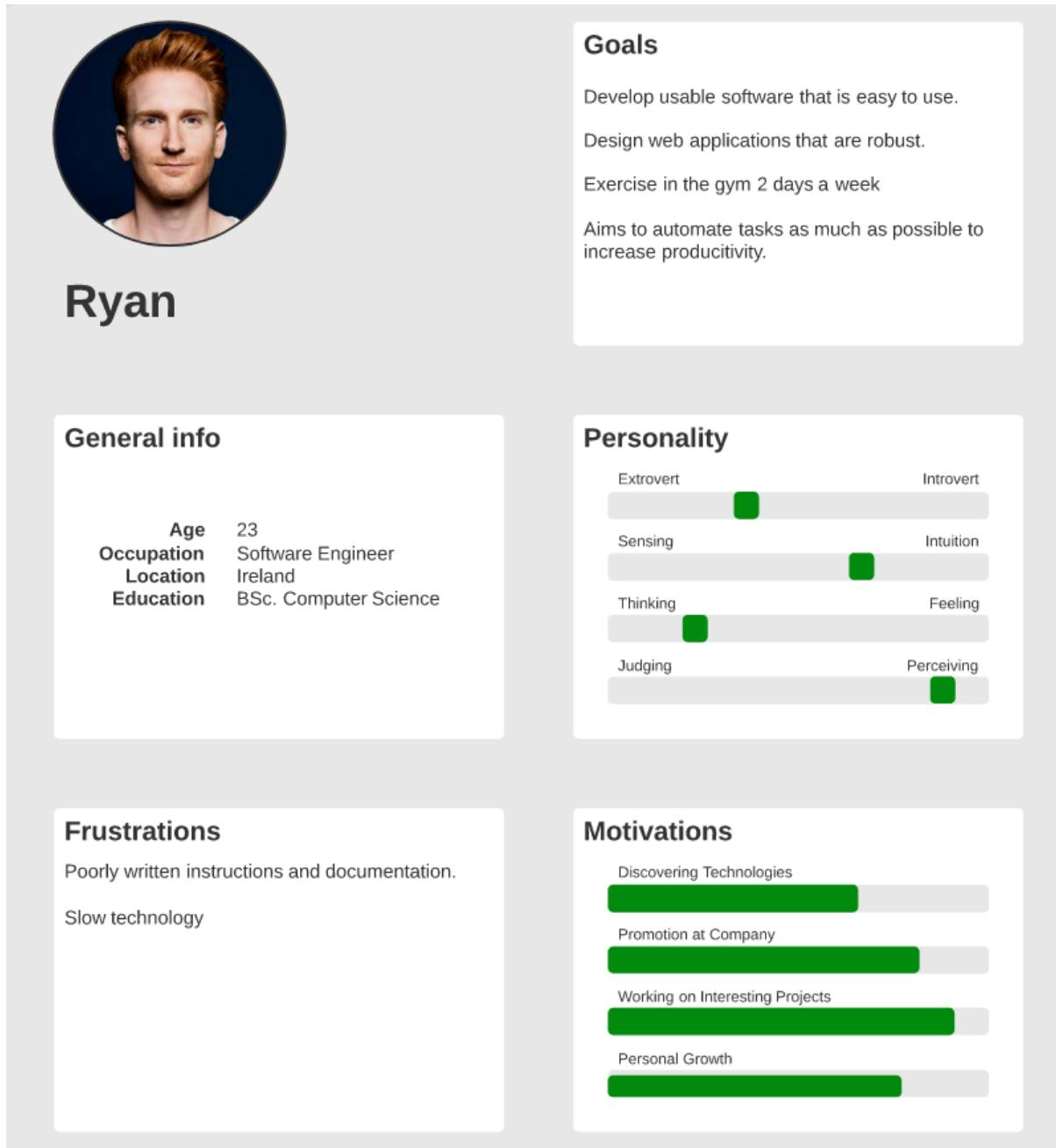


Figure 129 - Persona 1 (Developer)

FER Model Evaluation Overview

Three aspects of the system were evaluated. Firstly, the FER model was evaluated on metrics such as accuracy, precision, recall, F1 score. These were displayed using a confusion matrix, tables, and line charts. The FER model has been trained on the training set, validated on the validation set during training and tested on the FER2013 test set as well as other FER datasets to ensure that the evaluation is accurate as the model has not seen that data before.

These metrics will be compared to the state-of-the-art metrics for that particular dataset and model. Finally, the model will be given data from the camera in real-time to truly see how it performs in a real-world environment. The most successful model will also be evaluated with the data collected from these participants to determine the optimal angle of the camera and the detection accuracy of the model in a usability testing environment.

Local Application Evaluation Overview

Both personas will be used in the evaluation of the local app. A usability test for a website was 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 [83].

Web Application Evaluation Overview

The web application was evaluated by an expert throughout the development of the system and the web application was also evaluated with Nielsen's Heuristics . The web application will be tested and evaluated only with the developer type individuals. This is important as these individuals are part of the demographic that will use such an application and their feedback is deemed more relevant. Additionally, setting up a usability test and viewing the data is unlike other applications and websites designed for the general population such as shopping websites.

5.7. Web Application Evaluation

5.7.1. Expert Evaluation

The web application was constantly evaluated by a usability testing expert throughout the development of the project. The feedback was recorded and the web application was improved according to the feedback. The feedback given and the solution implemented in response to the feedback is shown in Table 6.

Table 6 - Expert Evaluation Web App

Feedback	Implemented Solution
The page you are on should be clearer. The dashboard title was “Projects” which made it confusing.	The title of the page was renamed to “Dashboard” and it was made sure that all pages display the page name at the top.
The placement of the “Create Usability Test” button should change after the usability test has been created.	The position of button was changed after the usability test is created. Initially it shown
Button text should be bigger	The styles across all of the applications was changed and text was set to be at the minimum of 16px. This number was recommended by [Source]
Give context for why a pre-test question is needed.	An icon with a question mark was added next to many of the user inputs and labels to give the user context and help them understand what input to give. Examples of the input were provided as well.
Consider the spacing between the components and grouping of components on the page.	The components on the “View Test Results” page in Overview tab were grouped in containers with a grey background. Gestalt Principles were researched and implemented.
There should be some information about what the colours of emotions and tasks in the video bars on the “Recordings” page represent.	A legend was added the below the video bars which shows what colours map to what emotion. Labels were also added that explain what each bar is. For expert users there is an option to hide the labels.
There should be some indication that the first two video bars show the entire timeline and the last video bar is the zoomed in bar.	The first two bars were placed close to one another and the zoomed in bar was spaced further away from the first two to indicate that it is different.
More information should be given on the “Overall Task Success Rate” bar chart.	The number of total task completed by all participants was displayed above the chart to give additional context.

5.7.2. Nielsen Heuristics [[Source](#)]

Nielsen Heurists Followed/Implemented

The web application was evaluated with Nielsen's Heuristics. The features of the app that have been implemented and follow the heuristics are shown in Table 7.

Table 7 - Web App Nielsen Heuristics (Implemented)

Heuristic	Implemented
Visibility of system status	The name of each page is displayed on top of each page. Whenever user creates or deletes something a popup appears that notifies them whether or not their action was successful. The user is told what action they should take such as "Create a usability test" if they haven't already.
Match between system and the real world	The icon to see more information is a question mark which makes it intuitive and obvious. Whenever possible simple language is used. The colours and emotion association has been researcher to make it as intuitive as possible. Colours attempt to match the action or result. For example, red for "Fail" or "Danger" and green for "Success".
User control and freedom	Whenever a user wants to perform an irreversible action such as deleting a project or a test they go through a two-step process. Should they be ever stuck or wish to return to the main screen they "Dashboard" button is always present on the navigation bar.
Consistency and standards	The language used attempts to be as unambiguous as possible. The style of the application aims to be as predictable as possible.
Error prevention	Whenever a user wants to perform an irreversible action such as deleting a project or a test they go through a two-step process and are informed in red text that they are about to perform a "dangerous" action. In other words, they need to press the delete button twice.
Recognition rather than recall	Whenever displaying data on a task or question the task or question is displayed above the results. This helps the users not having to remember what a task or question was about. A user can also click on a (?) icon to learn more about a specific label or input field.
Flexibility and efficiency of use	A "Hide Labels Button" has been implemented to hide the labels for expert users who know how to use the application. The labels are displayed by default for novice users.
Aesthetic and minimalist design	The web application has been designed with the intention to be minimalist and visually appealing.
Help users recognize, diagnose, and recover from errors	When creating a usability test, if the user has not input information that is required they will be informed that they should enter text into that field.
Help and documentation	The (?) information icons provide help right next to the content which saves the user from having to search for the information.

Nielsen Heuristics in Need of Implementation

The features that are lacking and need to be implemented according to Nielsen's Heuristics are shown in Table 8.

Table 8 - Web App Nielsen Heuristics (Need Implementation)

Heuristic	Need to Implement
Help users recognize, diagnose, and recover from errors	The errors that come from the backend are not always intuitive. Especially unexpected errors such as error 500. More error checking could be implemented and error messages should be more constructive.
Help and documentation	Although there is information provided on how to use the application and some information on how to go about creating a usability test the assumption is that the individual using the web application is familiar with how to conduct a usability test. The objective of the application is more about allowing them to conduct the usability study digitally. However, there could be more help and documentation to help a novice use who has no experience with usability testing.

5.7.3. Usability Testing with Participants

As mentioned in the beginning of the chapter, after the users have tested the FER model and the local application they will participate in one more study to test the web application. The usability test which tests the various features of the web application is shown in Table 9. The participant is given tasks to do and are asked questions after each task.

Table 9 - Usability Study to Test the Web App

Scenario	You are a researcher who has discovered UsabCheck and you will be looking for a website that you have built. You will go through the process of creating a usability study and viewing the data of that usability study etc.
Pre-Test Question	Have you conducted a usability study before?
Pre-Test Question	What is your current profession?
Task	Create an account and login
Multiple-Choice	Please rate the difficulty of the task (Easy, Moderate, Difficult)
Task	Follow the instructions on the website, the goal is to navigate to the "Create Usability Test" page
Multiple-Choice	Please rate the difficulty of the task (Easy, Moderate, Difficult)
Task	Imagine that you are going to test an online shopping website that you have built. Try to create some tasks and question to test the imaginary shopping website. The website is imaginary so ask any questions or give any instruction for the user to complete. When you are finished configuring the tasks and questions, create the usability test
Question	What features did you like when creating the usability test?
Question	What features do you think need improvement when creating the usability test?
Question	Are there any features you feel are missing when creating the usability test?
Task	View the details of the test you have created
Multiple-Choice	How satisfied are you with the displaying of the test details?
Task	Delete the test you have created
Task	Delete the project you have created
Task	Logout
Task	Login as: Username: OfficialUser1 Password: password123

Task	Select Project1 in Dashboard Select view the test results of "Blinkee" View the results in both the overview tab and recordings
Task	Enter fullscreen of the video Exit fullscreen of the video
Question	Do you have any opinion regarding the sequence of the information/data displayed in the overview tab?
Question	Do you have any suggestions for how to improve the overview tab?
Question	Do you feel there was anything missing?
Question	Do you have any suggestions for how improve the timeline bars below the video?
Question	What features did you like?
Question	What features did you dislike?

Web App Usability Test Results Summary

The information gathered from the questions asked of the participants has been summarised. The features that stood out to the participant are displayed in Table 10.

Table 10 - Features of Web App Liked by Participants

Features Participants Particularly Liked
The info bubbles explaining each component
The simplistic design when creating a usability test
The emotion timeline bars are a great are showing what emotions occurred

The main suggestions that the participants offered are displayed in Table 11.

Table 11 - Improvements/Suggestions for Web App

Impact on Usability	Feedback
Moderate	In the “entire” emotion timeline bar the slider which represents the portion that is zoomed in could be draggable with the mouse. Currently the only way to move the slider is with the zoomed in emotion timeline scroll bar. (User tried to click on the slider to drag it)
Moderate	When clicking on the emotions/tasks in the video bars the time in the video jumps. To avoid having to scroll up to the video it would be nice if the page automatically scrolled to the top to where the video is when the emotion is clicked.
Low	Minimize the questions and tasks in the create test page to avoid scrolling
Low	The delete button for a task or question in the dropdown of the usability create page could be coloured red
Low	In the overview tab when viewing results add a general emotion felt by all the participants.

5.7.4. NoCoffee (Impaired Vision Simulator)

The NoCoffee Vision Simulator Google Chrome extension [Source] was used to view the website through vision similar that to a visually impaired person. The limitation as stated by the creators are that the “The simulations are not medically/scientifically accurate.”. However, the extension still provided some insight. The screen was blurred to a certain extent and “floaters” (the shadows) were added.

The dashboard simulation of what would be seen by a visually impaired person is shown in Figure 130.

The screenshot shows the UsabCheck Dashboard interface. At the top, there is a navigation bar with icons for user profile, 'UsabCheck', 'Dashboard', 'OfficialUser1', and 'LogOut'. Below the navigation bar, the word 'Dashboard' is displayed in a large, bold font. Underneath, there are two project cards:

- Blinkee**:
 - View Test Details
 - View Test Results
 - Settings

Reference Code: OHSP4VAF
Launched: 2021-03-25
No. of Tasks: 3
No. of Questions: 12
No. of Participants: 1
Status: Open
- UsabCheck Web App**:
 - View Test Details
 - View Test Results
 - Settings

Reference Code: OBNVVGV
Launched: 2021-03-26
No. of Tasks: 10
No. of Questions: 14
No. of Participants: 0
Status: Open

Figure 130 - Dashboard (Impaired Vision)

The video timelines/bars simulation of what would be seen by a visually impaired person is shown in Figure 131.

