UsabCheck – A Usability Testing Suite With Facial Expression Recognition

DT228
BSc in Computer Science

Damian Wojtowicz

C17413722

Andrea Curley

School of Computer Science

Technological University, Dublin

05/04/2021

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.

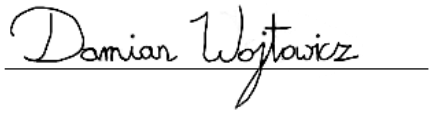
UsabCheck aims to provide a usability testing suite which can be used to conduct usability studies and to investigate the use of facial expression recognition technology to improve software usability testing and user experience testing. The project involves two applications, one of which is used by the 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.

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

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

1.1. Overview

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 application 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 complete the tasks efficiently and effectively 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 in a research paper by Bastien *et. al* (2003) [5] indicated that the first 5 participants only uncovered 35% of the usability problems.

1.2. Project Description

The purpose of this project is to provide a software usability testing suite which can be used to conduct usability studies and view the results of the study as well as investigate if usability testing can be improved upon with the use of machine learning and more specifically facial expression recognition.

UsabCheck is usability testing suite which consists of two applications. The first application is a web application that will be used by a researcher who can be a developer or designer of a website or other software. The researcher will access the *UsabCheck* web application and create a usability study consisting of tasks and questions to completed by the participant of the study. The second *UsabCheck* application will run locally on the machine the participant is using and it will display the tasks and questions for the participant to complete. The local application will also record the participants screen and collect data on the participant's facial expressions. After the usability test is conducted the video screen recording and other data will be uploaded to the web application where it is stored.

This data is then presented in the forms of bar charts, pie charts and lists in the web application for the researcher to analyse. Additionally, the researcher can view the video recording of the usability study along with the emotions which the participant has expressed via their facial expressions on a timeline located below the video. The timestamps of when the participant has started and finished a task is also displayed and this data can be used by the researcher to identify the problem areas in the software more easily.

A significant part of this project was to train and evaluate deep learning models which recognise facial expressions. The processes involved researching and selected the appropriate models and

datasets and evaluating the trained models with metrics such as precision, recall, accuracy, and f-score. The most successful model was further evaluated with data from a live camera feed and tested rigorously to discover the effects of face occlusion, facial features, and camera position. Data on the performance of the model was collected and the strengths and weaknesses of the model were identified. Additionally, usability studies were conducted and heuristic evaluation was used to test and evaluate the systems and sub-systems in this project.

1.3. Project Aims and Objectives

As mentioned above, the project objective is to create a complete usabilty testing suite and implement a machine learning model for facial expression recognition. To accomplish the final objective, more specific aims and objectives had to be achieved and these are outlined below.

Web Application

1. Create a usability testing web application that allows the researchers to create usability tests. This involves a web application frontend, backend, and database.
2. Allow the researchers to view the results of the tests on the website. This includes viewing of the videos, answers to questions, viewing the participant's progression, the participant's screen recording and the participant's facial expressions.
3. The results of the study should be presented in a clear manner. This includes charts, graphs, and widgets.

Local Application

4. Provide a local (runs on the machine the participant is using) usability testing application that will record the screen, record the participants face with a camera, predict the participants facial expression in real-time, show the participant the test instructions, ask the participants questions and upload the data to the cloud.
5. The videos need to be uploaded to a secure storage location and embedded on the website.
6. Evaluate the web application and local application with participants and heuristics.

Machine Learning

7. Research, implement, and evaluate the machine learning algorithms/models and datasets for the facial expression recognition side of the project.
8. Data pre-processing, cleaning, and loading
9. Determining the best dataset for the face expression recognition which will be a very important factor in the success of the classification.
10. Conduct comprehensive evaluation of machine learning models and datasets by using various metrics such as accuracy, recall, precision, and f-score. The model needs to be tested with participants in a usability testing environment.
11. The result data needs to be displayed in a clear and easy to understand manner in the forms of graphs, charts, and tables.

Project Scope

The scope of this project involves designing, implementing, testing, and evaluating a usability testing suite with facial expression recognition for desktop machines. The operating system which the testing suite is primary designed for is Windows and the focus will be on website usability testing. However, in theory any software can be tested with the UsabCheck testing suite. The software suite is not be designed and not intended for usability testing on mobile devices. The aim is to investigate how facial expression recognition can be applied to usability testing and how the data, which includes emotion data from a conducted usability study can be presented for analysis.

1.4. Thesis Roadmap

Chapter 2: Research

This section of the document will discuss the research conducted in the areas of emotions in usability testing, facial expression recognition, machine learning, usability testing, and software development. Firstly, the applicability of emotions in usability testing is explored, followed by research in the area of facial expression recognition. Next, usability testing research is presented, followed by research in the area of machine learning in relation to the metrics used in evaluation, and the various algorithms. The facial expression recognition datasets, GDPR, and existing usability testing application are discussed next. The technologies such as programming languages, libraries and frameworks are discussed after that and finally the previous final year project are reviewed.

Chapter 3: UsabCheck Design

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

Chapter 4: UsabCheck Development

The development chapter is split into three sections. These are the web application, local application, and facial expression recognition model. The chapter begins with an overview followed by the development of the web application which consist of the backend and frontend. The local application which includes the integration of the FER model is discussed next and finally, the machine learning aspect of the project which involves two FER models is discussed.

Chapter 5: Testing and Evaluation

The testing and evaluation chapter is split into two sections. Firstly, the testing section is discussed and this section is further split into the testing of the web application and local application. The machine learning model testing is under the evaluation section along with the evaluation of the models. The testing of the web app and local app includes system testing and automated unit testing. The second section of the chapter is evaluation and it is similarly split into the same three sections of web application evaluation, local application evaluation and FER model evaluation. The evaluation methods include expert evaluation, Nielsen's heuristics, impaired vision simulation, and usability testing with participants. The testing and evaluation of the machine learning models includes evaluation with metrics such as accuracy, evaluation with participants, occlusion testing, facial feature testing, testing model on multiple datasets and live camera feed evaluation.

Chapter 6: Conclusions and Future Work

This chapter begins with the main conclusions which give on overview of the completed project and answer the research questions. Following that a conclusion for each section of the report is given. The risks and challenges are discussed next and finally, the chapter finishes with improvements and future work.

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.

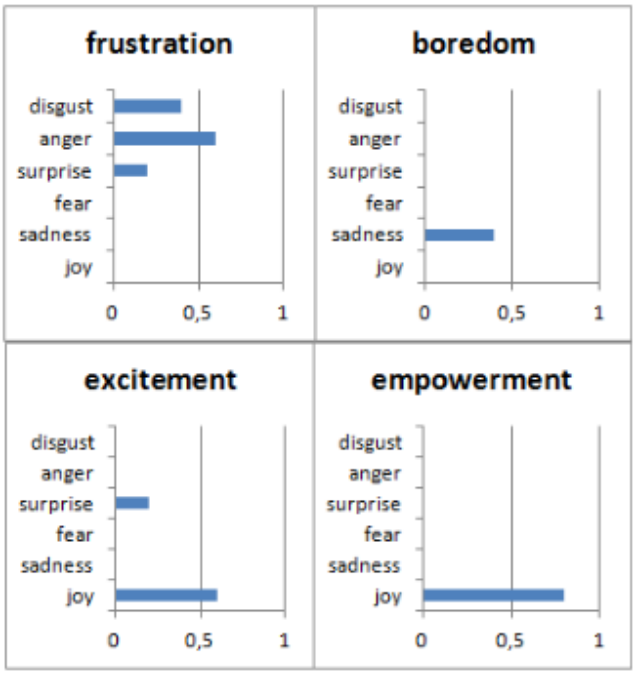


Figure 1 – Ekman's Six Basic Emotion Model

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.

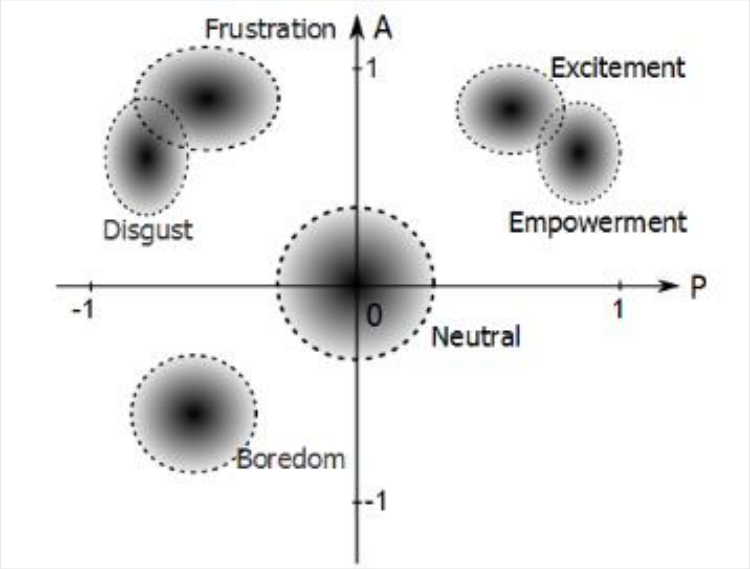


Figure 2 – Whissel's Wheel Emotion Model

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 valence of 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 vertical 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.

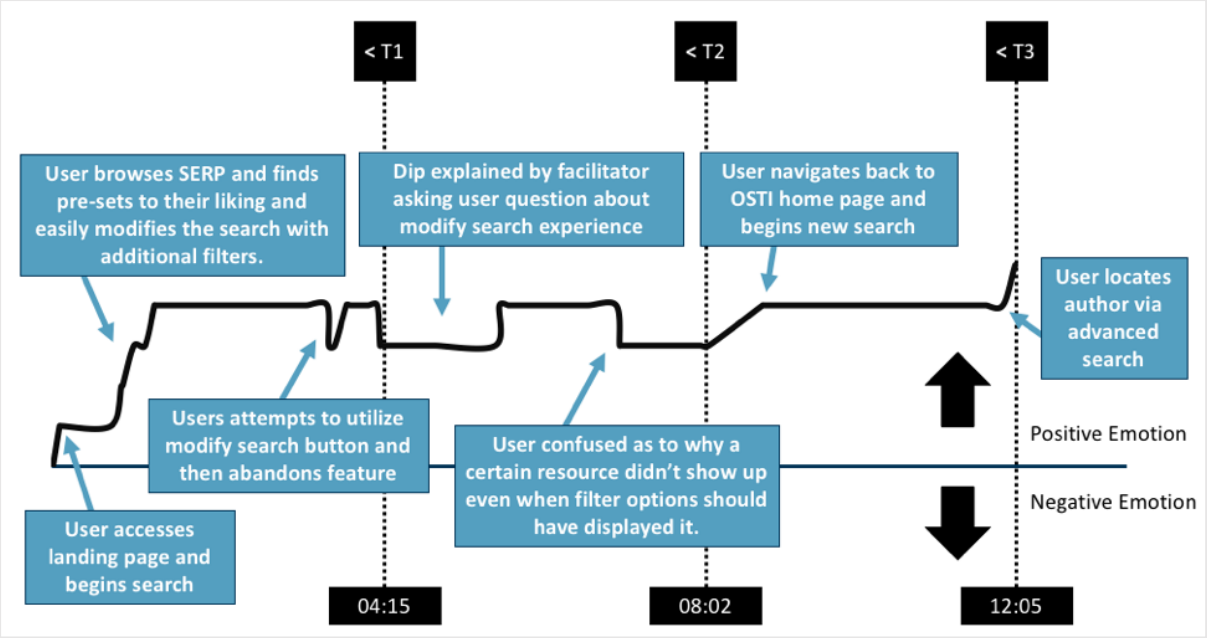


Figure 3 - Example Journey Map

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 not always clear.

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) [8] "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) [9] 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 combing 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.

Gaurav (2018) [10] 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*. 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.

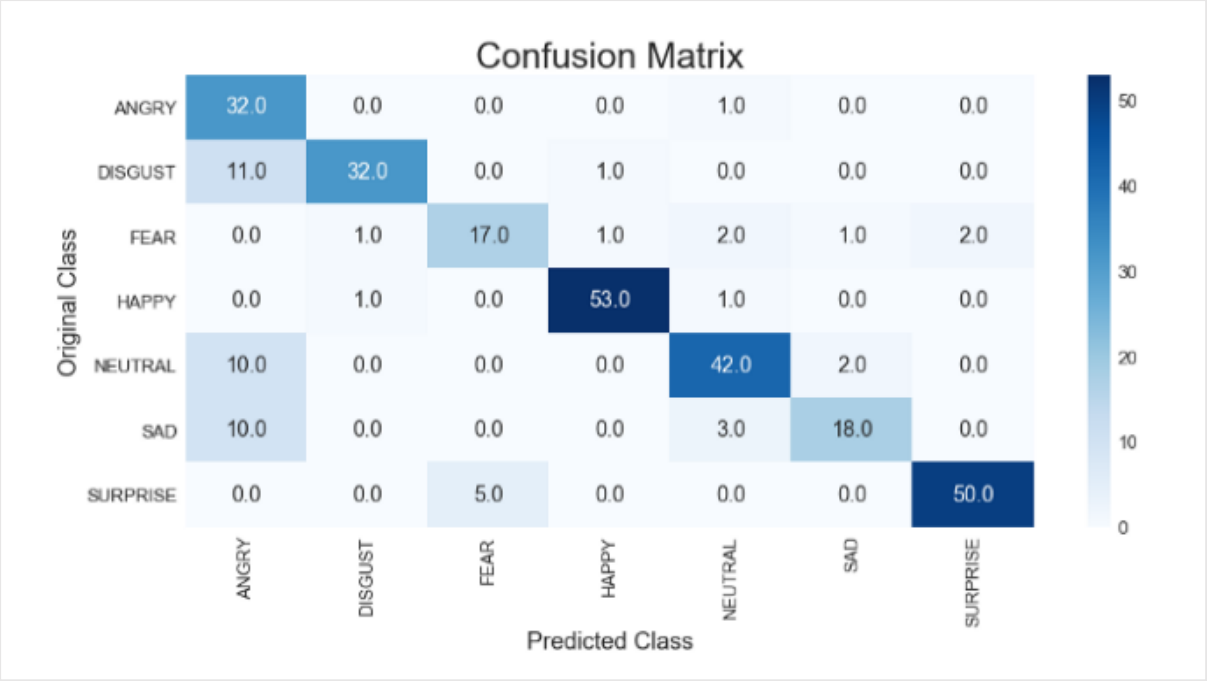


Figure 4 - Confusion Matrix Example

2.1.3. Usability Testing Research

Bastien (2009) [11] 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, making this research still 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 participant'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.

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) [12] 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".

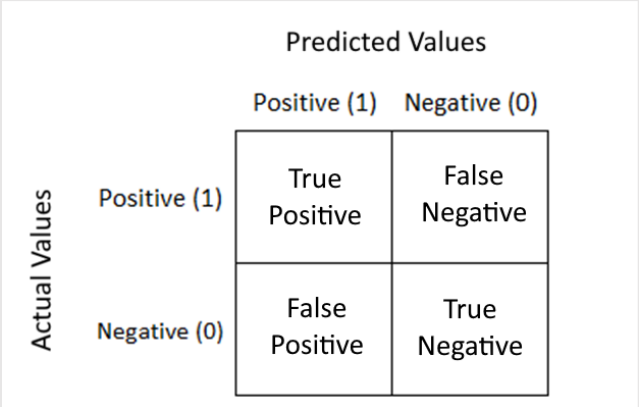


Figure 5 – Confusion Matrix Labels

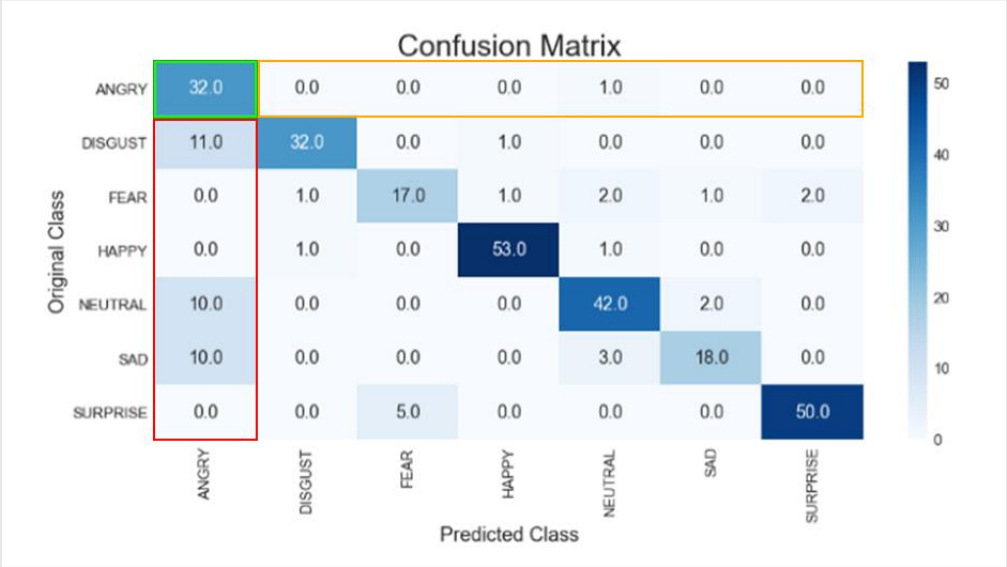


Figure 6 – Confusion Matrix

The article by Koehrsen (2018) [13] 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.

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

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

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

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

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

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

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

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 [14] and face recognition [15], 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% [16] which shows how effective CNNs can be 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 [17]. The fully connected layer is the final layer and uses the results from the previous layer to classify an image [18][19].

Visual Geometry Group (VGG) Model

The VGG-16 Model [20] is a famous CNN model that achieved a 92.7% test accuracy on the ImageNet dataset [21], making it the winner of the ILSVRC (ImageNet Large Scale Visual Recognition Competition) in 2014 [22]. The dataset the model was trained on consisted of 14 million labelled images belonging to 1000 classes [23]. This model can be used to train a facial expression recognition model as shown by Kusuma *et. al* 2020 [24]. 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.

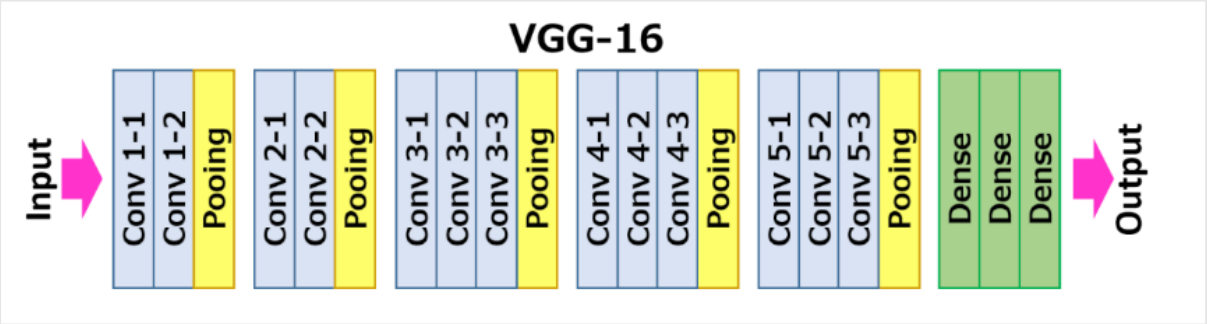


Figure 7 – VGG-16 Architecture

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 [25].

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 [26][27].

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 BOWV, 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 [28][29].

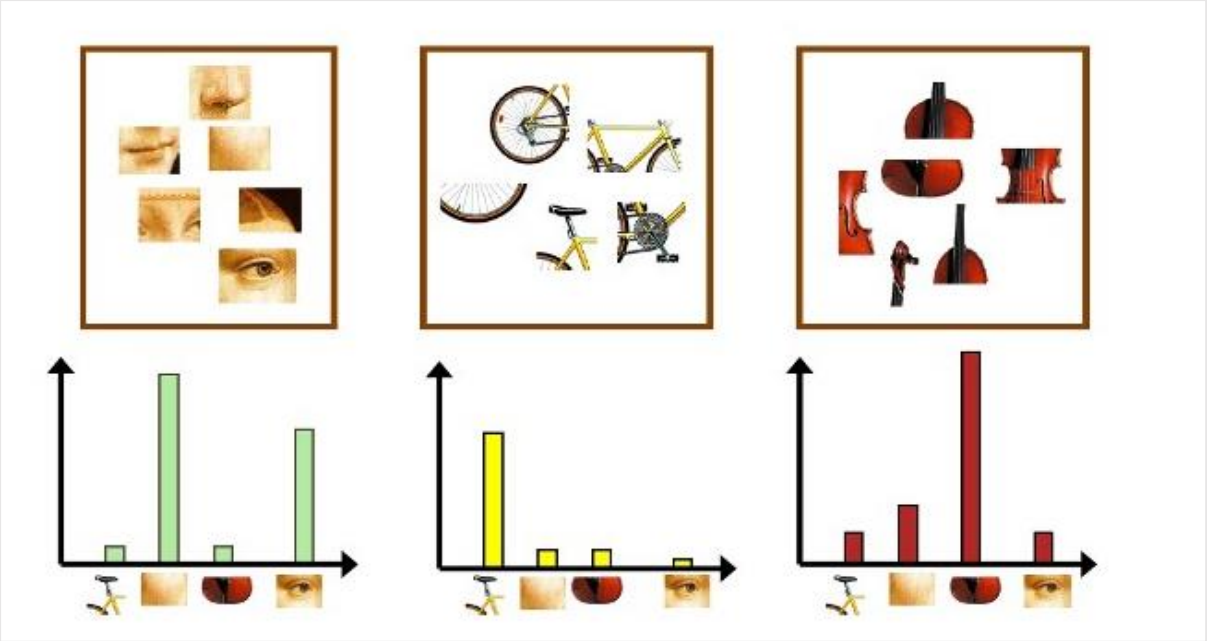


Figure 8 – Bag of Visual Words Histogram Example

Face Detection

Several methods for face detection are discussed in the article by Dwivedi (2018) [30]. 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 [31] 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 [32] 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 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 [33] 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

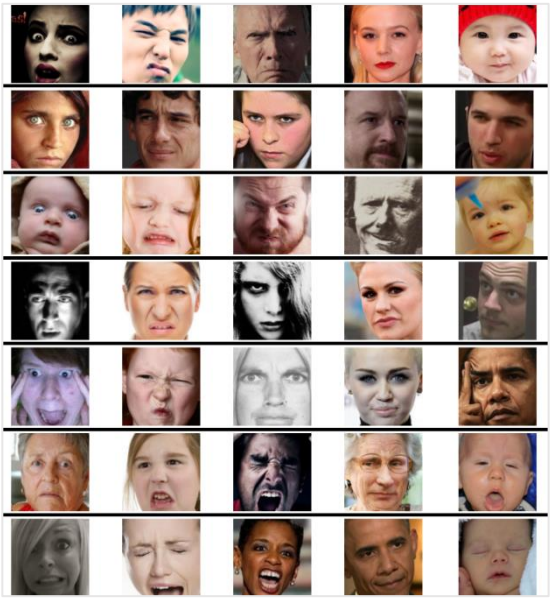


Figure 10 – AffectNet Sample Images

FacesDB

The FacesDB dataset [34] consists of 38 individuals who are both male and female and each individual makes one of each facial 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) [35] 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 and 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 need be downloaded and cropped using the bounding rectangle coordinates. The process by hand is simply unfeasible and a GitHub repository under the name "FEC Dataset Downloader" [36] 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" [37] 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 odd angles in many different environment. 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 [38] 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 suitable 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 [39] 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 functionality. 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) [40] 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 [41] 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 participant 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.

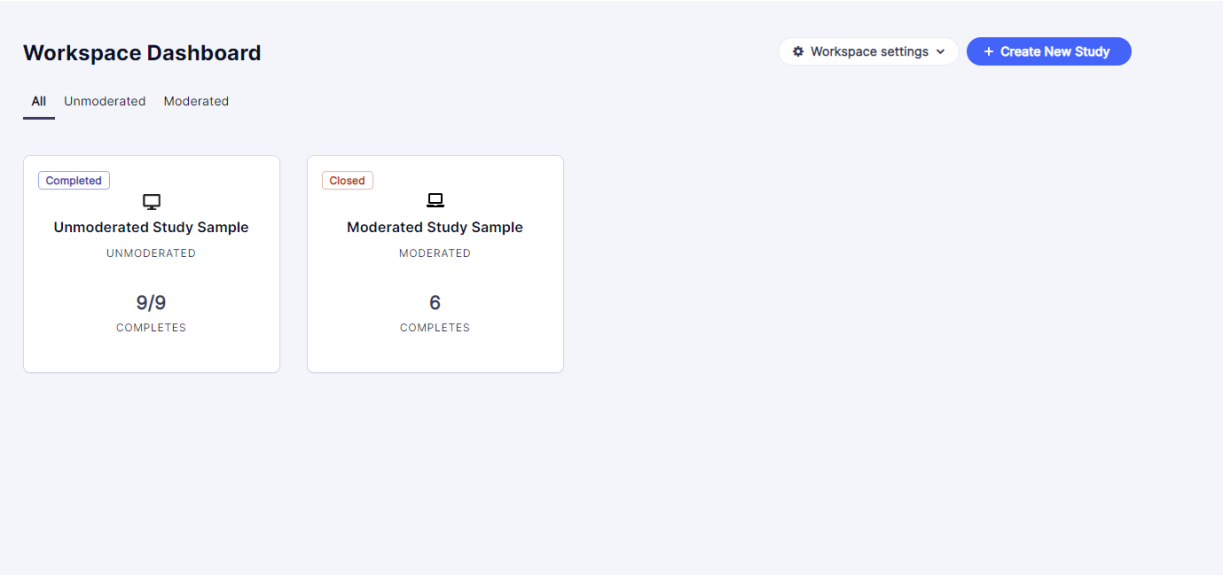


Figure 13 – UserZoom Go Dashboard

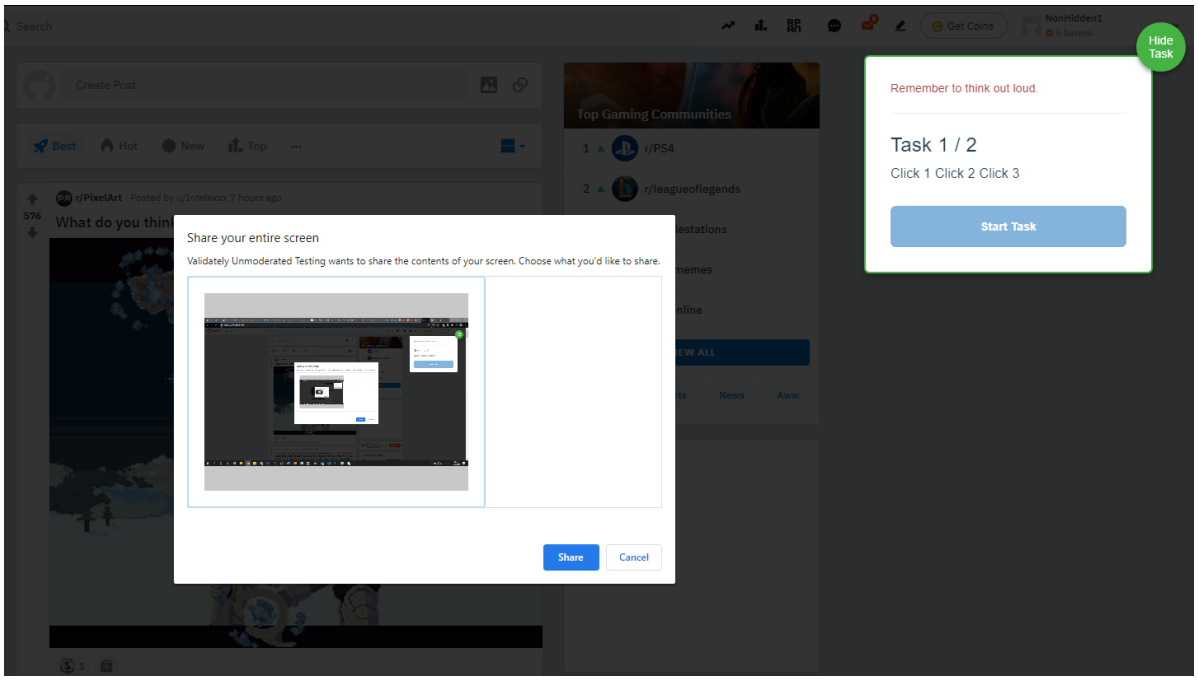


Figure 14 – UserZoom Go Usability Test Start

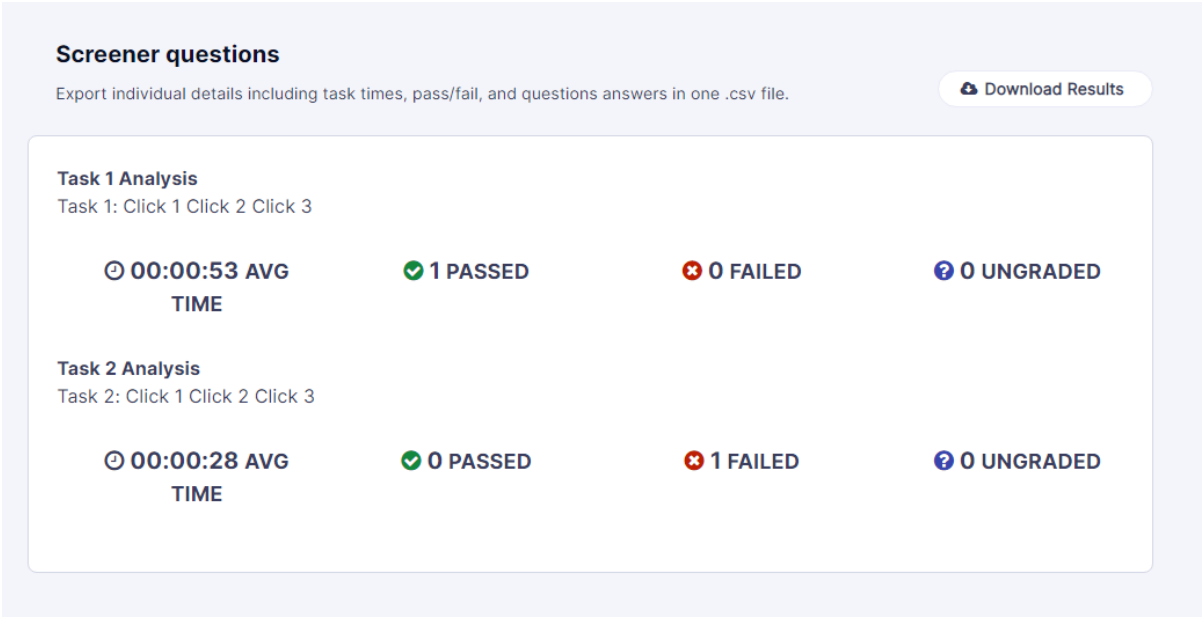


Figure 15 – UserZoom Go Usability Study Results

2.2.2. UserTesting

UserTesting [42] 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 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 which caused confusion.

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 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. 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.

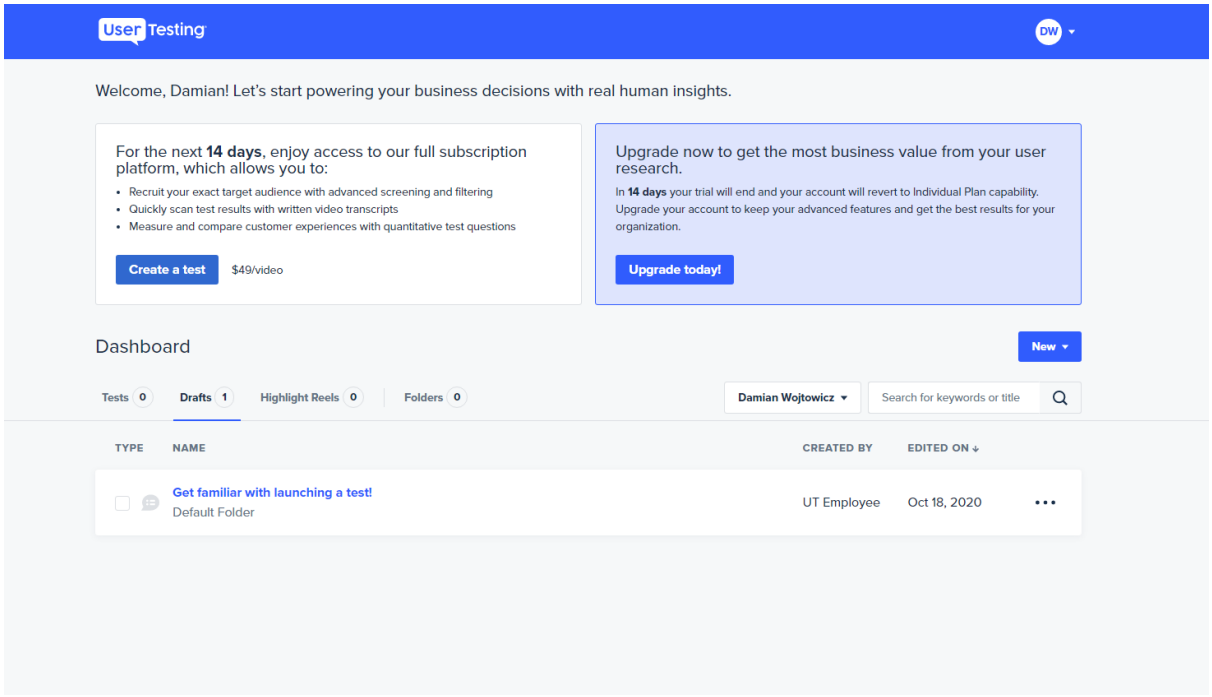


Figure 16 – UserTesting Dashboard

The test creation options included selecting the audience (See Figure 17) by demographic such as age, gender, income, country, employee status etc.

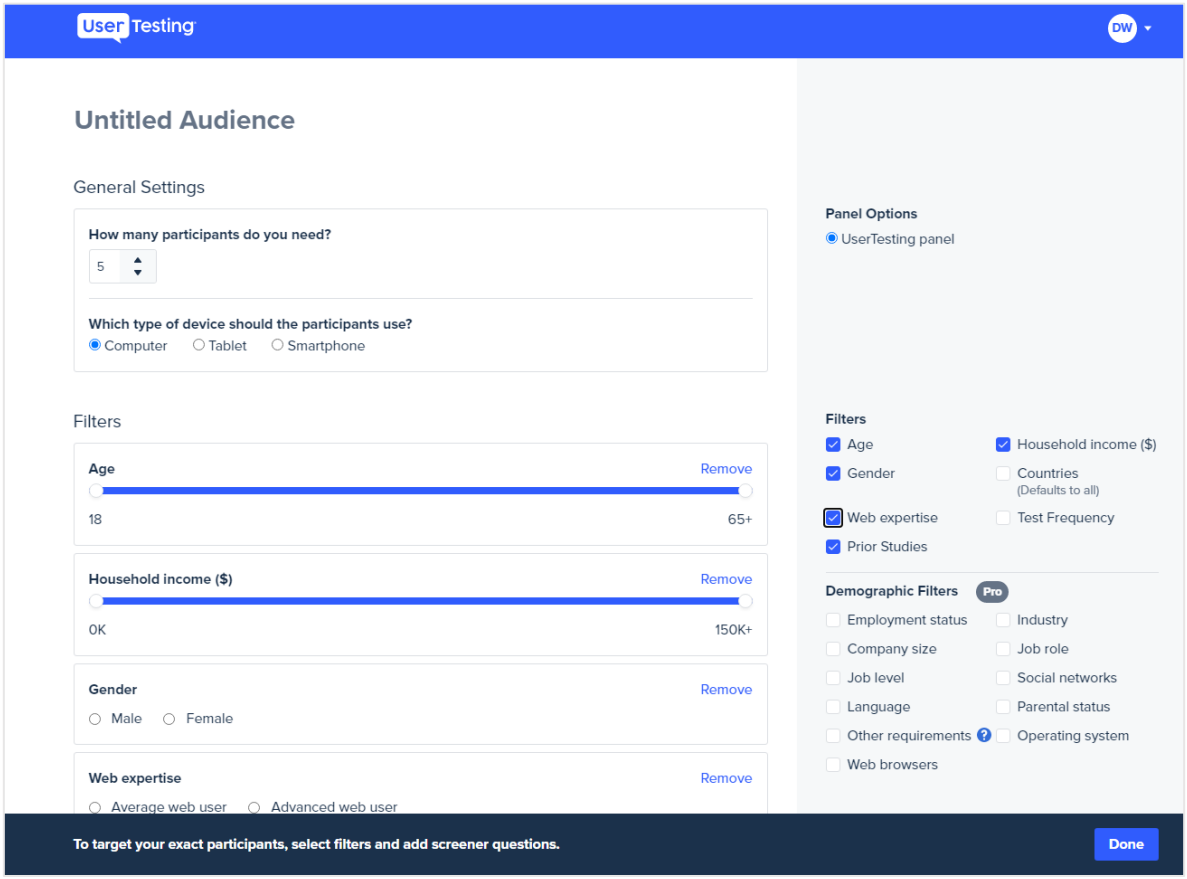


Figure 17 – UserTesting Select Audience

The test creation screen is illustrated in Figure 18.

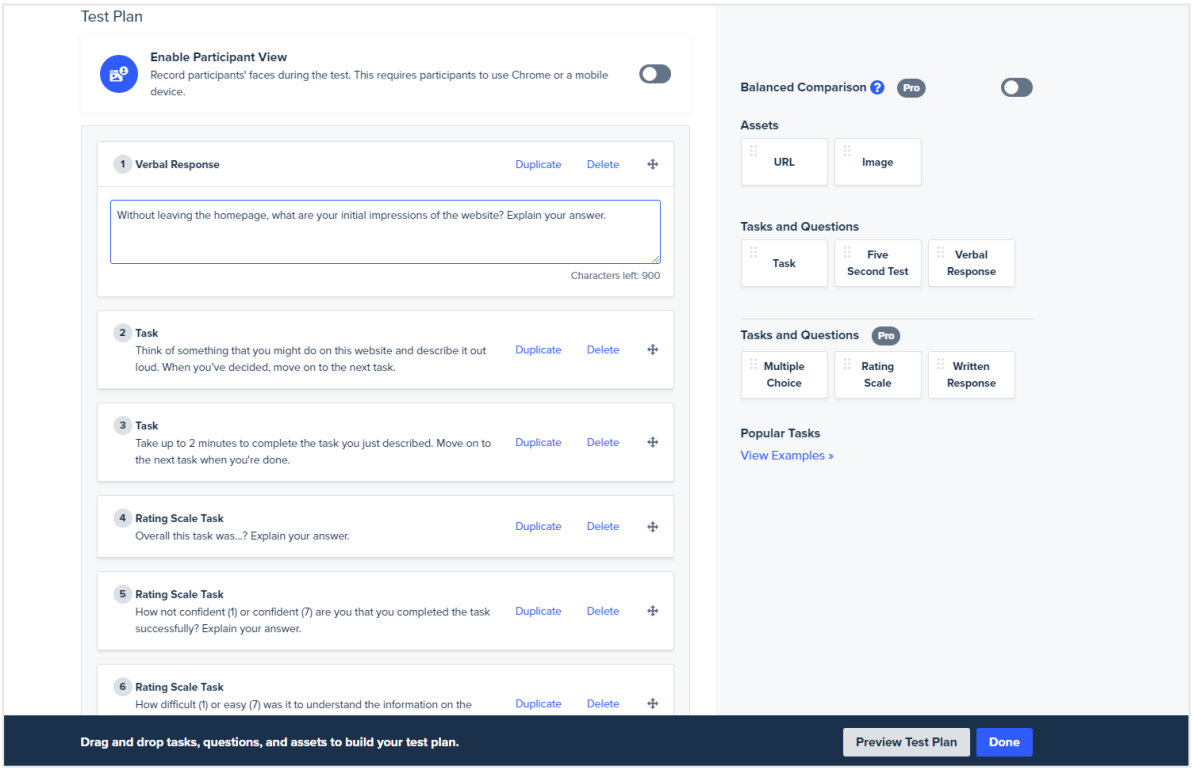


Figure 18 – UserTesting Create Usability Test

2.2.3. Loop11

Loop11 [43] 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 clicksteam & 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.