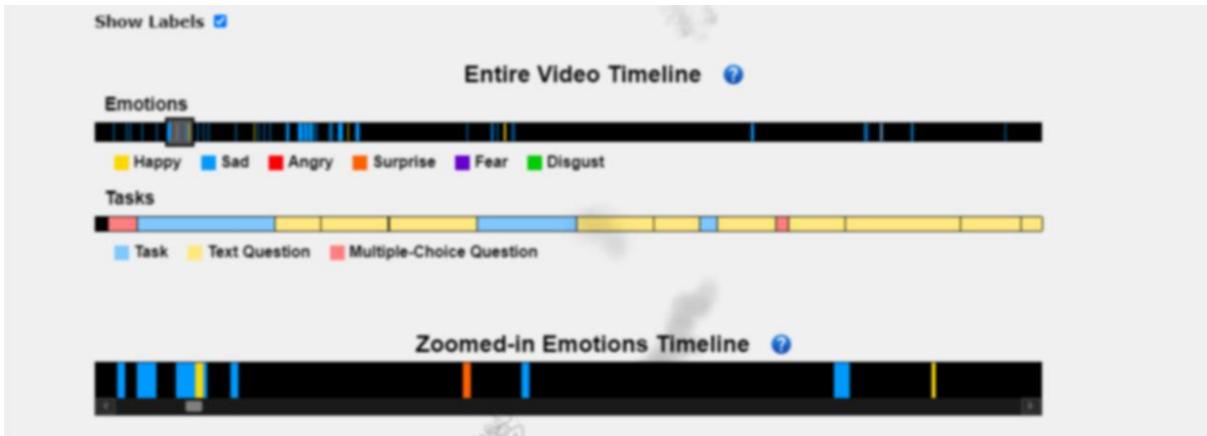


Figure 131 - Video Timelines (Impaired Vision)

In conclusion, although the screen was blurry the text was very readable and the floating shadows did not pose much problems, if any. The zoomed-in emotion timeline has been very useful as the emotion in the entire video bar were difficult to see. The only possible problem was the viewing of the video of the participant's screen recording. In many cases the content in the video was barely readable.

5.8. Local App Evaluation

5.8.1. Expert Evaluation

The local application was evaluated by an expert in usability testing throughout its development. The feedback given by the expert and the improvement in response to the feedback is displayed in Table 12.

Table 12 - Local App Expert Evaluation

Feedback	Implemented Solution
Have a progress bar or some indication to show how far the user is into the usability study.	The stage which the user is on is shown. For example, [2/15] tells the user that they have completed two out of a total of 15 tasks.
Introduce the local app and show how it can be used and the features involved before the test begins.	A tutorial button was added on the initial screen. It shows information and a GIF of the feature that is being demonstrated.
Confirm with the user that they want to proceed to the next task in case they misclick.	When the user clicks on the “STOP” button on the recording slider or they click “Finish” on the task, a prompt shows up to ask them if they are sure they want to proceed.

5.8.2. Nielsen Heuristics

The local application has been evaluated with Nielsen’s Heuristics. The features of the app that have been implemented and follow the heuristics are shown in Table 13.

Table 13 - Local App Nielsen Heuristics (Implemented)

Heuristic	Implemented
Visibility of system status	A loading screen has been implemented when the camera and machine learning model is loading. The status is shown to inform the user of the loading status. The title of the window is shown, for example, a task is labelled “Task” and a multiple-choice question labelled “Multiple-Choice Question”. The stage which the user is on is also displayed.
Match between system and the real world	The text is designed to be as simple as possible. A stop button is in red and a submit button is either blue or green which the user will expect.
User control and freedom	A cancel button is provided to allow to ensure that the action the user is performing is one they are sure they want to perform. Should they not want to proceed with the test the “STOP” button is always visible. The user can also shrink the window and/or move it wherever they want.
Consistency and standards	The user can expect that windows without the title bar can be moved by clicking anywhere on the window and dragging.
Error prevention	A cancel button is provided to allow to ensure that the action the user is performing is one they are sure they want to perform.

Recognition rather than recall	The design of the application is simple and the user simply has to follow the instructions on screen. For example, when answering a question, the window will be titled "Question", that combined with the question above the input box makes for an obvious course of action.
Flexibility and efficiency of use	The user can move any of the windows where they see fit to ensure that it does not interfere with their actions.
Aesthetic and minimalist design	The design is simple but could be more aesthetically pleasing.
Help users recognize, diagnose, and recover from errors	When it comes to error recovery there is not much to recover from as the application is designed to display data and let user's select options, enter their own input, or view tasks. In other words, there is no wrong input.
Help and documentation	There is a tutorial provided before the usability test begins to give more context to the participant.

Nielsen Heuristics in Need of Implementation

The features that are lacking and need to be implemented according to Nielsen's Heuristics are shown in Table 14.

Table 14 - Local App Nielsen Heuristics (Need Implementation)

Heuristic	Need to Implement
Visibility of system status	A loading bar for uploading the video could be added to show progress.
Match between system and the real world	The upload button goes from blue to grey when the user clicks it. The gray colour should be darker as a light gray button can be a standard for some of the buttons.
Consistency and standards	The question window always appears in the center. The task window reappears appears in the previous position. The same should happen for the question window.
Flexibility and efficiency of use	The task window should be resizable.

5.8.3. Usability Testing with Participants

Testing Local Application With Another Website

A blog by HubSpot has provided insight into creating a usability test [\[Source\]](#).

The website used to test the local application was <https://blinkee.com/>. It is an online shopping website which sells party and holiday items for events such as St. Patrick's day and Halloween. This website was chosen for the reason that it was rated as one of the websites with bad design according to [\[Source\]](#). The website was visually unappealing with an assortment of fonts and contained missing broken links. The purpose of choosing a website with bad design is to test the both the functionality of the local app but also see if there is a correlation between bad design and the emotions users have expressed.

The usability study which the participants have taken to test Blinkee.com with the help of the local app is shown in Table 15.

Table 15 - Local App Usability Study (With Blinkee)

Scenario	Binkee is a shopping website that contains various party items, your job will be to browse the website for various items. Website URL: https://blinkee.com/
Pre-Test Question	How often do you use online shopping websites? (Never, Sometimes, Often)
Task	Find necklace glowsticks for a Patrick's day party. <ol style="list-style-type: none">1. View the Patrick's day items2. Sort by price high to low3. Select option to view 100 products on the page4. Find "Glow Necklace Red and Green Pack Of 25"5. View the quantity discount of the item6. Add item to cart7. Return the home page
Question	When searching for the Patrick's day items where did you search first?
Question	Have you encountered any problems during the task?
Question	Did you feel any particular emotion throughout the duration of the task? Please state which ones.
Task	Browse Easter items <ol style="list-style-type: none">1. Navigate to the page with Easter items2. Find and add 1 mood changing egg to cart3. Return to the home page
Question	Are you satisfied with the navigation to the Easter items page? Please explain what you like or dislike.
Question	Did you feel any particular emotion throughout the duration of the task? Please state which ones.
Task	View items in cart <ol style="list-style-type: none">1. View the items you have added to your cart
Question	When searching for the cart page where did you first expect it to be?

Question (Multiple Choice)	Please select the difficulty of finding the items in cart page (Easy, Moderate, Difficult)
Question	Did you feel any particular emotion throughout the duration of the task? Please state.
Question	In the entire application are there any features you would change or features you dislike?
Question	Which feature did you think was well designed?
Question	Please describe the emotion, if any, that you felt throughout the usability study.

Interviewing User About the Usability of the Python Local App

After the participant has completed the usability study of Blinkee they are familiar with how the local python application works. They were then asked questions about the functionality of the app.

In a usual usability testing scenario, the user could be asked questions right after they complete a task. However, due to the nature of the application being a usability testing application the questions were asked afterwards to avoid confusion. As the participant was asked questions they were shown images of the local app to help them remember what it looked like.

The questions asked of the participant with regards to the functionality of the local app are shown in Table 16.

Table 16 - Local App Interview Questions

Q. Num.	Question
1.	Does the order of the information on the initial screen follow an order that makes sense?
2.	Was the tutorial provided in the beginning helpful? Do you think there is anything missing?
3.	How did you feel about the loading time of the application when the “Begin” test button was clicked.
4.	Do you have any suggestions regarding the text question window?
5.	Do you have any suggestions regarding the multiple-choice question window?
6.	Was the size of the task window adequate or would you have made it smaller or bigger?
7.	Regarding the upload data window do you have any suggestions?
8.	Which feature(s) do you think are well designed. Please state why.
9.	Which feature(s) do you think are poorly designed. Please state why.
10.	Are there improvements or features you would like added.
11.	How would you rank the difficulty using the application? (Easy, Moderate, Difficult)

The answers to the questions were gathered and a summary is given below. The features the users particularly liked in particular are shown in Table 17.

Table 17 - Local App Like Features

Features Liked
Dragging window is nice
Like how the usability test is automatic
Camera window is in a good place
The simplistic design made it easy to use
Tutorial was useful, eased nervousness
Instructions are clear, liked the way they are numbered
Easy to use

User suggestions regarding features they would like changed or added are shown in Table 18.

Table 18 - Improvement Suggestion & Additions

Impact on Usability	Feedback
High	Scrolling for some users was inconvenient, for other users it was a problem. It would be nice to resize the task window, as it is the size is good (for some users) but it could be bigger.
High	The upload screen freezes for a moment when the “Upload” button is pressed.
Moderate	Uploading finished should be more obvious.
Moderate	Have the title of the question/task in bold which makes the text more defined and readable
Moderate	The camera feed window covers some of the screen (also showing the camera feed made the user slightly self-conscious).
Low	Add an upload progress bar
Low	It might make more sense to have buttons in the bottom right corner instead of bottom left
Low	Give a warning that it'll open the camera/what stage it opens
Low	Have a title in the first screen. E.g. “UsabCheck Application”
Low	The tutorial button could be shown again after the test begins . That way the user has seen the UI for themselves and might want to view the tutorial then.
Low	Some users liked the confirm window when moving on to the next task. Other did not.
Low	Could say “View Tutorial” instead of “Open Tutorial”
Low	An additional progress bar would be better than the current progress text. E.g. [2/15]

5.8.4. Emotion Timelines From Blinkee Usability Study

After the participant has completed the usability study on Blinkee.com the data was uploaded to the web application where it was analysed. The answers to all of the questions by participants can be viewed in Appendix 1: Blinkee Usability Study Data.

Participant – Initials RB

The emotions predicted with this participant were most successful. The lighting conditions can be considered acceptable, there were no background interferences and the participant did not wear glasses. Additionally, the participant had a beard, however this did not seem to interfere with prediction of facial expressions.

The participant has stated that they have felt frustrated and confused when using the Blinkee shopping website. From the emotion timeline in Figure 132 it can be seen that the participant expressed emotions most significantly when answering questions and when they have encountered a serious issue. It should also be noted that the camera position was changed from the top of the screen to the bottom of the screen at 1/3 of time into the usability study which had an effect on the detection sensitivity.

The participant expressed the most significant emotion when they clicked on a broken link which caused them to frown and then smile. This was correctly detected by the algorithm. In conclusion, based on the data gathered from this participant it can be said that there some link between making facial expressions and issues encountered. However, emotions were not always expressed even when the user did not aspects the website.

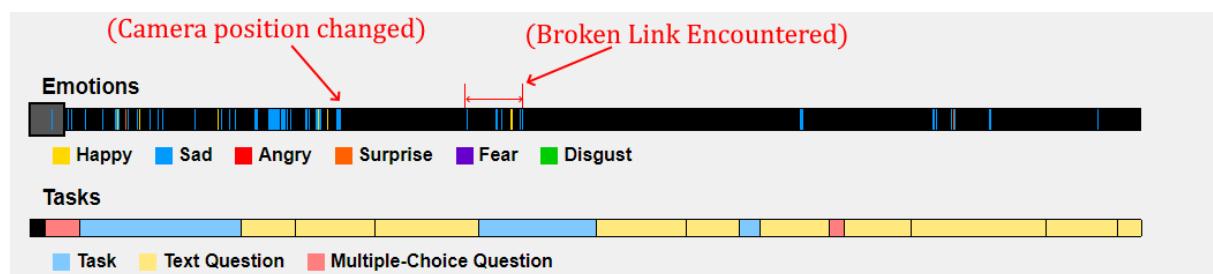


Figure 132 - Blinkee Emotions of Participant RB

Participant – Initials EW

This participant wore glasses and as a result the algorithm wrongly predicted “happy” when the participants face was neutral. The camera was positioned on the top of the screen which could have contributed to the lowered accuracy of the algorithm. As a result, the emotion data gathered from this participant could not be used to make any concrete conclusions. When the participant encountered the broken link on the web page they have not expressed any visible emotions.

Participant – Initials DS

This participant had a photo of their family in the background. The faces in the photograph were predicted by the algorithm and interfered with the usability study. As a result, the emotion data gathered from this participant could not be used to make any concrete conclusions. However, when the participant encountered a broken link on the web page the algorithm predicted “Angry” and “Sad”.