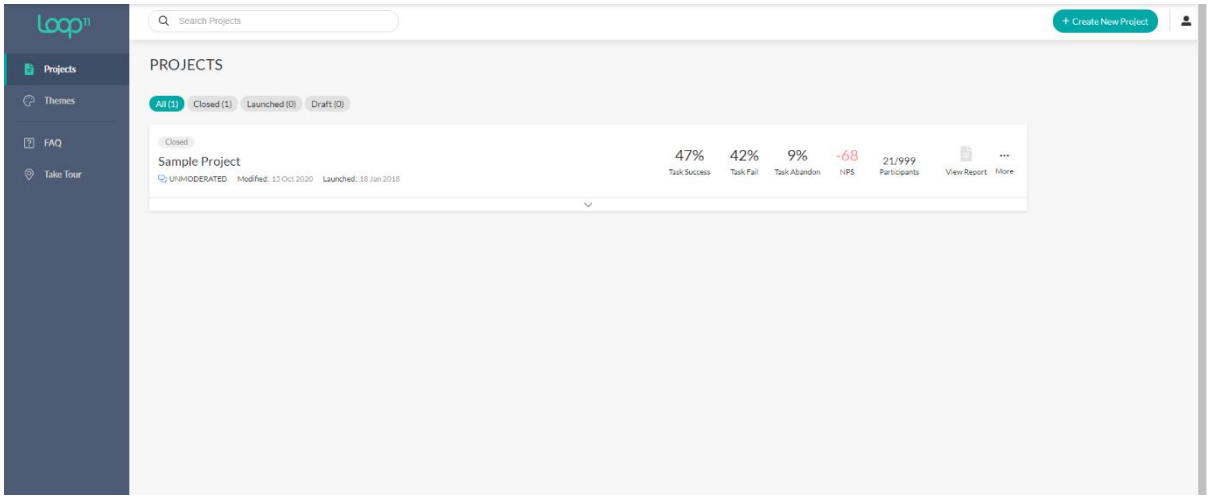


Figure 19 – Loop11 Main Dashboard

Usability study results page is shown in Figure 20.

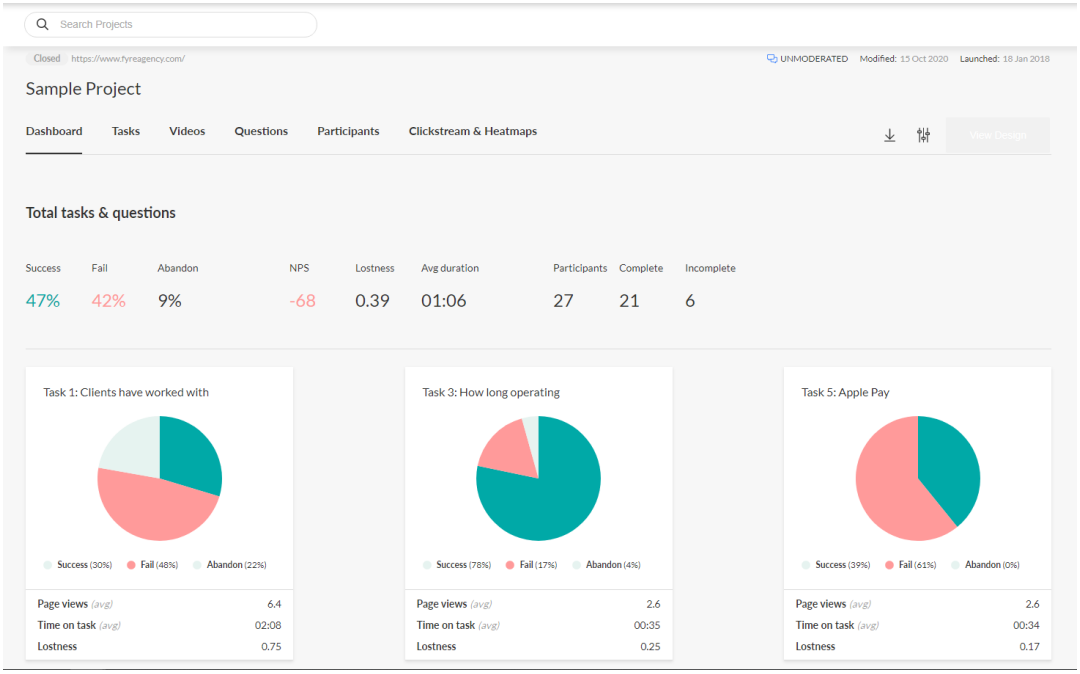


Figure 20 – Loop11 Project Dashboard

The participant information such as their demographic, operating system, device, and browser is displayed in Figure 21.

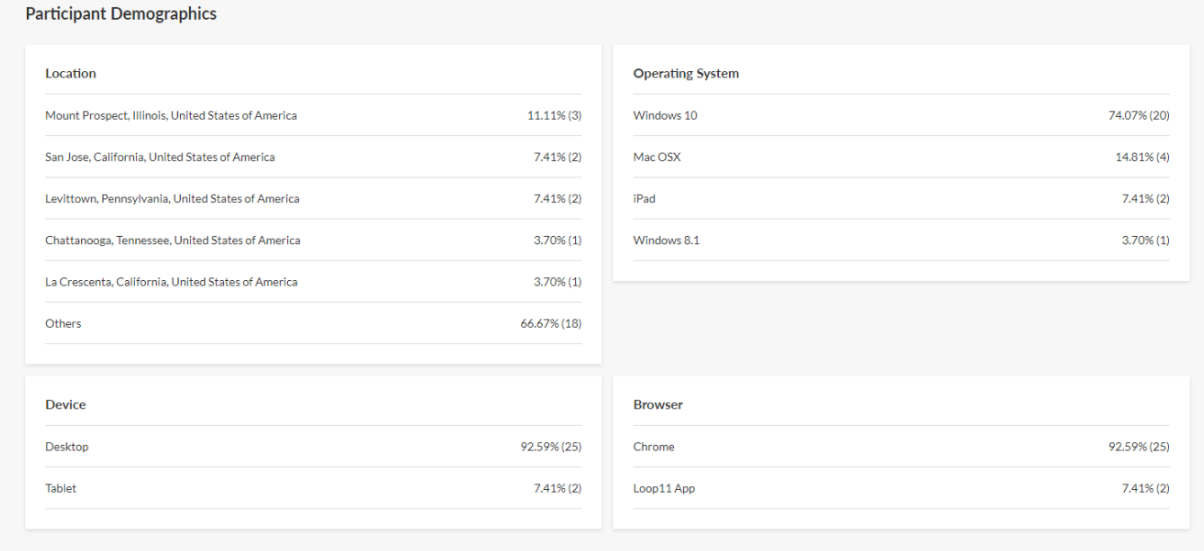


Figure 21 – Loop11 Participant Demographic

The types of questions that can be asked of the participant are displayed in Figure 22.

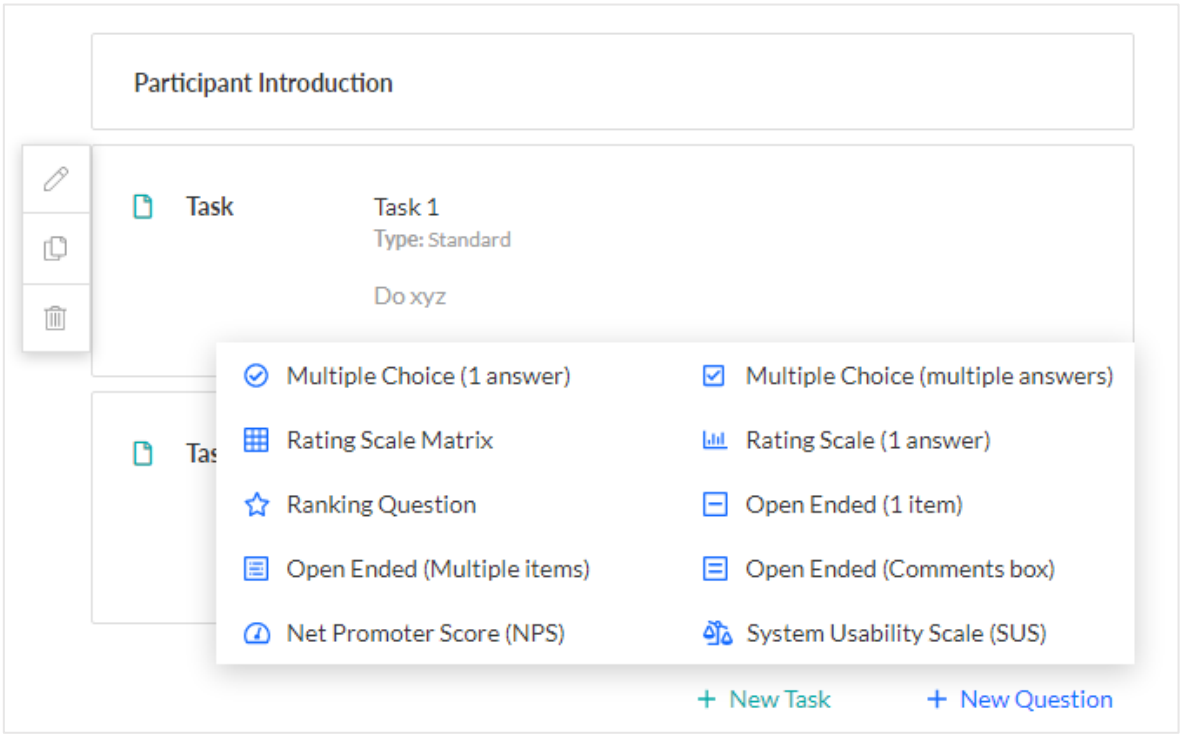


Figure 22 – Loop11 Tasks and Questions

The clickstream that shows how the participants have navigated from page to page is shown in Figure 23.

Figure 23 – Loop11 Clickstream

2.3. Technologies Research

2.3.1. Programming Languages

Python

Python [44] 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 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 [45]. Python can also be used to create applications that run locally on the machine with libraries such as PyQt5.

Django [46] 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 [47] 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 [48]. 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 [49] 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 backend and frontend framework named Node.js [50] that can be used in the development. There are also frontend frameworks and libraries such as AngularJS [51] and ReactJS [52] in the scenario that the design plan will be to implement a Java backend instead of a JavaScript backend.

2.3.2. Python Libraries

NumPy [53] 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 [54] provides functionality that can extract data out of a CSV file as an example and create data frames that are similar in structure to spreadsheet file in Python. This library can work with NumPy to manipulate the data.

Matplotlib [55] 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. This library can be

used to display images in Juypter 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.

Scikit-learn [56] 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 [57] 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 classification [58].

Keras API [59] 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 [60] 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 [61] 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 [62] 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) [9] and discussed in the research section.

2.4. Usability Testing Technologies

2.4.1. Usability Testing

The idea for the usability testing suite 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 need to be implemented and this may include a 3rd party service. The website has to be hosted and a database will need 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 flexible and easy to use method of uploading and streaming the videos reliably.

SwiftStack

SwiftStack [63] 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) [64] 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 [65] is an online video streaming site that allows users to upload their videos and these videos can be shared and embedded on websites. This service is often compared with YouTube and Vimeo has an API that can be used for uploading videos automatically.

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 [66] 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 [67] 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 [68].

Google Firebase

Google Firebase [69] 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, it should be noted that the database, the Firebase Realtime Database and Firestore are non-relational databases. Google Firebase is said to be a great option for rapid prototyping and development when building an application from scratch.

2.4.4. Databases

MySQL

MySQL [70] is an open-source relation database management that is well known for its high security, performance and on-demand scalability [71]. 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 is very fast and efficient, however, not all applications are suited to this type of storage and this will be further discussed in the design section.

2.5. Existing Final Year Projects

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 detect a bot account more accurately. 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. 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	3.6.12	Python will be used in the local application usability testing application and for training machine learning models. This is because Python offers a wide range of libraries for machine learning and GUI application development.
Java	11.0.3	Java will be used in the web application's backend. It is one of the most popular languages for a web server backend.
JavaScript	ES6	JavaScript will be used for the web application's frontend. It is compatible with the Java backend and very useful in frontend development.
ReactJS	17.0.1	ReactJS is a JavaScript library for the frontend 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 backend. It is a very useful tool for managing the libraries and dependencies which can be used to compile 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 backend of the web application. It is used to build decoupled systems.
NumPy	1.18.5	A python library for working with numbers and will be used in implementing the FER model.
Pandas	1.1.5	A python library for extracting data out of spreadsheets. Used in working with the dataset.
Matplotlib	3.3.4	A Python library for plotting charts. Used in the evaluation of the FER model.
Scikit-learn	0.24.1	A machine learning Python library. It is a combination of other libraries and can be used in evaluating the FER model.
TensorFlow	2.3.1	A high-level machine learning Python library. Will be used via the Keras API.
Keras	2.3.1	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	4.5.1.48	A Python library with functionality to manipulate images. It will be used when working with the FER dataset images.

MySQL	8.0.22	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 (Not Used in Production)	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. This dataset will be used for testing the FER model
FacesDB	Dataset	This dataset is of the front of the face. Once again, the size of the dataset is not large enough and this dataset will be used to test the FER model.
VGG-16	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 suitable 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 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 detail in this chapter.

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.

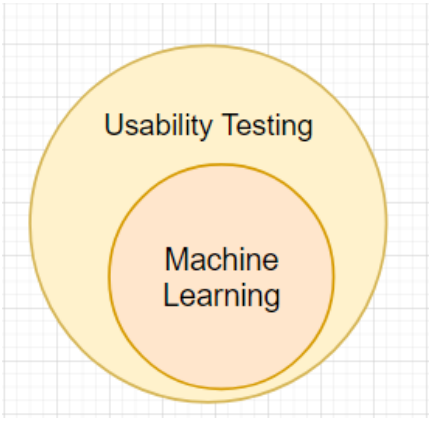


Figure 24 – Usability Testing & Machine Learning Subprojects

3.2. Software & Data Mining Methodologies

Due to the nature of the project consisting of two sub-projects 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 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 [72]. 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 (BPIs). This Product Backlog can change depending on the requirements as the project develops. These BPIs are used as input to the Sprint Backlog, which

is a list of tasks, user stories, bug fixes etc. which need to be completed by the end of the sprint. At the end of each sprint the team reviews and evaluates the work. The Scrum methodology also has "Increments" which is the usable end-product from a Sprint [73].

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 [74].

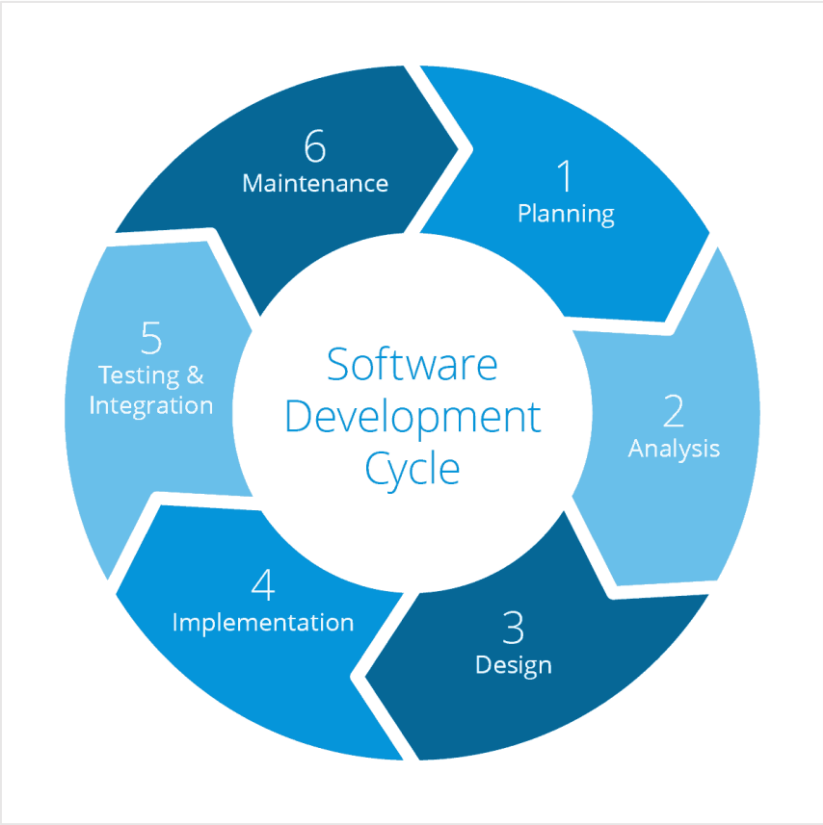


Figure 25 – Scrum Agile Software Development Cycle

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.

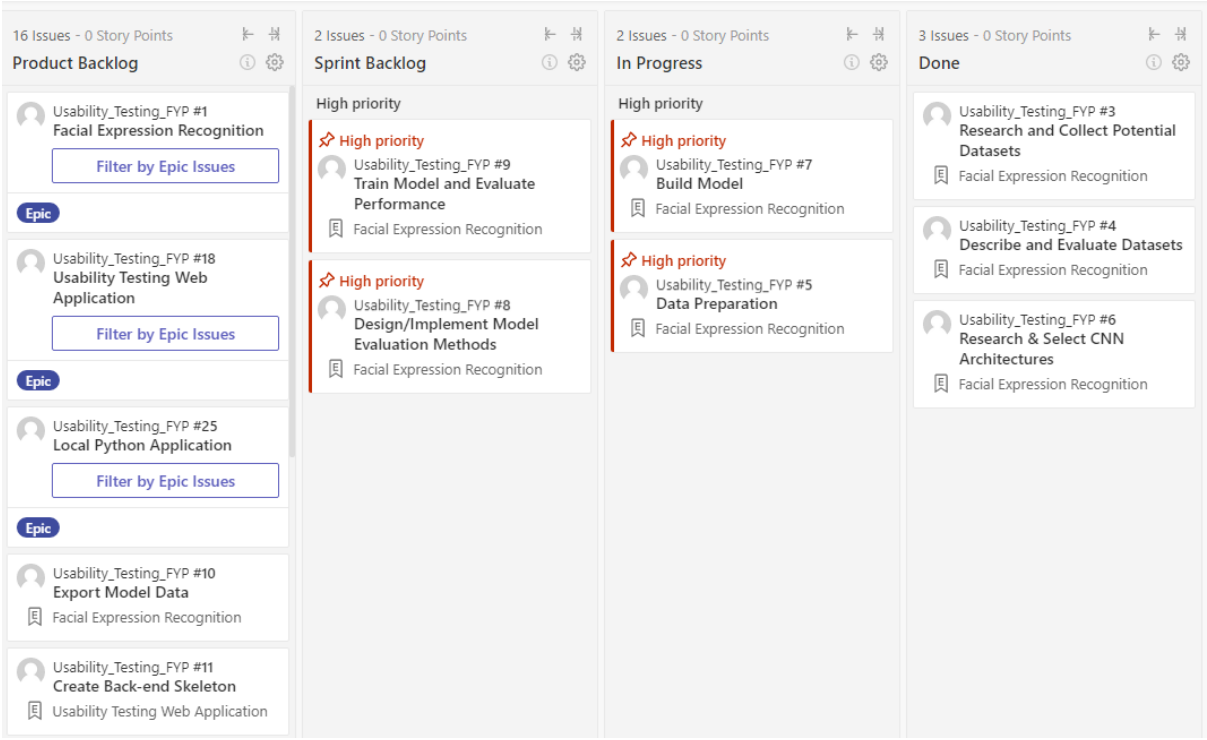


Figure 26 – ZenHub Epic and Sprint Planning (06/11/2020)

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.

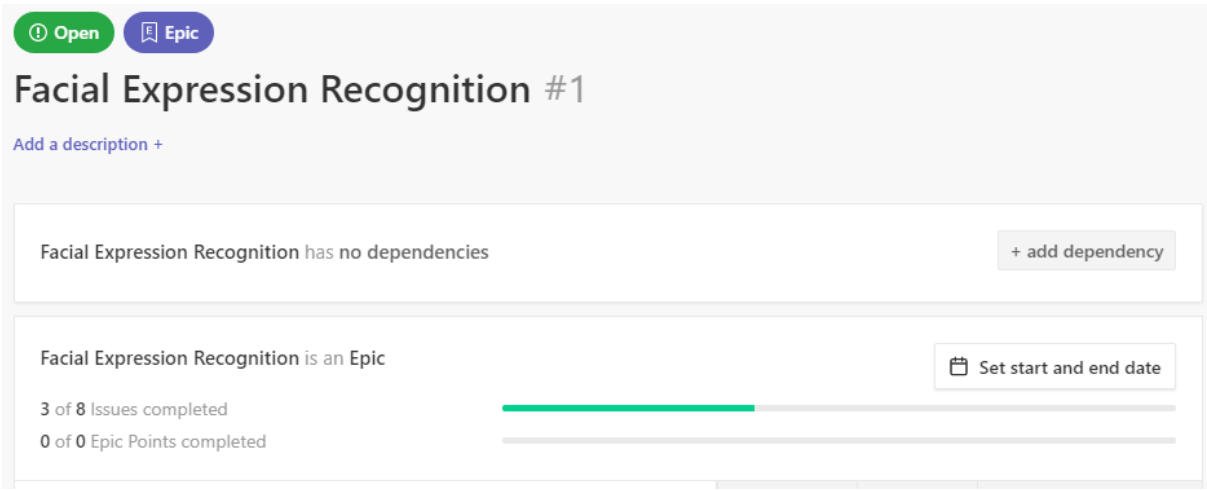


Figure 27 – ZenHub Facial Expression Recognition Epic

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 is focused on iterative progress. The cyclical iterative process consists of 6 stages which are illustrated in Figure 28 [75][76].

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 traditional 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.

The CRISP-DM methodology cycle diagram is displayed in Figure 28.

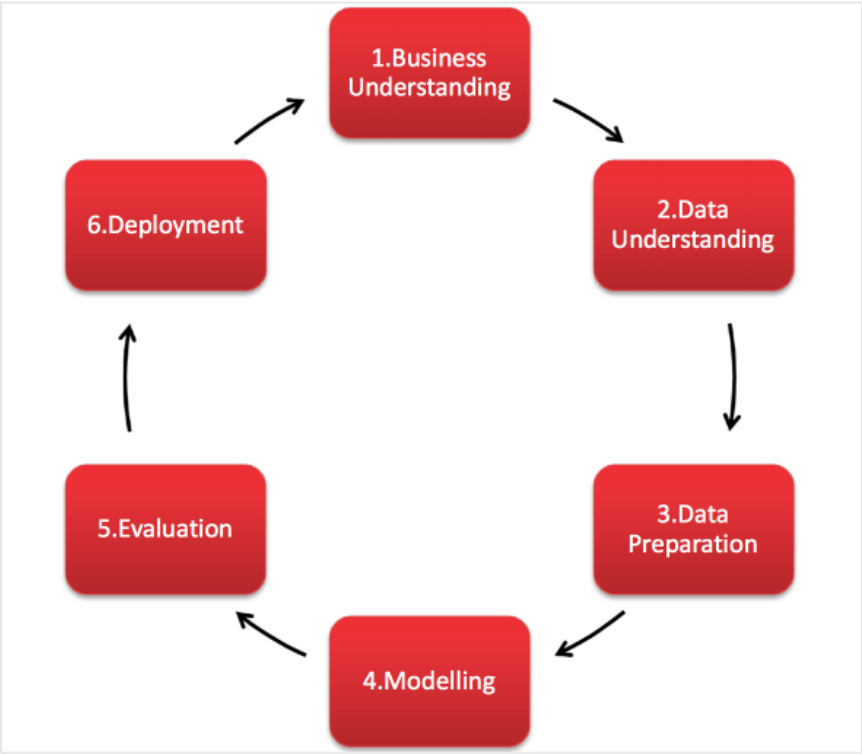


Figure 28 – CRISP-DM Methodology Cycle

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 30 and Figure 30 illustrates the high-level steps that are taken in each subproject, showing how each 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.

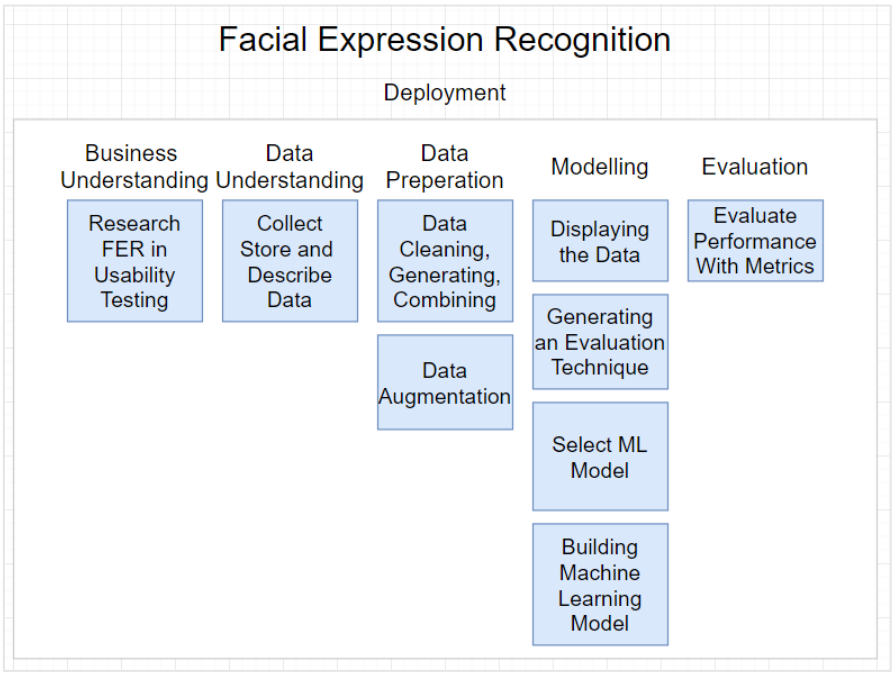


Figure 29 FER Subproject High Level Steps

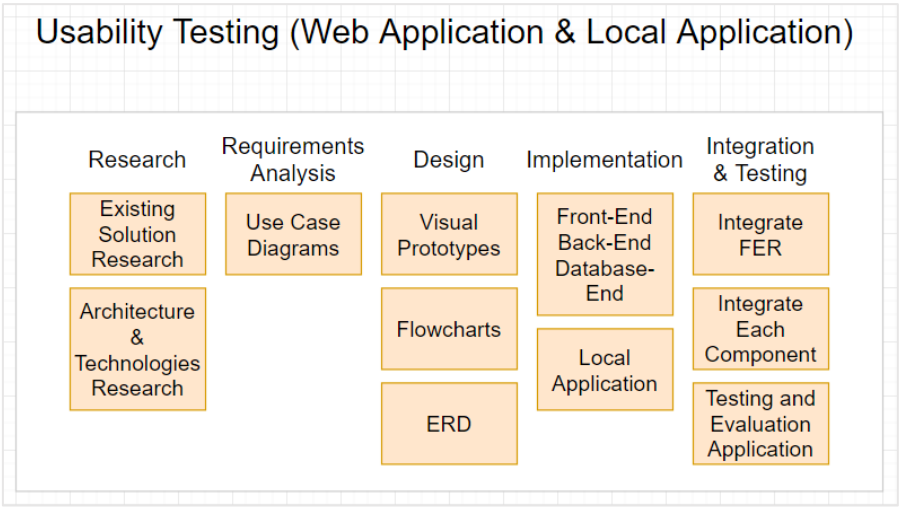


Figure 30 – Web App & Local App Subproject High Level Steps

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 frontend, backend 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. The diagram is designed to specifically show how the machine learning aspects fits into the system.

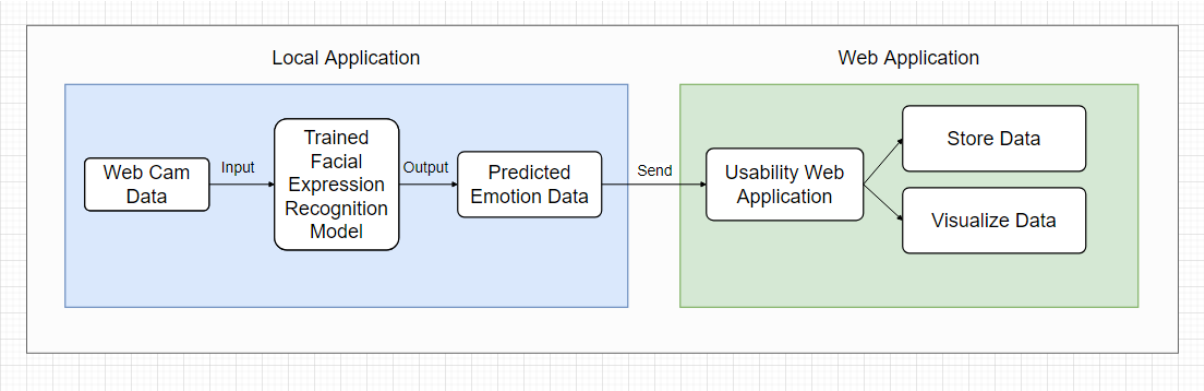


Figure 31 – High Level View of System

A more in-depth view of the architecture of the web application and the local Python application is shown in Figure 32. The web application is a 3-tier architecture with a client, server, and database. The frontend 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 frontend 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 for react that provide charting UI which is very necessary to display the data in this project.

The backend 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 with a steep learning curve given the time constraints. It would simply be an unnecessary risk to the project given that there is familiarly 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. It should be noted however, that the author has not worked with the Spring framework before.

The local python application connects to the backend via the REST API and the same goes for the frontend of the web application. The data is processed in the service tier and passed to and 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.

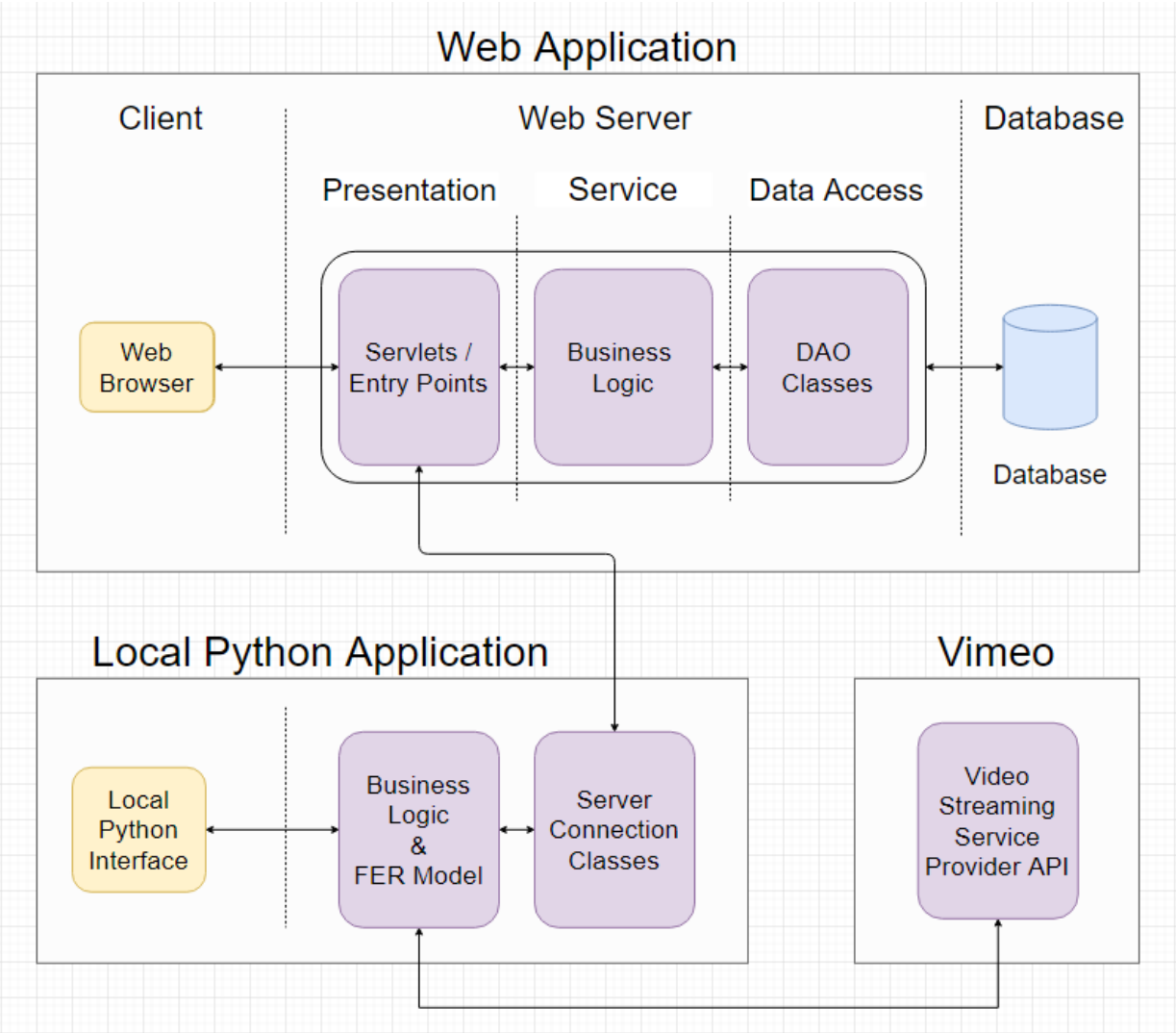


Figure 32 – Architecture of Applications

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 33. 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 system 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.

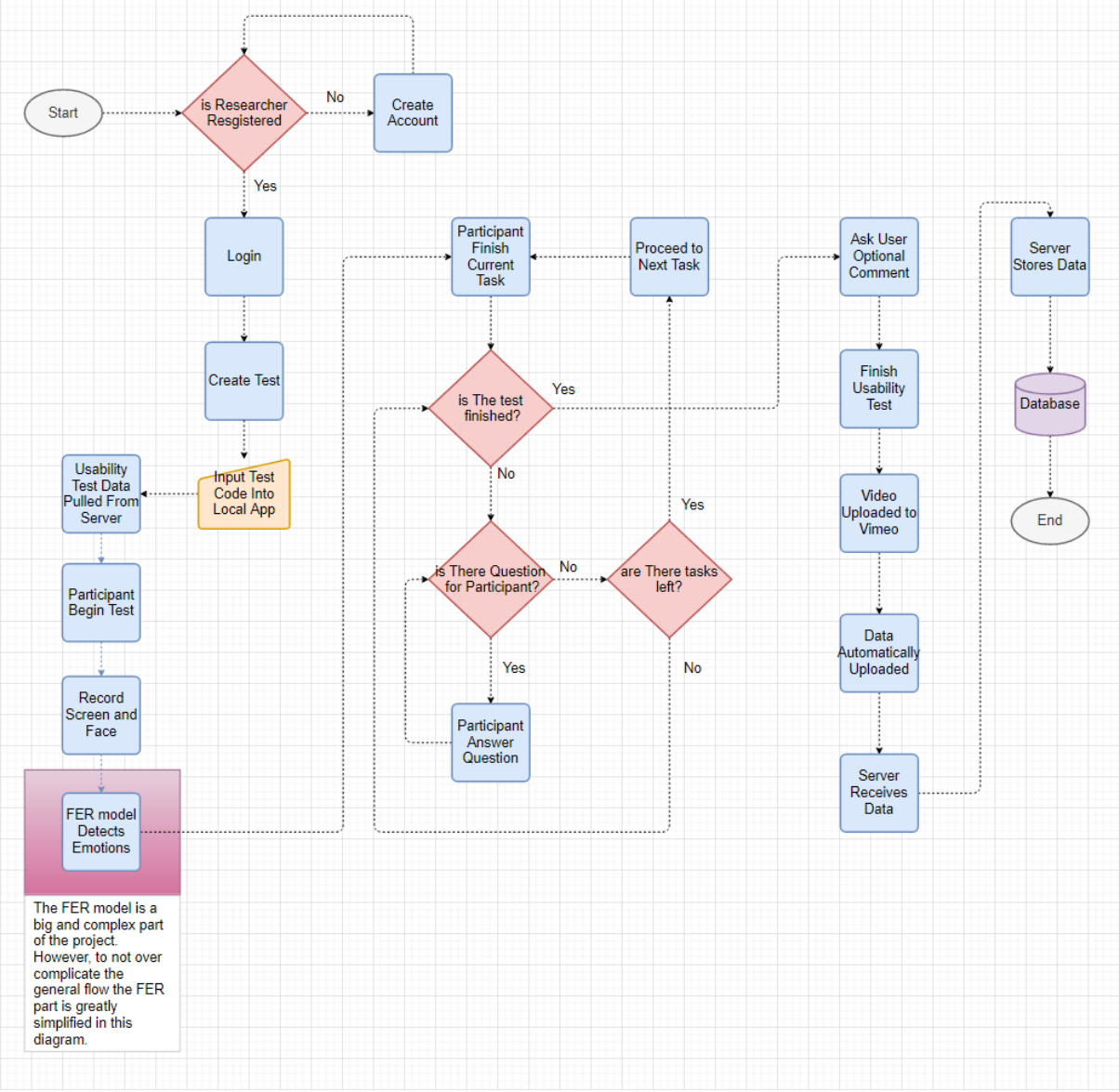


Figure 33 – High Level Flowchart of System

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 34. 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 participants failed one task and the 2nd participant failed two tasks then the success rate would be 7/10 as three tasks were failed amongst the two participants out of a total of 10 tasks.

The 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.

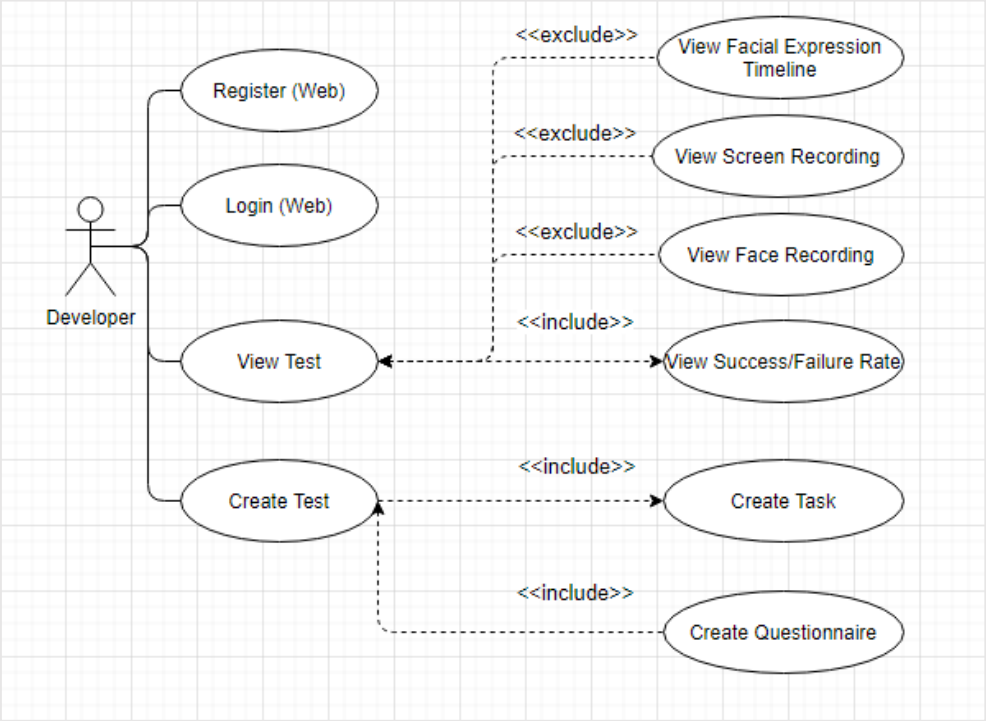


Figure 34 – Researcher Setting Up Usability Test Use Case

The next iteration of the researcher use case diagram is displayed below in Figure 35. 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

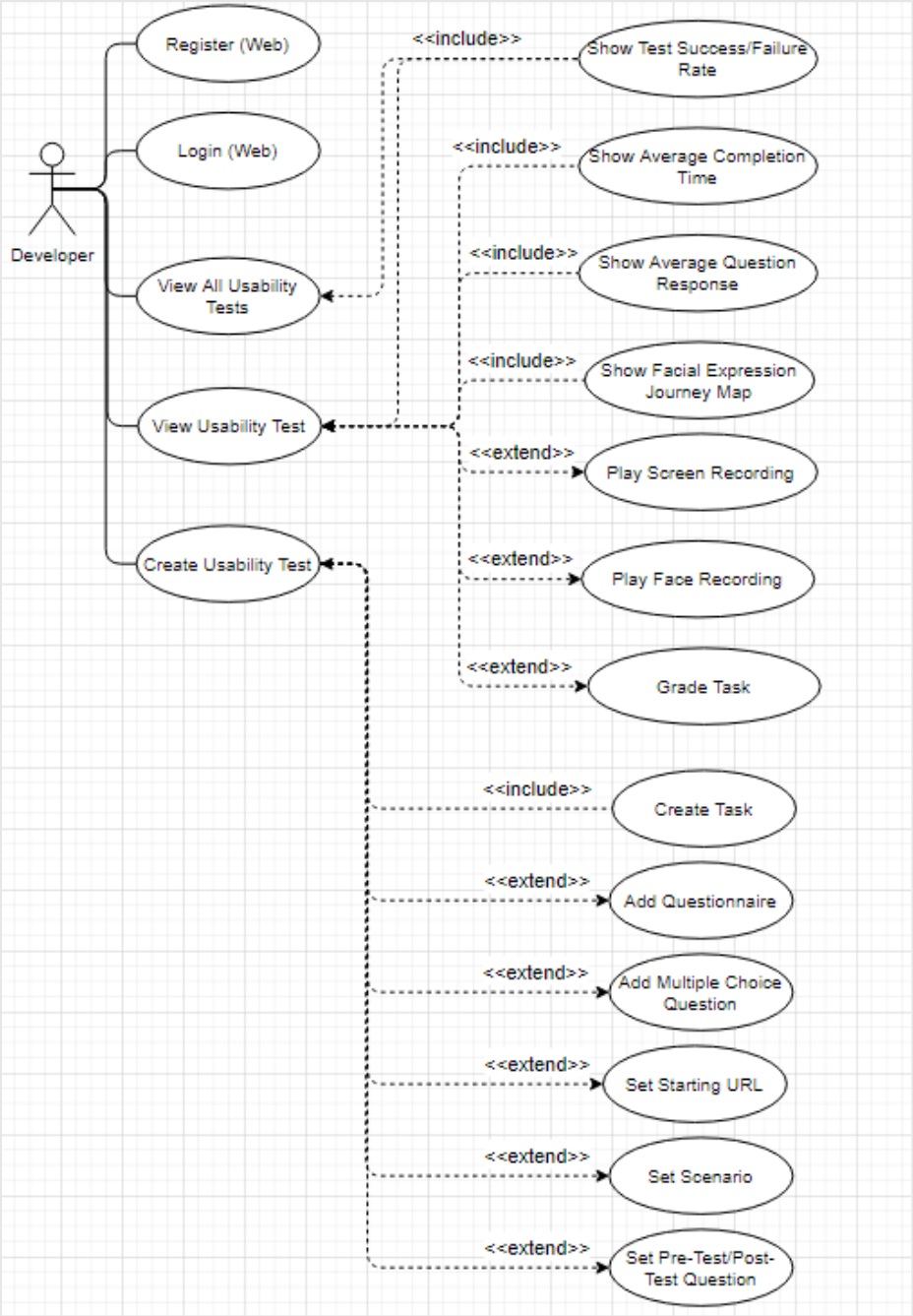


Figure 35 – Researcher Setting Up Usability Test Use Case (Second Iteration)

The use case diagram of the participant who is going to be using the local python application is illustrated below in Figure 36. First the test details are obtained from the web server and then this data is used by thr the local application to execute correct instruction. For example, displaying the tasks and questions. 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 after the test is finished if they would like to add anything.. Once the test is finished the data is uploaded automatically to the web server.

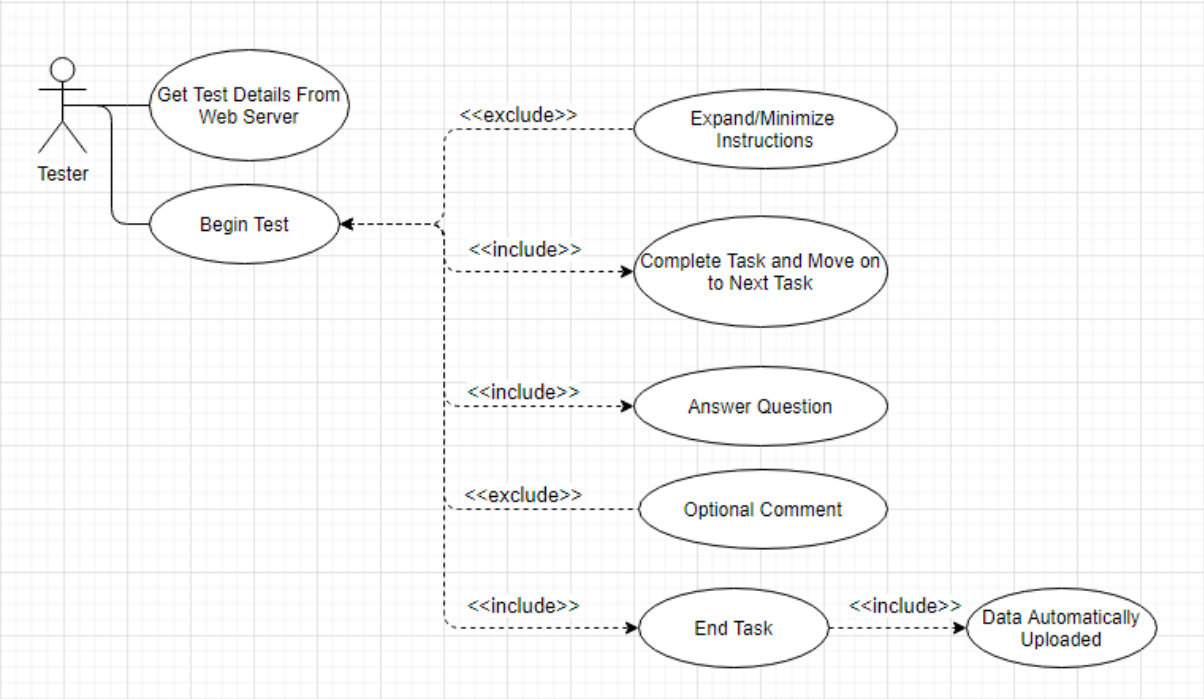


Figure 36 – Participant Use Case Diagram

Screen Prototypes

The "Dashboard" of the web application is the main screen the researcher will see after login and is illustrated below in Figure 37. The prototypes are designed with JustinMind. This screen corresponds 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 an application. 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%).

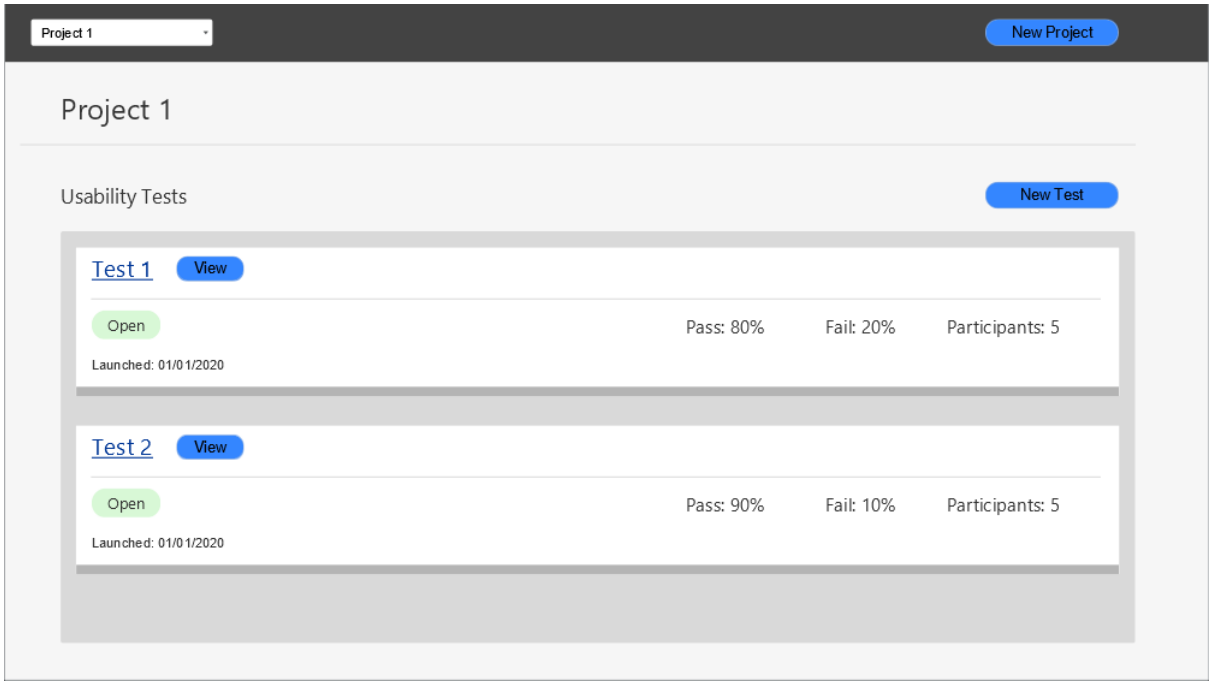


Figure 37 – Web Application Dashboard

The usability test creation screen which corresponds to the "Create Usability Test" use case is illustrated below in Figure 38. The researcher will name their test and insert questions and 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.

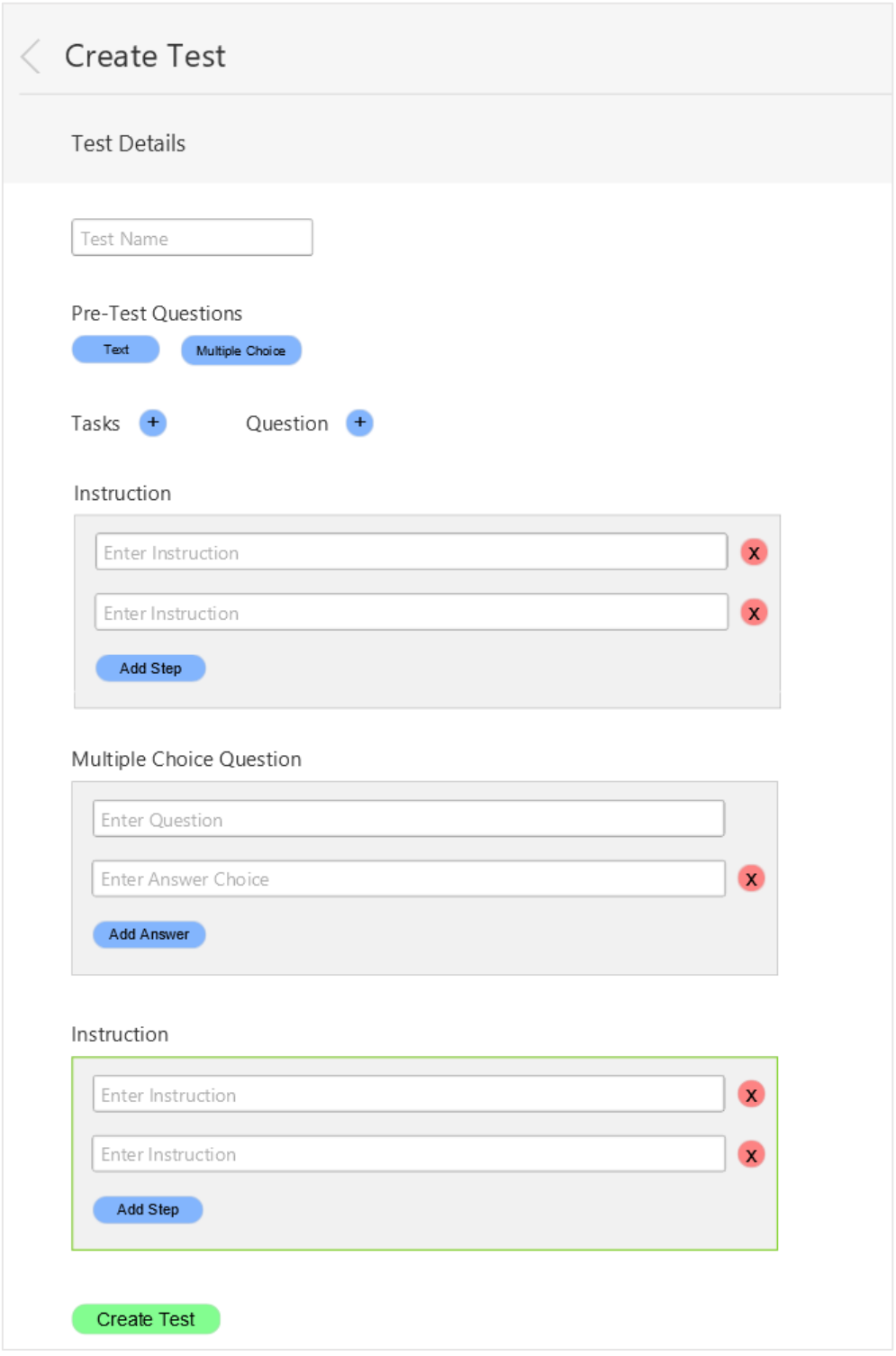


Figure 38 – Web Application Usability Test Create

The test results screen that the researcher is going to see after the usability test is conducted is illustrated below in Figure 39. 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 secton. 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 persis on the partipants 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".

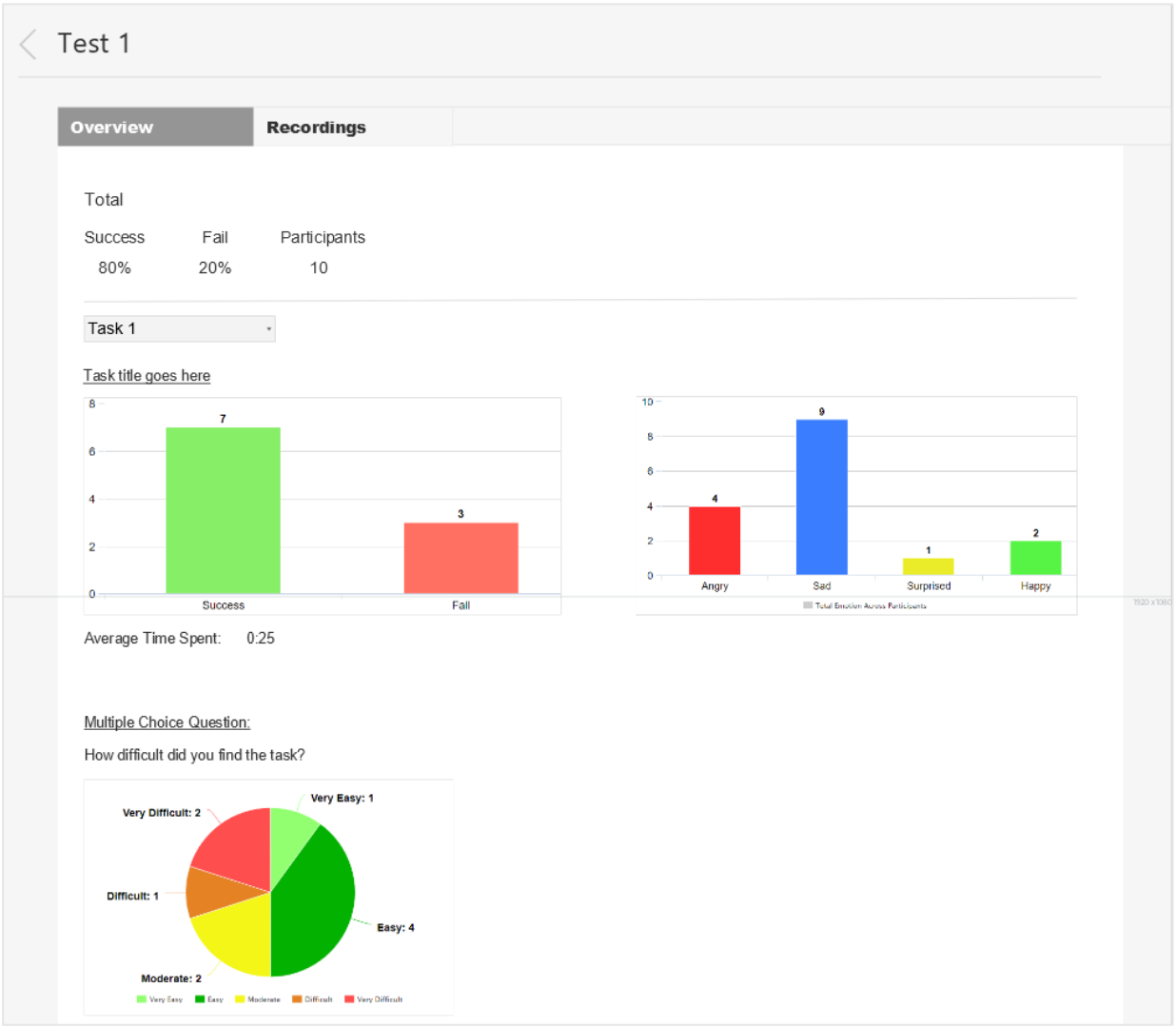


Figure 39 – Web Application View Usability Results (Overview)

The page where the researcher can view the recordings and grade the usability test is illustrated below in Figure 40. The tasks will be graded with either a “Pass” or a “Fail” depending on whether or not the participant 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 researcher 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 [77]. 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, resulting in the chart being less clear.

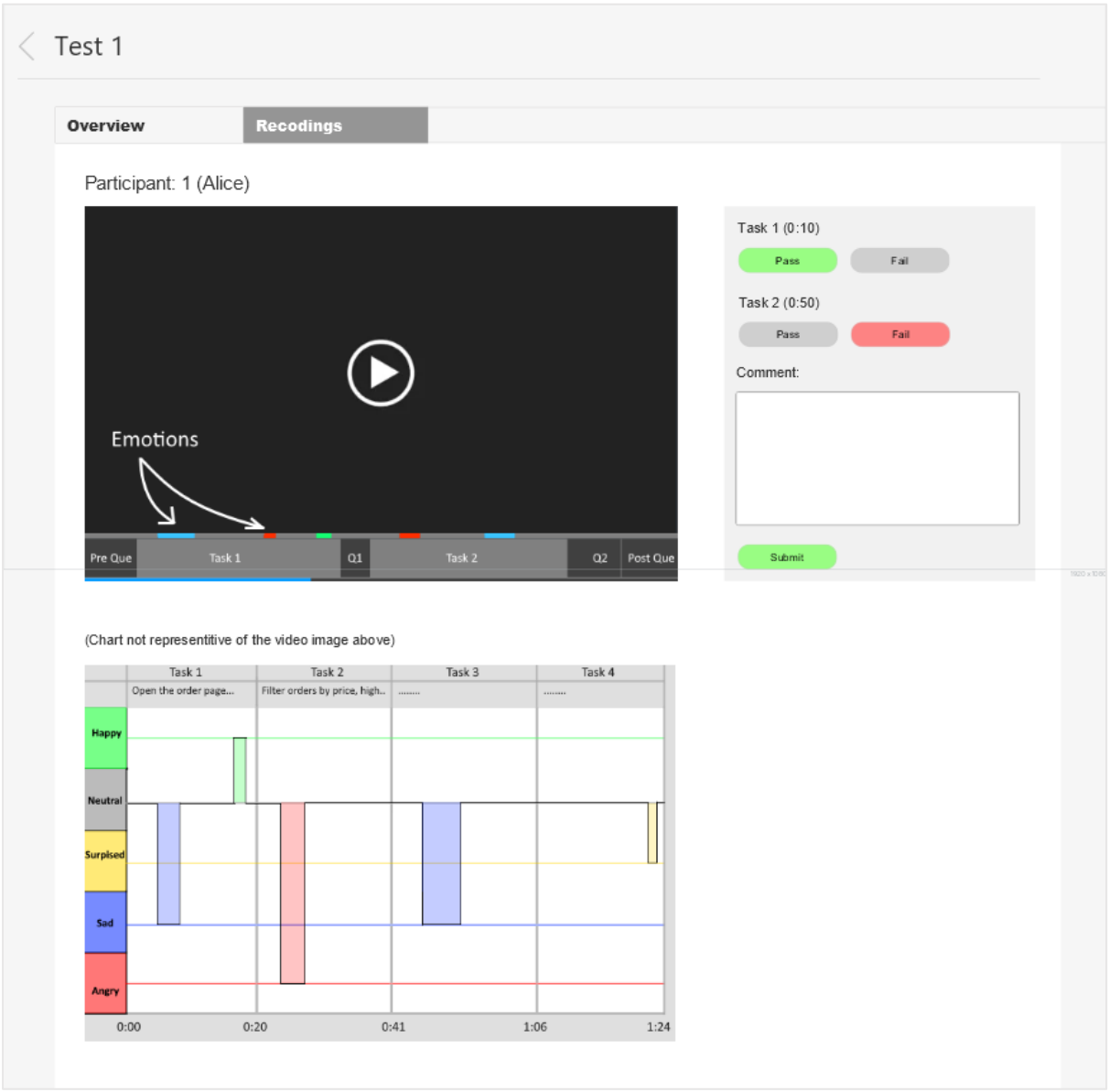


Figure 40 – Web Application View Recordings

The second design of the journey map after the evaluation is illustrated in Figure 41. 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.

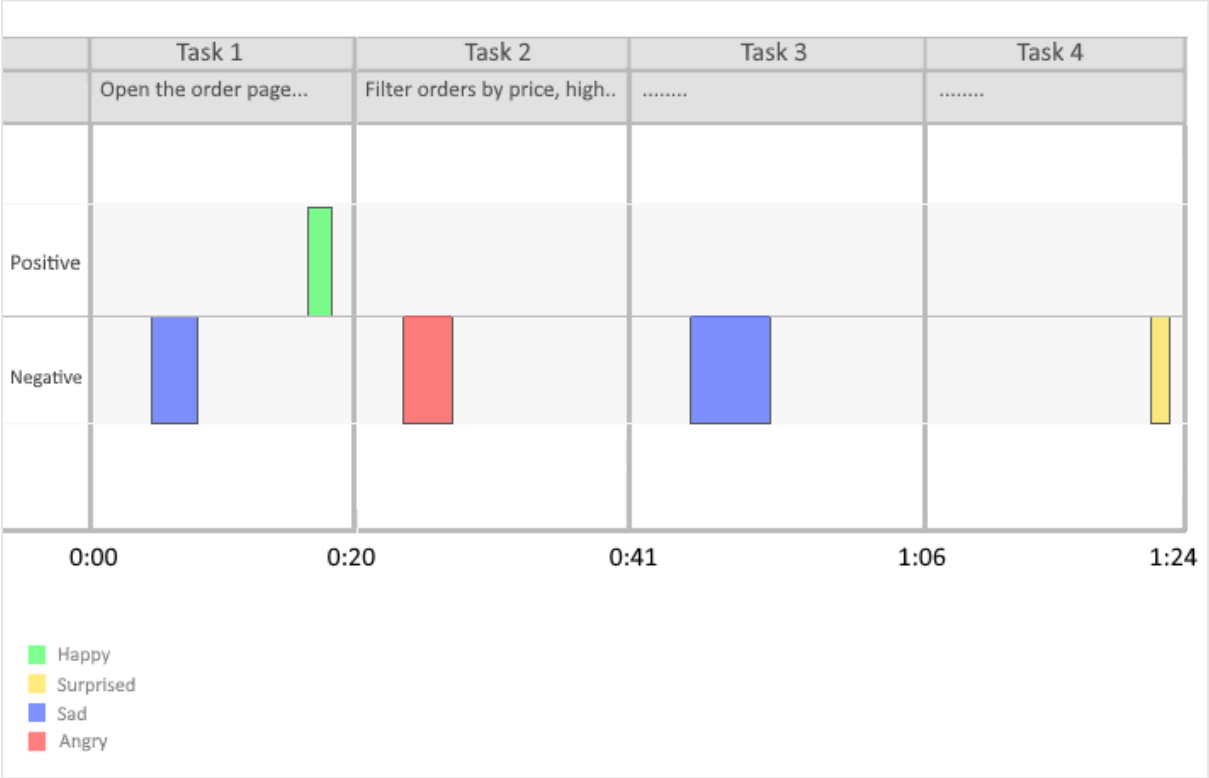


Figure 41 – Journey Map v2

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 or Firestore.

The design of the database is shown in Figure 42. 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 considered was to use inheritance and create a table for each question and answer.

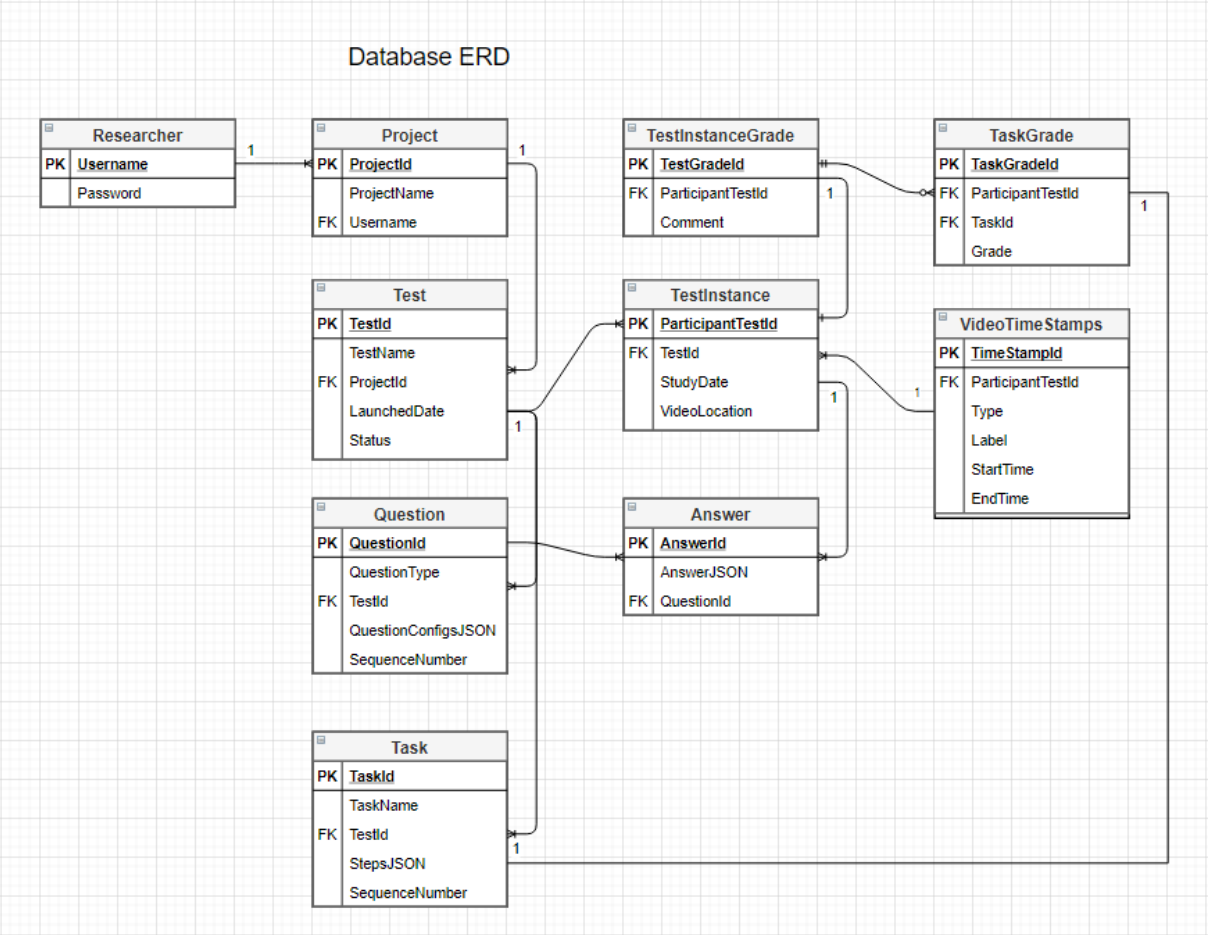


Figure 42 – Web Application ERD v1

The ERD below in Figure 43 - Web Application ERD (Final in Production) is the final design that is used in the deployed and complete application.

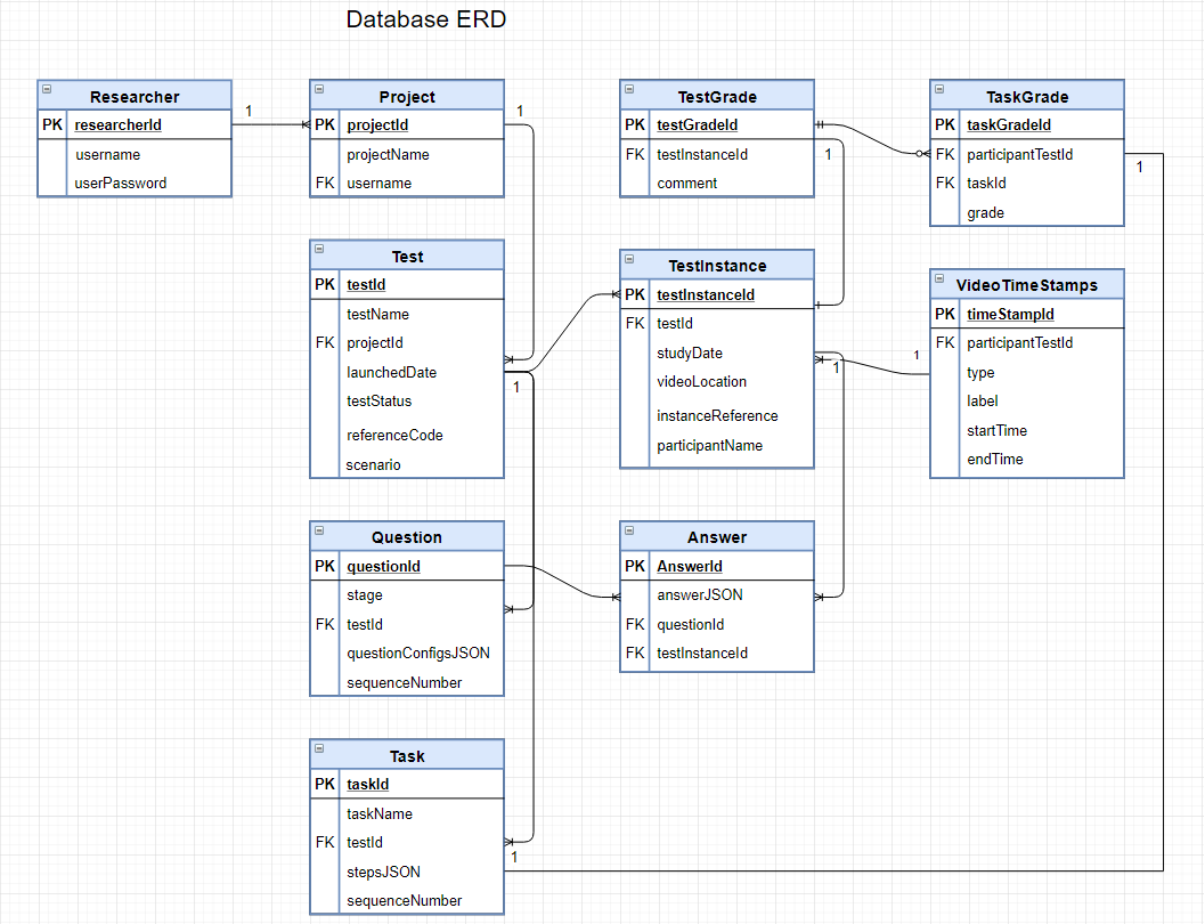


Figure 43 - Web Application ERD (Final in Production)

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 44. 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. This window corresponds to the “Get Details from Web Server” use case in Figure 36. Once the details are verified as correct the participant will enter their name and click “Begin” to begin the text.

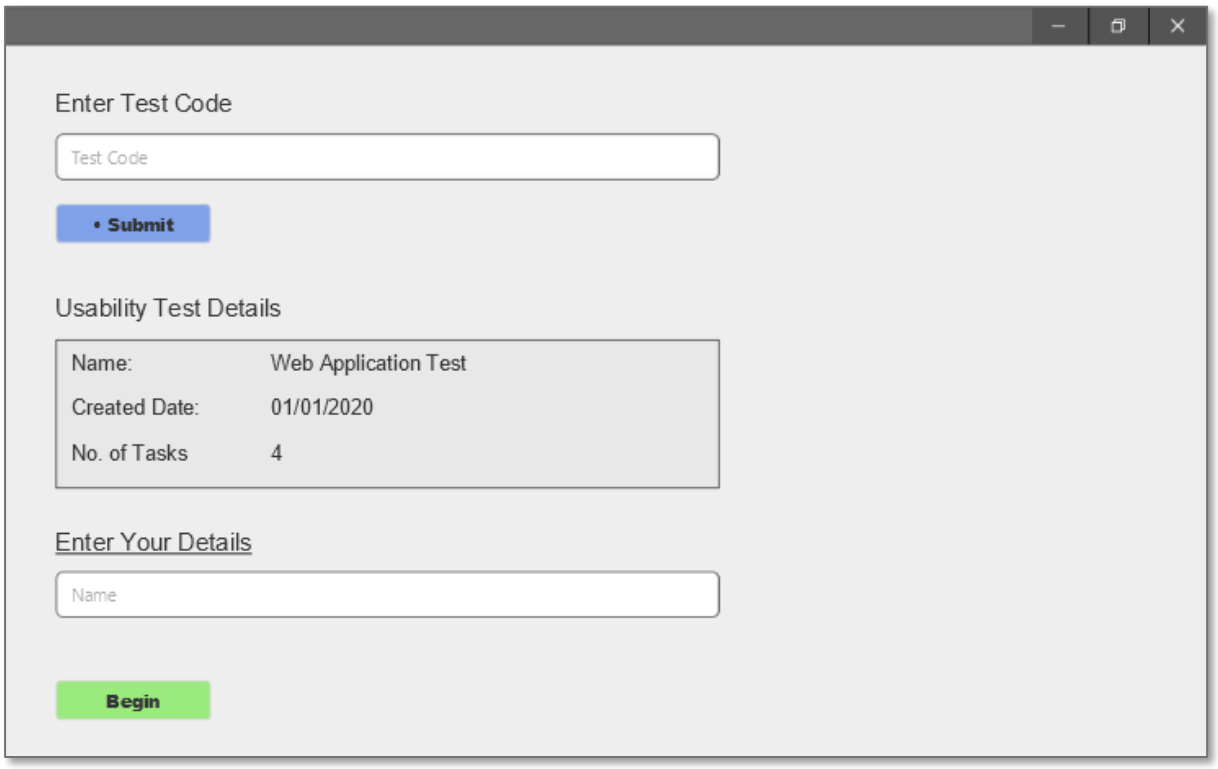


Figure 44 – Local App First Screen

Once the test begins the participant will see a similar screen to what is illustrated in Figure 45. 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 with “Recording” text to ensure the participant knows they are being recorded. They may press the “Stop” button at any moment 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.

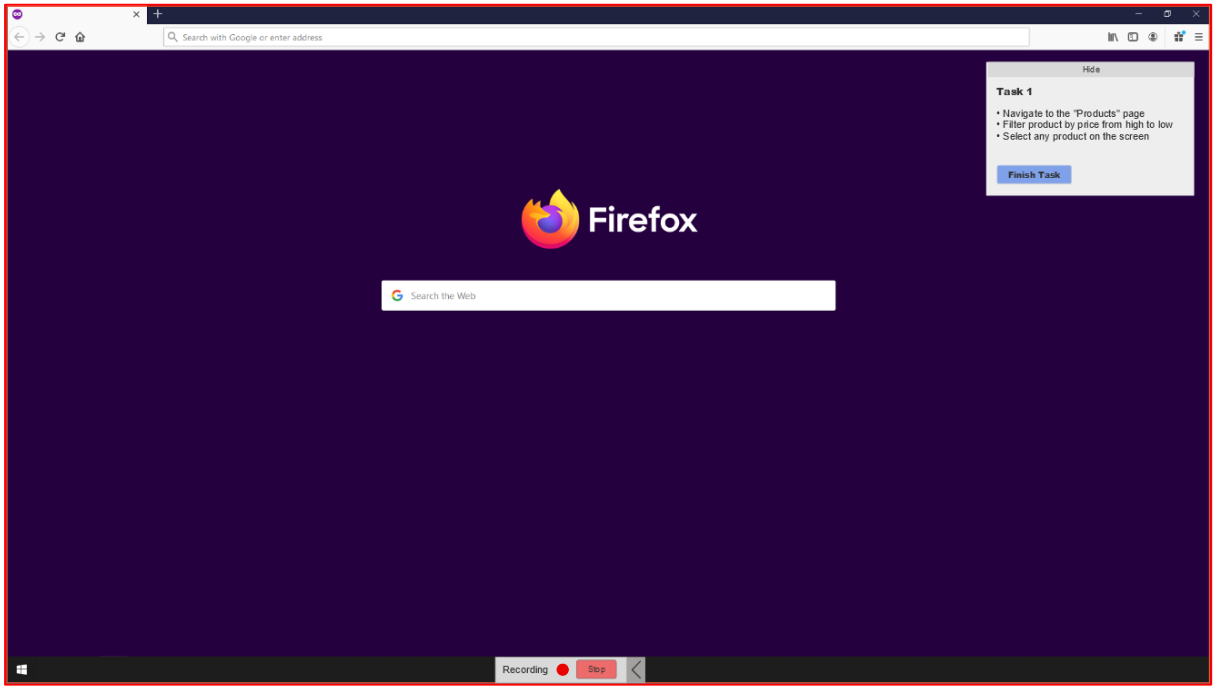


Figure 45 – Local App Usability Test Started

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 researching the answers to 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 classified 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 a 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 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 implement a method for taking the data out of the CSV file but not storing it and feeding it directly into the machine learning algorithm for training. However, for consistency reasons 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 46. 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.