5.9. FER Model Evaluation (CRISP Evaluation Stage)

5.9.1. Model 1

Test Data Preparation (Face Extraction)

A face extraction function which detects the face in the image and crops it has been implemented. It utilizes the Haar Cascade algorithm for face detection and works in a very similar way to how predictions are made on the images from the camera feed. This function iterates over each image in the dataset and recreates the dataset except that the images are only of the face/head.

This was needed for several reasons. Firstly, the result of the test on the cropped face dataset will more closely resemble what the result will be in the usability testing application. Secondly, the FER2013 dataset (example images in Figure 133) trains on images of the head/face which means that the input into the model for prediction should be similar. Thirdly, the images in the FacesDB dataset are not square and cannot be resized to a 48x48 image that the model expects without distortion. By cropping the face out of the image not only is the background removed but the image more closely resembles the input and is of the correct size. The before extraction FacesDB images are shown in Figure 134 and the images after extraction are shown in Figure 135. The code to the face extraction is in the “Model Testing.ipynb” file.

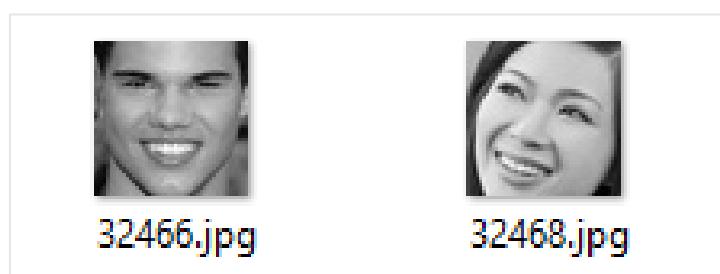


Figure 133 - FER2013 Dataset

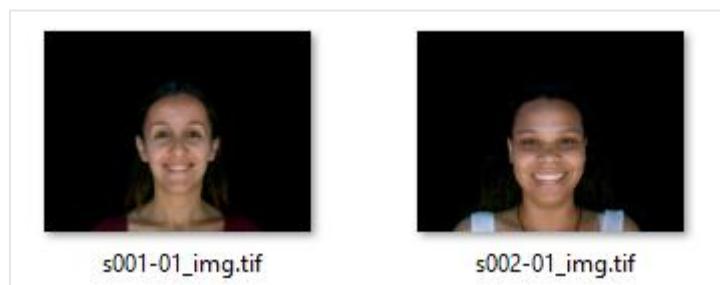


Figure 134 - FacesDB Original Images

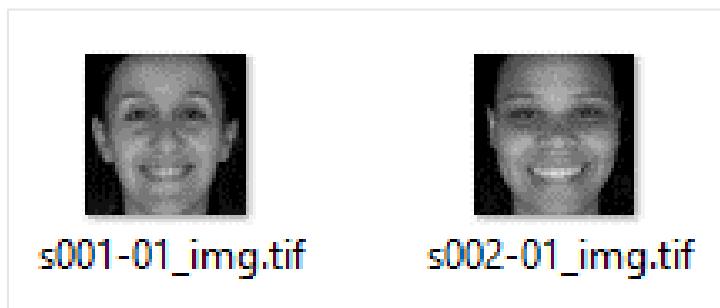


Figure 135 - FacesDB Post Face Extraction

Testing Model 1 on Other Datasets

The purpose of this section is to test the model with the images in the JAFFE and FacesDB datasets.

Model 1 which was trained on the FER2013 dataset has been tested on the JAFFE (entire dataset), FacesDB (entire dataset), and FER2013 (test set). The confusion matrices are shown in Figure 136. The model has been tested on the JAFFE dataset twice. The first is on the original resized images and again with the images after the face extraction which is explained in the previous section. The metrics such as the precision, accuracy, recall and F1 score are shown in Figure 137. The comparison of performance of the model on each dataset in terms of accuracy is displayed as a bar chart in Figure 138.

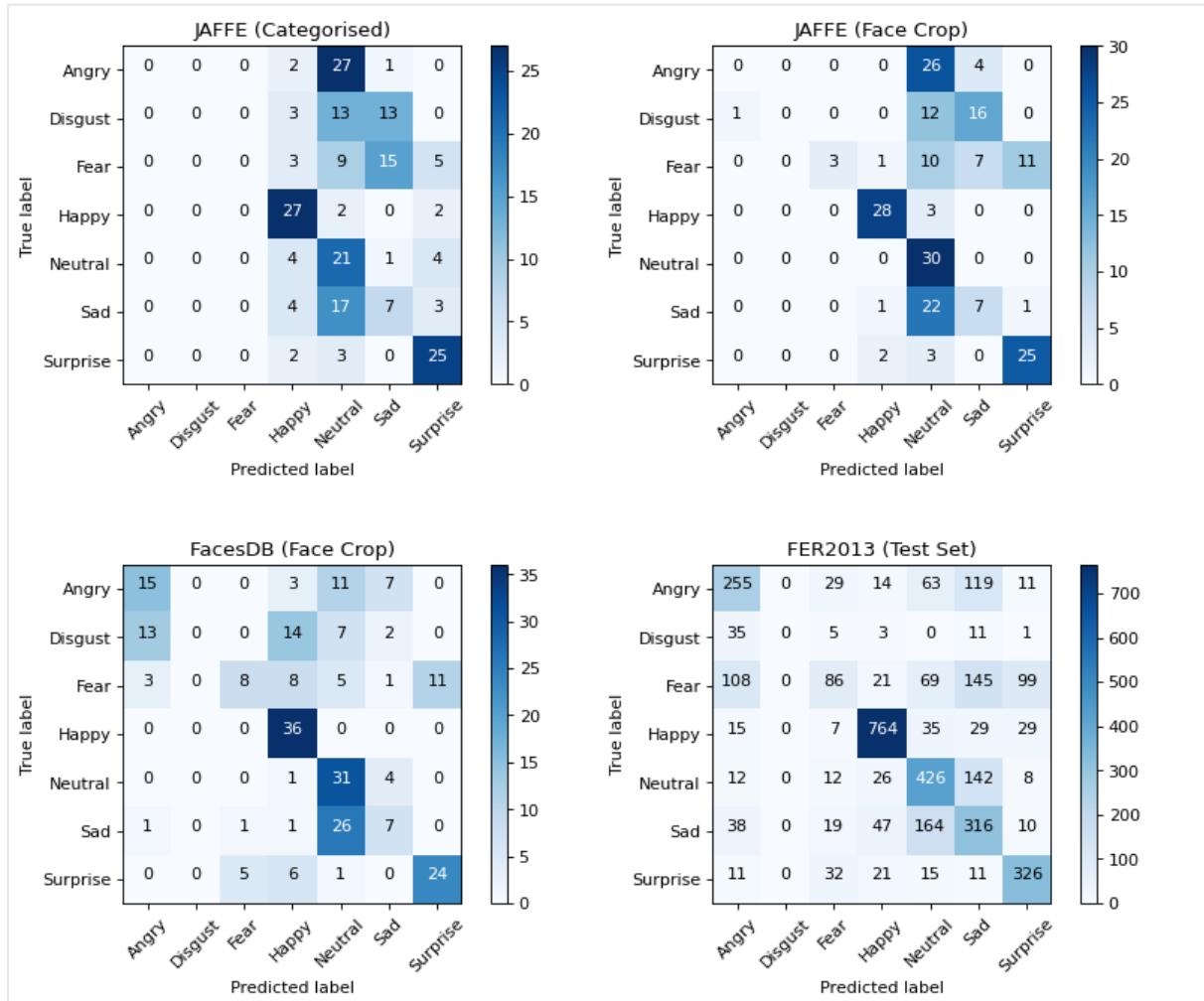


Figure 136 - Confusion Matrices (Model 1 train on FER2013)

----- JAFFE (Categorised) -----					----- JAFFE (Face Crop) -----				
	precision	recall	f1-score	support		precision	recall	f1-score	support
0	0.00	0.00	0.00	30	0	0.00	0.00	0.00	30
1	0.00	0.00	0.00	29	1	0.00	0.00	0.00	29
2	0.00	0.00	0.00	32	2	1.00	0.09	0.17	32
3	0.60	0.87	0.71	31	3	0.88	0.90	0.89	31
4	0.23	0.70	0.34	30	4	0.28	1.00	0.44	30
5	0.19	0.23	0.21	31	5	0.21	0.23	0.22	31
6	0.64	0.83	0.72	30	6	0.68	0.83	0.75	30
accuracy			0.38	213	accuracy			0.44	213
macro avg	0.24	0.38	0.28	213	macro avg	0.43	0.44	0.35	213
weighted avg	0.24	0.38	0.28	213	weighted avg	0.44	0.44	0.35	213
----- FacesDB (Face Crop) -----					----- FER2013 (Test Set) -----				
	precision	recall	f1-score	support		precision	recall	f1-score	support
0	0.47	0.42	0.44	36	0	0.54	0.52	0.53	491
1	0.00	0.00	0.00	36	1	0.00	0.00	0.00	55
2	0.57	0.22	0.32	36	2	0.45	0.16	0.24	528
3	0.52	1.00	0.69	36	3	0.85	0.87	0.86	879
4	0.38	0.86	0.53	36	4	0.55	0.68	0.61	626
5	0.33	0.19	0.25	36	5	0.41	0.53	0.46	594
6	0.69	0.67	0.68	36	6	0.67	0.78	0.72	416
accuracy			0.48	252	accuracy			0.61	3589
macro avg	0.42	0.48	0.41	252	macro avg	0.50	0.51	0.49	3589
weighted avg	0.42	0.48	0.41	252	weighted avg	0.59	0.61	0.59	3589

Figure 137 – Classification Report (Model 1 train on FER2013)

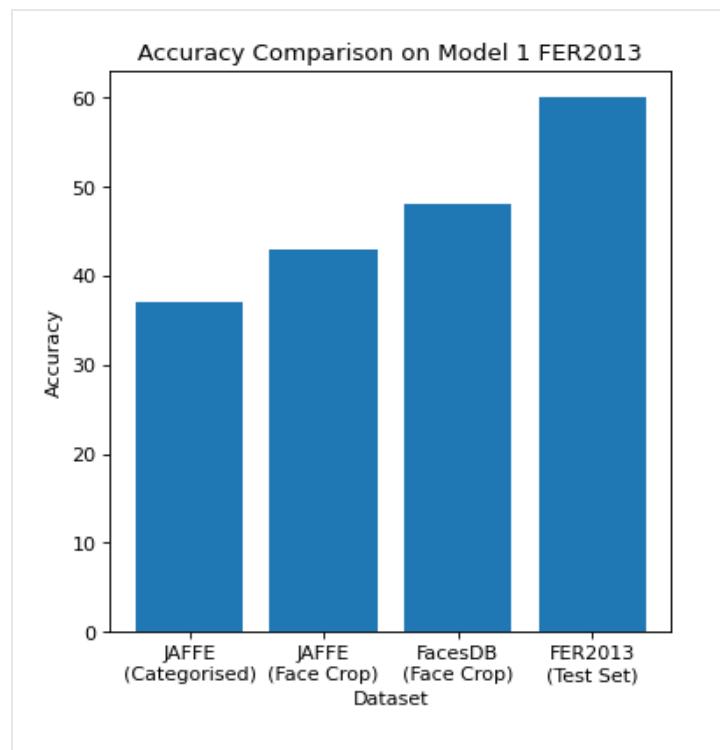


Figure 138 - Accuracy Comparisons (Model 1 train on FER2013)

Seeing the performance of the model on the JAFFE dataset raised some questions. Upon further inspection of the images within the JAFFE dataset it was clear why the model has performed in the way that it did. Emotions such as Angry, Disgust, and Fear which have a near 0% accuracy appeared far more neutral or sad than they did appear angry, disgusted, or fearful.

Below are some images of the angry, disgusted, and fearful faces in the JAFFE dataset. These are shown in Figure 139, Figure 140, and Figure 141. The emotions in the image are at times subtle and could be mistaken for other images in some cases. The angry emotions may not involve much eyebrow changes and the fearful faces appear surprised in some cases.



Figure 139 - JAFFE (Angry)



Figure 140 - JAFFE (Disgusted)

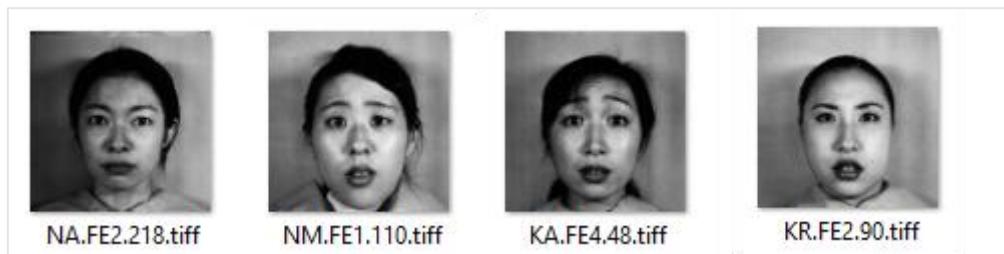


Figure 141 - JAFFE (Fearful)

Live Camera Testing (Initial Testing)

The model is finally put to practice in a real-world testing environment. This is illustrated in Figure 142. The emotions of fear and disgust were rather 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 initially yielded good results. In the research paper by Landowska (2015) the camera was at the bottom of the screen which later became the standard as it proved to be more accurate than having the camera at the top of the screen.

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.