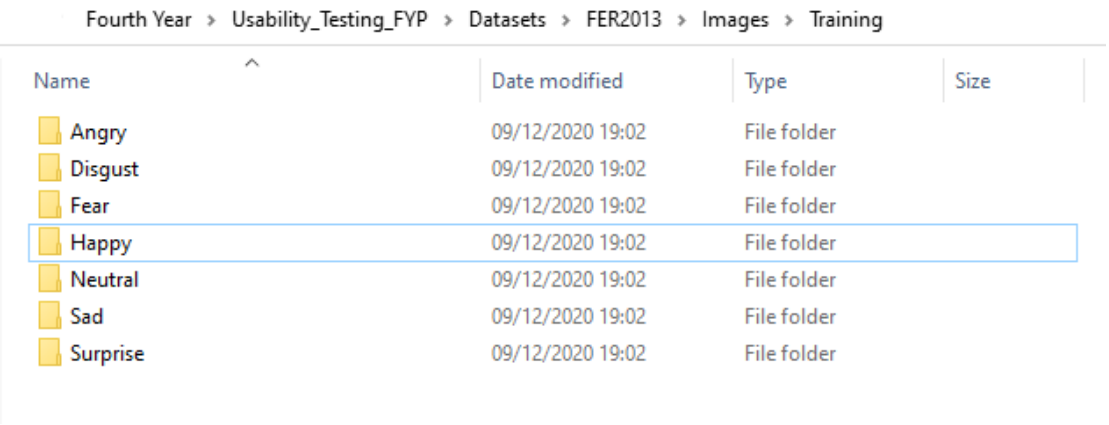


Figure 46 – Dataset Folder Structure (Training)

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 within a certain range.

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 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. The evaluation techniques for the model were also previously discussed in the research section 2.1.4.. To recap the metrics used to evaluate the model will be the confusion matrix, accuracy, recall, precision, and f-score. The 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 in described in research section. 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 a face 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 47. 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 47 – Project Tasks & Requirements

4. UsabCheck Development

4.1. Introduction

Links and Resources

GitHub Repository: https://github.com/damianazur/Usability_Testing_FYP

Web Application: https://usabcheck.herokuapp.com/

Local Application:
https://mega.nz/file/Ychi0QQI#aD7zslp4JQst6FBlv8lgLm5iD5Ww_PkxoIbXiC4BV2U

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 made it very suitable for data science and machine learning.

Development Sections

The sections of the development chapter are shown in Figure 48. The number next to the name is the order in which they are discussed.

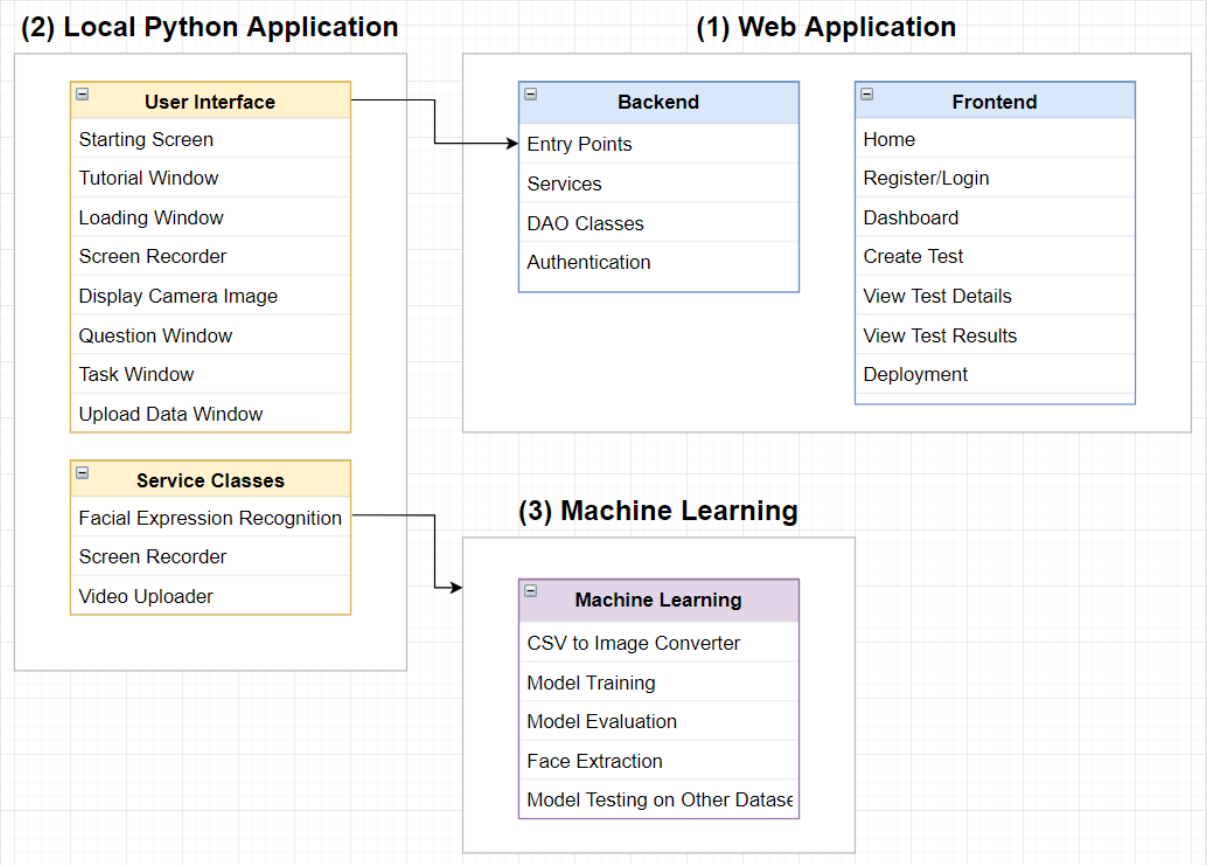


Figure 48 - Development Sections

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 the Java classes and packages that are in the web application which currently deployed on a server is shown below in Figure 49. The packages and classes are explained in greater detail in the following sections.

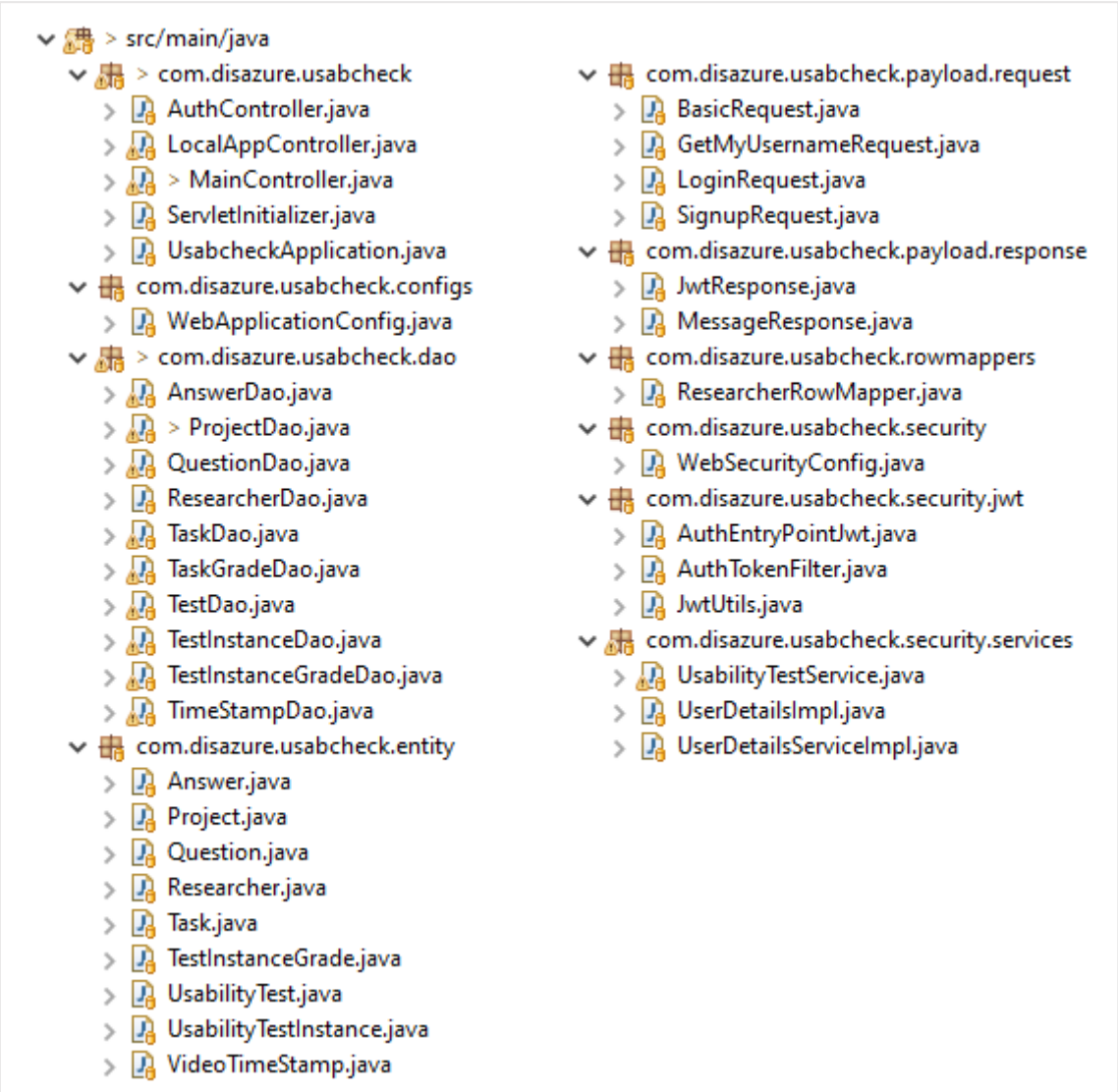


Figure 49 – Web App Backend Java Classes

The relationships between all of the classes in each package is shown below in Figure 50.

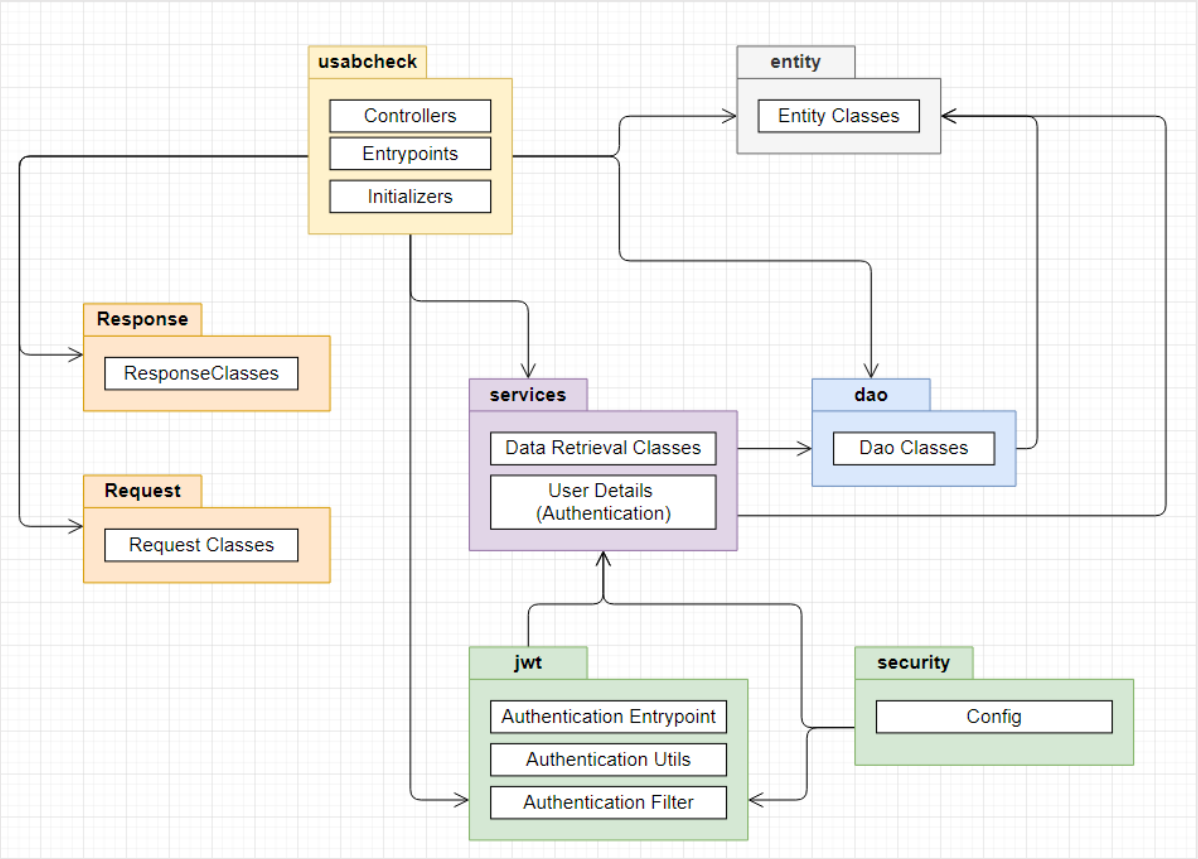


Figure 50 – Relationship Between Packages

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.

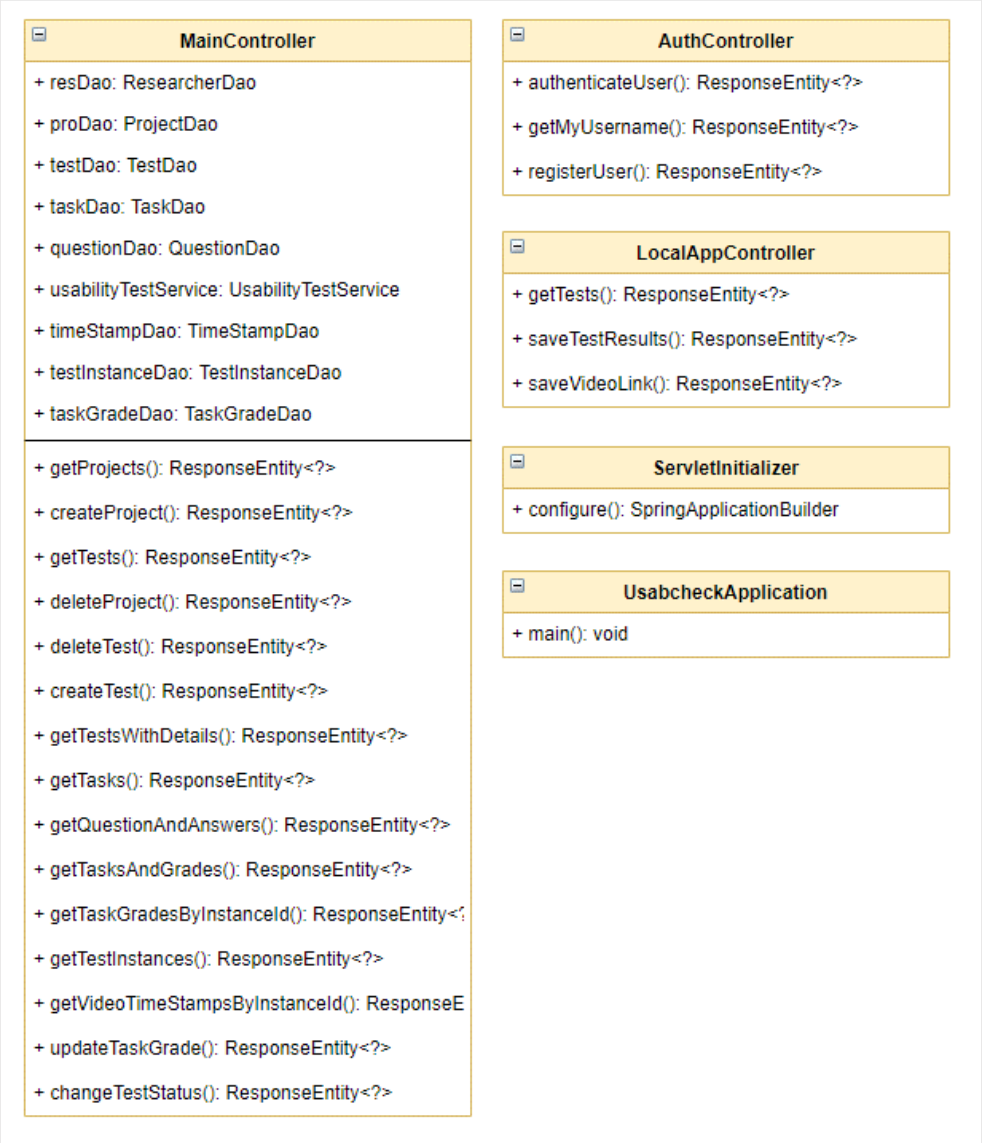


Figure 51 - Controller Classes

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/combining 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.

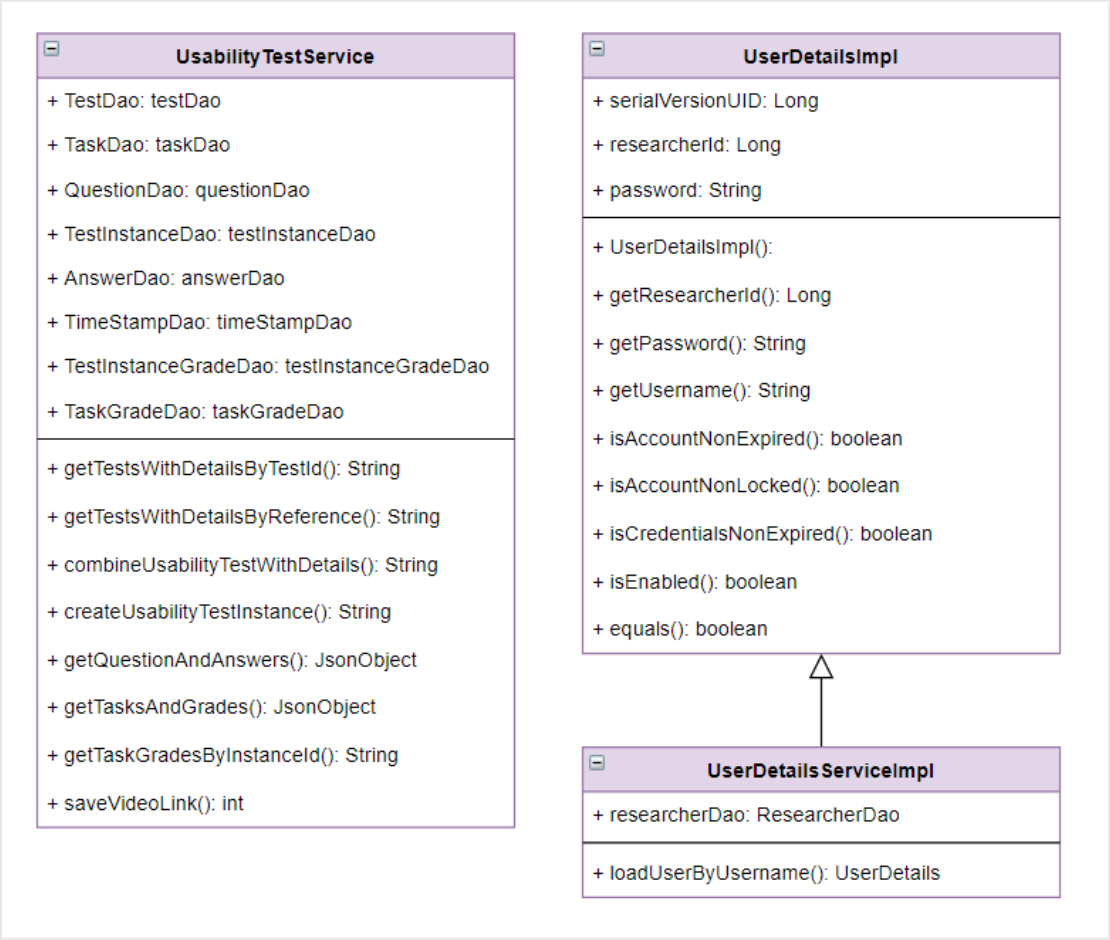


Figure 52 - Service Classes

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 a post by Karibasappa G C [78] is that Hibernate 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 53.

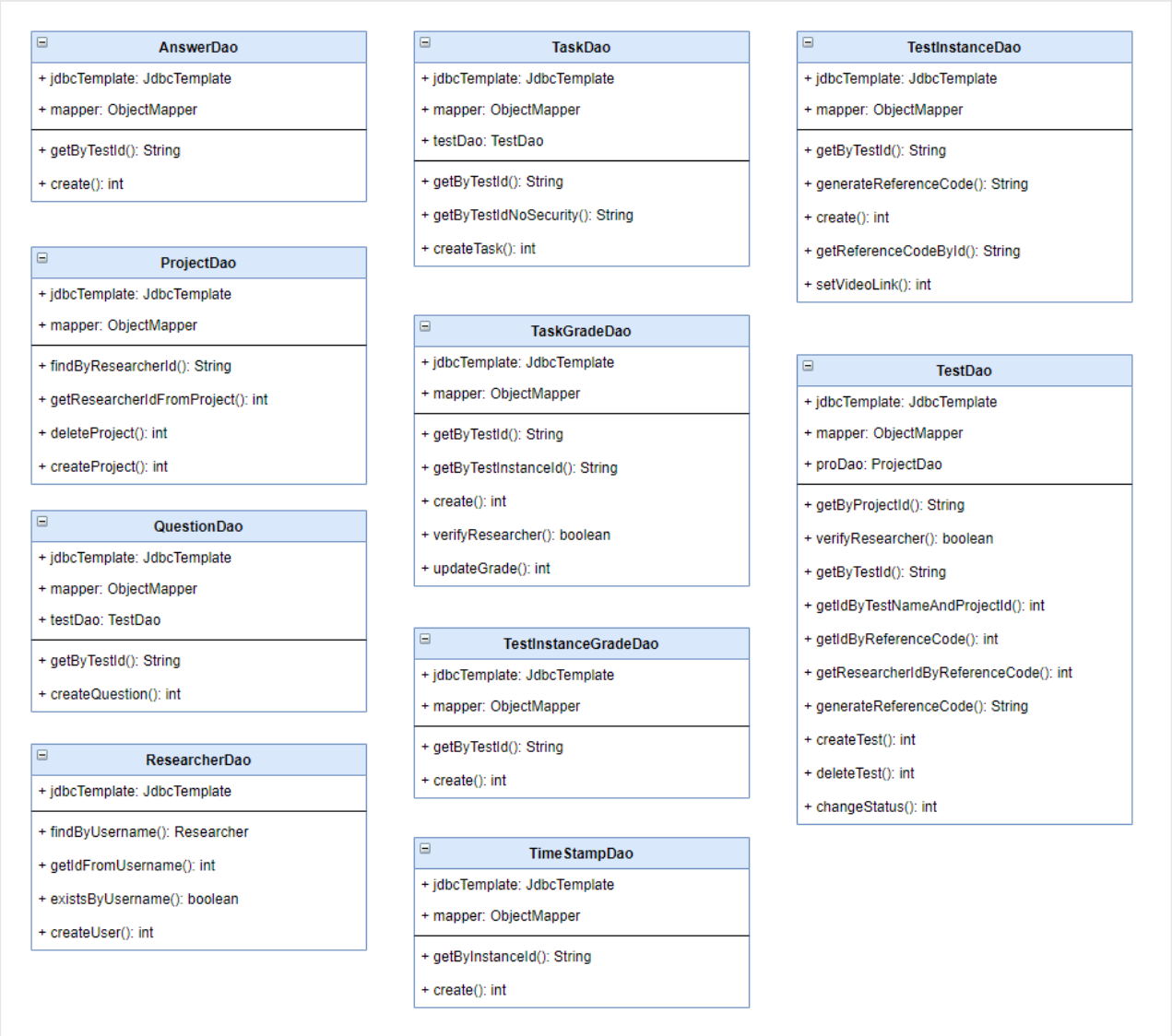


Figure 53 – DAO Classes

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 54. 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 a question mark.

```
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 54 – 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” could be simply “delete”.

Authentication

JWT (JSON Web Token) was used to authenticate the user. A tutorial by Bezkoder [79] 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 [80].

The Security classes are shown below in Figure 55.

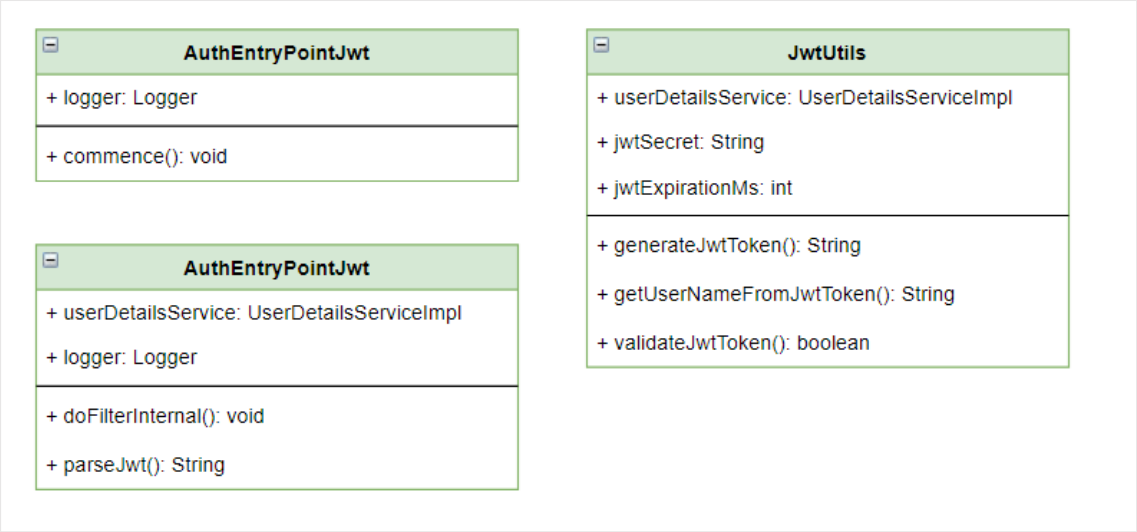


Figure 55 – Security Classes

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 56.

researcherId	username	userPassword
4	testuser123	$2a$10$wpxYsMF58Jro4e3Zc0WH..dLYrG2WX/HeJi3fbjGH.ZiGiq0zHZL.
5	testuser1234	$2a$10$kouiReUkkt57KQsjnfsEVe.dGlA2dUJ1l9P/kWM75nz535BzuWECm
6	SampleUser1	$2a$10$N9HIgayqeEEKeBQIklU.0uHOI5gWFSQc8rsg4Z5HFbVNlAAsouX8a
11	SampleUser2	$2a$10$Dm8TyoR40MC4GOVTBmj6i.X9nULzF.N4/I9NRqtJLB57KZxMyfPv.
12	SampleUser3	$2a$10$5307Eb0k/WPNv3Tr2tcTlOpEmd9jbkJgBIuGjd8bEymVlOhZBdIH2
13	SampleUser4	$2a$10$k5G1PswiLIv21.1Pwq5J1ObOzlg3Nfx9AXE2kuv7iUUybJ5T5A3X6

Figure 56 – 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 57. 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.

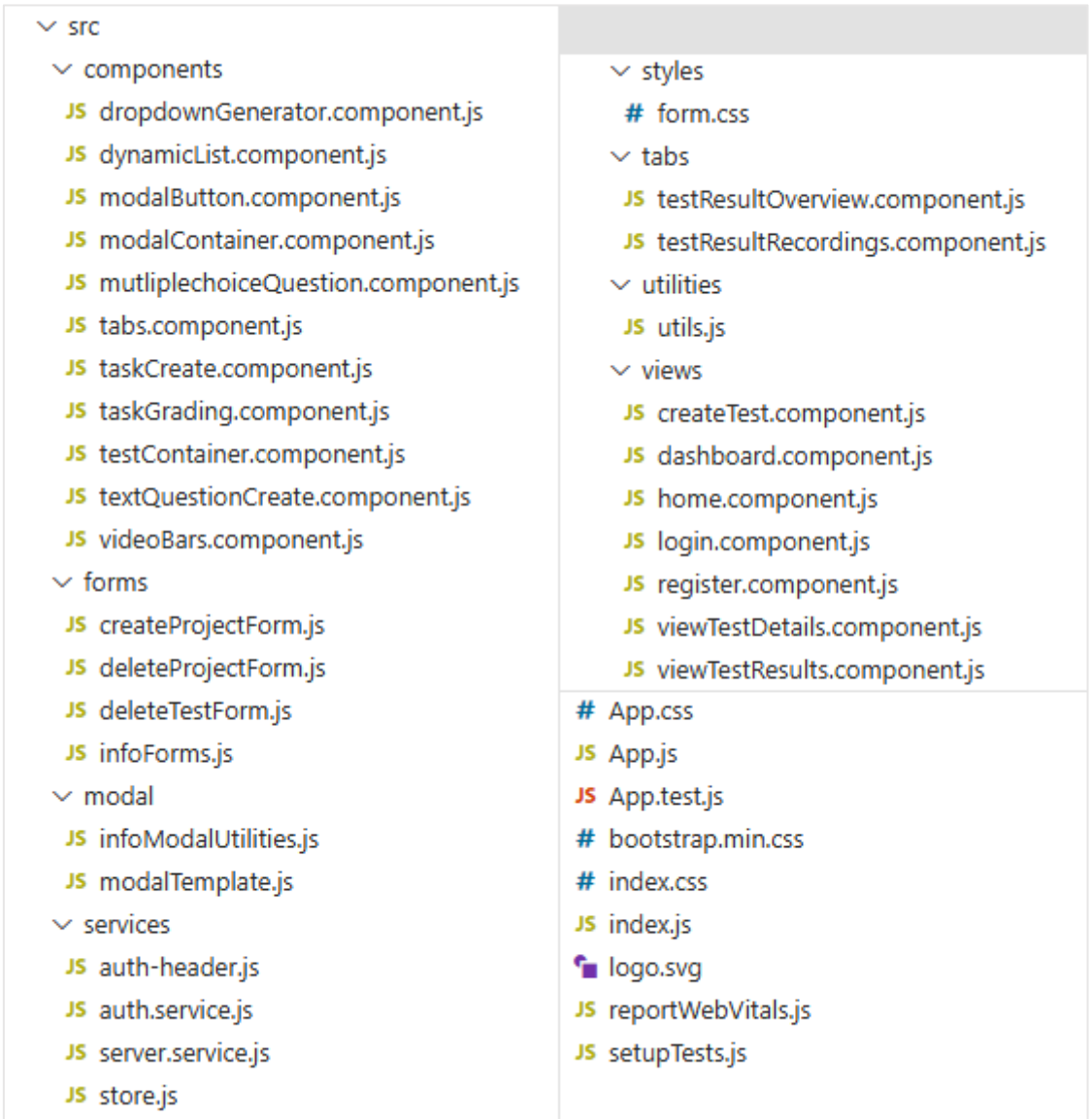


Figure 57 - Web App Front-end Classes

The interaction between all of the folders can be seen in Figure 58. 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.

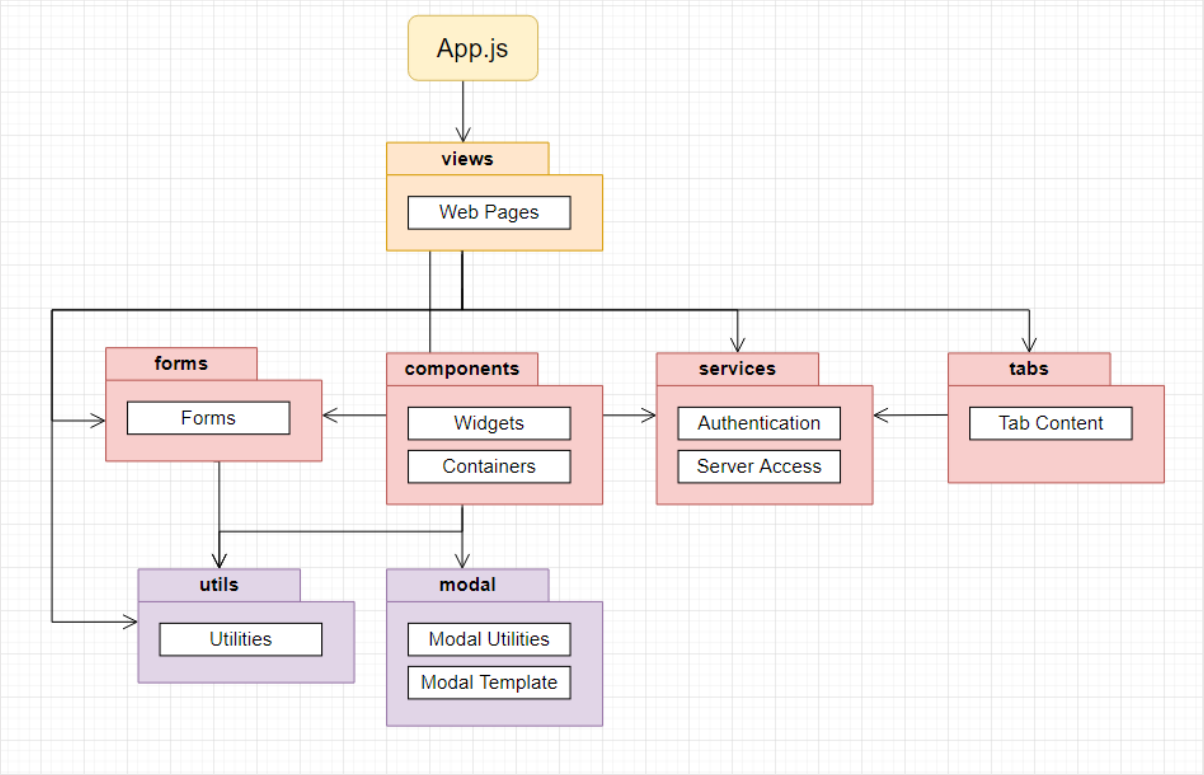


Figure 58 - Web App Class Front-end Relationships

The relationships between pages is shown in displayed in Figure 59. 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.

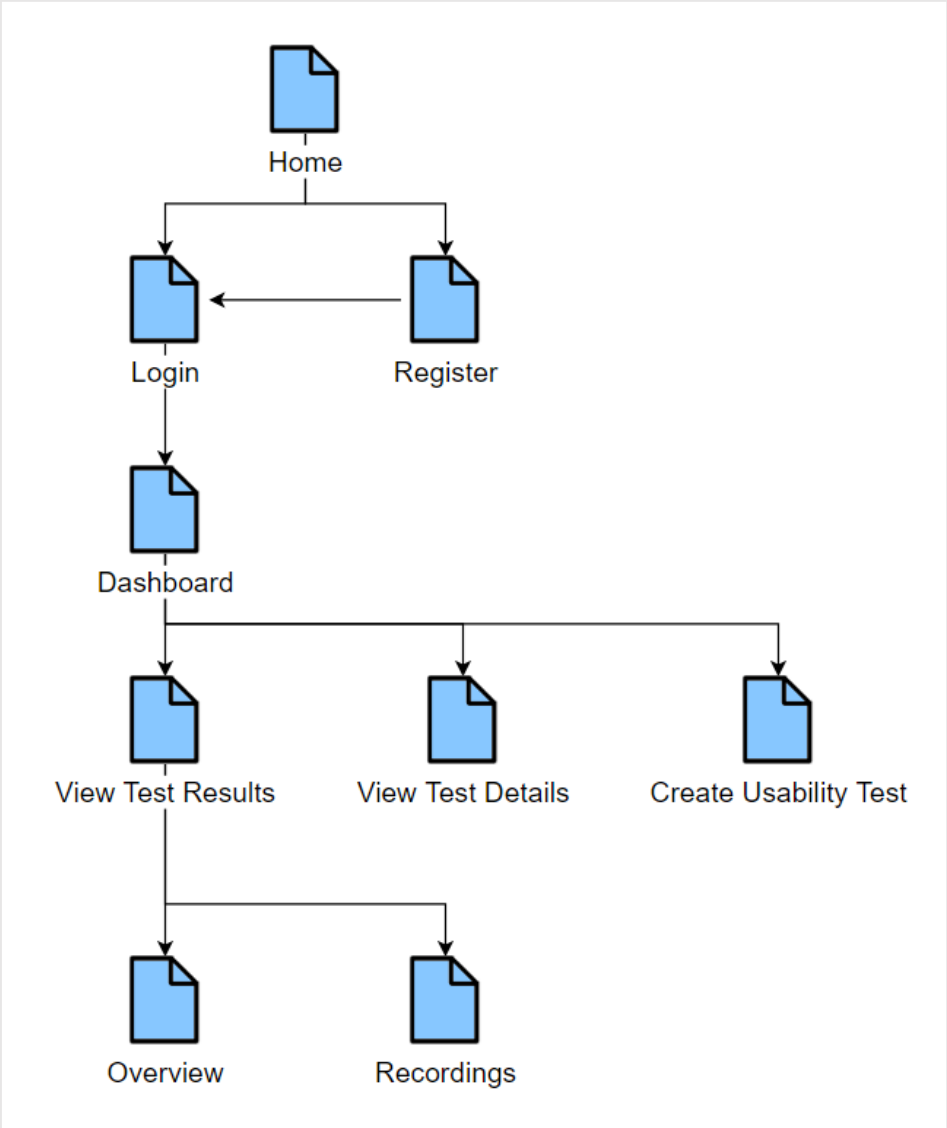


Figure 59 - Webapp Page Relationship Diagram

Home Page

The home page contains some general information about the application as shown in Figure 60. From there the user can navigate to the sign up and login page.

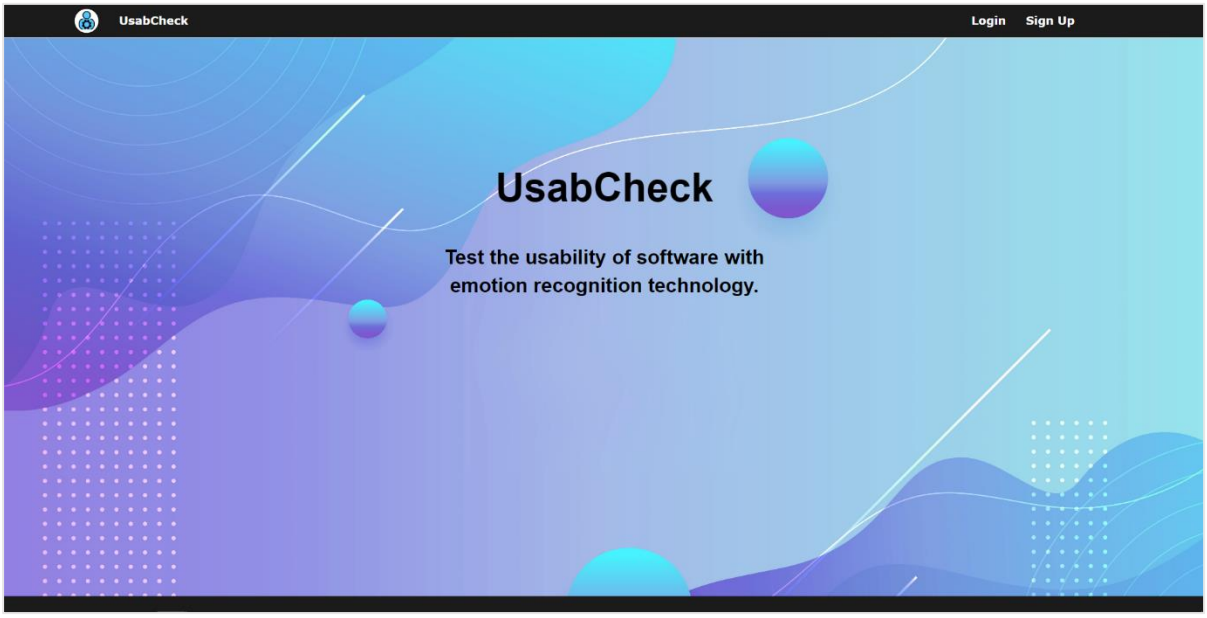


Figure 60 - Home Page

Register/Login

The login and register pages are separate pages and the UI is shown in Figure 61. For simplicity there is no email address and no password changing as the complexity of the project does not stem from this part.

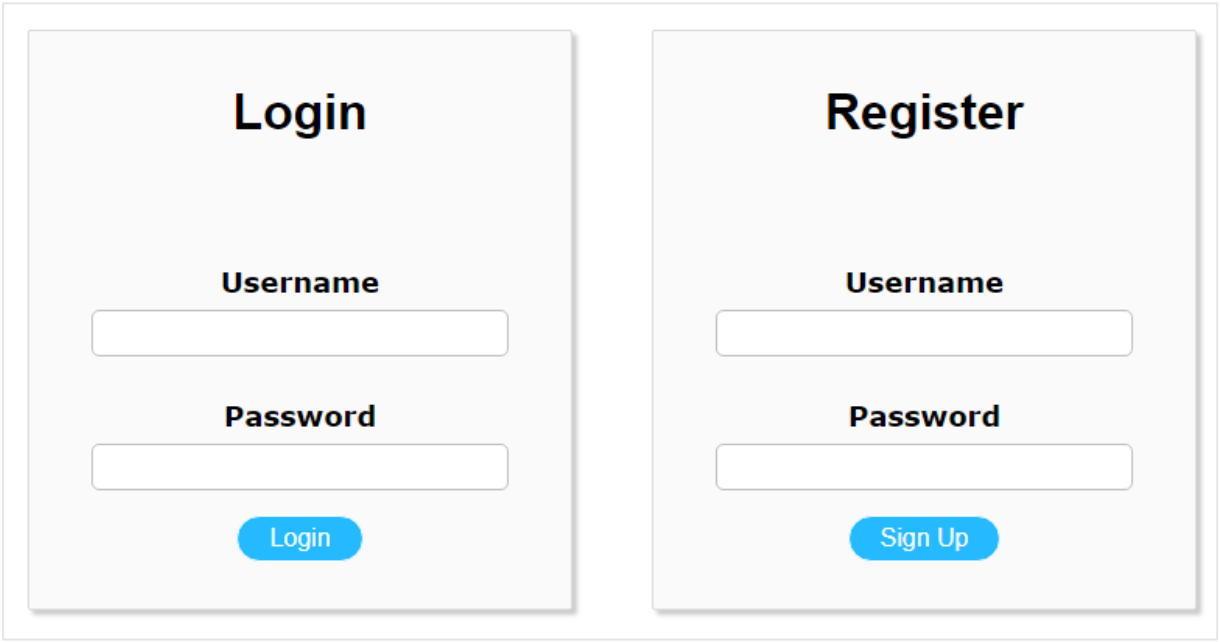


Figure 61 – Login and Register Pages

Dashboard

When a new user first logs in the dashboard page will indicate that they need to create a project as shown in Figure 62.

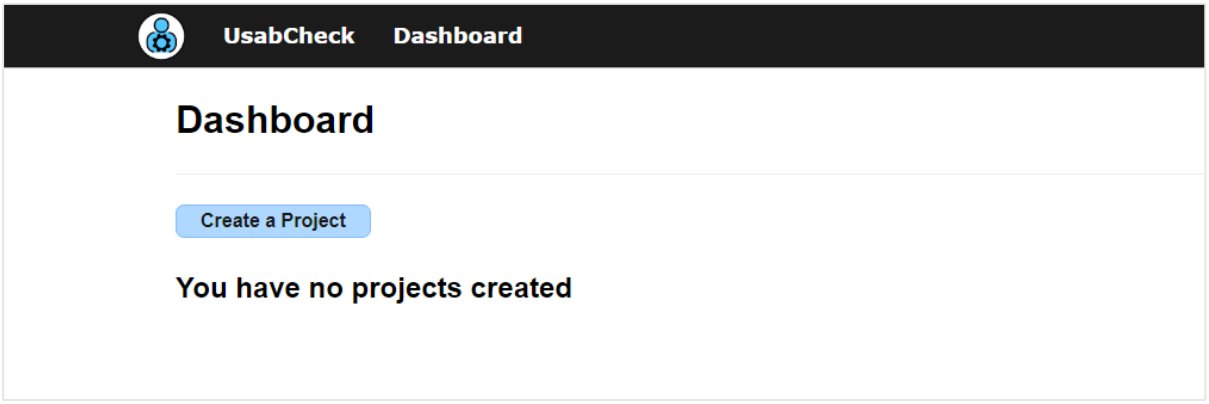


Figure 62 - Dashboard (No Project)

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 63. The starter code for the modal was obtained from a tutorial by Krissanawat Kaewsanmuan [81]. The code was modified to suit the needs of the project by adding additional parameters and entirely restyled 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 63 - Dashboard (Project Create)

Once a project is created additonal options show up on screen as shown in Figure 64. 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 utitilized 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.

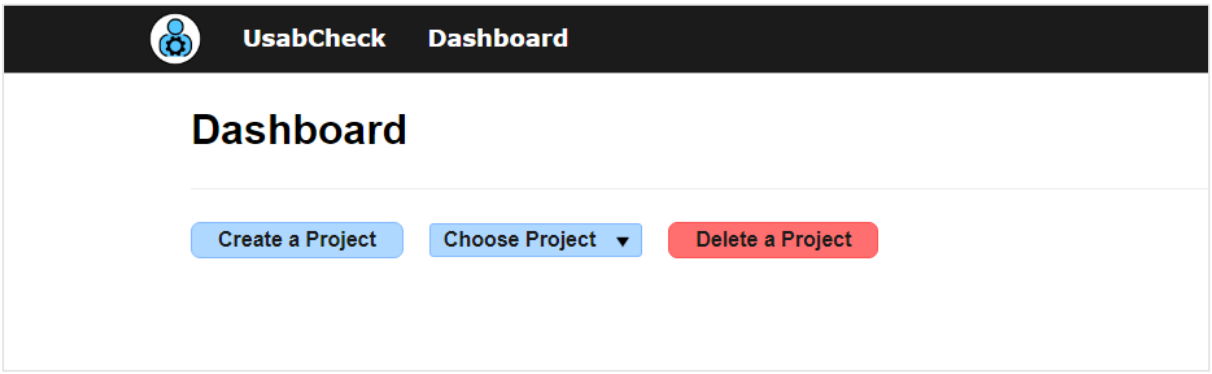


Figure 64 - Dashboard (Project Created)

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 65.

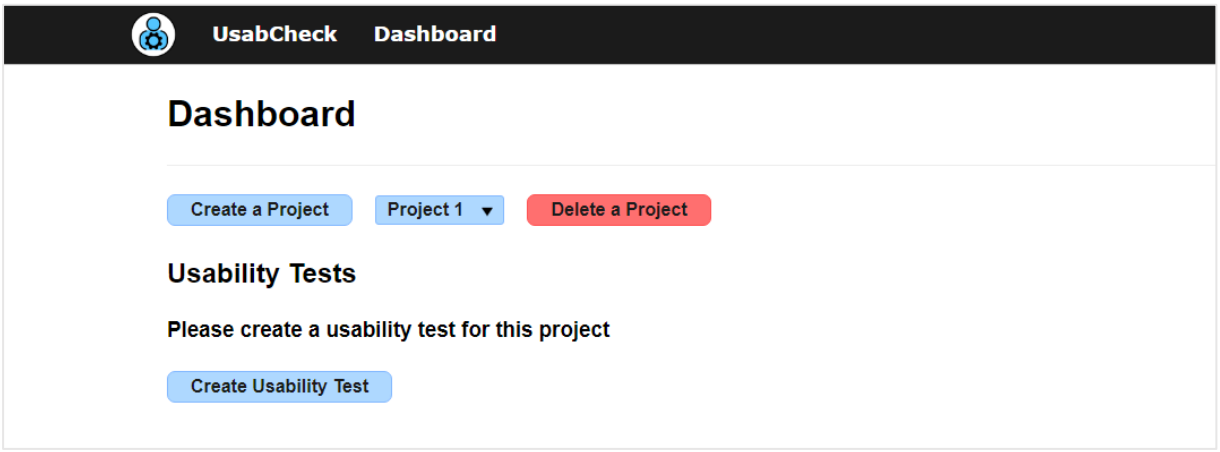


Figure 65 - Dashboard (Project Selected & No Usability Tests)

When the researcher clicks on the “Delete a Project” button a popup will appear 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.

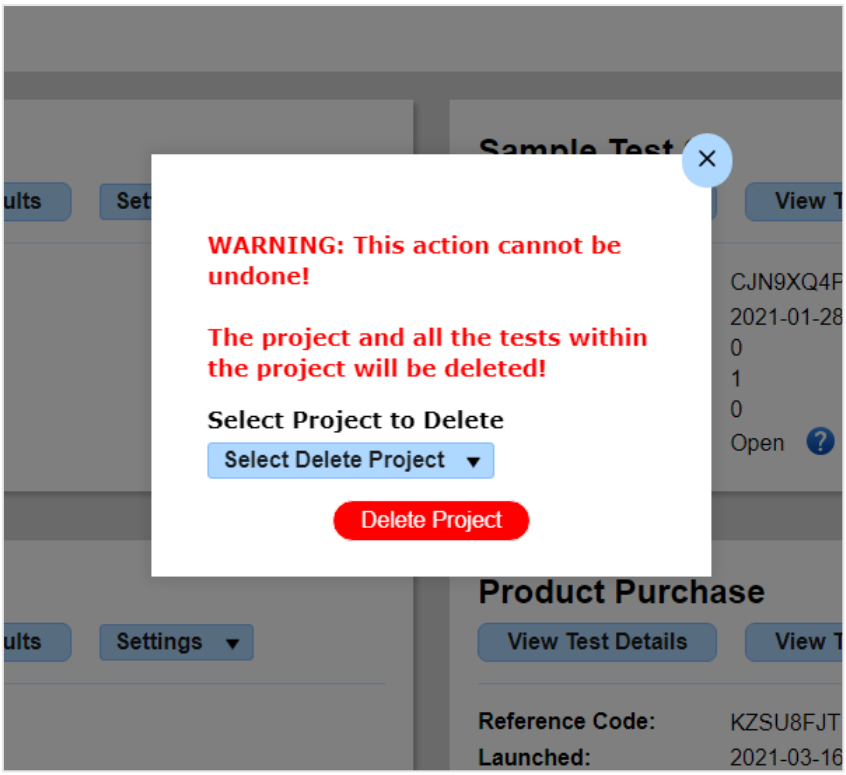


Figure 66 - Project Delete Modal

When a researcher has tests created and they select a project, the usability tests for that project will appear under the heading "Usability Tests" as can be seen in Figure 67. 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 test component box used to display the data is named *testContainer.component.js*.

A reference code/number is generated when a usabilty 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 generated 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 participate in 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.

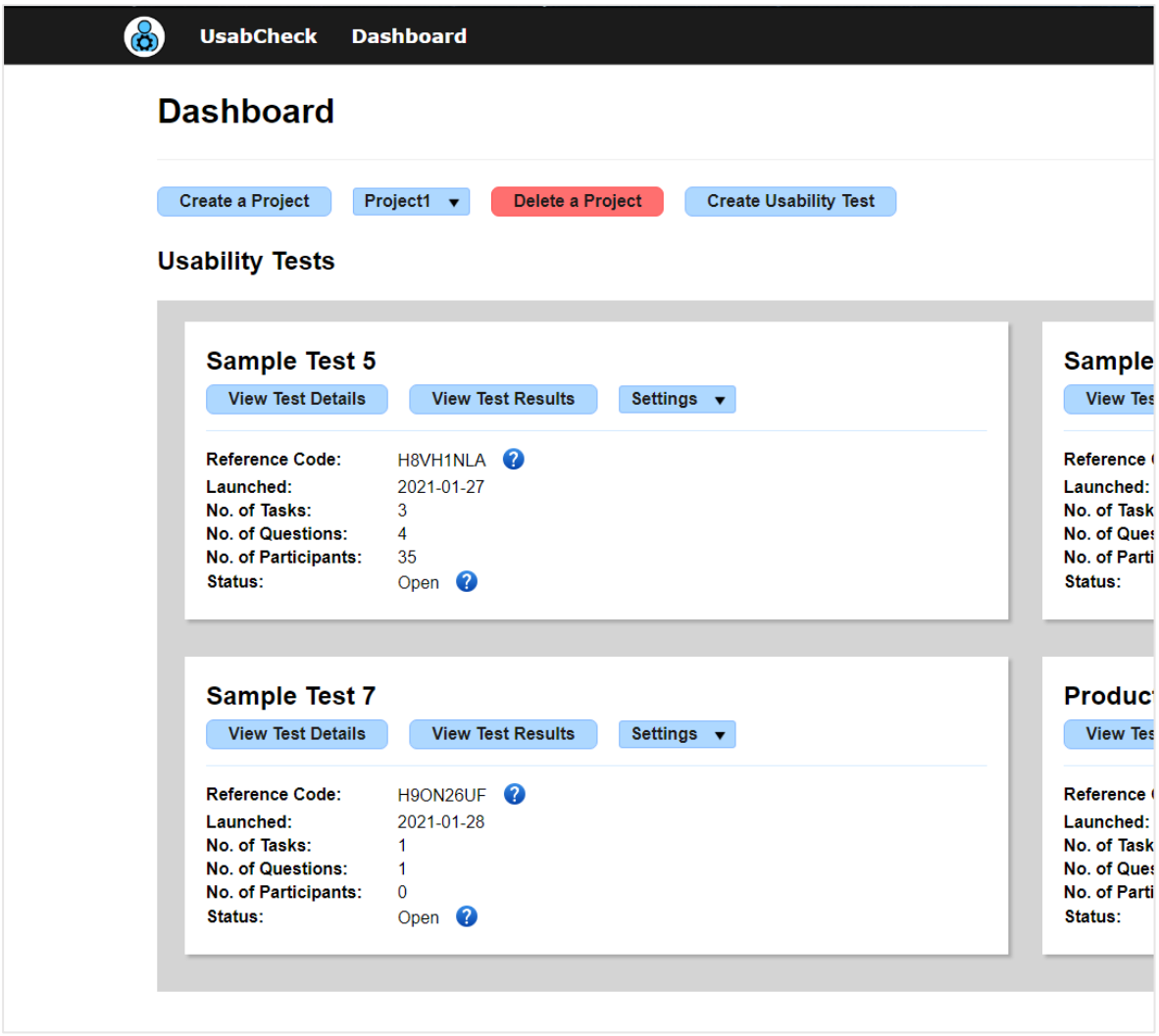


Figure 67 - Dashboard (Project Selected & There are Usability Tests)

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 68. 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 each field is and an example of the input that can be enterd. 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.

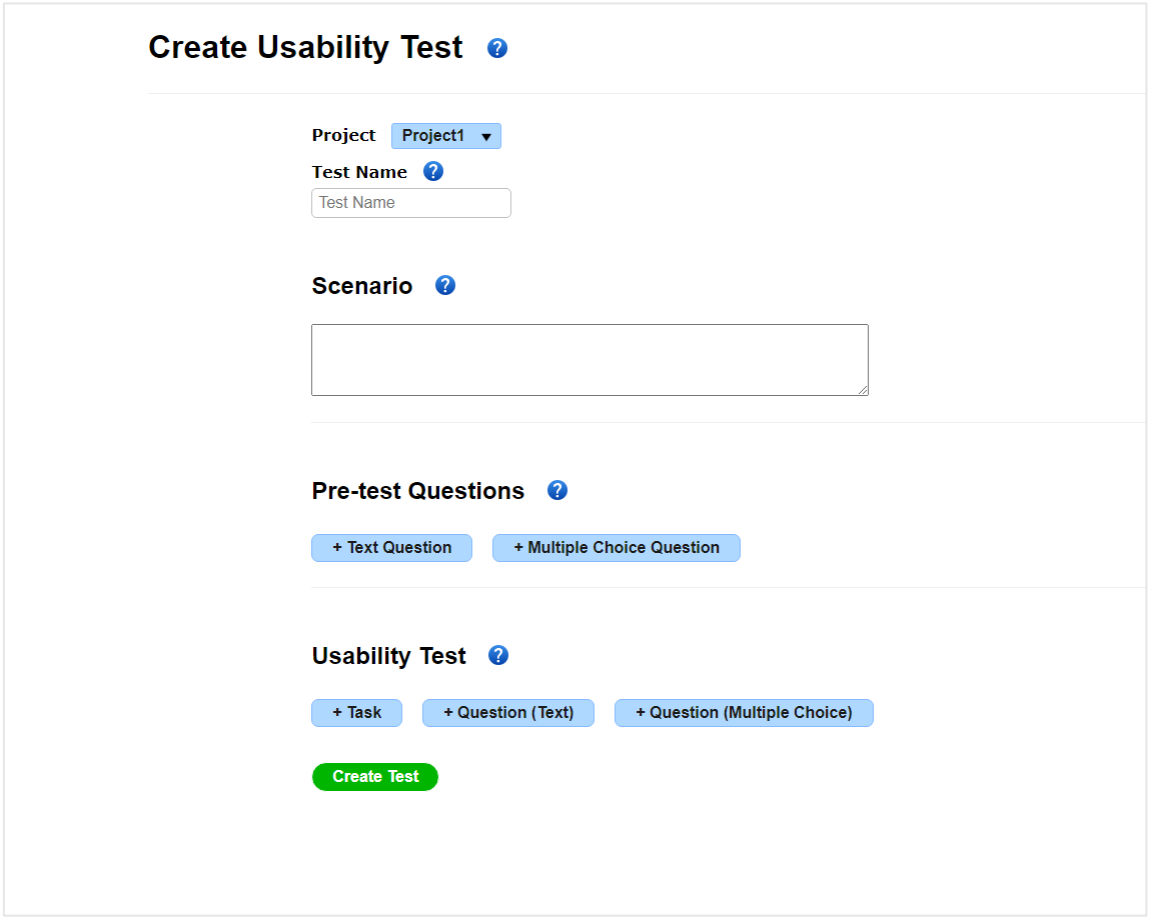


Figure 68 - Create Test (Initial Screen)

Examples of the information popups when the (?) button is pressed are shown below in Figure 69. 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).

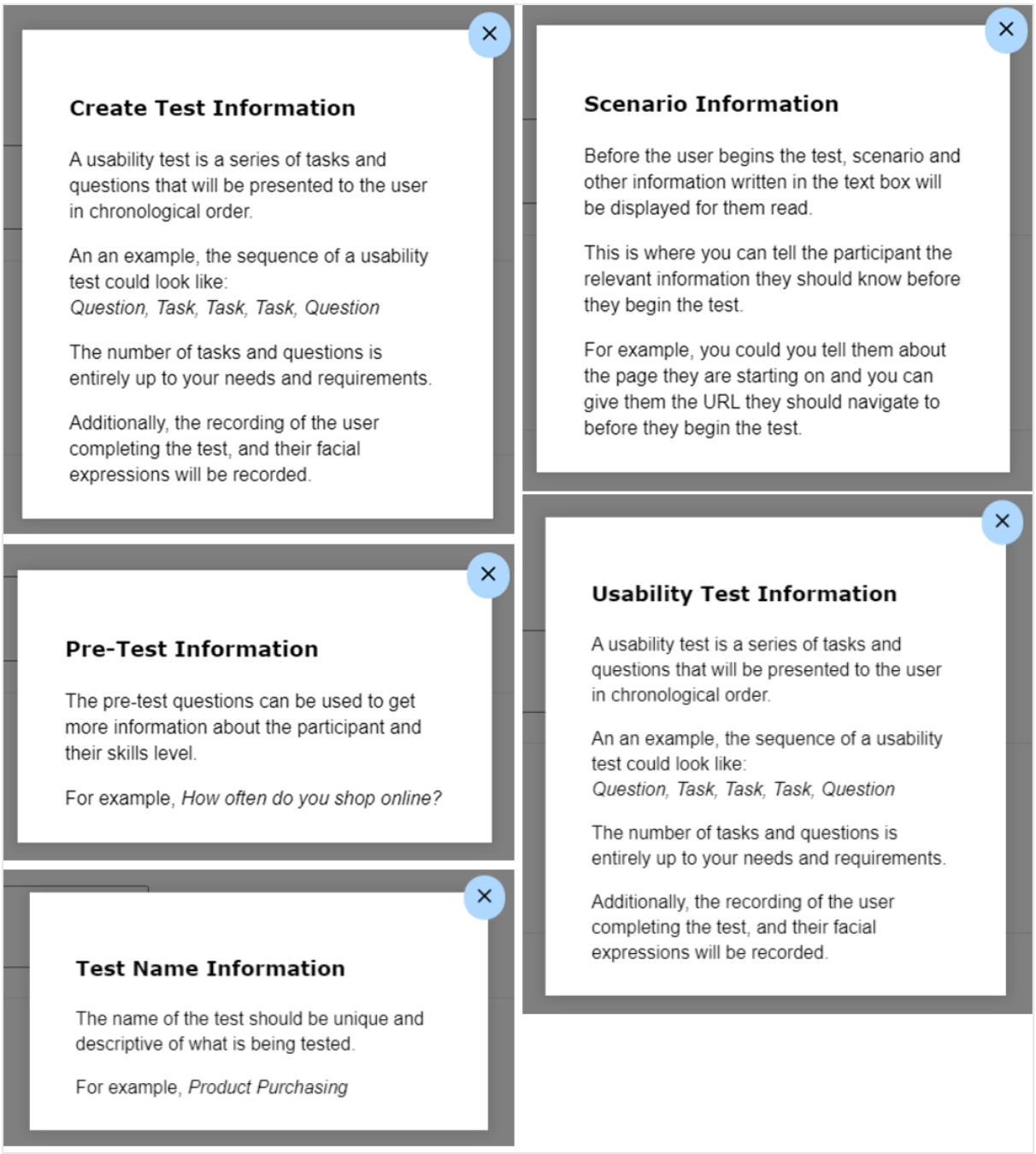


Figure 69 - Information Popups

The choice of questions within the usability study are “Text Answer” and “Multiple-Choice” questions. These can be seen in Figure 70. 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.

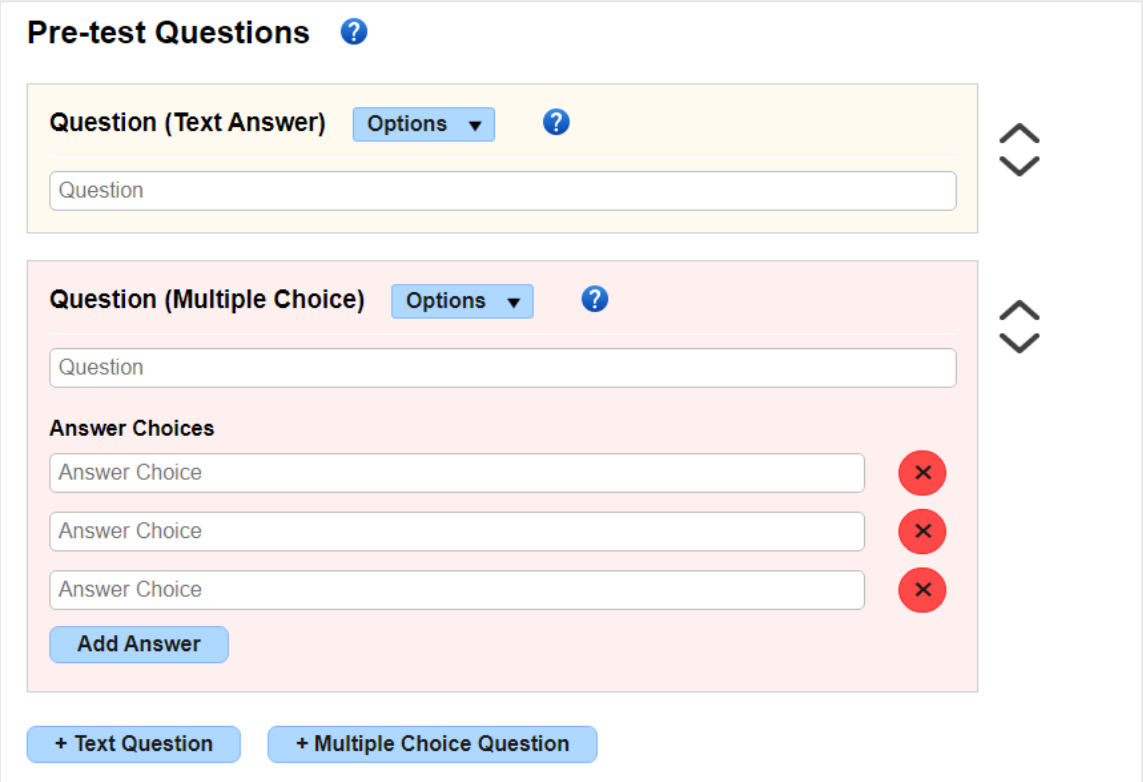


Figure 70 - Create Test (Pre-Test Questions)

The body of the usability study consists of tasks and questions for the participant to complete as can be seen in Figure 71. 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 any number of instructions for the participant to complete.

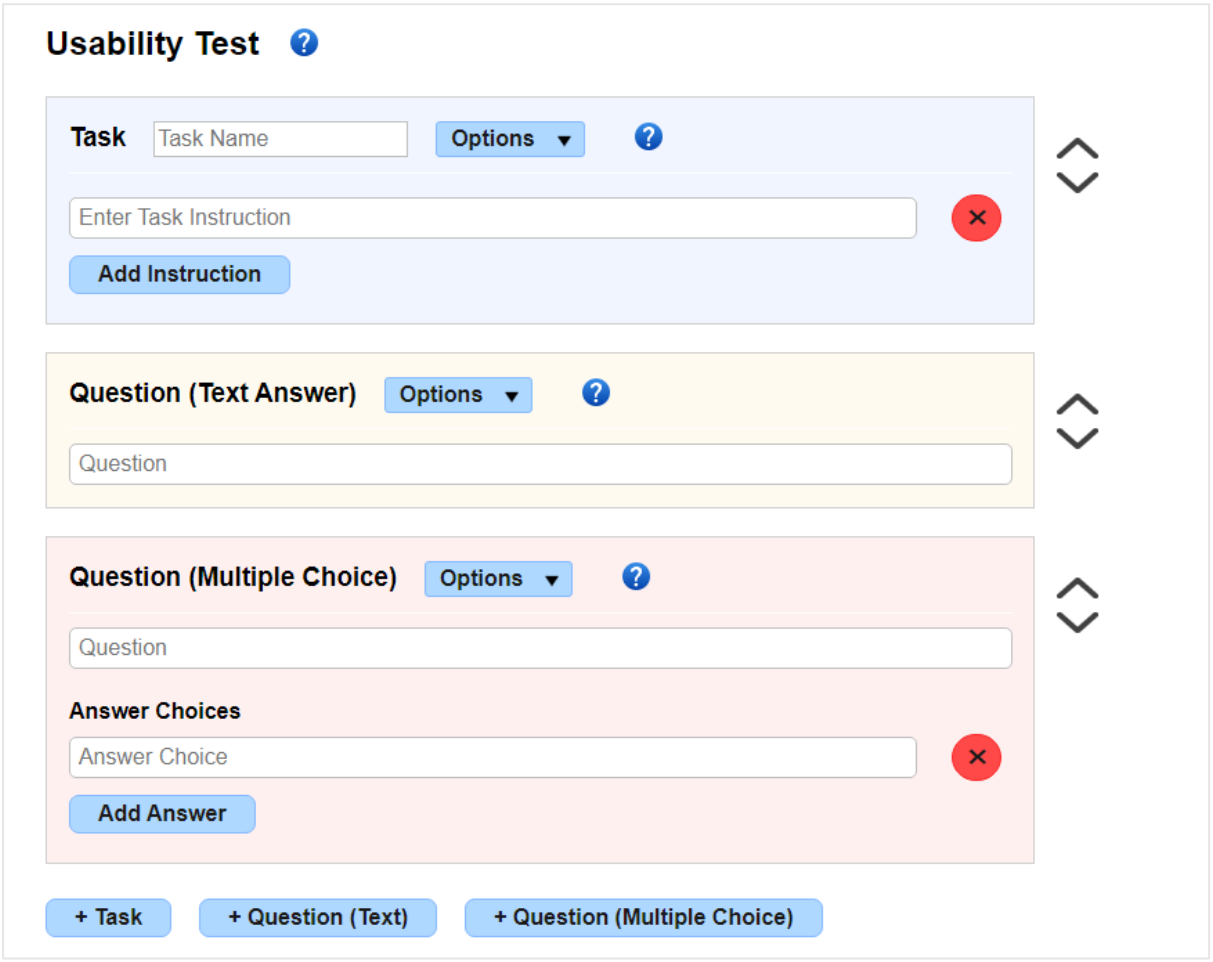


Figure 71 - Create Test (Body of Usability Study)

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 72.

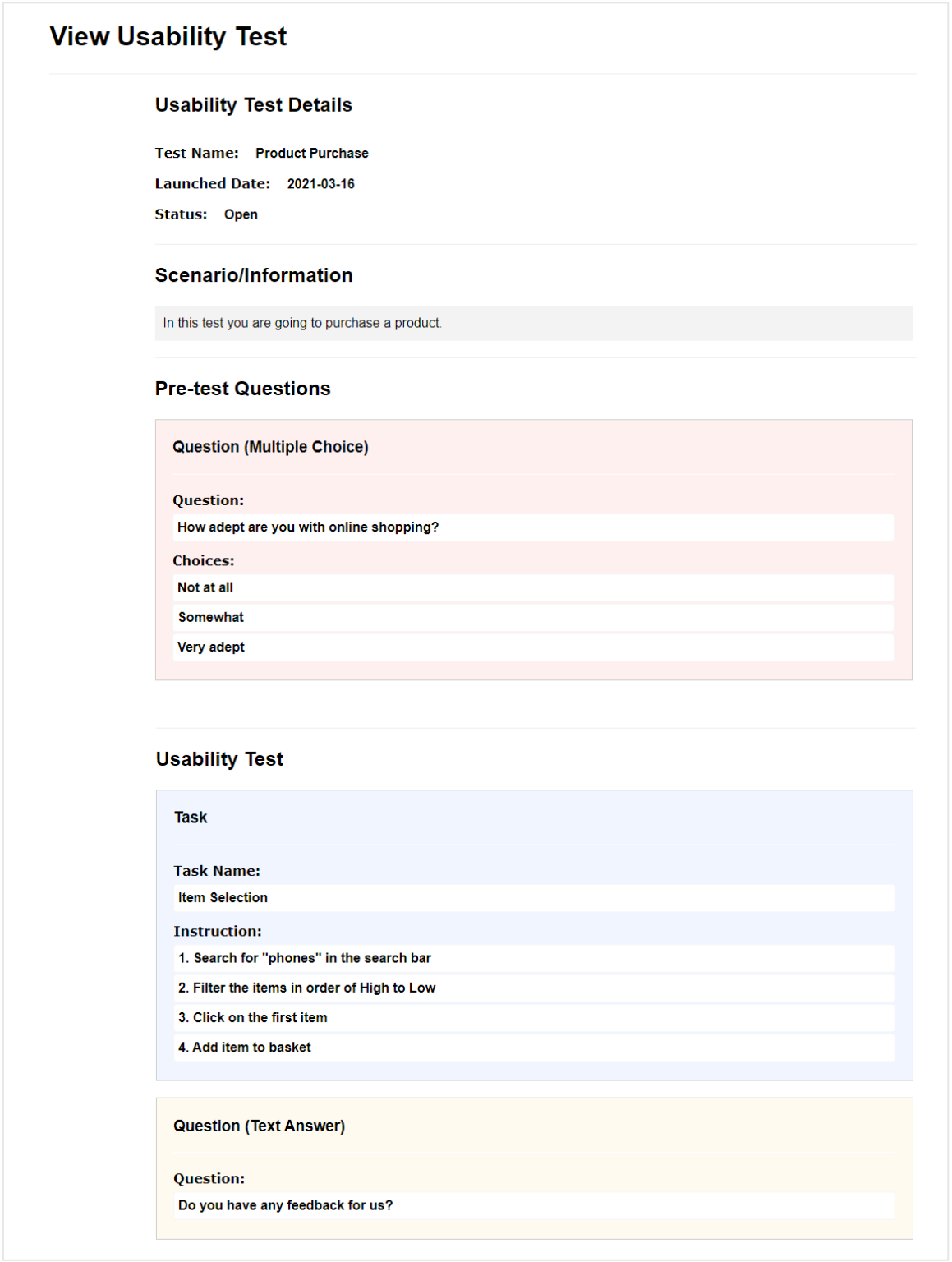


Figure 72 - View Test Details

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 first.

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 73 and Figure 74.

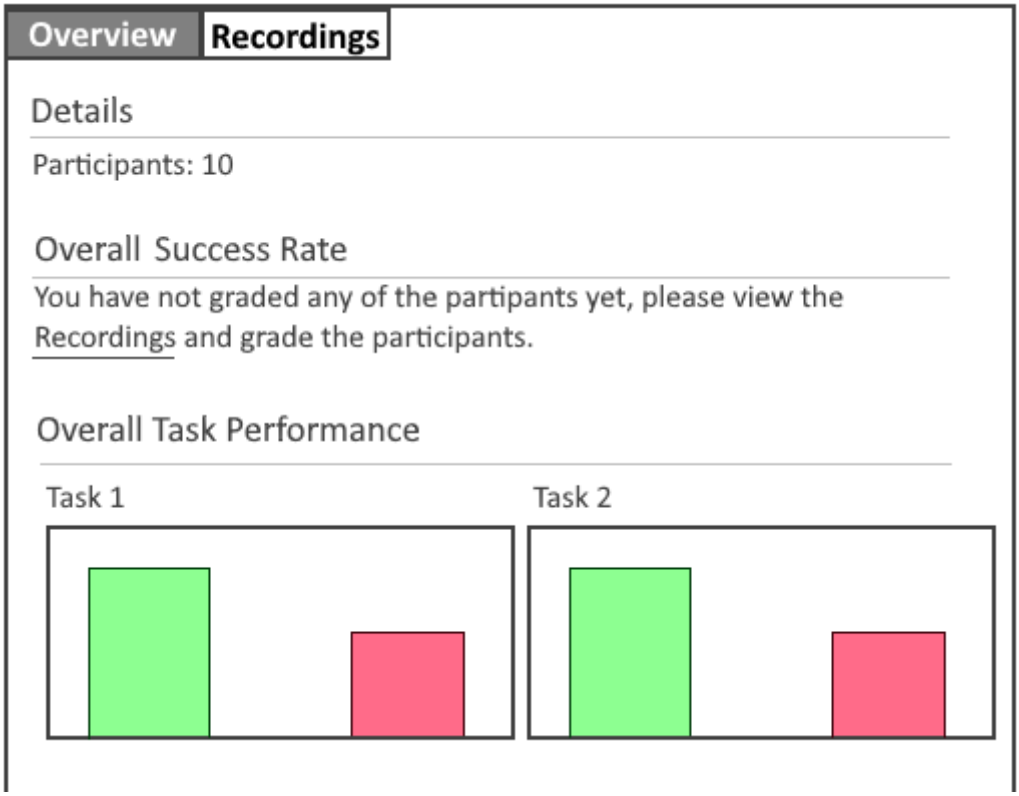


Figure 73 - Overview Tab New Design (Part 1)

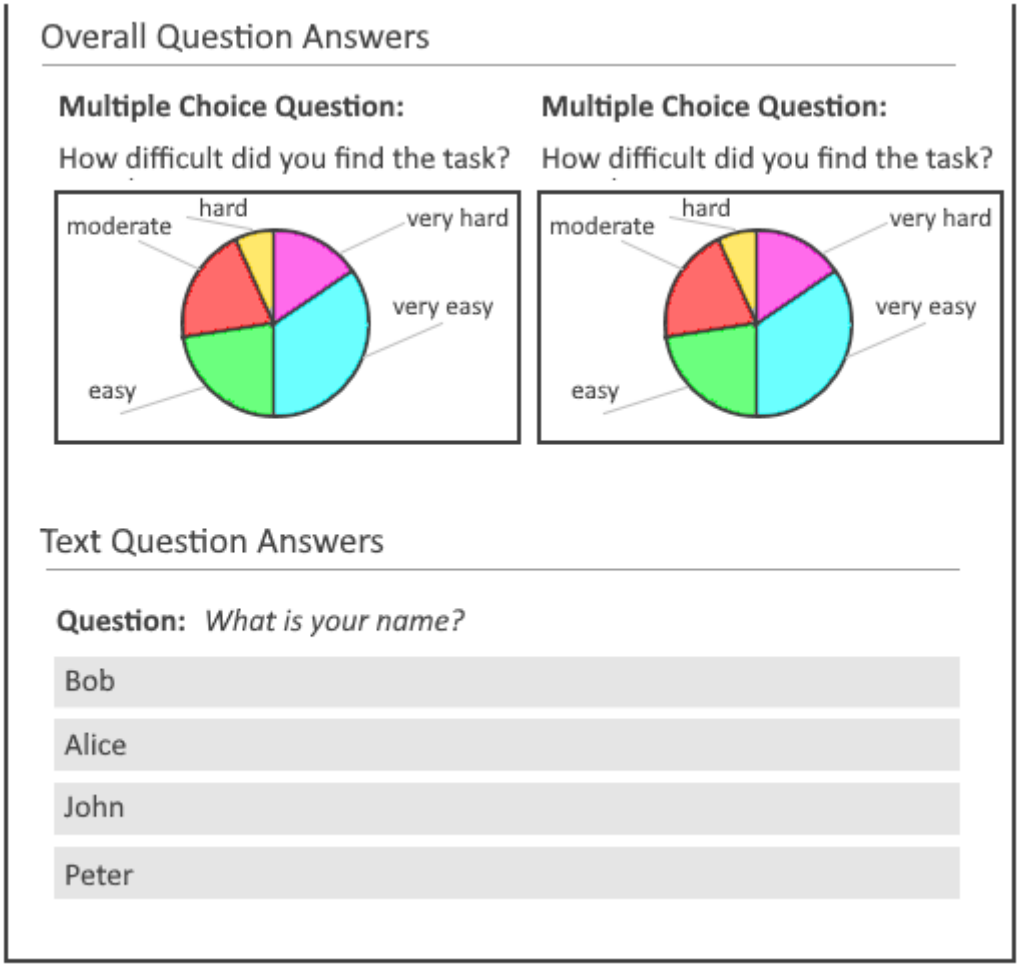


Figure 74 - Overview Tab New Design (Part 2)

The design shown in Figure 73 and Figure 74 is further improved upon after expert evaluations and research. For example, Gestalt Laws of Perceptual Organization [82] 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 75.

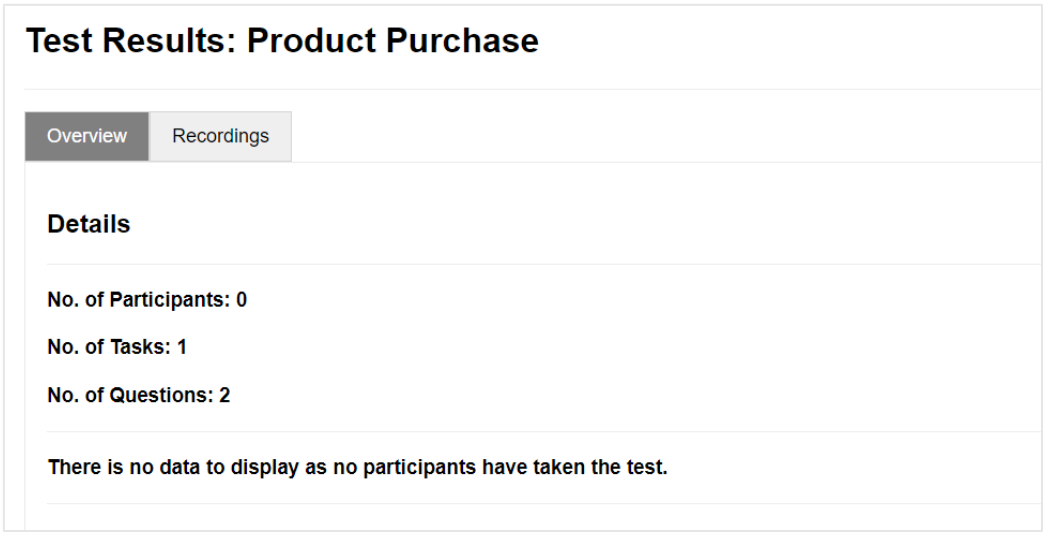


Figure 75 - View Test Results (No Data)

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 76 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*.

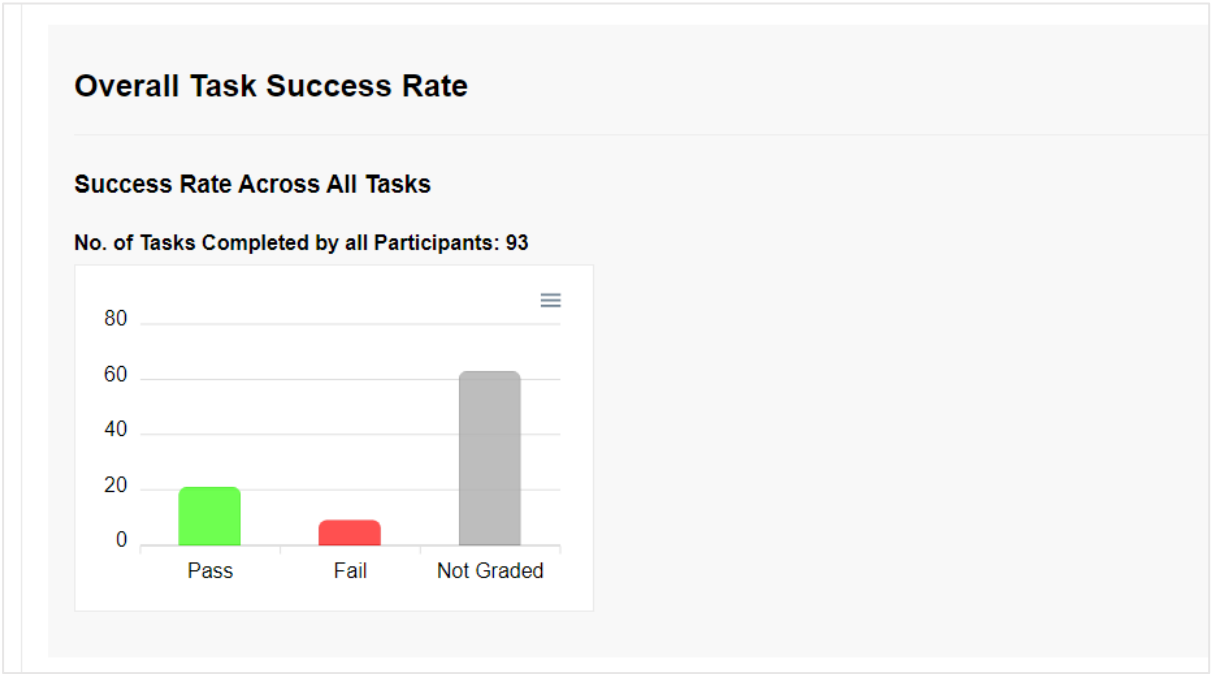


Figure 76 - View Test Results Overview (Overall Task Success)

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 correctly. 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 77. 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.

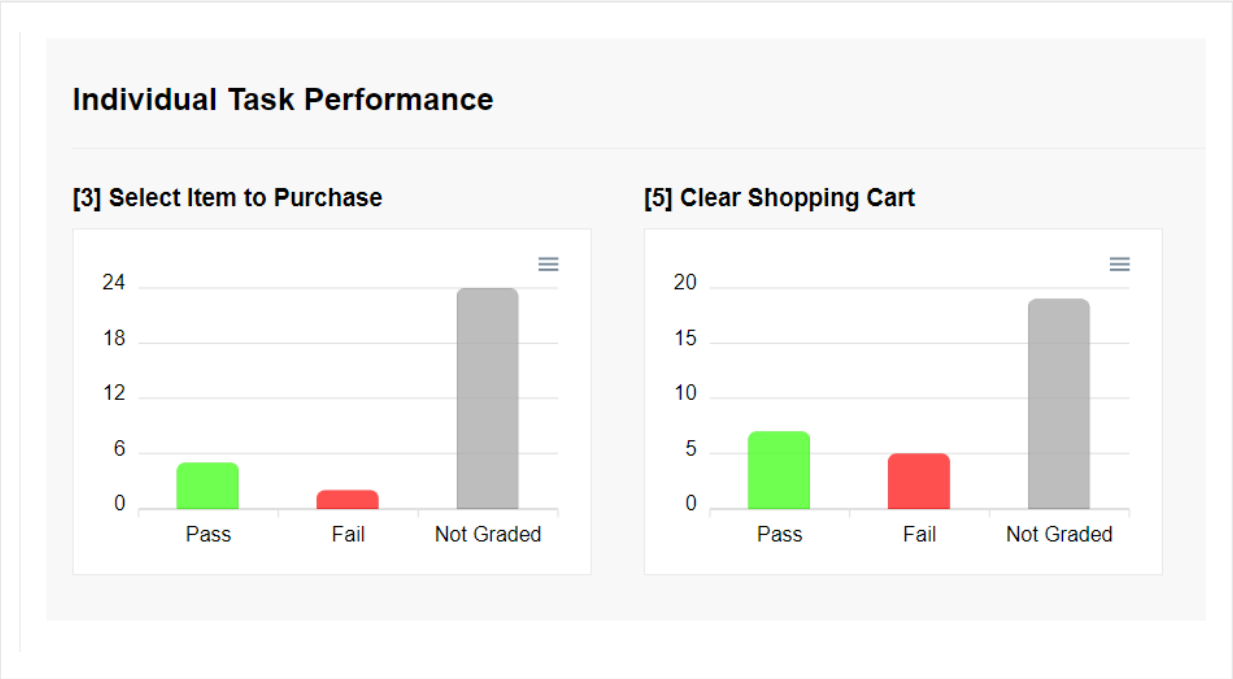


Figure 77 - View Test Results Overview (Individual Task Performance)

The answers to the multiple-choice questions is displayed as a pie chart as shown in Figure 78. The researcher can hover over the segments in the charts to get the exact value. For example, 38.7% is 12 out of 31.

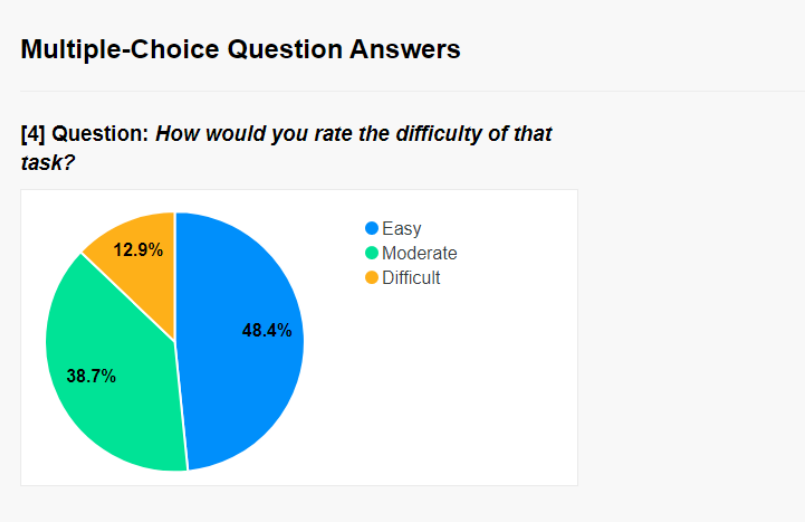


Figure 78 - View Test Results Overview (Multiple-Choice Question Answers)

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 79. The researcher can scroll and read the reply each participant has given.

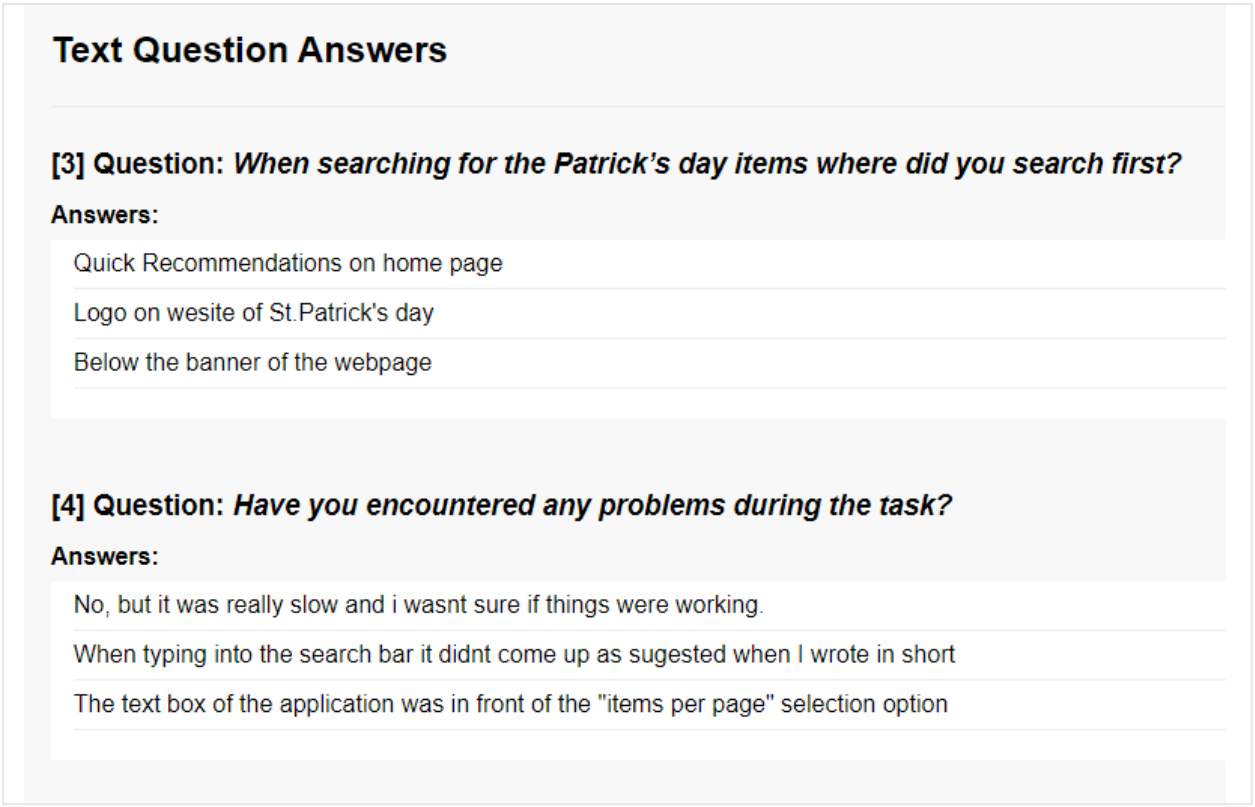


Figure 79 - View Test Results (Answers to Text Questions)

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 80 which closely resembles the final result.

Figure 80 - Recordings Tab New Design

The recording of the video, the emotion timestamps and task timestamps can be seen in Figure 81. The JavaScript class which contains the tab content is *testResultsRecordings.component.js*.

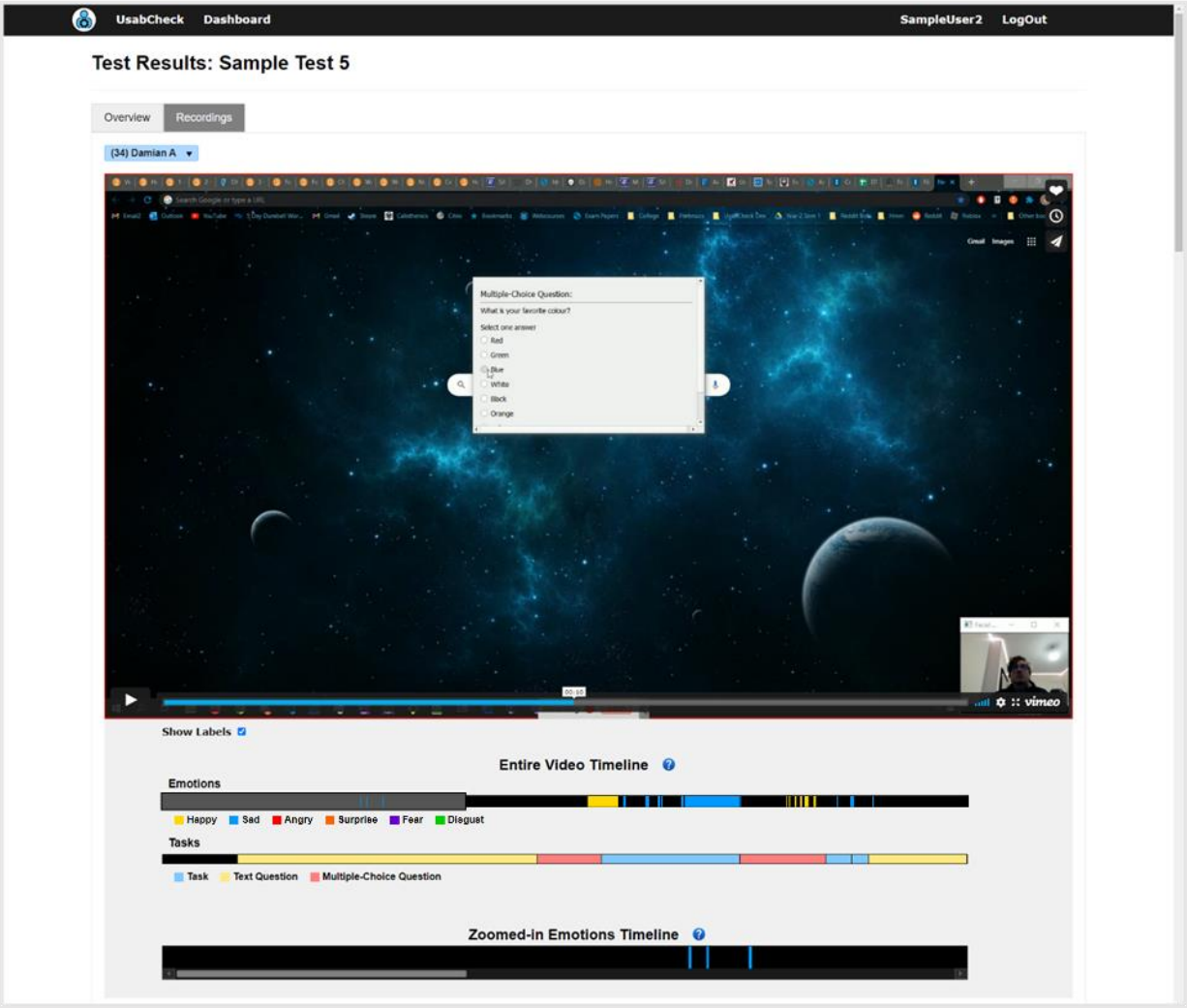


Figure 81 - View Test Results Recordings (View Video)

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 type 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 locatated in the bottom right corner of the video.

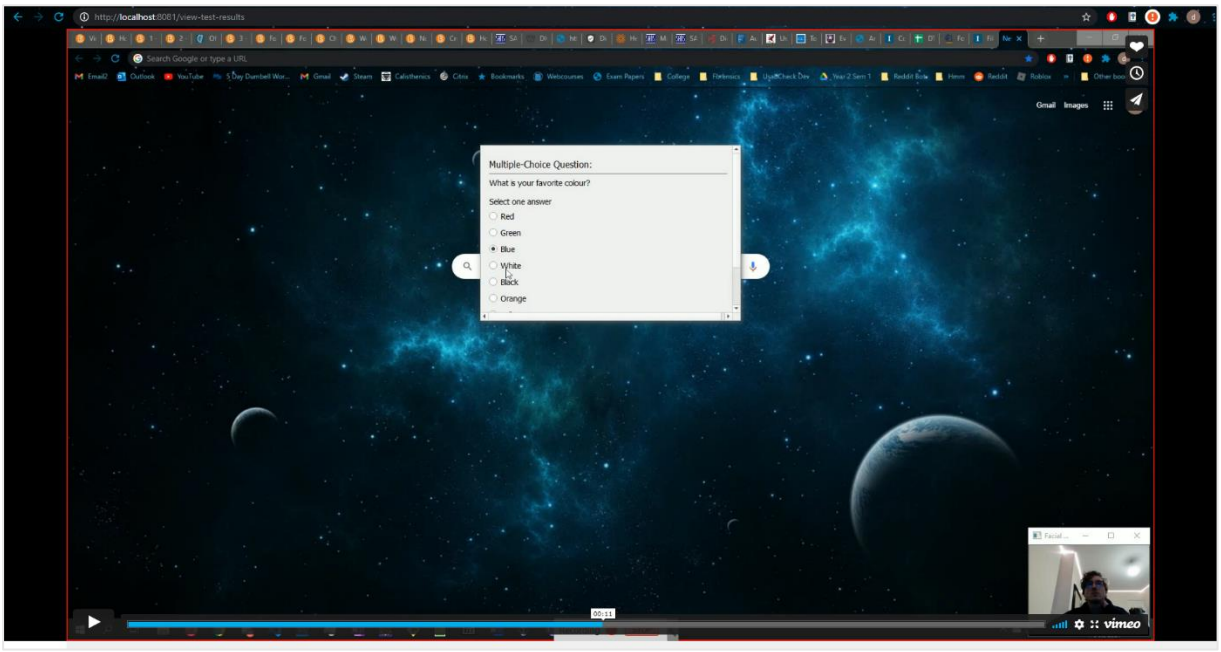


Figure 82 - Video Test Results Recordings (Video Full-Screen)