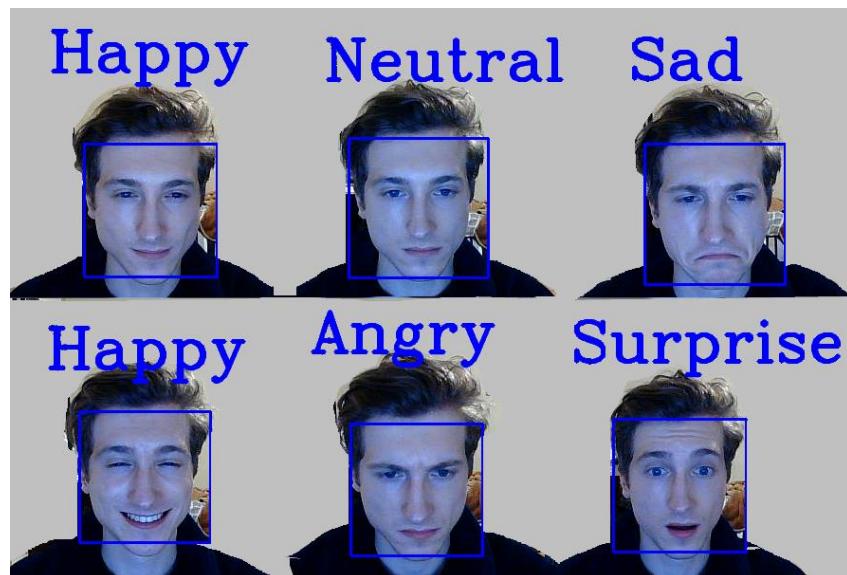


Figure 142 – Model 1 Camera Feed Predictions

Live Camera Testing (Extensive Facial Expressions Testing)

Various facial features, namely the eyebrows and mouth states were experimented with to gather data on the prediction of the model. For example, two facial expressions can be classified as “surprised” and can involve raised eyebrows, an open mouth or both. The data is presented in Table 19.

Table 19 - Model 1 Facial Expression Prediction Testing

Predict Emotion	Expected	Eyebrows State	Mouth State	Pass/Fail
Neutral	Neutral	Neutral	Neutral	Pass
Neutral	Sad	Neutral	Deep Frown	Fail
Neutral	Surprised	Raised	Neutral	Fail
Sad	Sad	Frown	Neutral	Pass
Sad	Sad	Frown	Frown	Pass
Sad/Angry	Angry	Frown	Deep Frown	Pass
Angry	Angry	Frown	Open	Pass
Disgust	Sad/Angry	Frown	Disgust Open	Fail
Surprised	Surprised	Raised	Open	Pass
Surprised	Surprised	Neutral	Open	Pass
Fear	Fear	Frown	Open	Pass
Fear	Fear	Raised	Frown	Pass
Happy	Happy	Neutral	Smile	Pass
Happy	Happy	Neutral	Open Smile	Pass
Happy	-----	Frown	Smile	-----

In conclusion, the model performs with relatively high accuracy and the classification of facial expressions in some cases depends largely on a single feature. For example, the “Sad” facial expression relies more on the eyebrows than the mouth and the “Surprised” facial expression relies more on the mouth than the eyebrows. Disgust would fall under the “Sad” or “Angry” category.

Occlusion Testing

Various ways of occlusion were tested for which includes glasses and placing a hand over the face as well as the effects that the camera position had when combined with occlusion. It was found that glasses had negatively impacted the prediction of the algorithm. When the camera was on the top the algorithm predicted happy when the face was neutral. When the camera was at the bottom it correctly predicted neutral, however, it was unable to detect a frown. This can be seen in Figure 143.

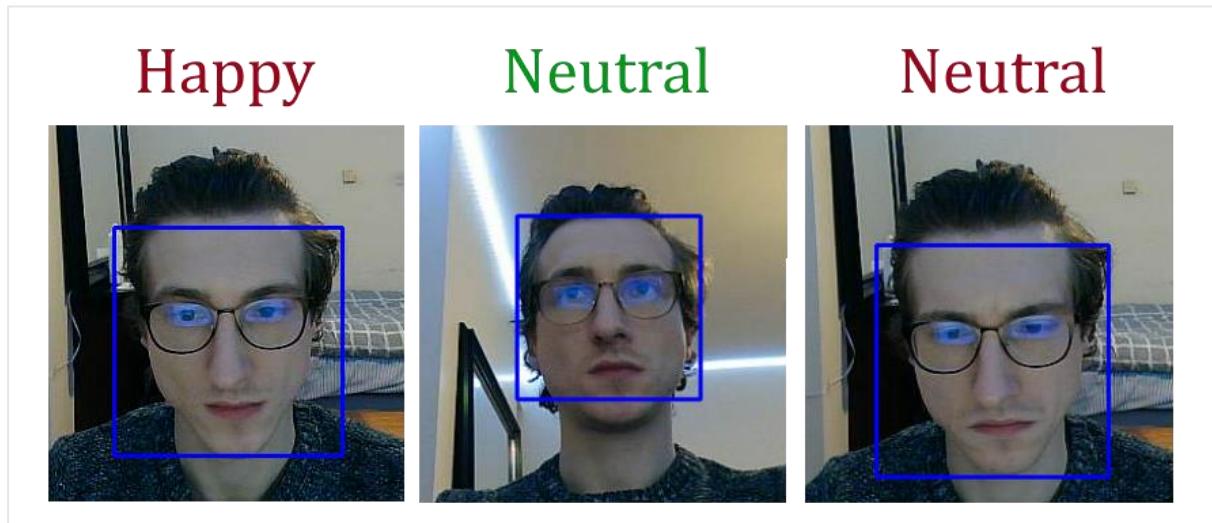


Figure 143 – Face Occlusion (Glasses)

Placing the hand over the mouth region only has not posed too much of a problem for the algorithm. It was still able to correctly predict with similar accuracy as before. However, when the bottom half or top half of the face was covered the Haar cascade face detection algorithm could not detect the face. This can be seen in Figure 144.



Figure 144 – Face Occlusion (Hand)

Model 1 Evaluation with Participants

Individuals who have no prior knowledge of the system will come in and test the FER model as a singular component. The setup will be simple and will involve a window displaying the input from the camera and the prediction of the facial expression of the participant. 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?

The data has been collected and the complete data can be viewed in Appendix 2: FER Model 1 – Testing & Evaluation With Participants. The combination of the emotion data is displayed in Table 20.

In conclusion the emotions best detected were neutral, happy, and surprised at 100% accuracy. The emotion that was worst detected was disgust at 0% accuracy and fear at 50% accuracy. Negative emotions such as sad, angry and disgust fell under the sad or angry category which shows that negative emotions are still detected even if the prediction label is not entirely correct. It was also discovered that the algorithm would make many different predictions when the participant was speaking. This is due to the fact that the face changes a lot as the person talks.

Table 20 - Model 1 Evaluation Summary

Facial Expression	Predicted Emotion	Pass	Fail	Pass %
Neutral	Neutral	4	0	100
Happy (Smile)	Happy	4	0	100
Happy (Grin)	Happy	4	0	100
Sad	Sad	3	1	75
Angry	Angry	3	1	75
Fear	Surprised	2	2	50
Fear (Exaggerated)	Fear	3	1	75
Surprised	Surprised	4	0	100
Disgusted	Sad	0	4	0

Model 1 Conclusion

The state of the eyebrows was very important to the prediction of the sad emotion. Even if the person's mouth was in the frowning state, if their eyebrows were not furrowed the model did not detect that they were sad. The mouth also needed to be opened for the model to detect the person was surprised. The disgust emotion was very difficult to detect as it would fall under the sad or angry category. The fear emotion can be considered a combination of the sad and surprised facial expression. If the person had their eyebrows furrowed and they were smiling the algorithm predicted happy. The prediction of the happy emotion depended mainly on the state of the mouth.

4.4.2. Model 2

The second model, as mentioned previously has achieved an accuracy of 56%. Looking at the metrics in Figure 146, 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 model could 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 confusion matrix in Figure 145 we can also see that the model has often predicted "Sad" when the true label was natural and vice versa.

The data from the camera was fed to the model and the predictions would change very rapidly and could definitely not be used in a production environment. The only emotion that the model could capture consistently was happiness. In conclusion, although the accuracy of the second doesn't appear to be that much lower than the first model from looking at the confusion matrices it is incomparable in terms of real-world applications and cannot be used. However, it provides some insight into how emotions are predicted by looking at the confusion matrices and seeing which types of emotions are likely to overlap.

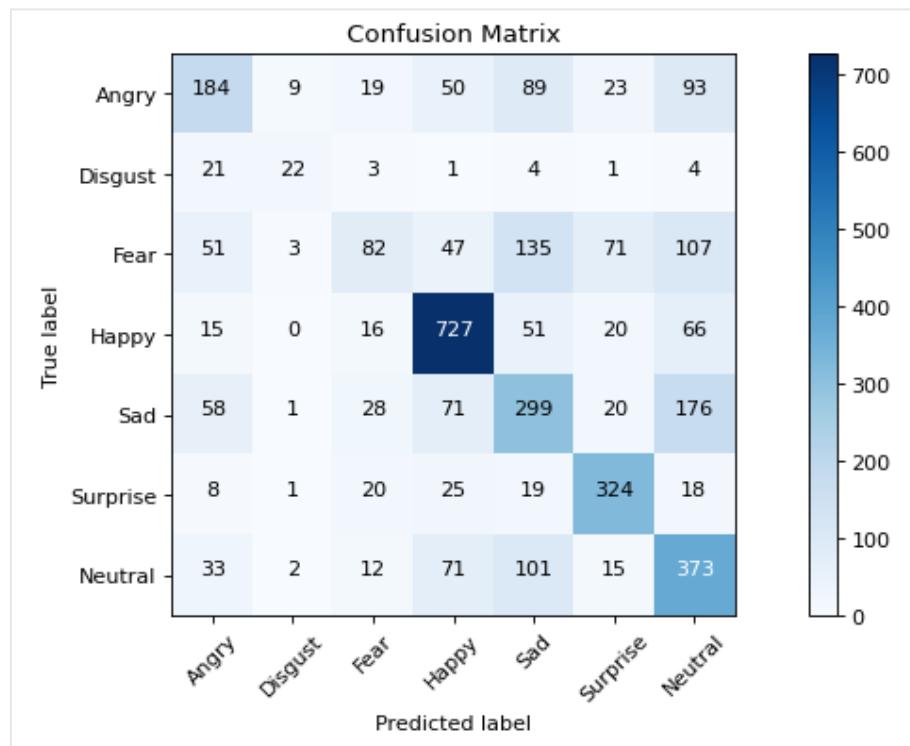


Figure 145 – Confusion Matrix of Second Model (Validation Set)

		precision	recall	f1-score	support
Angry	0	0.50	0.39	0.44	467
Disgust	1	0.58	0.39	0.47	56
Fear	2	0.46	0.17	0.24	496
Happy	3	0.73	0.81	0.77	895
Sad	4	0.43	0.46	0.44	653
Surprised	5	0.68	0.78	0.73	415
Neutral	6	0.45	0.61	0.52	607
accuracy				0.56	3589
macro avg		0.55	0.52	0.52	3589
weighted avg		0.55	0.56	0.54	3589

Figure 146 – Metrics for the Second Model (Validation Set)

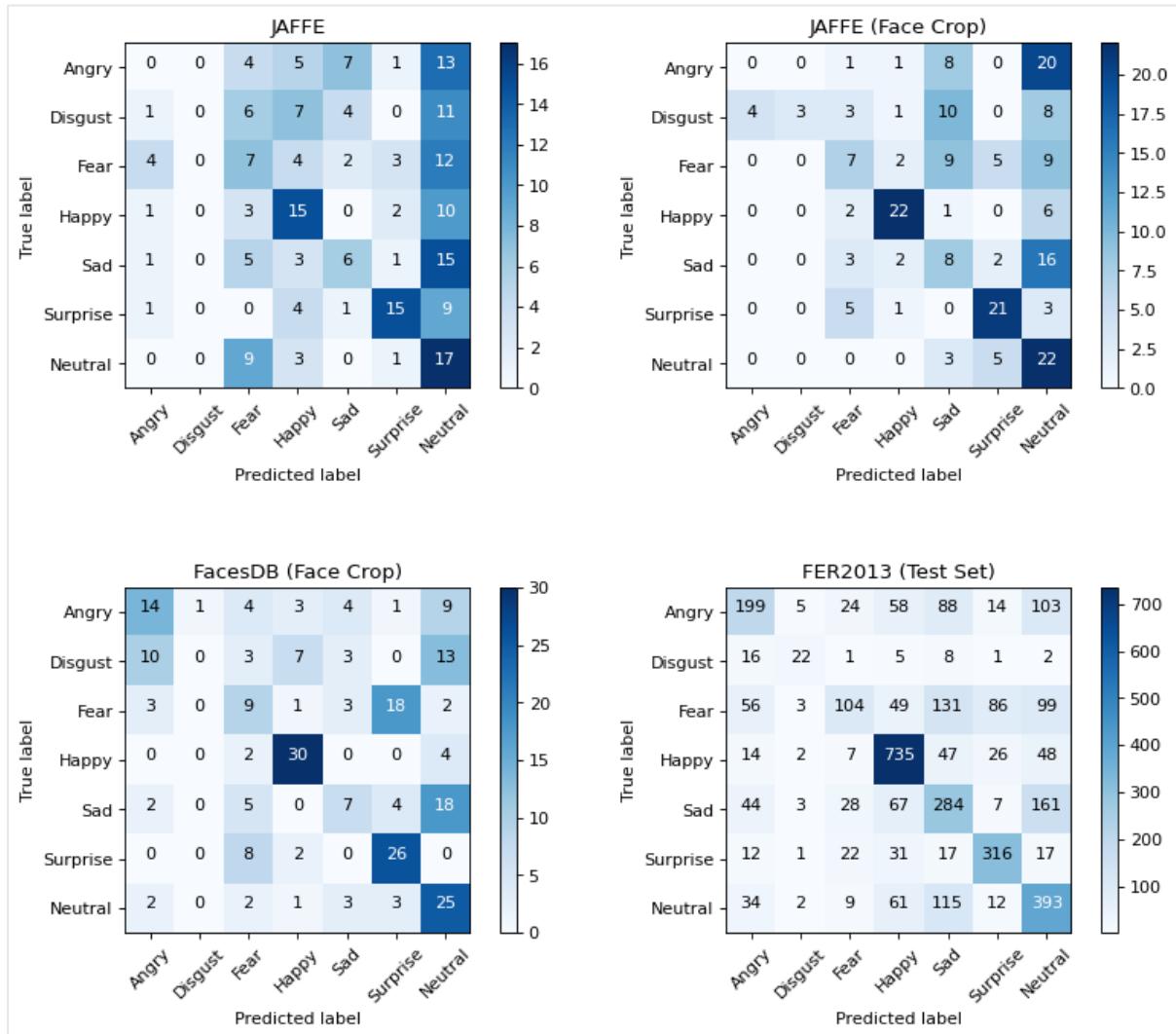


Figure 147 - Confusion Matrices (Model 2 train on FER2013)

----- JAFFE -----					----- JAFFE (Face Crop) -----				
	precision	recall	f1-score	support		precision	recall	f1-score	support
0.0	0.00	0.00	0.00	30	0.0	0.00	0.00	0.00	30
1.0	0.00	0.00	0.00	29	1.0	1.00	0.10	0.19	29
2.0	0.21	0.22	0.21	32	2.0	0.33	0.22	0.26	32
3.0	0.37	0.48	0.42	31	3.0	0.76	0.71	0.73	31
4.0	0.30	0.19	0.24	31	4.0	0.21	0.26	0.23	31
5.0	0.65	0.50	0.57	30	5.0	0.64	0.70	0.67	30
6.0	0.20	0.57	0.29	30	6.0	0.26	0.73	0.39	30
accuracy			0.28	213	accuracy			0.39	213
macro avg	0.25	0.28	0.25	213	macro avg	0.46	0.39	0.35	213
weighted avg	0.25	0.28	0.25	213	weighted avg	0.45	0.39	0.35	213
----- FacesDB (Face Crop) -----					----- FER2013 (Test Set) -----				
	precision	recall	f1-score	support		precision	recall	f1-score	support
0.0	0.45	0.39	0.42	36	0.0	0.53	0.41	0.46	491
1.0	0.00	0.00	0.00	36	1.0	0.58	0.40	0.47	55
2.0	0.27	0.25	0.26	36	2.0	0.53	0.20	0.29	528
3.0	0.68	0.83	0.75	36	3.0	0.73	0.84	0.78	879
4.0	0.35	0.19	0.25	36	4.0	0.41	0.48	0.44	594
5.0	0.50	0.72	0.59	36	5.0	0.68	0.76	0.72	416
6.0	0.35	0.69	0.47	36	6.0	0.48	0.63	0.54	626
accuracy			0.44	252	accuracy			0.57	3589
macro avg	0.37	0.44	0.39	252	macro avg	0.56	0.53	0.53	3589
weighted avg	0.37	0.44	0.39	252	weighted avg	0.57	0.57	0.55	3589

Figure 148 - Classification Report (Model 2 train on FER2013)

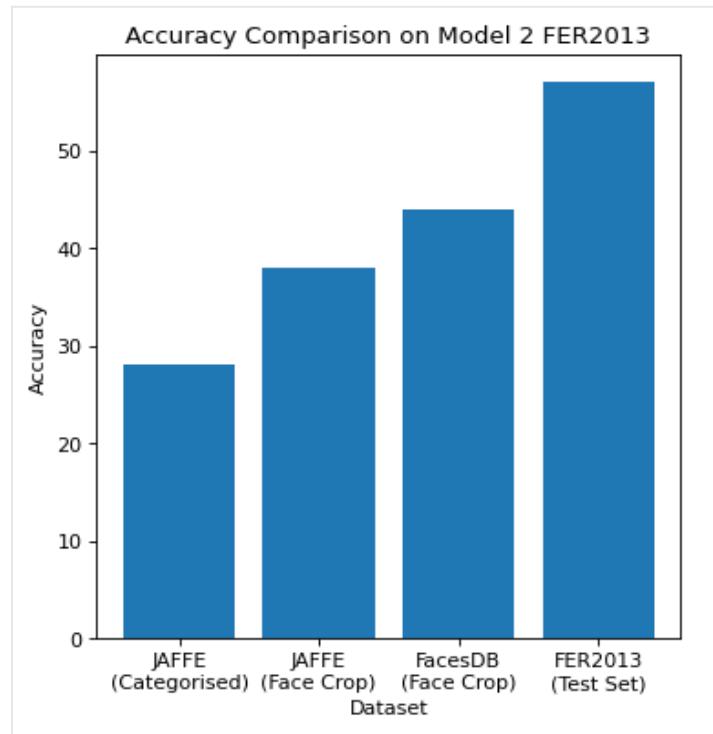


Figure 149 - Accuracy Comparisons (Model 2 train on FER2013)

6. Conclusions 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. Project Conclusions

All unit tests which tested the web application, local application and the integration of components have passed. The evaluation with participants revealed that improvements can be made to the design of the UsabCheck web application, UsabCheck local application, and the FER model. Although there is room for improvement. both the web application and local application can be used in conducting usability studies. The most notable findings are discussed below.

The FER model that has been more successful is Model 1. This model has been used in the production version of the project and evaluated in many ways. When the camera was at the bottom center of the screen the emotion predictions were more resistant to interference. Glasses in any case had a major effect on the prediction accuracy of the model and so did faces of other individuals in the same frame as the participant. This includes both photographs containing faces or other individuals in the background. Facial expressions of fear were difficult to capture and disgust was always classified under either the “sad” or “angry” category. It was also observed that certain facial features such as the mouth or eyebrows have more of an effect on the detection accuracy of certain emotions.

A website with poor design was selected to frustrate the participant and obtain the emotion data. As a result of the poor design, some participants have expressed amusement and laughter when testing the application. One participant has expressed little emotion throughout the duration of the study which was attributed to nervousness and seriousness.

From the limited data obtained it can be concluded that in some cases there was a correlation between any emotion being expressed and issues encountered by the participant. For example, when trying to access a web page with a broken hyperlink, two out of the three participants have expressed emotions. The emotions of “anger”, “sadness”, and “happiness” were prominent during that movement. However, the participants have not always expressed emotions when encountered with subtle inconveniences which need improvement. The findings show that the presence of both positive and negative emotions can be a sign of usability problems, however, the lack of emotion does not imply that there are no usability issues.

6.3. Challenges and Risks

6.3.1. Risks

Having little to no experience with the vast majority of this project such as Spring-Boot, React, Machine Learning, Python Application Development with GUIs and Video Uploading to name a few there were many risks and contingency plans put in place. For example, if the video uploading with Vimeo did not work as intended there was a plan to use YouTube which also has an API. In the beginning there was uncertainty in implementing the emotion timestamps and the journey map. The possibility, which eventually became the reality of implementing a new widget to display the data in the web application was factored into the sprint planning and research was conducted in the area of widget creation.

6.3.2. Machine Learning Challenges

One of the issues encountered was the displaying of a confusion matrix for the first model. The values did not match the accuracy of the model. The issue was with the data loader of the images and to solve this problem a function was implemented which loaded the images from a directory and classified each image at time. Once the predictions were made the confusion matrix displayed correctly.

Another issue encountered was with regards to 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 was not used in a production environment. This was not a problem as the first model has performed very well.

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 problem was solved by only installing with pip. 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, the AffectNet dataset could not be obtained as the researchers who published it encountered bandwidth problems as a result of the demand of the dataset. The dataset would have been useful in this project and a request was sent; however, a response was not received.

6.3.3. Web Application & Local Application Challenges

There were many challenges involving the web application's frontend. Having never used React before it was challenging to grasp the syntax and the flow of execution. It was challenging configuring components and passing data between them. For example, the usability test create page contains a form which contains dynamic containers that contain dynamic lists. The reloading and updating of child and parent components was a challenge to manage.

A challenge encountered with the backend involved using Spring-Boot, configuring the entry points, and authentication. Initially entry points could not be accessed and needed to be configured to work correctly with the JWT authentication.

A challenge encountered was synchronising the time of the video and the frames in the video to match the real time. This challenge was solved by counting the frames and duplicating or removing frames to ensure that the length of the video is consistent with the real time. Additionally, having no experience with GUI in Python it was a challenge to ensure all components display correctly and run concurrently when necessary.

Before implementing the video emotion timestamps there were ideas of using a library. However, the libraries did not work at all and some even required an MP4 file instead of an embedded video. The solution was to implement the video timestamps from scratch and make it compatible with Vimeo. There were challenges encountered when it came to interacting with the embedded video due same-origin policies. In other words, the JavaScript code could not access or modify the embedded video unless its origin is the same website. The solution was to use the Vimeo API.

Deploying was a challenge that involved using dependencies and plugins to compile the project. The initial idea was to use Tomcat to deploy the application. However, Tomcat requires the compiled web application to be in the form of a WAR file which did not compile correctly and was not recommended by individuals online. The solution was to compile the project as a JAR file and deploy it to Heroku. Deploying the local application required some trial and error. Initially the entire local app was compiled into a single EXE file which resulted in problems with the boot-up time and importing assets. Both of these were overcome by compiling the local app into a directory structure.

6.4. System Improvements & Future Work

6.4.1. Web Application

Improvements to the web app include more understandable error messages when unexpected errors occur and more sophisticated error handling. Additional support and information could be provided on the website to explain in even more detail what is involved in a usability test. Many examples are provided; however, novice researchers might need help with what questions to ask and how to configure tasks. There are many online resources that explain how to create a usability study and providing such detailed information is outside the scope of this project.

In response to the feedback and suggestions improvements can be made to the emotion timelines that are located under the video in the web application. Such improvements include clicking on the slider window to move it instead of using a scroll bar and auto-scrolling to the top where the video is when an emotion timestamp is clicked. Other improvements include colouring and minimizing of sections in the usability create page. The feedback for the web app is displayed in Table 11 in the evaluation section. Another improvement is implementing a filter for emotions which allows the researcher to selectively display emotions of a minimum duration and to not display the emotions for the sections when the participant is answering questions. For security reasons a password can be implemented to prevent users with the test reference code from taking the test.

6.4.2. Local Application

Improvements based on the feedback from participants and Nielsen's heuristics include loading bars to show progress of data upload, colouring inactive buttons a dark gray colour instead of light gray, reappearing a question window in the position it was last moved to as it is currently done with the task window, and implement the possibility of resizing the task window.

Resizing the task window and making it customizable mitigates the problem some user's had with scrolling. The data upload to the web application should run on a separate thread so as to not temporarily freeze the application when a large quantity of data is sent. The upload complete status should be more obvious and this can be achieved with a loading bar or a more prominent message or alert. Titles of tasks and the question text can be made bold to make it easier for the user to focus.

The camera feed window takes up a portion of the screen and to tackle this inconvenience image processing techniques can be used to alter the video frames and paste the camera feed frames on top of the video. The FER algorithm can run in the background without interfering with the participant, however, the problem with that is that it still blocks part of the screen in the video just not for the participant. Another solution could be to upload two videos, one of the screen recording and the other of the camera feed and display them separately on the web application. The new challenge would be displaying both side by side in the web application.

Additional improvements include notifying the user that their camera feed is going to be opened and possibly even add an option to disable the FER aspect of the usability study. The screen recording does not record audio and an issue encountered is that the FER model makes seemingly random predictions when the participant speaks due to their facial features changing. A potential mitigation would be to implement a push-to-talk button to not detect emotions when the user is speaking. This would however interfere with the usability test. Another idea is to not detect emotions when audio is detected.

6.4.3. Facial Expression Recognition

Model 1 which is the VGG-16 model variation that has trained on the FER2013 dataset encountered issues with detecting facial expressions of participants who wore glasses. To counter this problem the model should be trained on a FER dataset which includes faces of individuals who wear glasses and other accessories. The FER algorithm which obtains images from the camera and extracts the face needs to be reconfigured to detect only the biggest face in the frame. This is to counter the issues involving other persons or faces of persons in the background. Other machine learning/deep learning models can be used to increase the prediction accuracy of facial expressions such as the Resnet model, Visual Bag of Words and SVMs. Further research can involve training the model(s) on a usability testing FER dataset which consists of faces from individuals in a usability testing environment.

6.4.4. Final Reflections

This project has provided a tremendous learning experience in the areas of machine learning, web development, Python application development, usability testing and software engineering which the author is very grateful for. It has been an enormous effort to design and develop the usability testing suite as well as a true joy developing the project and contributing to the field of software testing. Overcoming the many challenges and achieving the final end result has been very rewarding and encouraging.

6.4.5. Sprint Chart

Initial Plan

The initial sprint plan for the months September until April is shown in Figure 150. As the text is not very legible, the sprint plan figure has been split into two figures (Figure 151 and Figure 152).