The displaying of the emotion data and the timestamps of the participant progressing through the usability study is shown in Figure 83. The design in Figure 83 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 during 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 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*.

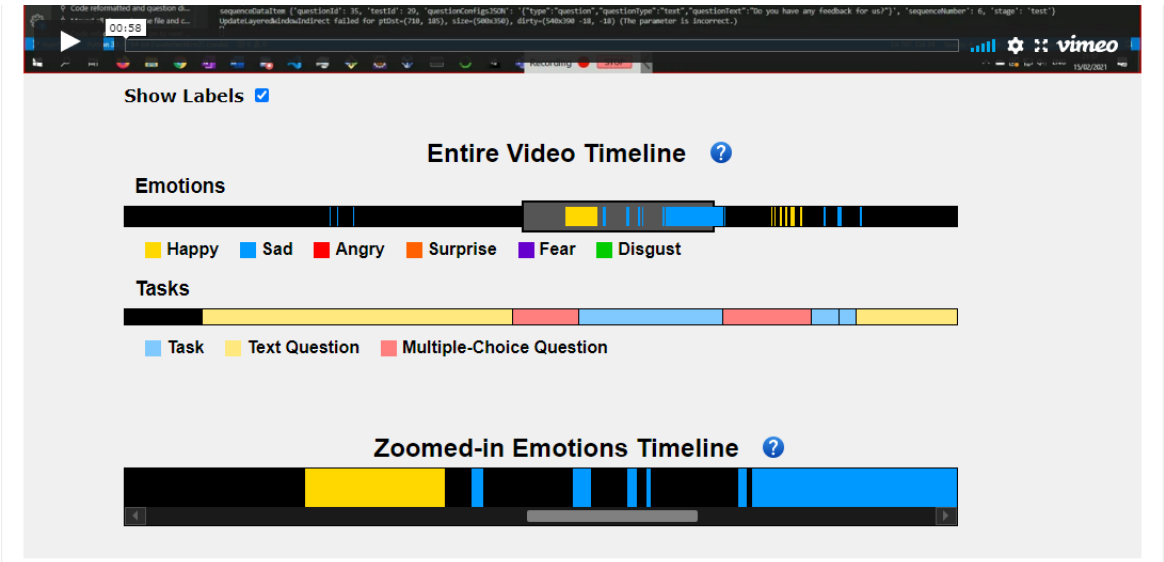


Figure 83 - Video Timeline Bars

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 from the video to the video bars. The timeline charts with hidden labels are shown in Figure 84. It should also be noted that the video bars scale automatically with the size of the screen and are always in line with the progress bar in the video.

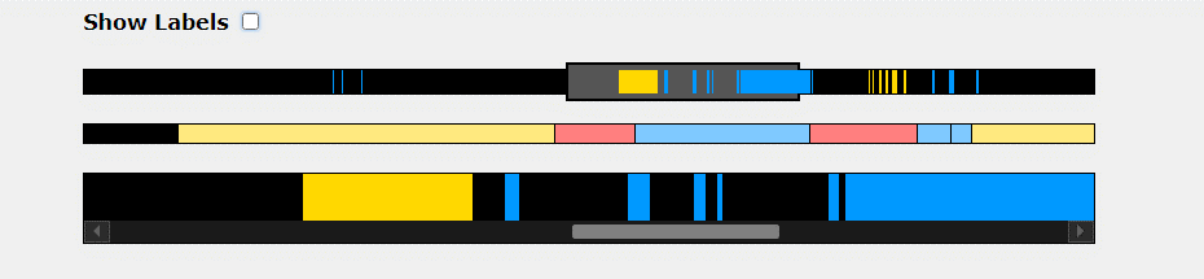


Figure 84 - Video Timeline Bars (No Labels)

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 as separate from the zoomed in bar.

To better understand how people, associate emotions and colours a dissertation by Gohar Kadar, Navit 2008 [83] was reviewed. Below in Figure 85 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.

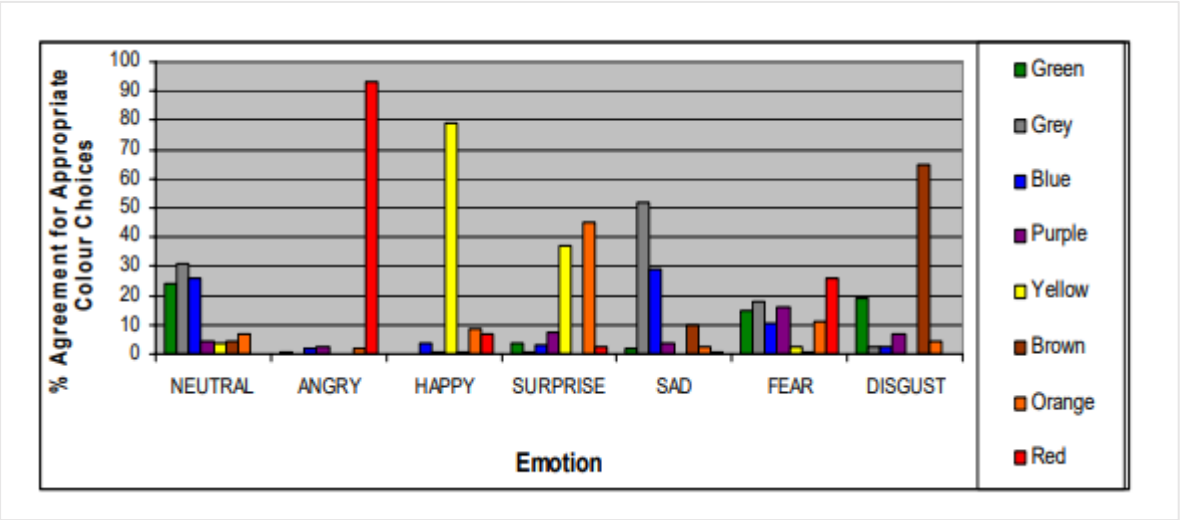


Figure 85 - Percentage agreement of Appropriate colour choices for each of the emotional categories

When developing the web application and before conducting any research the colours were assigned as shown in Figure 86. After researching colours and emotions the final colour association used is displayed in Figure 87.

Figure 86 – Colours Personal First Guess

Figure 87 - Colours After Research

The task grading is displayed in Figure 88. 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.

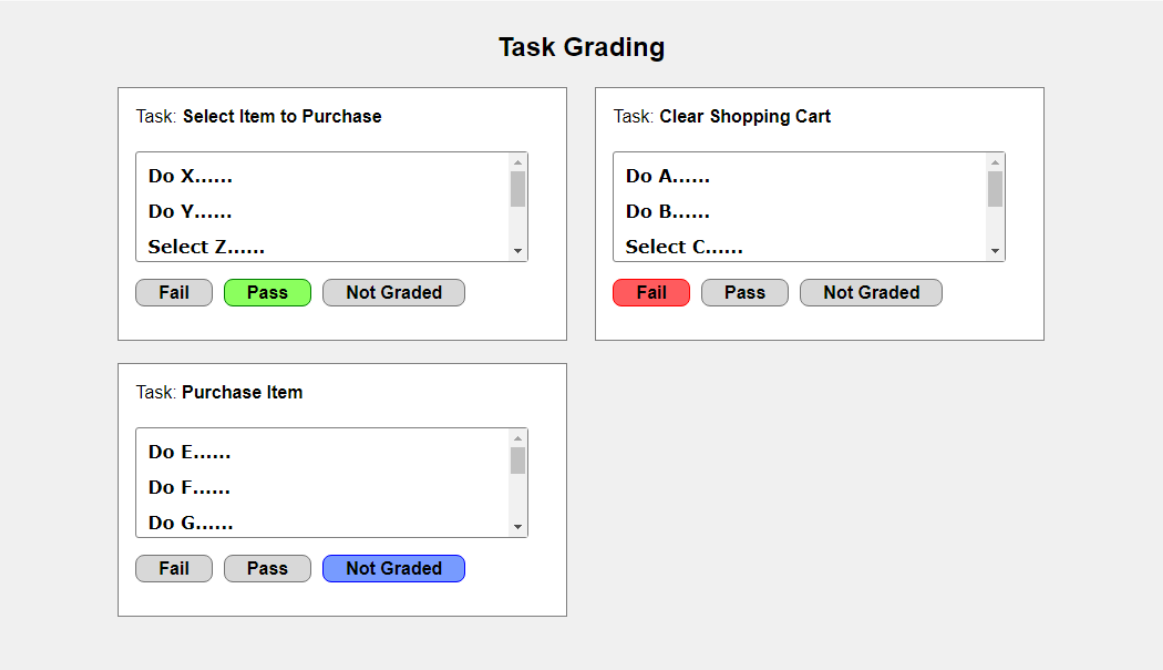


Figure 88 - Test Results Recording Tab (Task Grading)

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 89 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 90. A library was used to display these.

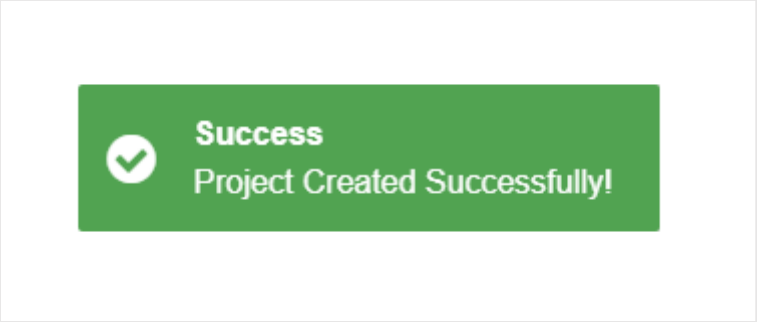


Figure 89 - Notification Updates (Successful)

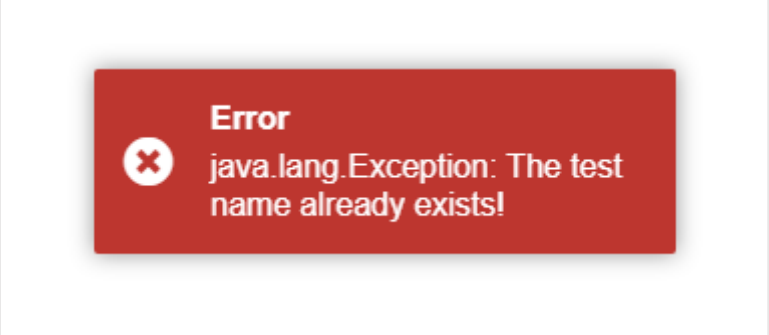


Figure 90 - Notification Updates (Failed)

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 91. This was achieved with a html attribute.

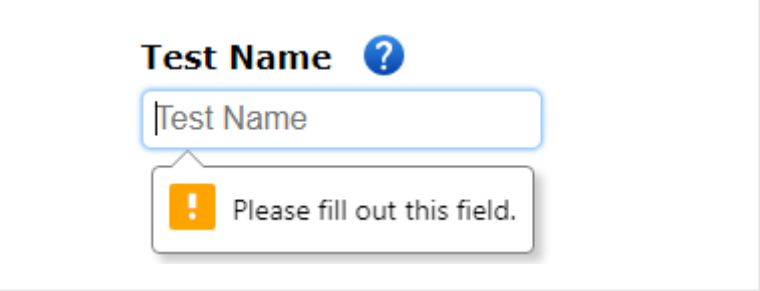


Figure 91 - Form Validation

Web App Deployment

The web application was first built and compiled with the help of the frontend-maven-plugin [84] that runs the build of the frontend React application and with the help of the maven-antrun-plugin [85] 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 [86] had to be used to allow for deploying from a sub-directory. The web application was deployed to https://usabcheck.herokuapp.com/.

The reason Tomcat has not been used to deploy the project as initially planned is because the project did not compile correctly into the form of a WAR file which tomcat required and Heroku was a better alternative in terms of the automation and cost. Database hosting was also required as the development database was localhost. CloudClusters.io provided a free trail 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 92. The main program, *Main.py* is responsible for spawning of the windows and components, retrieving data, and uploading data.

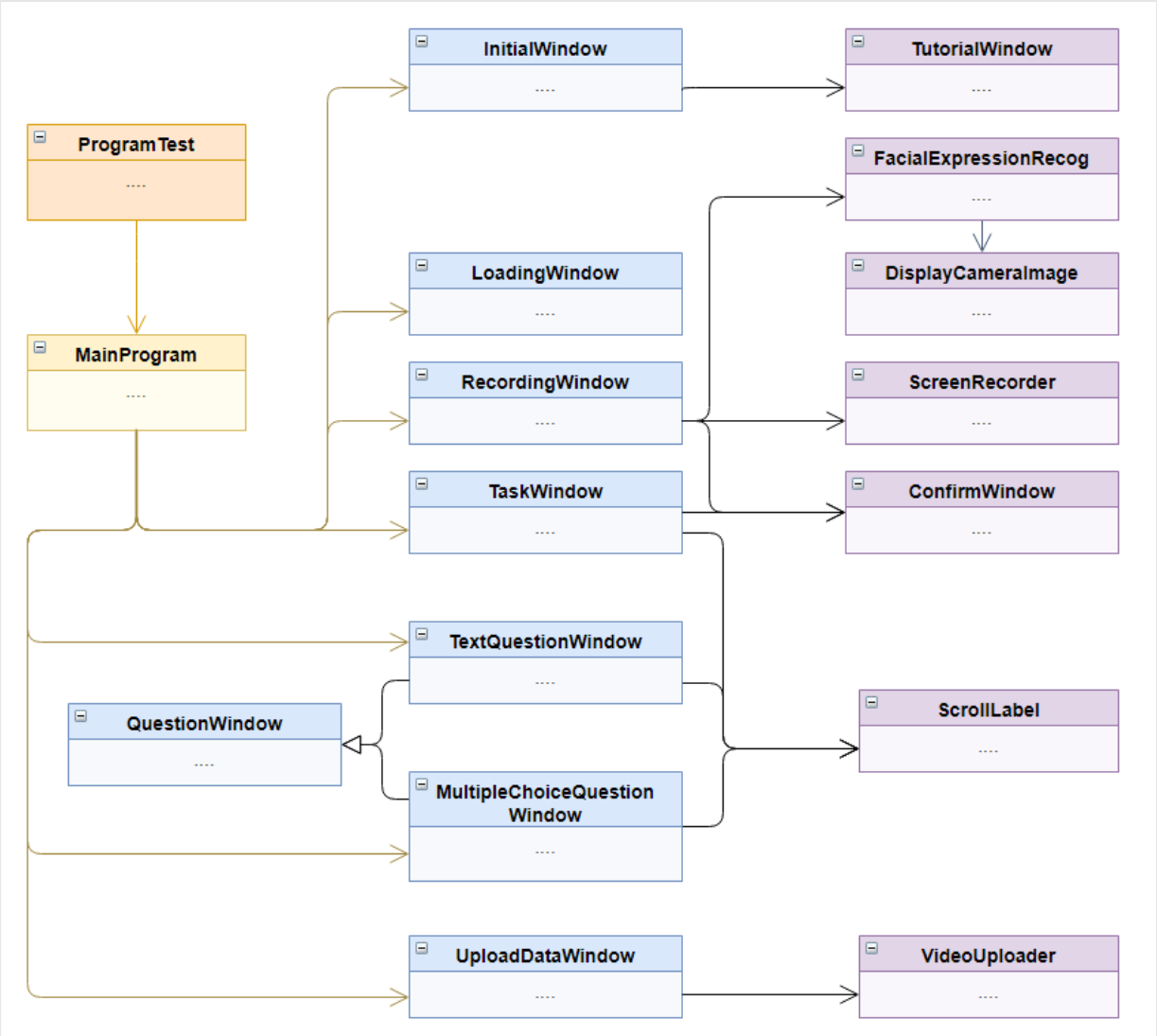


Figure 92 - Local Application Classes

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 93. After the data has been retrieved from the server it is displayed as shown in Figure 94. The test details shown include the test name, created date, the number of tasks and questions, and the scenario. 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 for this window is *InitialWindow.py*.

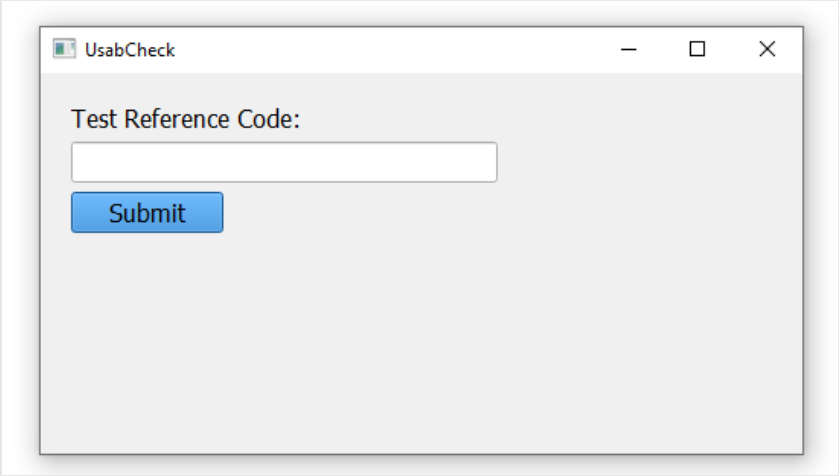


Figure 93 - Initial Screen

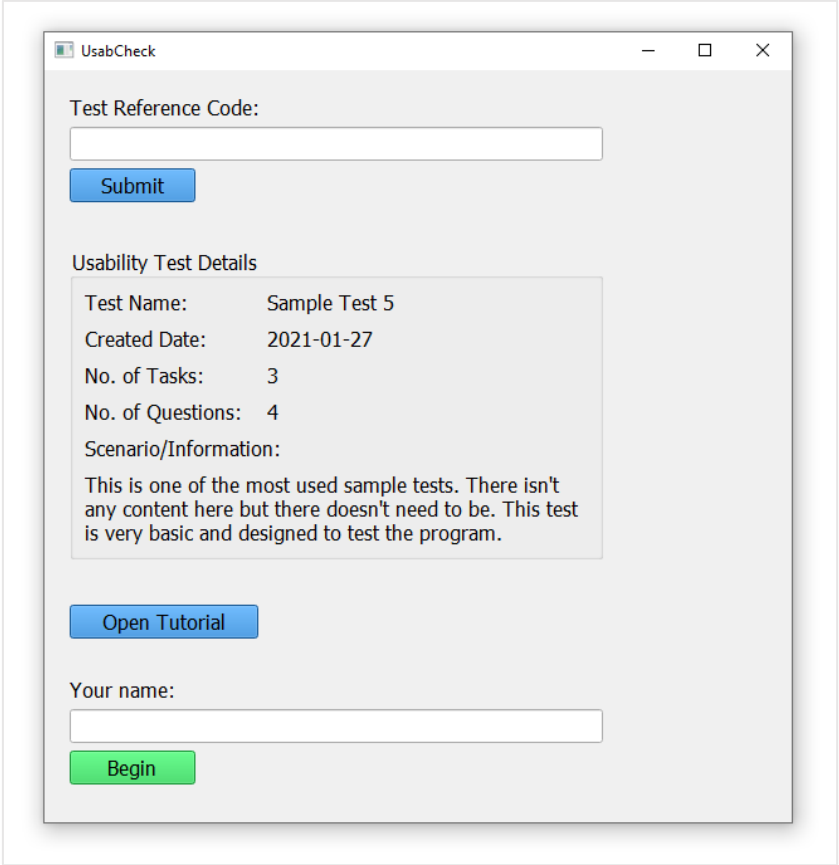


Figure 94 - Initial Screen (After Data Retrieval)

Tutorial Window

The user can click the “Open Tutorial” button shown in Figure 94 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 95 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.

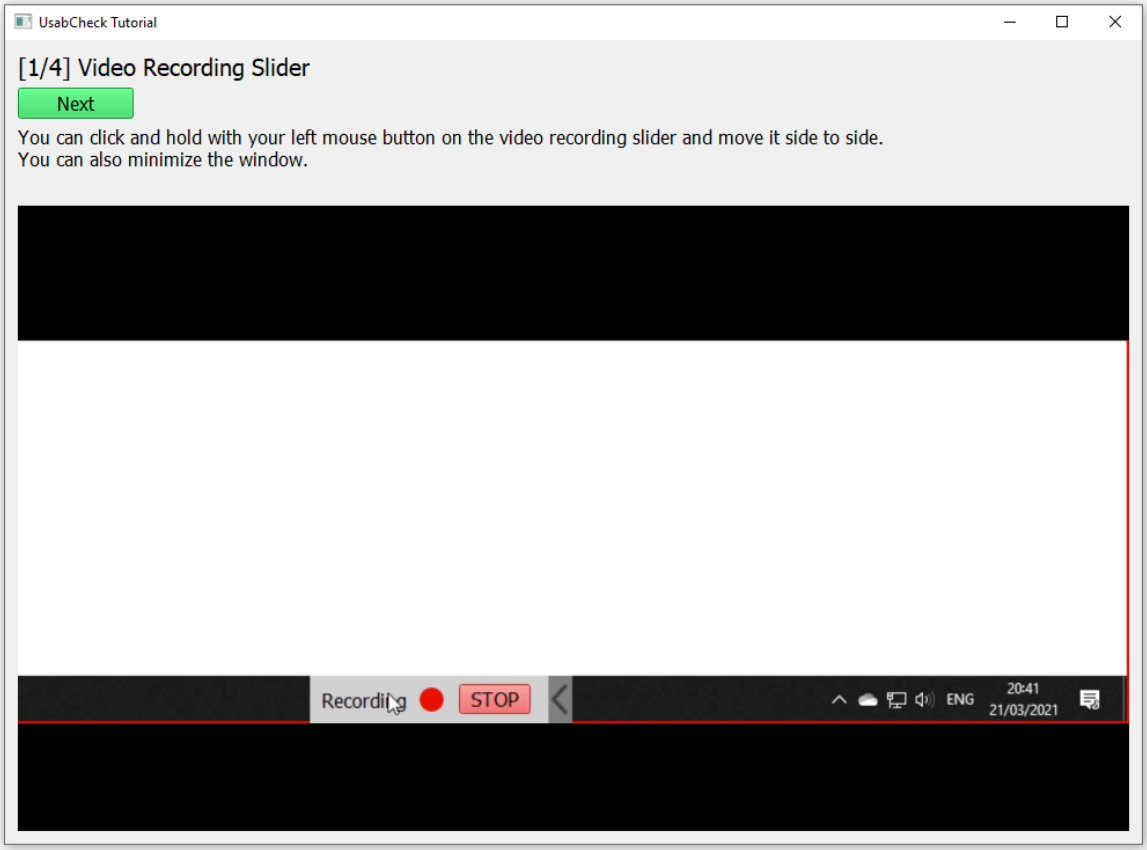


Figure 95 - Tutorial Window (Video Recording Slider)

Loading Screen

After the "Begin" button is clicked the application is going to connect to the camera and start 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 96 and Figure 97. 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.*

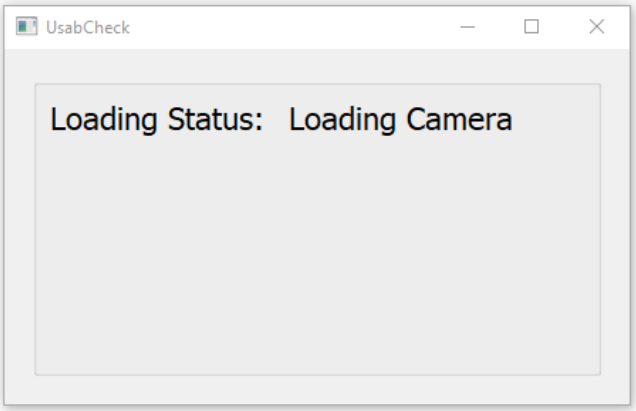


Figure 96 - Loading Screen (Loading Camera)

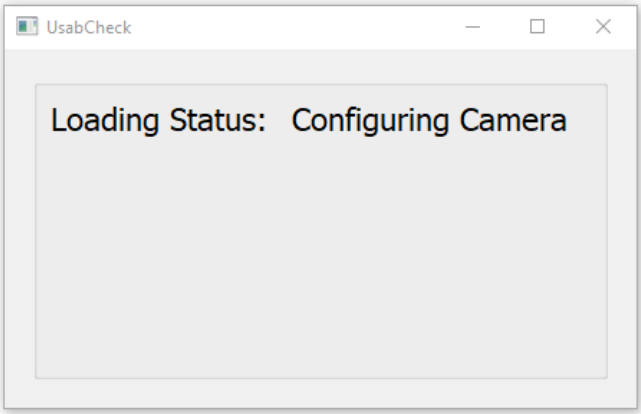


Figure 97 - Loading Screen Configuring Camera

Screen Recorder

The screen recorder GUI is shown in Figure 98. 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 98 - Screen Recorder (Full Desktop View)

A close-up of the screen recorder GUI is shown in Figure 99. 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 100.

Figure 99 - Screen Recorder (Components)

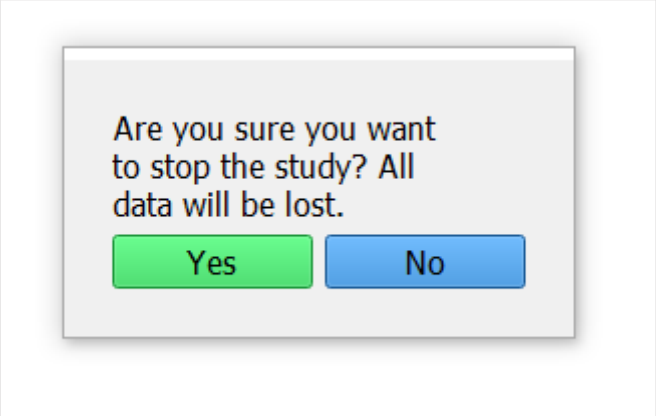


Figure 100 - Confirm Popup

Facial Expression Recognition

For reference, the development of the machine learning models is discussed in section 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, making the prediction, and finally storing the data in a variable.

An example of the data which was recorded is shown in Figure 101. 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 101 - 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 102.

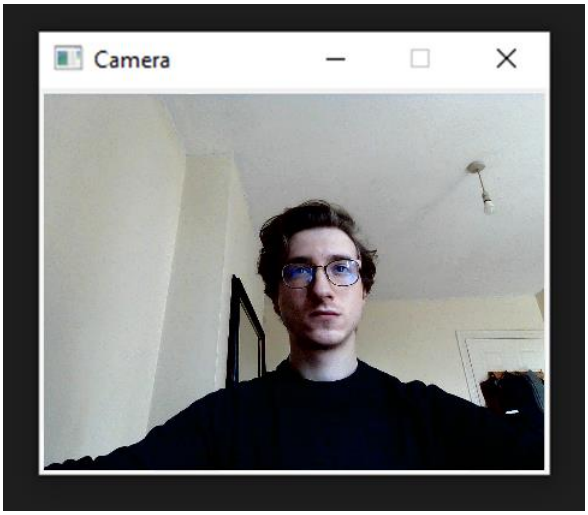


Figure 102 - Camera Feed Window

Question Window

One type of question that can be asked is multiple-choice questions which is shown in Figure 103 and Figure 104. 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).

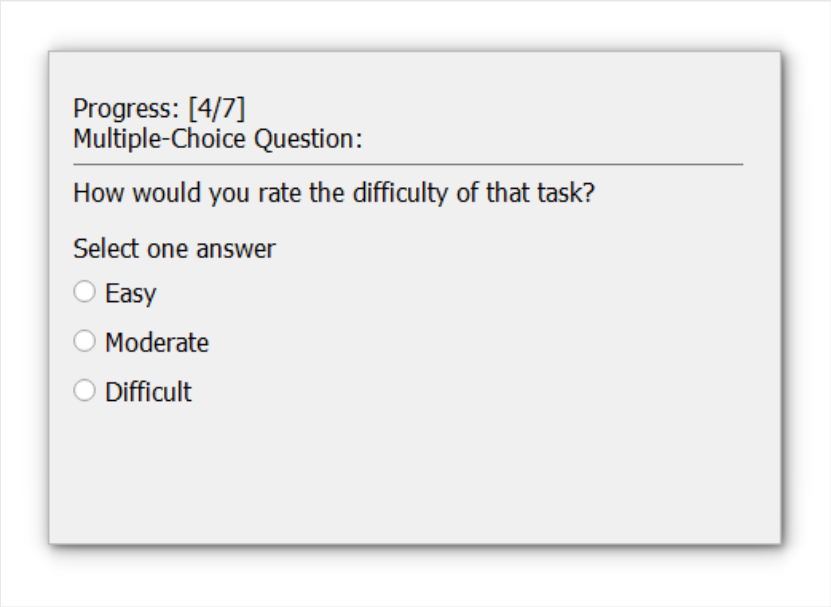


Figure 103 - Multiple-Choice Question Window

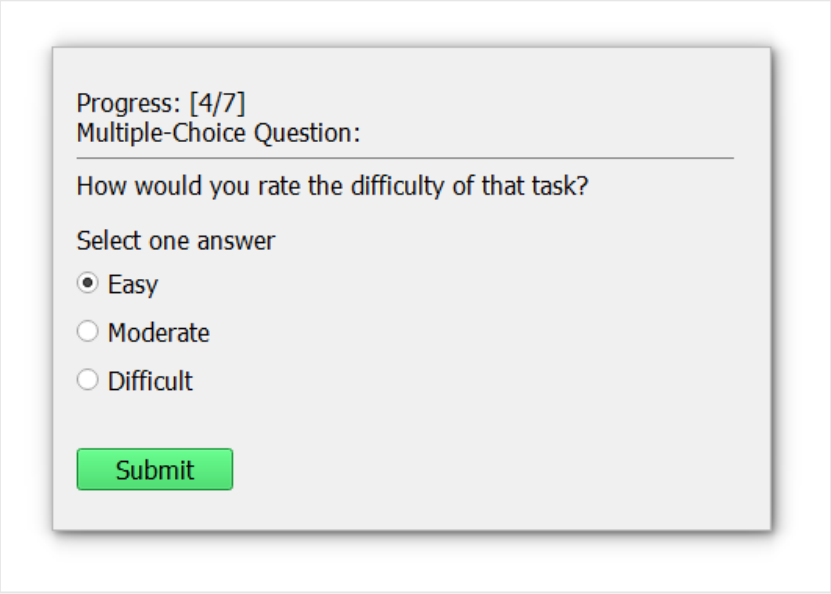


Figure 104 - Multiple Choice Question Window (After Selection)

The text answer question is displayed in Figure 105. The user enters any text into the window as their answer and this question type is intended for open-ended questions.

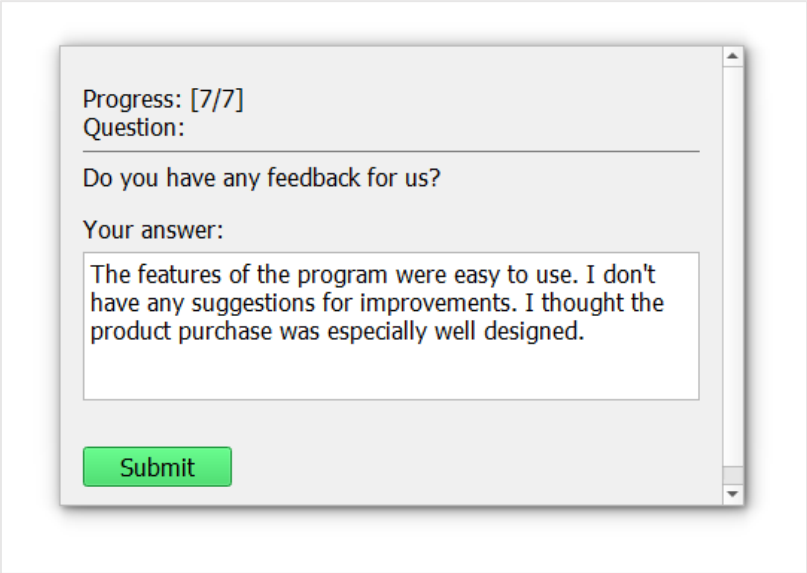


Figure 105 - Text Question Window

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 competes 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 the "show" 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.

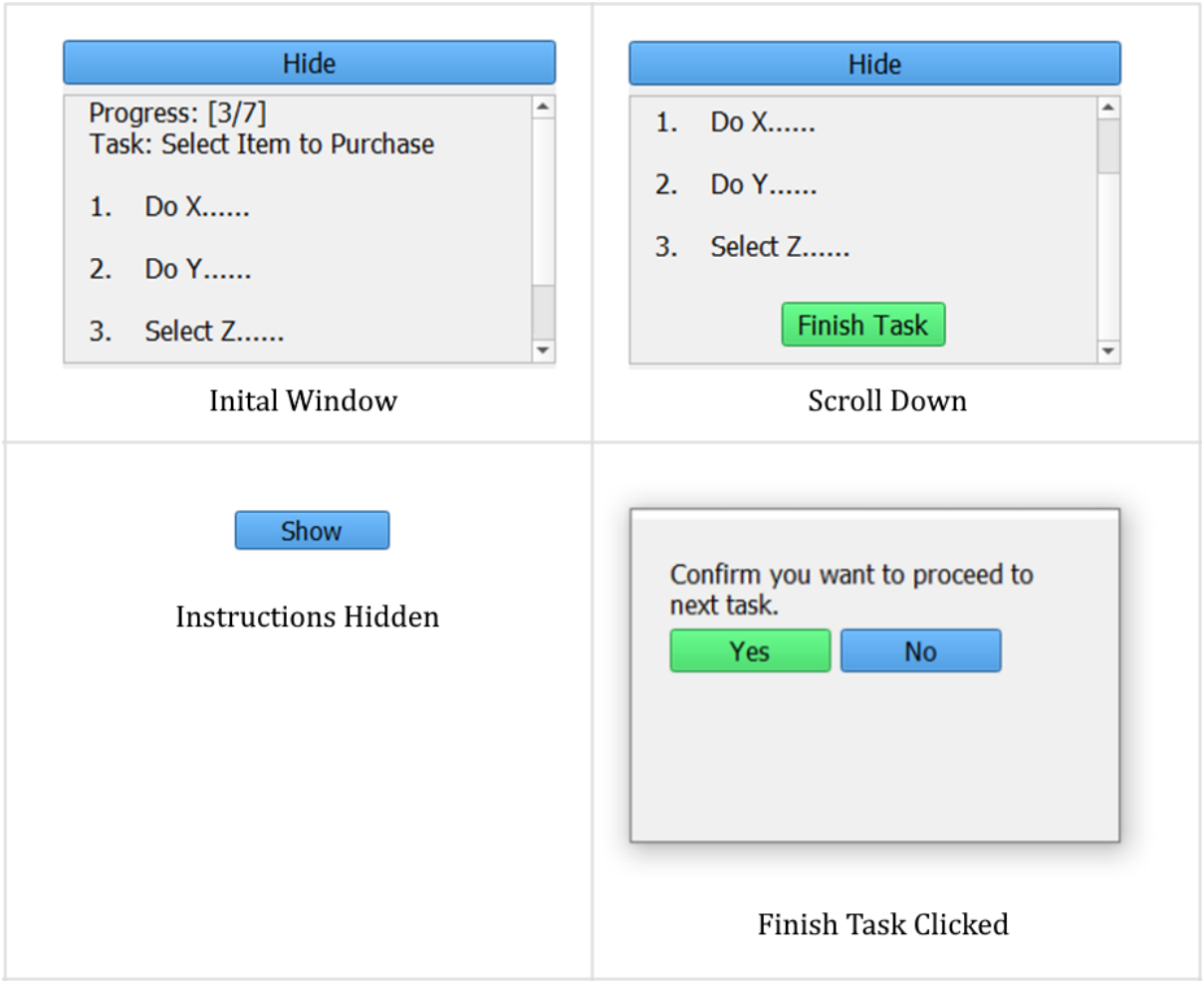


Figure 106 - Task Window States

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 107. 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 completed 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.

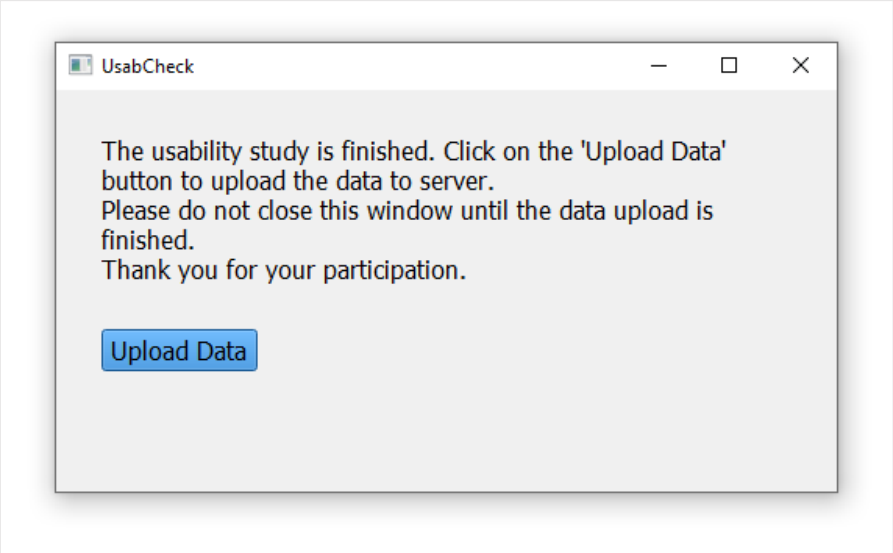


Figure 107 – Data and Video Uploading (Initial Screen)

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 108. Once the video has finished uploading the message changes and notifies the user that the window can now be closed as shown in Figure 109.

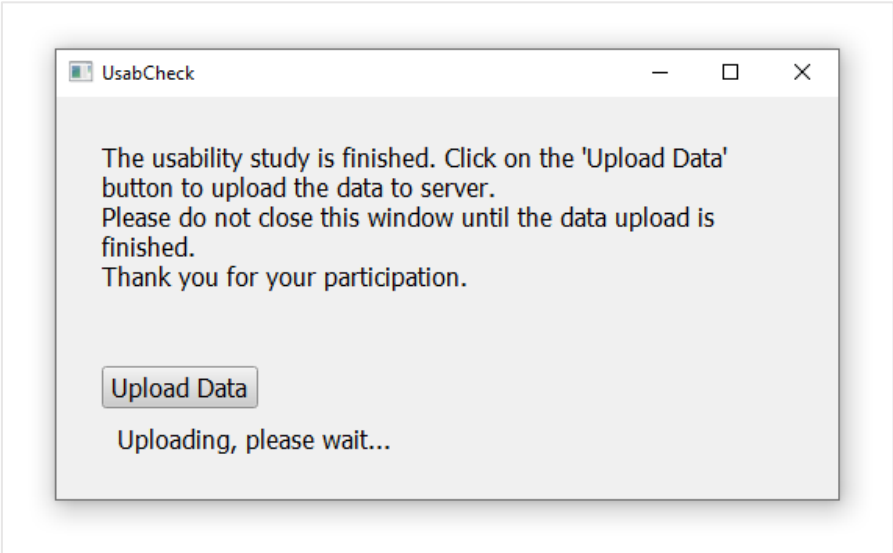


Figure 108 - Video Uploading (Uploading Stage)

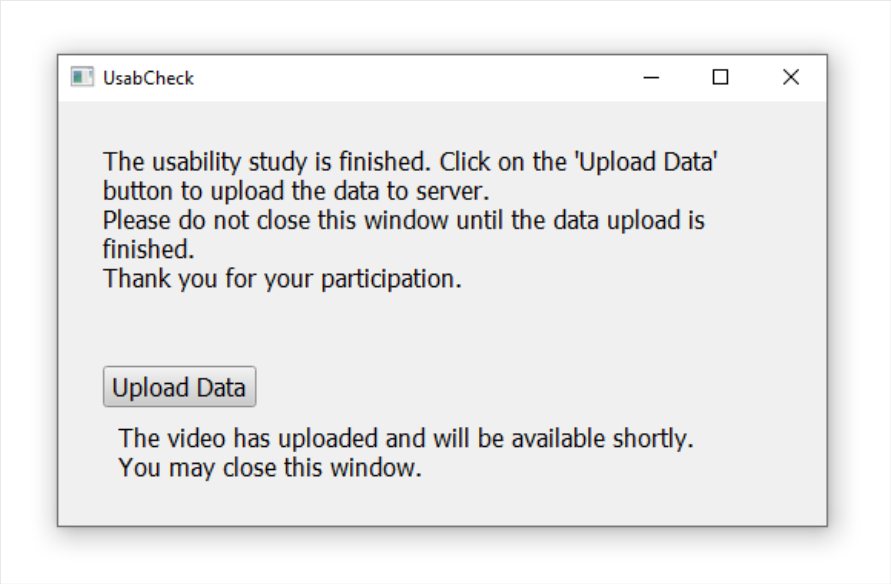


Figure 109 - Video Uploading (Finished Stage)

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 reasons 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.

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 111. The *pixels* column consists of 2304 (48 x 48) pixel values which are converted into an image. Figure 110 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 machine training input consistency reasons a CSV to image converter was implemented to convert the pixel values to a grayscale image.

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

Figure 111 – FER2013 Dataset CSV File

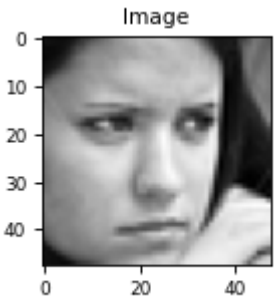


Figure 110 – FER2013 Pixel Values as Image

Below is the code for opening the CSV file, obtaining the relevant information, and saving the image as a .jpg file. 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 112.

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

Figure 112 – 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 113.

```
# 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 113 – 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 114.

```
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 114 - CSV to Image Converting and Exporting

The final end results is images stored in their respective folders as shown in Figure 115 below. For example, All the images labelled “Happy” are in the “Happy Folder”.

Figure 115 – 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) [10] 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 [87]. 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% [88]. The model was trained for 25 epochs.

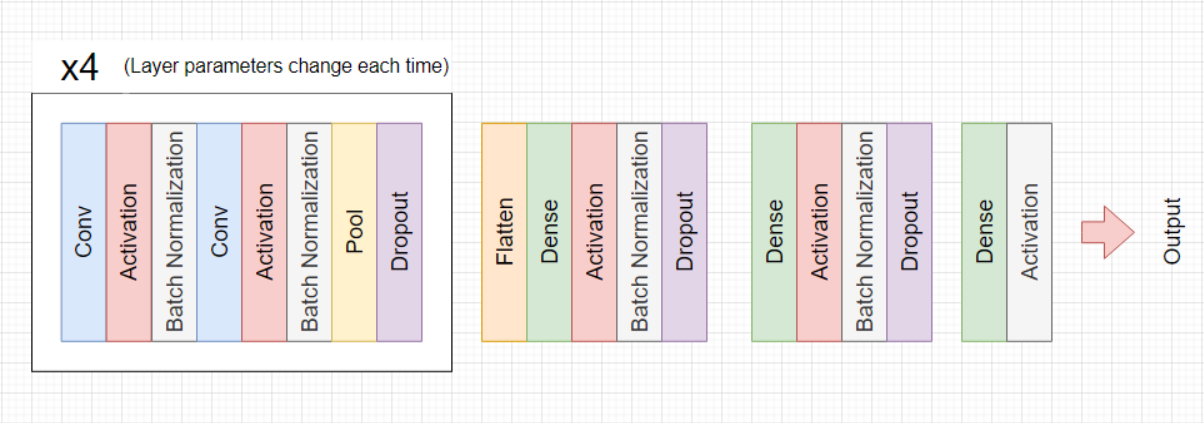


Figure 116 - Model 1 Diagram

The model has been trained to recognise 7 different emotions. Below Figure 117 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 118. 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 117 – First Model Training

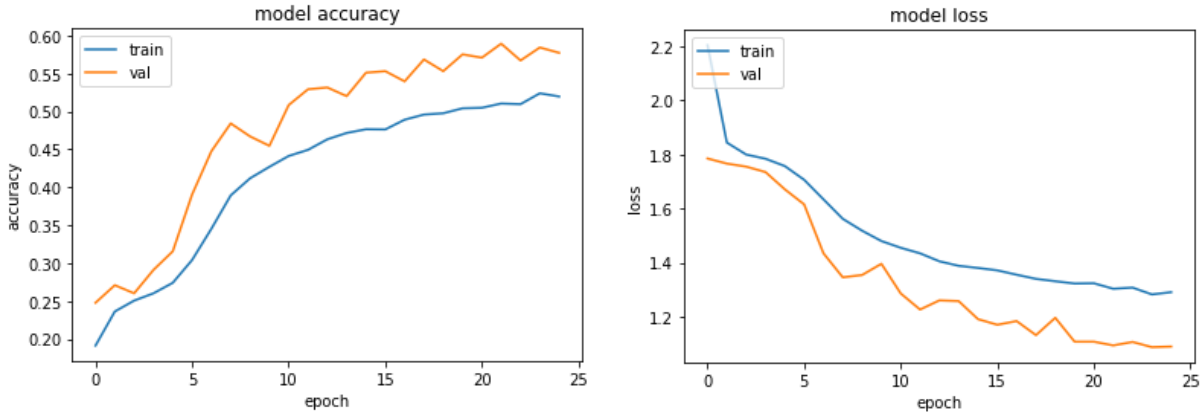


Figure 118 – First Model Accuracy and Loss

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 [89] the CNN architecture was different to the first model and is illustrated in Figure 119. 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 120. 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 119. The accuracy increase as the model trains is displayed in Figure 121.

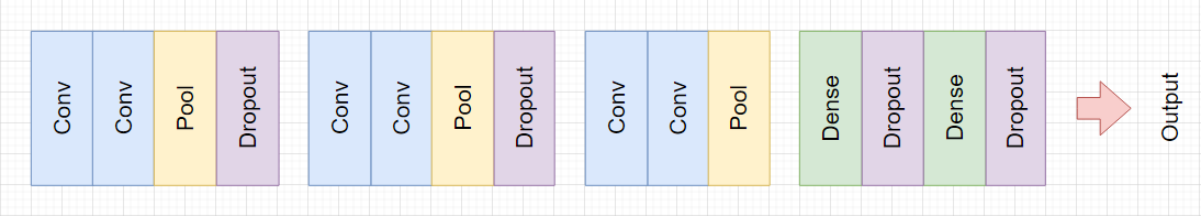


Figure 119 Second Model Architecture

```
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 120 – Second Model Training

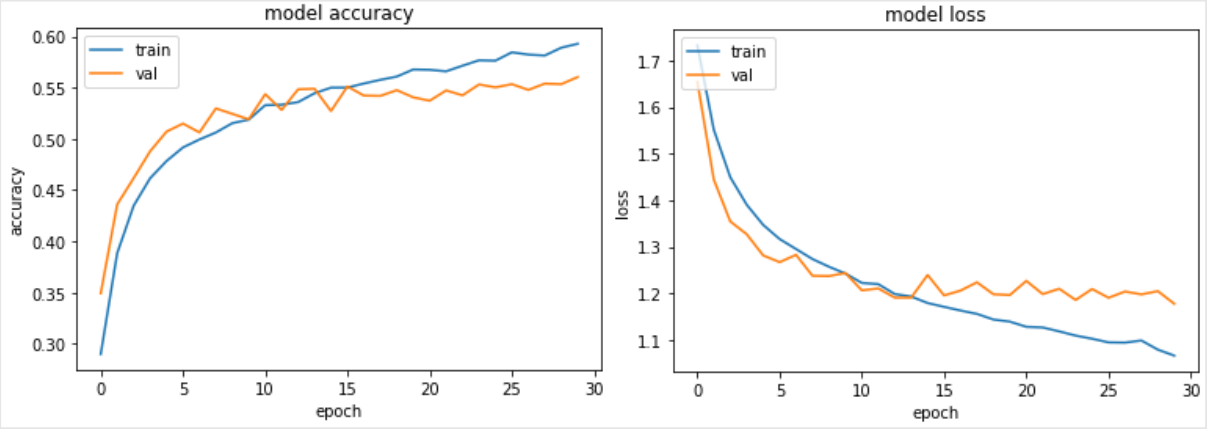


Figure 121 – Training Accuracy and Loss Charts for the Second Model

To explain the code, first the paths to the train, validation and testing data is initialized below in Figure 122. 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 122 – 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 123.

```
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_categorial_valid_y = valid_y.copy()
```

Figure 123 – Loading Dataset from File

An arbtrary number of features and batch size is chosen in Figure 124. The number of labels corresponds to the number of facial expressions and the number of epochs. It has been observed that after 25 epochs the accuracy on the validation dataset does not increase by much, if at all. The code in Figure 124 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 124 – Configurable Variables and Data Normalization

The code for the architecture illustrated in Figure 119 is displayed below in Figure 125. The code for the model has been taken directory from and 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 125 – Model 2 - 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 section mainly involves system 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 126.

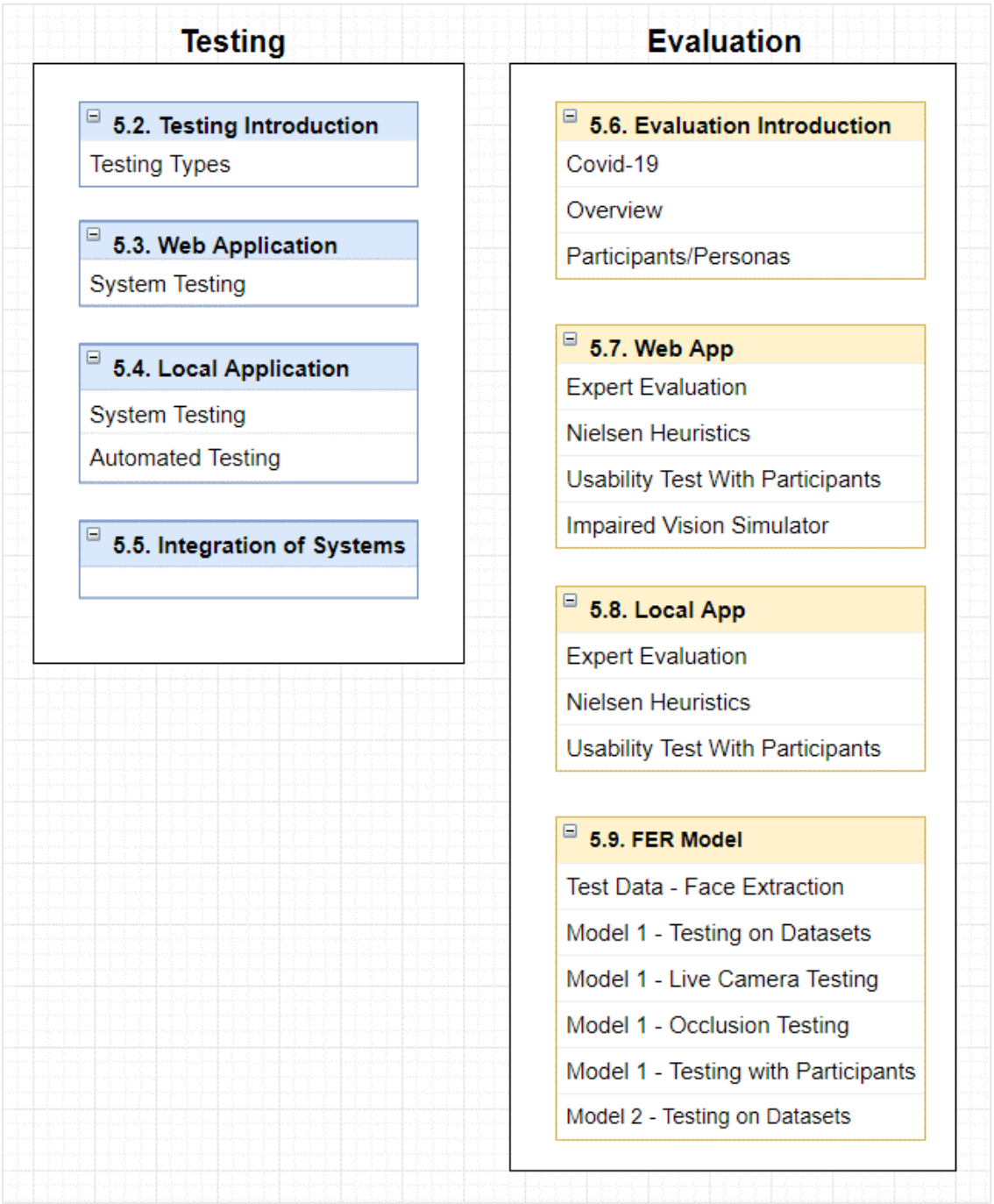


Figure 126 - Testing and Evaluation Overview

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 component 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. [90]. 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 and was automated with methods and classes that input 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 [91]. 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 [92].

The following testing sub-sections include system testing of the web application, local application, and integration of components.

5.3. Web Application Testing

5.3.1. System Testing

System testing was used to test the functionality of each component of the system and ensure that it has been correctly integrated with other components. System testing is black-box testing and involves the user interface. These will help find the bugs in the system and ensure that the functionality is working. Below in Table 3 are some of the system tests designed to test the components of the web application.

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. System Testing

A portion of the system tests used to test the components of the local application are displayed below in Table 4.

Table 4 - Local Application Unit Tests

Test Description	Expected Outcome	Pass/Fail
Reference number entered	The correct details and data of that usability study is received from the web server	PASS
Tutorial gifs display	The gif and information for a tutorial displays correctly	PASS
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
Recording slider minimised button pressed	The recording slider is minimised and the position is correct	PASS
Recording slider moved	The recording slider moves only along the x-axis	PASS
Task Window “Hide” button pressed	The task window is minimized	PASS
Task Window “Show” button pressed	The task window is displayed	PASS
User selects an option from multiple-choice list	Submit button appears	PASS
User clicks anywhere and drags on a question window	Window is moved	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 timestamps are consistent with emotions predicted	The emotion data from the participant is timestamped at the exact time of the prediction.	PASS
Video saved locally	The video will be saved to a local directory under the correct name	PASS
User clicks upload button	The data is uploaded to the backend web server and an instance reference number is returned. Video is uploaded to Vimeo	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

Automated tests have been implemented to test the local python application. The library used to create the UI is called PyQt5 and it includes a unit testing module [93] which allows for automated clicking on the GUI components such as buttons. The QTest clicking simulation code is shown in Figure 127. To run the tests the Python unittest library [94] was used and the file in which the tests reside in is named *ProgramTest.py*. The test in Figure 127 automatically enters a usability study's reference code, the data is pulled and processed and then verified.

```
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 127 - Local App Automated Unit Testing (Part 1)

The test in Figure 128 tests the multiple-choice question window. Sample data is passed, processed and an option is selected. The options, selected option and other text is tested for.

```
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 128 - Local App Automated Unit Testing (Part 2)

5.5. Integration of Systems

Integration testing involves connecting the units, components and sub-systems and testing them as a group. This type of testing is conducted to find flaws in the interaction between components. The integration of systems is displayed in Table 5.

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

5.6.1. 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 the test the FER model that number has been reducted 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 reducted to 2.

5.6.2. Evaluation With Participants

The evaluation procedure with participants is shown below in Figure 129. First the FER model was evaluated on its own. Next the local app which has the FER model integrated into it was evaluated by using it to test Blinkee.com, a poorly designed online shopping website. Finally, the web application was evaluated. With this procedure the participants were introducted to the various systems and functionalities of UsabCheck a step at time. When evaluating the local app they were already familiar with the machine learning aspect and when evaluating the web application they were already familiar with how a usability test is conducted as they have participated in one.

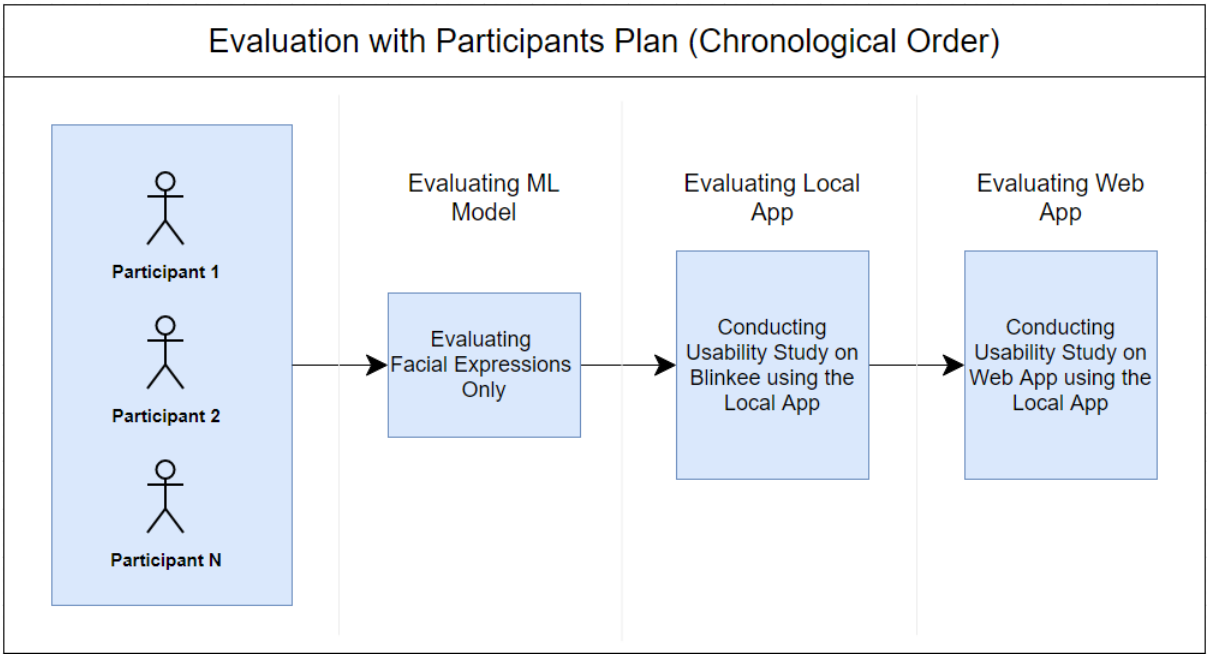


Figure 129 - Evaluation Procedure in Chronological Order

Two personas (excluding expert evaluator) were involved in testing and evaluating of the UsabCheck applications. Firstly, there are software developer type individuals who are tech savvy and understand software engineering and software development. This persona is shown in Figure 130. 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 they have close to no understanding of software development.

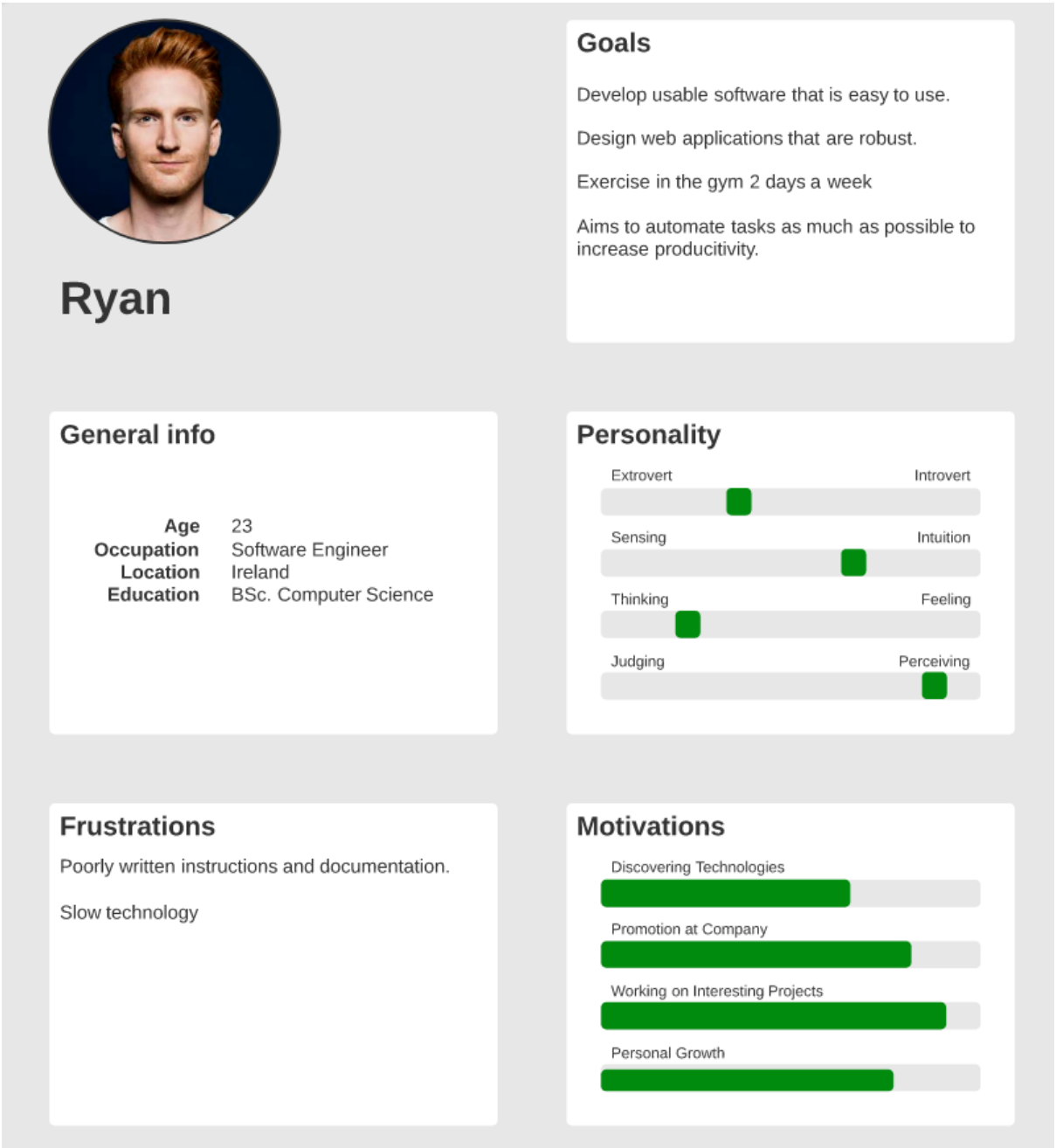


Figure 130 - Persona 1 (Developer)

5.6.3. Evaluation Overview

FER Model Evaluation

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 (Model 1) was additionally evaluated with 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, Blinkee.com was created and the participants will follow the instructions. The facial expression recognition data and other data will be obtained. The participants were asked questions about their emotions to determine if the data reflects how they felt. Other questions about the usability of the local application were asked. These questions will include questions based on Nielsen's 10 Heuristics [95].