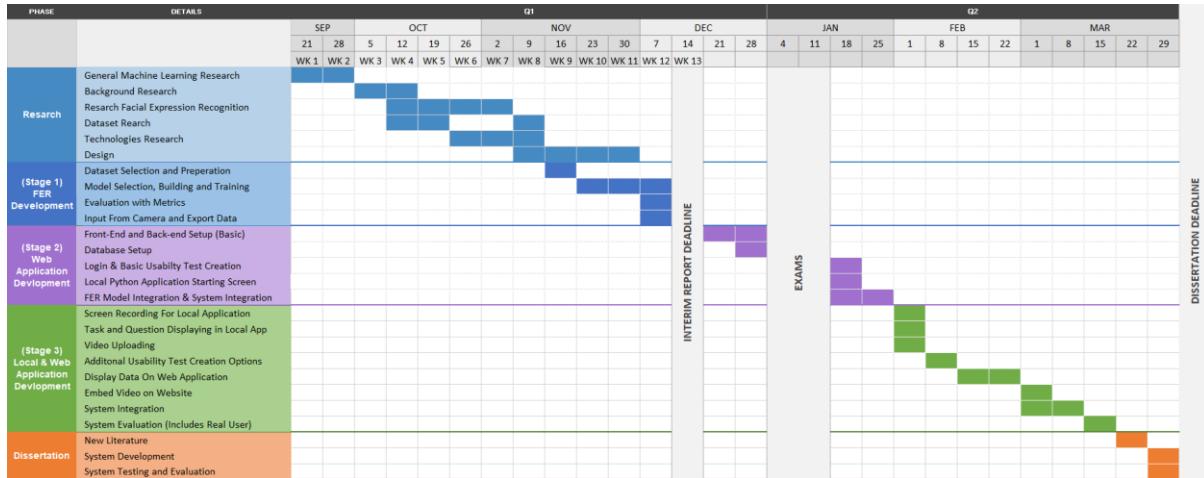


Figure 150 – Sprint Chart Overview (All Stages)

The first two stages are Research and FER Development and illustrated in Figure 151. This sprint chart is the representation of the research and FER development.

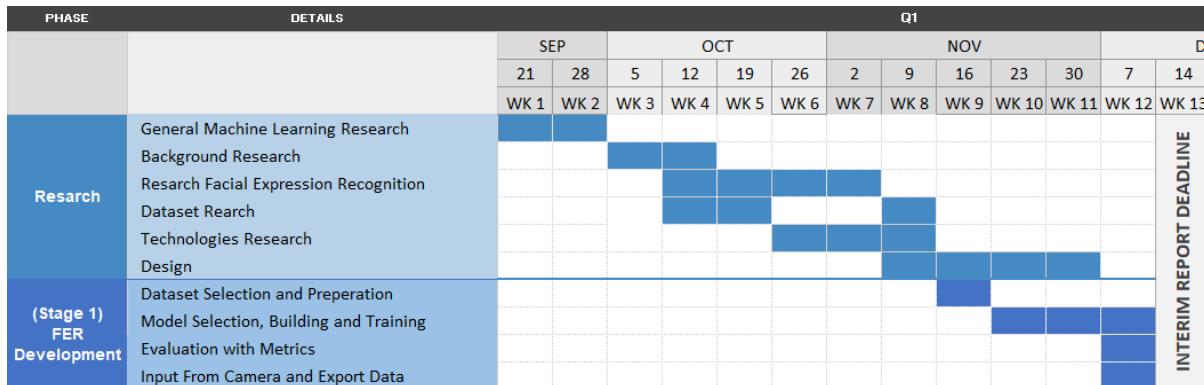


Figure 151 – Sprint Chart First Two Stages (Zoomed in for Readability)

The sprint plan for the last three stages which includes the Web Application Development, Local Application Development and the Dissertation report are illustrated in Figure 152. This sprint plan was designed when the interim report was submitted.

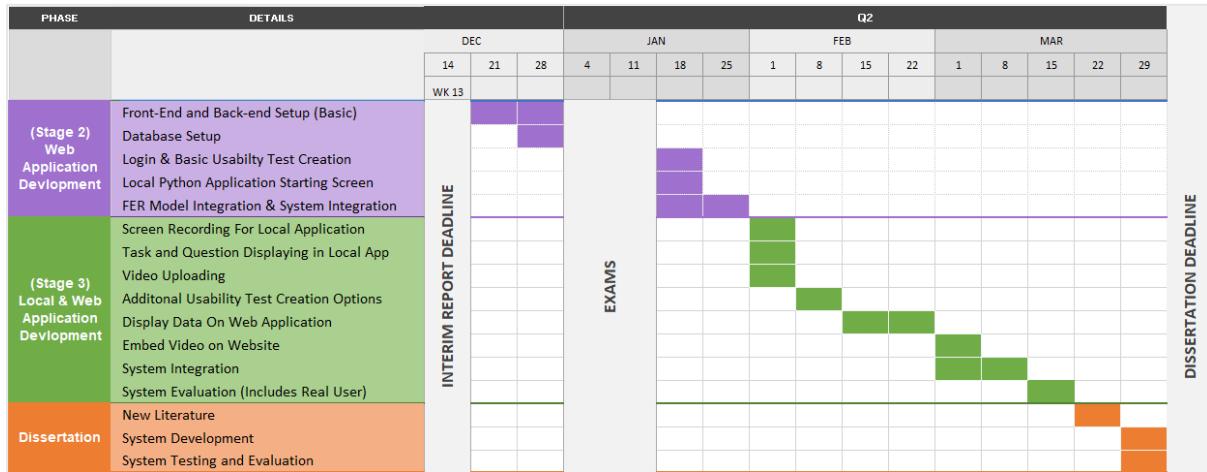


Figure 152 – Sprint Chart Last Three Stages (Zoomed in for Readability)

Plan Followed

The development that occurred from January until April is displayed in Figure 153.

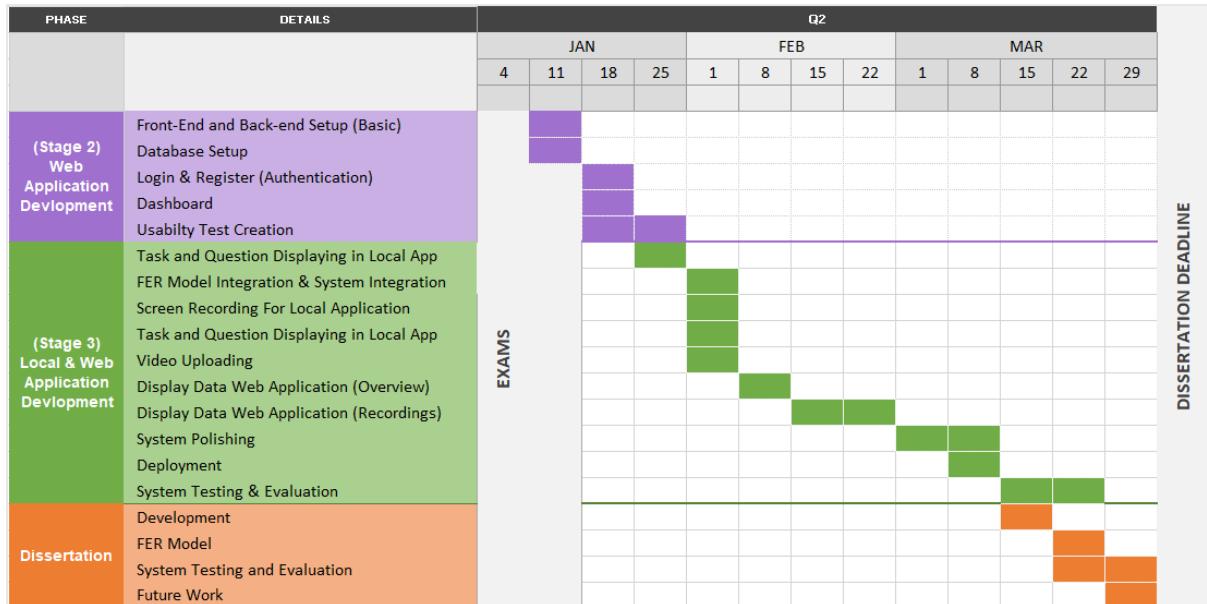


Figure 153 – Sprint Plan Followed

The tasks were broken up into small manageable chunks and planned by the day. The exact development procedure that was followed is shown in Figure 154 and Figure 155. Throughout the development when a task was complete it was coloured green. Finally, when all tasks within a cell were complete the cell was coloured green to indicate that it was finished. This list of tasks was easy to view and manage.

FYP Development

15 January 2021 14:52

Month	Week	Day	Development
JAN		15 -> 16:	Setup backend and front-end of web application (very basic environment setup)
JAN		17:	Setup database (including tables) and connect backend to database
JAN		18:	Implement register and login (front-end and back-end)
JAN		19:	Finish authentication: <ul style="list-style-type: none"> • Change User to Researcher • Check if user exists before creating • Repository vs Dao • Entity vs Model • Request & Response • Change Front-end appearance • Change entry point names
JAN		20 -> 23:	Create Dashboard <ul style="list-style-type: none"> • Project selection (Drop down menu) • Create Project (Modal pop-up) • Delete Project • Display Tests
JAN	WK 1	24 -> 26:	Test creation in web application
JAN	WK 1	27 -> 28	View tests in local application <ul style="list-style-type: none"> • Create local application UI and pull data (Basic)
JAN	WK 1	29 -> 30	Task Rendering (Local Application)
JAN	WK 1	31	Question Rendering (Local Application)
FEB	WK 2	1 -> 2	Screen recording video <ul style="list-style-type: none"> • Record screen • UI working and integrated
FEB	WK 2	3 -> 4	Integrate FER
FEB	WK 2	5 -> 6	Uploading with Vimeo
FEB	WK 2	6	Upload data to backend and store it
FEB	WK 2	7 -> 8	Task select (View Test in Web app)
FEB	WK 3	9 -> 10	Bar chart (View Test in Web app)
FEB	WK 3	11 -> 12	Pie chart (View Test in Web app)
FEB	WK 3	13 -> 14	View video including with time stamps (Recording in Web app)
FEB	WK 5	24	Task grading (Recording in Web app)

Figure 154 - Project Development Schedule (Part 1)

FEB & MAR	WK 5 WK 6	25 -> 12	<p>System Polishing Web App:</p> <ul style="list-style-type: none"> • Refactor code • Test creation form validation • Increase font size • Show test details in dashboard instead of N/A • Add status settings • Grouping in test result overview • Implement all status updates for web app <ul style="list-style-type: none"> ◦ Create project ◦ Delete project ◦ Create Test ◦ Delete Test ◦ Change Test Status • Make changes to emotions bars (add spacing) • Give context <ul style="list-style-type: none"> ◦ What is a project ◦ Pre-test, Test, Task ◦ What is a status ◦ What is a reference code ◦ Text Question/Multiple-choice question ◦ How does the FER data display works • Show examples in web app • Add scenario in test create • Make a logo • Implement scenario in web app • Add loading bar in local app (camera, screen recorder etc.)
MAR	WK 7	14	Deploying Local application

Figure 155 - Project Development Schedule (Part 2)

The exact report development schedule that was followed is shown in Figure 156.

			Report
MAR	WK 8	15	<ul style="list-style-type: none"> • Draw updated DB schema • Draw Backend class diagram • Explain Back-end authentication • Explain Entry points and classes
MAR	WK 8	16	<ul style="list-style-type: none"> • Draw Front-end diagram • Explain how the pages and components work
MAR	WK 8	18 -> 19	<ul style="list-style-type: none"> • Get screenshots of all the web app and local app • Explain local app functionality • Explain recording functionality
MAR	WK 8	20 -> 22	<ul style="list-style-type: none"> • FER (Fix Metrics) • JAFFE & FacesDB Comparisons • Discuss the FER progress
MAR	WK 9	23	<ul style="list-style-type: none"> • Discuss Gestalt Laws (design/development/where relevant) • Discuss Colours and Emotions • Draw page relationship diagram
MAR	WK 9	25	<ul style="list-style-type: none"> • Test Local App (Participant 1)
MAR	WK 9	26	<ul style="list-style-type: none"> • Create the web app usability test • Create Web App Usability Test • NoCoffee Simulator • Local App Expert Evaluation • Local App Nielsen's Heuristics • Persona for evaluation (Lucid App)
MAR	WK 9	27	<ul style="list-style-type: none"> • Test web app with participants
MAR	WK 9	28	<ul style="list-style-type: none"> • Evaluate machine learning (occlusion etc.) • Local App Automated Testing
MAR	WK 10	29	<ul style="list-style-type: none"> • Add all FER model testing • Mention number of participants tested • Talking is not possible. • Review the emotion bars obtained with the local app recording (Blinkee) • See what problems came up in official publications/articles • Discuss problems that came up • Rewrite Evaluation Plan

Figure 156 - Project Report Schedule

Bibliography

1. Return on Investment for Usability [Internet]. Nielsen Norman Group. 2020 [cited 2020 Dec 14]. Available from: <https://www.nngroup.com/articles/return-on-investment-for-usability/>
2. What is Usability Testing? (and What it Isn't) | Hotjar [Internet]. Hotjar; 2020 [cited 2020 Dec 14]. Available from: <https://www.hotjar.com/usability-testing/>
3. Virzi RA. Refining the Test Phase of Usability Evaluation: How Many Subjects Is Enough? Human Factors: The Journal of the Human Factors and Ergonomics Society [Internet]. 1992 Aug [cited 2020 Dec 14]; Available from: <https://journals.sagepub.com/doi/10.1177/001872089203400407>
4. Faulkner L. Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. Behavior Research Methods, Instruments, & Computers [Internet]. 2003 Aug [cited 2020 Dec 14]; Available from: <https://link.springer.com/article/10.3758%252FBF03195514>
5. Christian Bastien J. Usability testing: some current practices and research questions [Internet]. Available from: <https://arxiv.org/ftp/arxiv/papers/1009/1009.5918.pdf>
6. Landowska, Agnieszka. (2015). Towards Emotion Acquisition in IT Usability Evaluation Context. [cited 2020 Dec 14]; Available from: https://www.researchgate.net/publication/301454217_Towards_Emotion_Acquisition_in_IT_Usability_Evaluation_Context
7. Connor Esterwood. Facial Recognition & Journey Mapping for Improved Usability Testing [Internet]. Medium. Medium; 2018 [cited 2020 Dec 14]. Available from: <https://medium.com/@cesterwo/facial-recognition-journey-mapping-for-improved-usability-testing-37269a50b71f>
8. Halder, Rituparna & Sengupta, Sushmit & Pal, Arnab & Ghosh, Sudipta & Kundu, Debashish. (2016). Real Time Facial Emotion Recognition based on Image Processing and Machine Learning. International Journal of Computer Applications. [cited 2020 Dec 14]; Available from: https://www.researchgate.net/publication/301335563_Real_Time_Facial_Emotion_Recognition_based_on_Image_Processing_and_Machine_Learning
9. Gaurav Sharma. Real Time Facial Expression Recognition - Data Driven Investor - Medium [Internet]. Medium. Data Driven Investor; 2018 [cited 2020 Dec 14]. Available from: <https://medium.com/datadriveninvestor/real-time-facial-expression-recognition-f860dacfeb6a>
10. Bastien, J.. (2009). Usability testing: A review of some methodological and technical aspects of the method. International journal of medical informatics. ResearchGate [Internet]. 2020 [cited 2020 Dec 14]; Available from: https://www.researchgate.net/publication/24256512_Usability_testing_A_review_of_some_methodological_and_technical_aspects_of_the_method

11. Sarang Narkhede. Understanding Confusion Matrix - Towards Data Science [Internet]. Medium. Towards Data Science; 2018 [cited 2020 Dec 14]. Available from: <https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>
12. Koehrsen W. Beyond Accuracy: Precision and Recall - Towards Data Science [Internet]. Medium. Towards Data Science; 2018 [cited 2020 Dec 14]. Available from: <https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c>
13. Patel R, Dhakad J, Desai K, Gupta T, Correia S. Hand Gesture Recognition System using Convolutional Neural Networks. 2018 4th International Conference on Computing Communication and Automation (ICCCA) [Internet]. 2018 Dec [cited 2020 Dec 14]; Available from: <https://ieeexplore.ieee.org/document/8777621>
14. Lawrence, Steve & Giles, C & Tsoi, Ah & Back, Andrew. (1997). Face Recognition: A Convolutional Neural Network Approach. ResearchGate [Internet]. 2020 [cited 2020 Dec 14]; Available from: https://www.researchgate.net/publication/3302251_Face_Recognition_A_Convolutional_Neural_Network_Approach
15. Cireşan D, Meier U, Schmidhuber J. Multi-column Deep Neural Networks for Image Classification [Internet]. arXiv.org. 2012 [cited 2020 Dec 14]. Available from: <https://arxiv.org/abs/1202.2745>
16. DeepAI. ReLu [Internet]. DeepAI. DeepAI; 2019 [cited 2020 Dec 14]. Available from: <https://deepai.org/machine-learning-glossary-and-terms/relu>
17. Sharma, Neha & Jain, Vibhor & Mishra, Anju. (2018). An Analysis Of Convolutional Neural Networks For Image Classification. ResearchGate [Internet]. 2018 [cited 2020 Dec 14]; Available from: https://www.researchgate.net/publication/325663368_An_Analysis_Of_Convolutional_Neural_Networks_For_Image_Classification
18. Stewart M. Simple Introduction to Convolutional Neural Networks [Internet]. Medium. Towards Data Science; 2019 [cited 2020 Dec 14]. Available from: <https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac>
19. VGG16 - Convolutional Network for Classification and Detection [Internet]. Neurohive.io. 2018 [cited 2020 Dec 14]. Available from: <https://neurohive.io/en/popular-networks/vgg16/>
20. ImageNet [Internet]. Image-net.org. 2017 [cited 2020 Dec 14]. Available from: <http://image-net.org/index>
21. Sik-Ho Tsang. Review: VGGNet — 1st Runner-Up (Image Classification), Winner (Localization) in ILSVRC 2014 [Internet]. Medium. Coinmonks; 2018 [cited 2020 Dec 14]. Available from: <https://medium.com/coinmonks/paper-review-of-vggnet-1st-runner-up-of-ilsvrc-2014-image-classification-d02355543a11>
22. ImageNet [Internet]. Image-net.org. 2016 [cited 2020 Dec 14]. Available from: <http://image-net.org/about-overview>

23. Kusuma GP, Jonathan J, Lim AP. Emotion Recognition on FER-2013 Face Images Using Fine-Tuned VGG-16. Advances in Science, Technology and Engineering Systems Journal [Internet]. 2020 [cited 2020 Dec 14];5(6):315–22. Available from: https://www.astesj.com/publications/ASTESJ_050638.pdf
24. Rohith Gandhi. Support Vector Machine — Introduction to Machine Learning Algorithms [Internet]. Medium. Towards Data Science; 2018 [cited 2020 Dec 14]. Available from: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>
25. Saptashwa Bhattacharyya. Support Vector Machine: Kernel Trick; Mercer's Theorem [Internet]. Medium. Towards Data Science; 2018 [cited 2020 Dec 14]. Available from: <https://towardsdatascience.com/understanding-support-vector-machine-part-2-kernel-trick-mercers-theorem-e1e6848c6c4d>
26. Analytics Vidhya. SVM | Support Vector Machine Algorithm in Machine Learning [Internet]. Analytics Vidhya. 2017 [cited 2020 Dec 14]. Available from: <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>
27. Aybüke Yalçınér. Bag of Visual Words(BoVW) - Aybüke Yalçınér - Medium [Internet]. Medium. Medium; 2019 [cited 2020 Dec 14]. Available from: <https://medium.com/@aybukeyalcinerr/bag-of-visual-words-bovw-db9500331b2f>
28. Image Classification using Bag of Visual Words Model | MLK - Machine Learning Knowledge [Internet]. MLK - Machine Learning Knowledge. 2020 [cited 2020 Dec 14]. Available from: <https://machinelearningknowledge.ai/image-classification-using-bag-of-visual-words-model/>
29. Divyansh Dwivedi. Face Detection For Beginners - Towards Data Science [Internet]. Medium. Towards Data Science; 2018 [cited 2020 Dec 15]. Available from: <https://towardsdatascience.com/face-detection-for-beginners-e58e8f21aad9>
30. Lyons M, Kamachi M, Gyoba J. The Japanese Female Facial Expression (JAFFE) Dataset. Zenodo [Internet]. 1998 Apr 14 [cited 2020 Dec 14]; Available from: <https://zenodo.org/record/3451524>
31. Rohit Verma. fer2013 [Internet]. Kaggle.com. 2013 [cited 2020 Dec 14]. Available from: <https://www.kaggle.com/deadskull7/fer2013>
32. AffectNet – Mohammad H. Mahoor, PhD [Internet]. Mohammadmahoor.com. 2017 [cited 2020 Dec 14]. Available from: <http://mohammadmahoor.com/affectnet/>
33. Home - FacesDB | VISGRAF [Internet]. Impa.br. 2012 [cited 2020 Dec 14]. Available from: <http://app.visgraf.imp.br/database/faces>
34. Google facial expression comparison dataset [Internet]. Google Research. 2018 [cited 2020 Dec 14]. Available from: <https://research.google/tools/datasets/google-facial-expression/>
35. Emotions in context [Internet]. Uoc.edu. 2019 [cited 2020 Dec 14]. Available from: <http://sunai.uoc.edu/emotic/>

36. Bechtel J. Two Major Concerns about the Ethics of Facial Recognition in Public Safety [Internet]. Design World. 2019 [cited 2020 Dec 16]. Available from: <https://www.designworldonline.com/two-major-concerns-about-the-ethics-of-facial-recognition-in-public-safety/>
37. What is GDPR, the EU's new data protection law? - GDPR.eu [Internet]. GDPR.eu. 2018 [cited 2020 Dec 16]. Available from: <https://gdpr.eu/what-is-gdpr/>
38. leonshting, leonshting/fec-dataset-loader [Internet]. GitHub. 2020 [cited 2020 Dec 14]. Available from: <https://github.com/leonshting/fec-dataset-loader>
39. alex-miller-0, alex-miller-0/Tor_Crawler [Internet]. GitHub. 2019 [cited 2020 Dec 14]. Available from: https://github.com/alex-miller-0/Tor_Crawler
40. Actionable UX insights for better digital experiences [Internet]. Userzoom.com. 2020 [cited 2020 Dec 14]. Available from: <https://www.userzoom.com/>
41. UserTesting: The Human Insight Platform [Internet]. UserTesting. 2020 [cited 2020 Dec 14]. Available from: <https://www.usertesting.com/>
42. Online User Testing Tool | Loop11 [Internet]. Loop11.com. 2020 [cited 2020 Dec 14]. Available from: <https://www.loop11.com/>
43. Welcome to Python.org [Internet]. Python.org. Python.org; 2020 [cited 2020 Dec 14]. Available from: <https://www.python.org/>
44. Python Features - javatpoint [Internet]. www.javatpoint.com. 2011 [cited 2020 Dec 14]. Available from: <https://www.javatpoint.com/python-features>
45. The Web framework for perfectionists with deadlines | Django [Internet]. Djangoproject.com. 2020 [cited 2020 Dec 14]. Available from: <https://www.djangoproject.com/>
46. Java.com. 2020 [cited 2020 Dec 14]. Available from: <https://www.java.com/en/>
47. What is Java and why is it important? - Code Institute [Internet]. Code Institute. 2014 [cited 2020 Dec 14]. Available from: <https://codeinstitute.net/blog/what-is-java/>
48. What is JavaScript? [Internet]. MDN Web Docs. 2020 [cited 2020 Dec 14]. Available from: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript
49. Node.js. Node.js [Internet]. Node.js. 2020 [cited 2020 Dec 14]. Available from: <https://nodejs.org/en/>
50. AngularJS — Superheroic JavaScript MVW Framework [Internet]. Angularjs.org. 2010 [cited 2020 Dec 14]. Available from: <https://angularjs.org/>
51. React – A JavaScript library for building user interfaces [Internet]. Reactjs.org. 2020 [cited 2020 Dec 14]. Available from: <https://reactjs.org/>

52. NumPy [Internet]. Numpy.org. 2019 [cited 2020 Dec 14]. Available from: <https://numpy.org/>
53. pandas - Python Data Analysis Library [Internet]. Pydata.org. 2020 [cited 2020 Dec 14]. Available from: <https://pandas.pydata.org/>
54. Matplotlib: Python plotting — Matplotlib 3.3.3 documentation [Internet]. Matplotlib.org. 2012 [cited 2020 Dec 14]. Available from: <https://matplotlib.org/>
55. scikit-learn: machine learning in Python — scikit-learn 0.23.2 documentation [Internet]. Scikit-learn.org. 2020 [cited 2020 Dec 14]. Available from: <https://scikit-learn.org/stable/>
56. Introduction to TensorFlow [Internet]. TensorFlow. 2020 [cited 2020 Dec 14]. Available from: <https://www.tensorflow.org/learn>
57. Asena Atilla Saunders. Top Five Use Cases of TensorFlow [Internet]. Exastax. Exastax; 2017 [cited 2020 Dec 14]. Available from: <https://www.exastax.com/deep-learning/top-five-use-cases-of-tensorflow/>
58. Keras Team. Keras: The Python deep learning API [Internet]. Keras.io. 2020 [cited 2020 Dec 14]. Available from: <https://keras.io/>
59. Welcome to fastai | fastai [Internet]. Fast.ai. 2020 [cited 2020 Dec 14]. Available from: <https://docs.fast.ai/>
60. Dmytro Mishkin. Fast.ai library: 1st impression - Towards Data Science [Internet]. Medium. Towards Data Science; 2019 [cited 2020 Dec 14]. Available from: <https://towardsdatascience.com/fast-ai-library-1st-impression-958cb52afc>
61. Home - OpenCV [Internet]. OpenCV. 2020 [cited 2020 Dec 14]. Available from: <https://opencv.org/>
62. Multi-cloud storage for your applications of today and tomorrow. SwiftStack Data Platform [Internet]. SwiftStack. 2020 [cited 2020 Dec 14]. Available from: <https://www.swiftstack.com/>
63. StreamingVideoProvider [Internet]. StreamingVideoProvider. 2020 [cited 2020 Dec 14]. Available from: <https://www.streamingvideoprovider.com/>
64. Vimeo [Internet]. Vimeo.com. 2020 [cited 2020 Dec 14]. Available from: <https://vimeo.com/>
65. YouTube Data API | Google Developers [Internet]. Google Developers. 2020 [cited 2020 Dec 14]. Available from: <https://developers.google.com/youtube/v3>
66. Apache Tomcat Project. Apache Tomcat® - Welcome! [Internet]. Apache.org. 2020 [cited 2020 Dec 14]. Available from: <http://tomcat.apache.org/>
67. Five Reasons You Should Use Tomcat (Updated for 2020) [Internet]. Future Hosting. 2014 [cited 2020 Dec 14]. Available from: <https://www.futurehosting.com/blog/five-reasons-you-should-use-tomcat/>
68. Firebase [Internet]. Firebase. 2020 [cited 2020 Dec 14]. Available from: <https://firebase.google.com/>

69. MySQL [Internet]. Mysql.com. 2020 [cited 2020 Dec 15]. Available from: <https://www.mysql.com/>
70. 8 Major Advantages of Using MySQL [Internet]. Datamation.com. 2016 [cited 2020 Dec 15]. Available from: <https://www.datamation.com/storage/8-major-advantages-of-using-mysql.html>
71. Atlassian. What is Scrum? [Internet]. Atlassian. 2018 [cited 2020 Dec 14]. Available from: <https://www.atlassian.com/agile/scrum>
72. Scrum methodology [Internet]. Digital.ai. 2020 [cited 2020 Dec 14]. Available from: <https://digital.ai/glossary/scrum-methodology>
73. Blueprint Software Systems. Benefits and Challenges of Agile Development [Internet]. Blueprintsys.com. 2017 [cited 2020 Dec 14]. Available from: <https://www.blueprintsys.com/agile-development-101/agile-benefits-and-challenges>
74. Scaling Data Science: How We Use CRISP-DM and Agile | AgileThought [Internet]. AgileThought. 2018 [cited 2020 Dec 14]. Available from: <https://agilethought.com/blogs/scaling-data-science-use-crisp-dm-agile/>
75. Crisp DM methodology - Smart Vision Europe [Internet]. Smart Vision Europe. 2020 [cited 2020 Dec 14]. Available from: <https://www.sv-europe.com/crisp-dm-methodology/>
76. Videojs-markers [Internet]. Sampingchuang.com. 2020 [cited 2020 Dec 14]. Available from: <http://sampingchuang.com/videojs-markers>
77. pydeveloperashish. pydeveloperashish/Facial-Expressions-Recognition [Internet]. GitHub. 2020 [cited 2020 Dec 16]. Available from: <https://github.com/pydeveloperashish/Facial-Expressions-Recognition>
78. Papers with Code - FER2013 Benchmark (Facial Expression Recognition) [Internet]. Paperswithcode.com. 2013 [cited 2020 Dec 16]. Available from: <https://paperswithcode.com/sota/facial-expression-recognition-on-fer2013>
79. neha01. neha01/Realtime-Emotion-Detection [Internet]. GitHub. 2020 [cited 2020 Dec 16]. Available from: <https://github.com/neha01/Realtime-Emotion-Detection>
80. What is Black Box Testing | Techniques & Examples | Imperva [Internet]. Learning Center. 2020 [cited 2020 Dec 14]. Available from: <https://www.imperva.com/learn/application-security/black-box-testing/>
81. White Box Testing: A Complete Guide with Techniques, Examples, & Tools [Internet]. Softwaretestinghelp.com. 2015 [cited 2020 Dec 14]. Available from: <https://www.softwaretestinghelp.com/white-box-testing-techniques-with-example/>
82. Rungta K. What is Grey Box Testing? Techniques, Example [Internet]. Guru99.com. Guru99; 2020 [cited 2020 Dec 14]. Available from: <https://www.guru99.com/grey-box-testing.html>

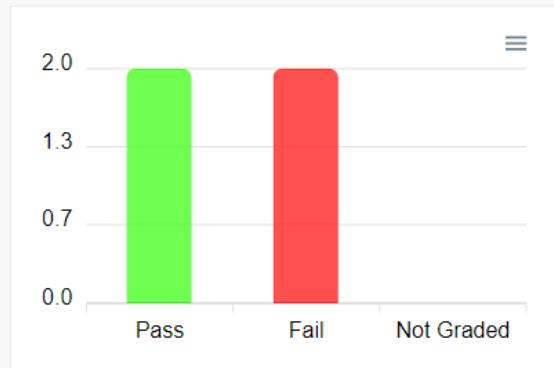
83. 10 Usability Heuristics for User Interface Design [Internet]. Nielsen Norman Group. 2020 [cited 2020 Dec 14]. Available from: <https://www.nngroup.com/articles/ten-usability-heuristics/>

Appendix

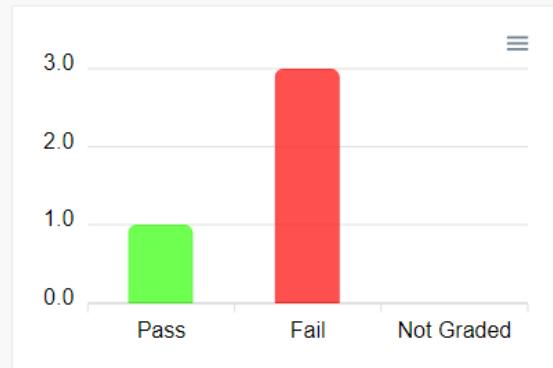
Appendix 1: Blinkee Usability Study Data

Individual Task Performance

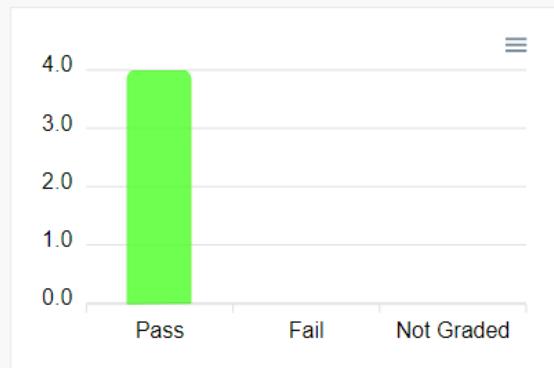
[2] Find necklace glowsticks for a Patrick's day party.



[6] Browse Easter items

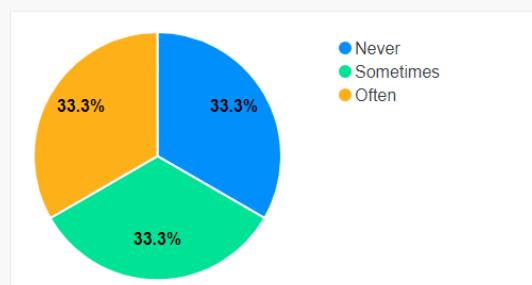


[9] View items in cart

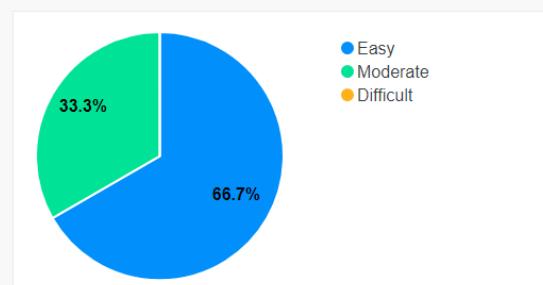


Multiple-Choice Question Answers

[1] Question: *How often do you use online shopping websites?*



[11] Question: *Please select the difficulty of finding the items in cart page*



Text Question Answers

[3] Question: *When searching for the Patrick's day items where did you search first?*

Answers:

Quick Recommendations on home page

Logo on website of St.Patrick's day

Below the banner of the webpage

[4] Question: *Have you encountered any problems during the task?*

Answers:

No, but it was really slow and I wasn't sure if things were working.

When typing into the search bar it didn't come up as suggested when I wrote in short

The text box of the application was in front of the "items per page" selection option

[5] Question: *Did you feel any particular emotion throughout the duration of the task? Please state which ones.*

Answers:

I was kind of confused by a lot of things as stuff wasn't in obvious places. I would be angry if I couldn't find the thing I wanted.

Curious of where the item was, concentrated

Happy when I couldn't find the select option

[7] Question: Are you satisfied with the navigation to the Easter items page? Please explain what you like or dislike.

Answers:

No, the quick link on the homepage didnt work and i had to search manually for what i wanted.

No, page crashed when I chose the easter logo to find my item.

It was awful, the easter page didnt load I had to search for the easter page to find it

[8] Question: Did you feel any particular emotion throughout the duration of the task? Please state which ones.

Answers:

I was frustrated and angry when it didnt work and i had to find a different way to search.

Disappointment that the page crashed

Frustration that I couldnt load the webpage

[10] Question: When searching for the cart page where did you first expect it to be?

Answers:

It the very top right or in a popout window on the right side.

Top right corner of the website

[12] Question: Did you feel any particular emotion throughout the duration of the task? Please state.

Answers:

It was frustrating that it wasn't where I expected but it only took a second to figure out so it wasn't too bad.

Curiosity and disappointed

The cart was not updated with the correct amount of items until clicked

[13] Question: In the entire application are there any features you would change or features you dislike?

Answers:

I would have sections like Easter more clearly labelled with regular font because the text is hard to read. The logo animation is also quite distracting and all the colours are very harsh and look bad.

Cleaner logos, they are very bright, in a way that's not bad but I prefer a cleaner aesthetic. Too many bright colours distracted me and I couldn't focus on what I wanted to find

To much flashing of colours, the webpage is laggy

[14] Question: Which feature did you think was well designed?

Answers:

The quick sections for certain holidays is nice to have on the homescreen at the very beginning.

search bar

N/A

[15] Question: Please describe the emotion, if any, that you felt throughout the usability study.

Answers:

Confused and frustrated

Distracted because of the bright colours

Frustration and laughter about how awful this webpage is

Appendix 2: FER Model 1 – Testing & Evaluation With Participants

Participant 1 Test (Name Initials: RB)

Facial Expression	Predicted Emotion	Pass/Fail
Neutral	Neutral	Pass
Happy (Smile)	Happy	Pass
Happy (Grin)	Happy	Pass
Sad	Sad	Pass
Angry	Angry	Pass
Fear	Surprised	Fail
Fear (Exaggerated)	Fear	Pass
Surprised	Surprised	Pass
Disgusted	Sad	Fail

Observations:

1. Sad is only predicted with when the eyebrows are involved, the mouth does not need to be in a frowning state.
2. The user closing their eyes while having a neutral face predicted Sad.

Question:

1. How well has the model predicted your facial expressions, generally speaking?

Answer: *It predicted well; some facial expressions were difficult to show such as fear. Doing emotions without a reason felt "forceful"*

2. Which expression do you feel was best predicted?

Answer: *Happy and Sad were predicted very well, Angry and Neutral too.*

3. Which expression do you feel was worst predicted?

Answer: *Disgusted didn't display correctly at all and Fear was difficult to express.*

Participant Comments:

1. With regards to the model predicting sad only when the eyebrows were frowning: "*This might be a good thing because when people fake emotions they tend to use their mouth and not their eyebrows*".

Participant 2 Test (Name Initials: SC)

Facial Expression	Predicted Emotion	Pass/Fail
Neutral	Neutral	Pass
Happy (Smile)	Happy	Pass
Happy (Grin)	Happy	Pass
Sad	Neutral	Fail
Angry	Angry/Sad	Pass
Fear	Fear	Pass
Fear (Exaggerated)	Fear	Pass
Surprised	Surprised	Pass
Disgusted	Sad	Fail

Observations:

1. The user closing their eyes did not cause issues.
2. The participant needed to change their background. They had a curtain with shapes that were picked up as faces by the Haar Cascade algorithm.

Question:

1. How well has the model predicted your facial expressions, generally speaking?

Answer: *Sad has not been predicted well, however, happy, and neutral worked very well. Fear and Surprise was okay.*

2. Which expression do you feel was best predicted?

Answer: *Happy, Neutral and Angry were predicted well.*

3. Which expression do you feel was worst predicted?

Answer: *Sad hardly displayed and neither did disgusted.*

Participant Comments:

1. *The camera being at the top has worked better than the camera being at the bottom.*

Participant 3 Test (Name Initials: DS)

Facial Expression	Predicted Emotion	Pass/Fail
Neutral	Neutral	Pass
Happy (Smile)	Happy	Pass
Happy (Grin)	Happy	Pass
Sad	Sad	Pass
Angry	Angry	Pass
Fear	Fear	Pass
Fear (Exaggerated)	Fear	Pass
Surprised	Surprised	Pass
Disgusted	Sad/Angry	Fail

Observations:

1. The user closing their eyes did not cause issues.
2. Participant had photos in the background, these were picked up on by the algorithm

Question:

1. How well has the model predicted your facial expressions, generally speaking?

Answer: *It predicted very well, it just picked up on the background photos*

2. Which expression do you feel was best predicted?

Answer: *All of them were well predicted, except disgusted*

3. Which expression do you feel was worst predicted?

Answer: *Disgusted was not predicted at all*

Participant 4 Test (Name Initials: EW)

Facial Expression	Predicted Emotion	Pass/Fail
Neutral	Neutral	Pass
Happy (Smile)	Happy	Pass
Happy (Grin)	Happy	Pass
Sad	Sad	Pass
Angry	Sad	Fail
Fear	Happy	Fail
Fear (Exaggerated)	Happy	Fail
Surprised	Surprised	Pass
Disgusted	Sad/Angry	Fail

Question:

- How well has the model predicted your facial expressions, generally speaking?

Answer: *Predicted okay when it came to happy and sad. It thought angry was sad and making a fearful facial expression was difficult*

- Which expression do you feel was best predicted?

Answer: *Happy was predicted best*

- Which expression do you feel was worst predicted?

Answer: *Disgusted and Fearful was not predicted at all*