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. This is because setting up a usability test and viewing the data is unlike other applications and websites such as shopping website which are designed for the general population.

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 "Create Usability Test" button after the test is created should no longer appear at the bottom.	The position of button was changed after the usability test is created. It now appears along with the rest of the buttons.
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 w3-lab [96]
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 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. Info buttons were added to give the user even more context.
There should be some intuitive 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

Nielsen Heurists [95] 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 the user performs an action such as a create, update, or delete a popup appears that notifies them whether or not their action was successful. Additionally, 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 is about to perform an irreversible action such as deleting a project or a test they go through a two-step process. Additionally, should they ever be stuck or wish to return to the main screen the "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 title is displayed above the results. This helps the users not having to remember what a task or question was about. A user can 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 minimalistic and visually appealing.
Help users recognize, diagnose, and recover from errors	When creating a usability test, if they 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 or can 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 rather than on paper. However, there could be more help and documentation to help users who have 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 to conduct a usability study for a website that you have built. You will go through the process of creating a usability study and viewing the result data of a usability study etc.
Pre-Test Question	Have you conducted a usabilty 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 and drag the slider to move 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 [97] 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 131.

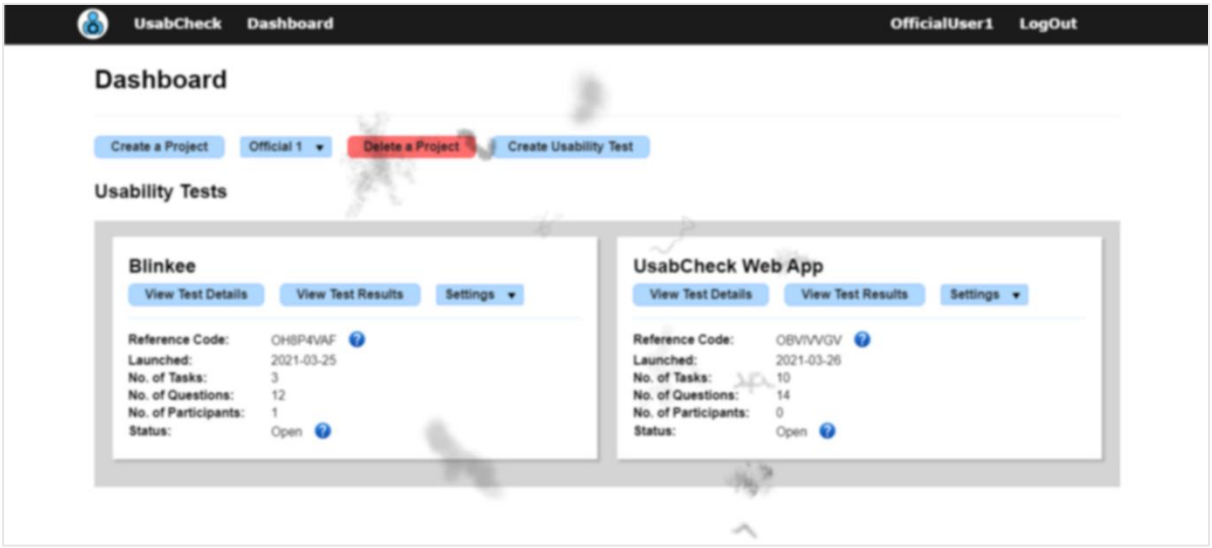


Figure 131 - Dashboard (Impaired Vision)

The video timelines/bars simulation of what would be seen by a visually impaired person is shown in Figure 132.

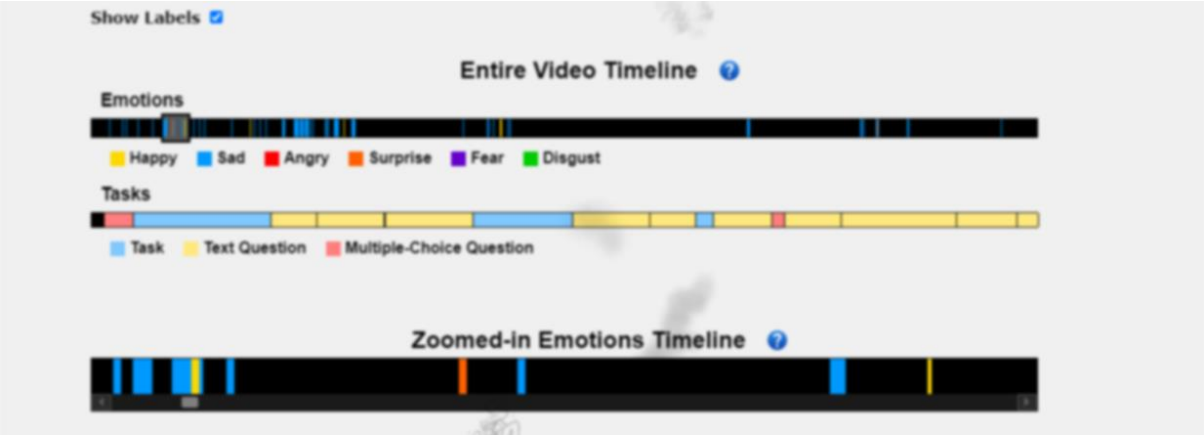


Figure 132 - Video Timelines (Impaired Vision)

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 displayed next to the task or question. 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 that the system is loading components. 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 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 move the task, question, and camera feed windows 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 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 and irreversible actions require two-steps.
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 colour for some of the buttons in other applications.
Consistency and standards	The question window always appears in the center. The task window reappears 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 [98]. It includes examples of the testing script and questions that could be asked. The blog also provided insight into how to ask questions that are helpful to the study and do result in biased responses.

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 [99]. The website was visually unappealing with an assortment of fonts and contained broken hyperlinks. The purpose of choosing a website with bad design is to test not only 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 use 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. 1. View the Patrick's day items 2. Sort by price high to low 3. Select option to view 100 products on the page 4. Find "Glow Necklace Red and Green Pack Of 25" 5. View the quantity discount of the item 6. Add item to cart 7. 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 1. Navigate to the page with Easter items 2. Find and add 1 mood changing egg to cart 3. 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 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. 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 responses of the participants to the questions were gathered and a summary of the most notable answers shown below 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 numered
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 not 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 more readable.
Moderate	The camera feed window covers some of the screen (also showing the camera feed made the user slightly self-conscious).
Low	Add a data upload progress bar
Low	For some windows it might make more sense to have buttons in the bottom right corner instead of bottom left
Low	Give a warning the application will open the camera or tell the participant that what stage the camera will be opened it
Low	Have a title in the first window. 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	The tutorial button text 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 usabilty study of 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.com shopping website. From the emotion timeline in Figure 133 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 facial expression detection sensitivity.

The participant expressed significant emotion when they clicked on a broken hyperlink 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 like parts the Blinkee.com website.

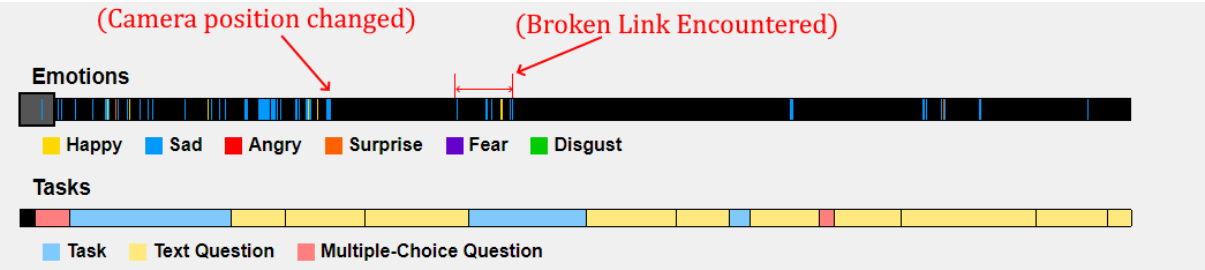


Figure 133 - Blinkee Emotions of Participant RB

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. Additionally, when the participant encountered the broken hyperlink 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 picked up by the algorithm and interfered with the usability study. As a result, the emotion data gathered from this participant could also not be used to make any concrete conclusions. However, when the participant encountered a broken link on the web page the algorithm correctly 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 results of the model 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 134) 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 135 and the images after extraction are shown in Figure 136. The code to the face extraction is in the "Model Testing.ipynb" file.

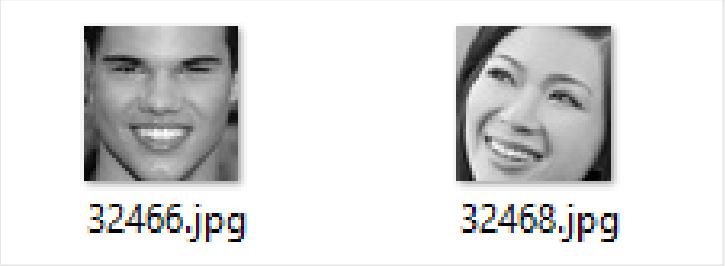


Figure 134 - FER2013 Dataset

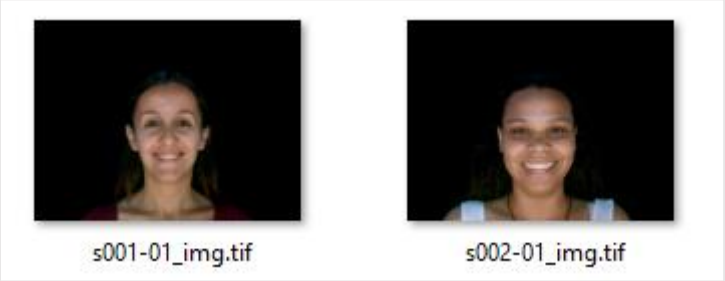


Figure 135 - FacesDB Original Images

Figure 136 - FacesDB Post Face Extraction

Testing Model 1 on Other Datasets

The purpose of this section is to test the model on 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 137. The model has been tested on the JAFFE dataset twice. Once on the original resized images and again with the images after the face extraction which was explained in the previous section.

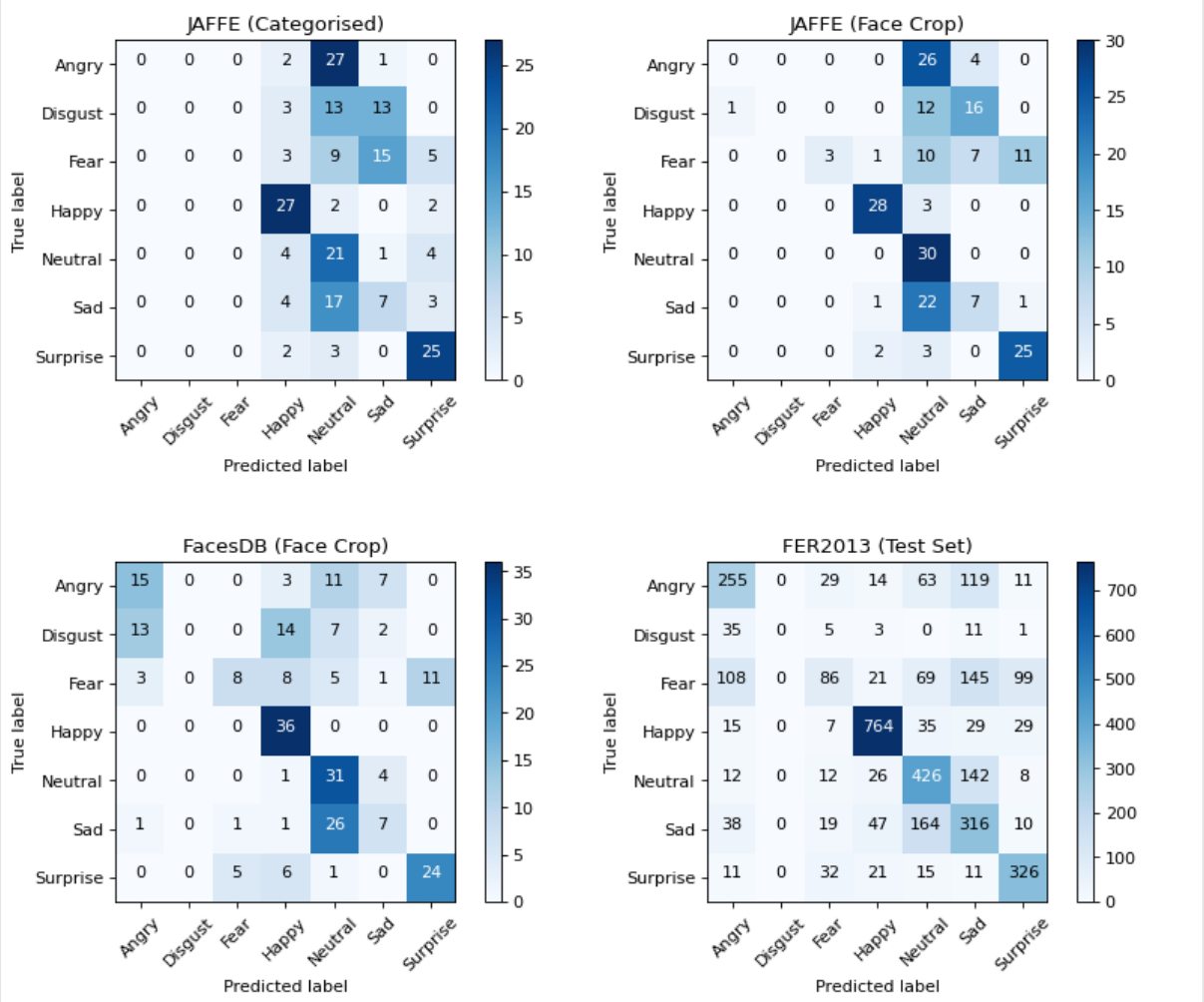


Figure 137 - Confusion Matrices (Model 1 train on FER2013)

The metrics such as the precision, accuracy, recall and F1 score are shown in Figure 137.

---------- JAFFE (Categorised) ----------

	precision	recall	f1-score	support
0	0.00	0.00	0.00	30
1	0.00	0.00	0.00	29
2	0.00	0.00	0.00	32
3	0.60	0.87	0.71	31
4	0.23	0.70	0.34	30
5	0.19	0.23	0.21	31
6	0.64	0.83	0.72	30
accuracy			0.38	213
macro avg	0.24	0.38	0.28	213
weighted avg	0.24	0.38	0.28	213

---------- JAFFE (Face Crop) ----------

	precision	recall	f1-score	support
0	0.00	0.00	0.00	30
1	0.00	0.00	0.00	29
2	1.00	0.09	0.17	32
3	0.88	0.90	0.89	31
4	0.28	1.00	0.44	30
5	0.21	0.23	0.22	31
6	0.68	0.83	0.75	30
accuracy			0.44	213
macro avg	0.43	0.44	0.35	213
weighted avg	0.44	0.44	0.35	213

---------- FacesDB (Face Crop) ----------

	precision	recall	f1-score	support
0	0.47	0.42	0.44	36
1	0.00	0.00	0.00	36
2	0.57	0.22	0.32	36
3	0.52	1.00	0.69	36
4	0.38	0.86	0.53	36
5	0.33	0.19	0.25	36
6	0.69	0.67	0.68	36
accuracy			0.48	252
macro avg	0.42	0.48	0.41	252
weighted avg	0.42	0.48	0.41	252

---------- FER2013 (Test Set) ----------

	precision	recall	f1-score	support
0	0.54	0.52	0.53	491
1	0.00	0.00	0.00	55
2	0.45	0.16	0.24	528
3	0.85	0.87	0.86	879
4	0.55	0.68	0.61	626
5	0.41	0.53	0.46	594
6	0.67	0.78	0.72	416
accuracy			0.61	3589
macro avg	0.50	0.51	0.49	3589
weighted avg	0.59	0.61	0.59	3589

Figure 138 – Classification Report (Model 1 train on FER2013)

The comparison of performance of the model on each dataset in terms of accuracy is displayed as a bar chart in Figure 139.

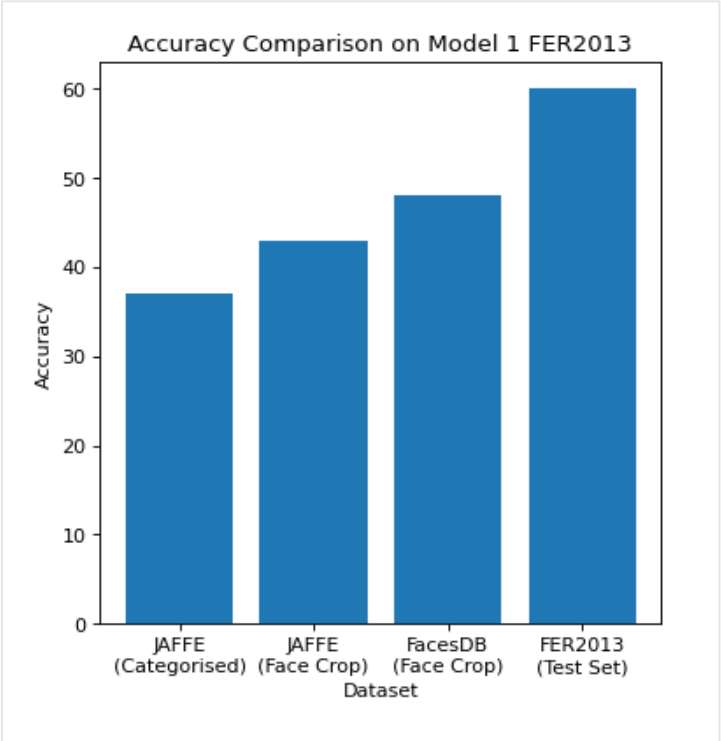


Figure 139 - Accuracy Comparisons (Model 1 train on FER2013)

Seeing the performance of the model on the JAFFEE 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 140, Figure 141, and Figure 142. The emotions in the image are at times subtle and could be mistaken for other emotions in some cases. The angry emotions may not involve much eyebrow changes and the fearful faces appear surprised in some cases.

Figure 140 - JAFFE (Angry)

Figure 141 - JAFFE (Disgusted)

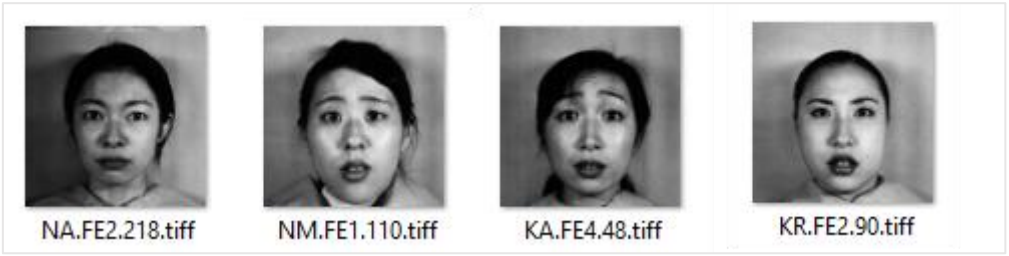


Figure 142 - JAFFE (Fearful)

Live Camera Testing (Initial Testing)

The model is finally put to practice in a real-world testing environment. This is illustrated in Figure 143. The emotions of fear and disgust were rather difficult to detect and would generally be classified under the "Sad" or "Angry" category. Generally, the model has performed very well in detecting the 5 basic emotions below in Figure 143. 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 resistant to noise 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.

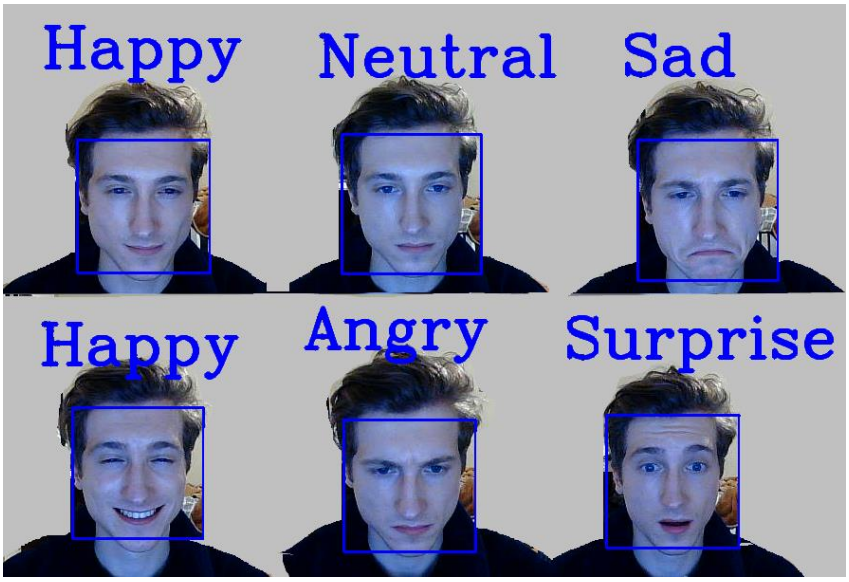


Figure 143 – Model 1 Camera Feed Predictions

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 criteria 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 expressions 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 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 wrongly predicted happy when the face was neutral and wrongly predicted neutral when the face was sad/angry. When the camera was on the bottom, below the screen and the participant wore glasses it correctly predicted neutral. This can be seen in Figure 144.

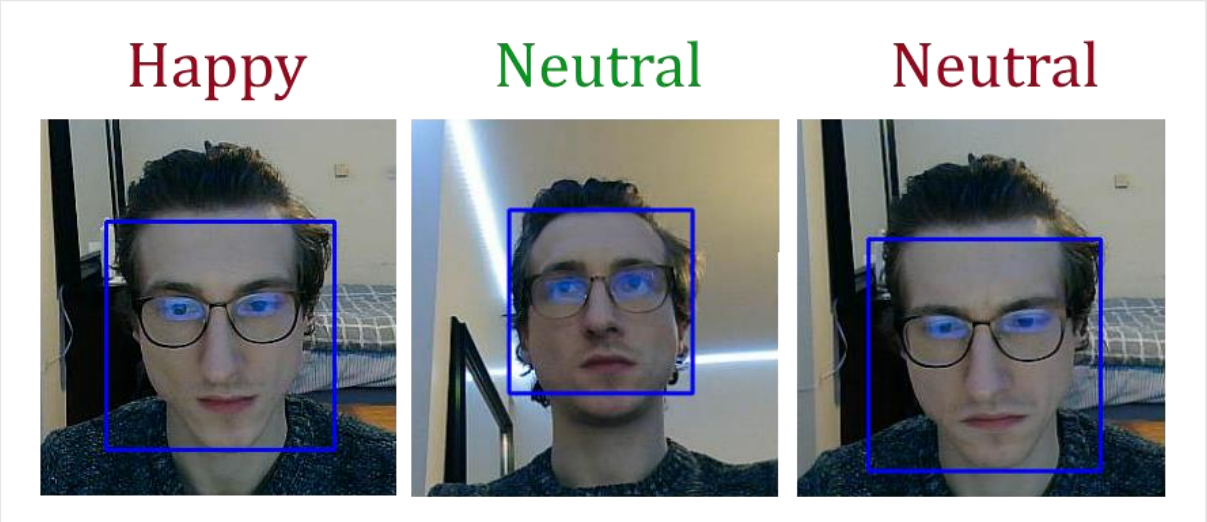


Figure 144 – Face Occlusion (Glasses)

Placing the hand over the mouth region has not posed too much of a problem for the algorithm when predicted emotions such as sad or neutral. It was still able to correctly 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 145.

Figure 145 – Face Occlusion (Hand)

Model 1 Evaluation with Participants

Individuals who have no prior knowledge of the system have tested the FER model as a singular component. The setup was simple and involved a window displaying the output from the camera and the prediction of the facial expression of the participant. The participant was asked to make various facial expressions and the accuracy of the model will be recorded. After the participant has expressed the facial expressions they were asked the below questions.

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 face changes a lot as the person is speaking.

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

Model 1 is a variation of the VGG-16 model trained on the FER2013 dataset and has performed very well on images from the camera. The model has performed the worst on the JAFFE dataset with a prediction accuracy of 44% (post face extraction) as the facial expressions in the dataset were more neutral. The model has performed the best on the FER2013 test set with the prediction accuracy of 61%.

It was discovered that 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 always detect that they were sad. In many cases 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. The prediction of the happy emotion depended mainly on the state of the mouth. Glasses had a negative effect on the prediction accuracy of the model.

4.4.2. Model 2

The second model, as mentioned previously has achieved an accuracy of 56%. Looking at the metrics in Figure 147, 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 146 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 as a result this model could 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.

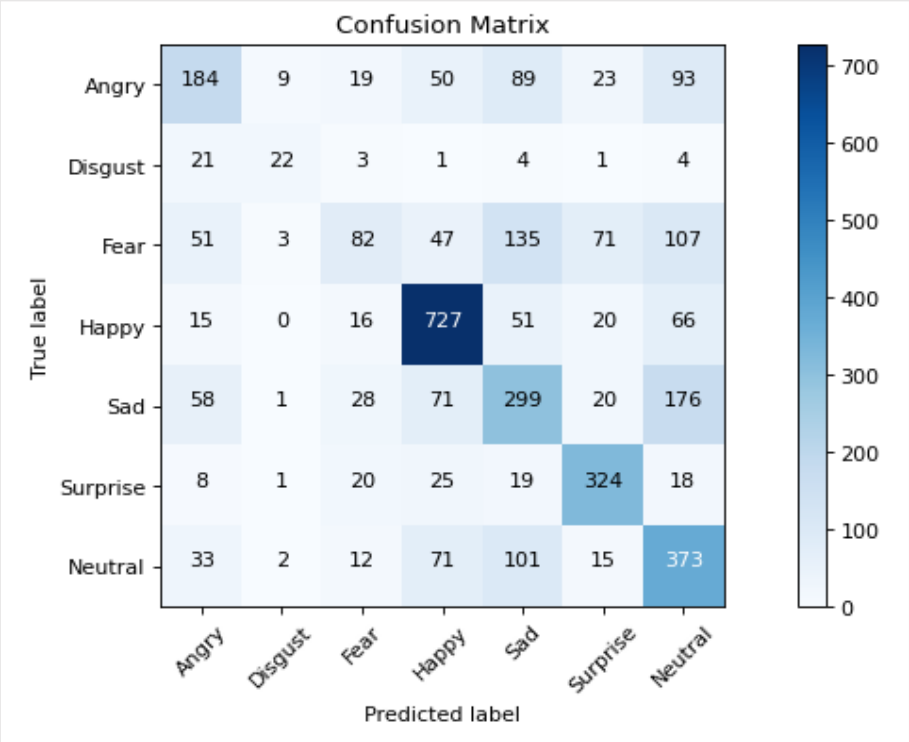


Figure 146 – Confusion Matrix of Second Model (Validation Set)

The metrics which includes the precision, recall and f1-score for Model 2 on the FER2013 validation set are shown in Figure 147.

		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 147 – Metrics for the Second Model (Validation Set)

The confusion matrices which show the prediction results of various datasets tried on Model 2 are displayed in Figure 148.

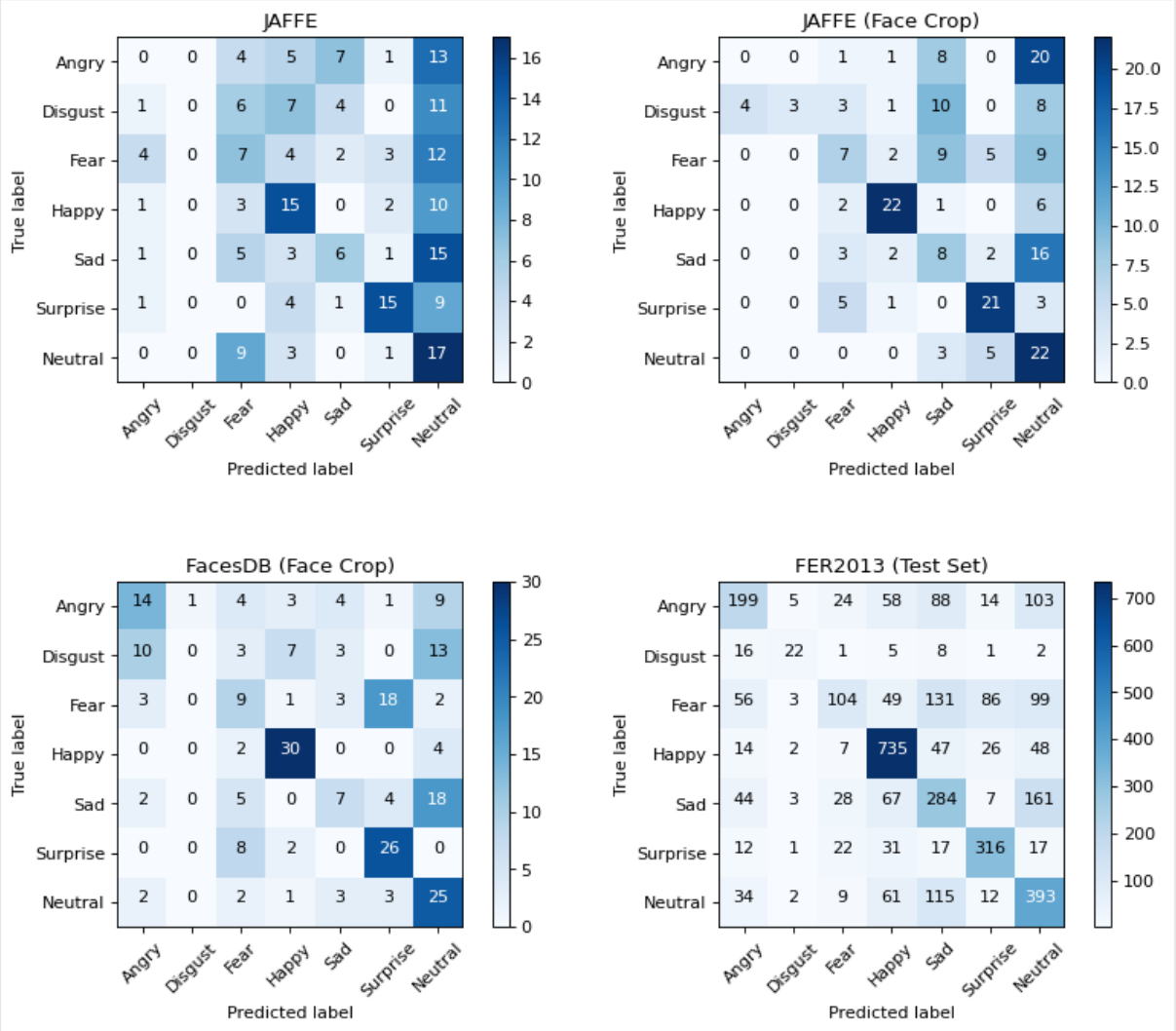


Figure 148 - Confusion Matrices (Model 2 train on FER2013)

The metrics for each dataset tested on Model 2 are displayed below in Figure 149.

---------- JAFFE ----------

	precision	recall	f1-score	support
0.0	0.00	0.00	0.00	30
1.0	0.00	0.00	0.00	29
2.0	0.21	0.22	0.21	32
3.0	0.37	0.48	0.42	31
4.0	0.30	0.19	0.24	31
5.0	0.65	0.50	0.57	30
6.0	0.20	0.57	0.29	30
accuracy			0.28	213
macro avg	0.25	0.28	0.25	213
weighted avg	0.25	0.28	0.25	213

---------- JAFFE (Face Crop) ----------

	precision	recall	f1-score	support
0.0	0.00	0.00	0.00	30
1.0	1.00	0.10	0.19	29
2.0	0.33	0.22	0.26	32
3.0	0.76	0.71	0.73	31
4.0	0.21	0.26	0.23	31
5.0	0.64	0.70	0.67	30
6.0	0.26	0.73	0.39	30
accuracy			0.39	213
macro avg	0.46	0.39	0.35	213
weighted avg	0.45	0.39	0.35	213

---------- FacesDB (Face Crop) ----------

	precision	recall	f1-score	support
0.0	0.45	0.39	0.42	36
1.0	0.00	0.00	0.00	36
2.0	0.27	0.25	0.26	36
3.0	0.68	0.83	0.75	36
4.0	0.35	0.19	0.25	36
5.0	0.50	0.72	0.59	36
6.0	0.35	0.69	0.47	36
accuracy			0.44	252
macro avg	0.37	0.44	0.39	252
weighted avg	0.37	0.44	0.39	252

---------- FER2013 (Test Set) ----------

	precision	recall	f1-score	support
0.0	0.53	0.41	0.46	491
1.0	0.58	0.40	0.47	55
2.0	0.53	0.20	0.29	528
3.0	0.73	0.84	0.78	879
4.0	0.41	0.48	0.44	594
5.0	0.68	0.76	0.72	416
6.0	0.48	0.63	0.54	626
accuracy			0.57	3589
macro avg	0.56	0.53	0.53	3589
weighted avg	0.57	0.57	0.55	3589

Figure 149 - Classification Report (Model 2 train on FER2013)

The accuracy prediction of the model on each dataset is compared in Figure 150.

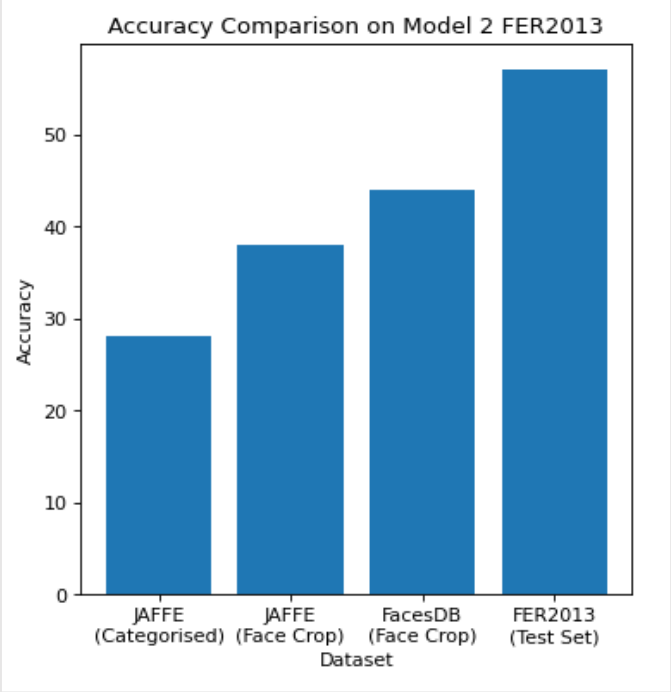


Figure 150 - Accuracy Comparisons (Model 2 train on FER2013)

6. Conclusions and Future Work

6.1. Introduction

This section will cover project conclusions, the challenges, risks, and improvements and future work. The project conclusions are divided in the sections of main conclusions, literature review, system design, system development, system testing, and evaluation.

6.2. Project Conclusions

6.2.1. Main Conclusion

The project aimed to provide a usability testing software suite and investigate the use and applicability of facial expression recognition in the area of usability testing. The project provides a web application service allowing researchers to create customized usability studies which can be conducted on the UsabCheck local application. The usability study data which includes the screen recordings, participant responses and emotions of the participant are presented to the researcher for analysis. 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. Additionally, as not all emotions experienced by the participants are expressed via their facial expressions and the FER alone cannot be used discover all usability flaws. Means of displaying emotions on a timeline has been developed and it allows the researcher to clearly view and analyse the participant's expressible emotions. The applicability of FER in usability testing is the improved navigation of the screen recordings of the usability study as the emotion timestamps provide the researcher with another layer by which they can navigate the video.

It should be also noted that the sample size of participants who participated in the usability study is only 3, all of whom where close friends and family of the author and took part in the usability study from the comfort of their own home. This sense of familiarity may have resulted in the participants expressing more visible emotion than they otherwise would in a professional usability testing setting. As a result of this small sample size and familiarity variable more research is needed.

6.2.2. Literature Review

Journey maps as means of displaying the emotions the participant has expressed during the usability study has been researched. However, a better solution involving interactive timelines for displaying emotions and tasks has been implemented. The advantage of the video timelines it provides an interactive interface for navigating the video. It can also be used to display any number of emotions instead of just positive and negative emotions. Research has shown that the six basic emotions can be combined to form more complex emotions such as boredom or frustration; however, this was not implemented or explored in-depth in this project.

Metrics such as accuracy, precision, recall and f-score which are used in evaluating machine learning models have been researched as well as the means of displaying these metrics which includes confusion matrices, These have proven very useful in the evaluation of the FER model.

Remote and onsite usability testing has been researched and the UsabCheck application can be used in both scenarios. All usability studies conducted with the UsabCheck applications were remote and synchronous, meaning that the participant's used their own machine at home and the researcher was present with them when they were participating in the usability study. As

expected, the varying standards of equipment and environment had a negative effect on the usability study.

The research of machine learning models includes Support Vector Machines (SVMs), Bag of Visual Words (BOVWs), and Convolutional Neural Networks (CNNs). In this project two CNNs were trained and evaluated. Several datasets which include JAFFE, FER2013, AffectNet, FacesDB and Emotic were researched and the dataset the models were trained on is the FER2013 dataset. Additionally, the models were tested with the JAFFE and FacesDB datasets.

Currently existing usability testing applications such as UserTesting, Loop11 and UserZoom Go were reviewed. These applications provided great insight into usability testing. The technologies researched and chosen in the researched chapter remained mostly the same. Python, Anaconda and Jupyter Notebook proved to be powerful tools in the areas of data science and machine learning with an array of libraries that provide functionality for training and evaluating machine learning models. Python and the PyQt5 library has been very useful in the development of the UsabCheck local application. JavaScript and React have allowed for flexible development of the web application's frontend. Spring-Boot and Java have been very suitable for the development of the web application's backend server. The video uploading service, Vimeo, has proven to be very useful with its services and API. The changes to the technologies that were made include deploying with Heroku instead of Tomcat and implementing a new widget to display the video timestamps instead of using the videojs-markers library.

6.2.3. System Design

The CRISP-DM methodology has been very helpful in managing the machine learning and data science side of the project. It specified the stages in its cycles which served as a guide to the design, development, evaluation, and deployment of the model. The Scrum Agile methodology was used in managing the project as a whole by dividing the work into sprints. The methodology was flexible and allowed for iterative design and development. The agile methodology kept the project on track and allowed for changes to be made throughout the development.

The high-level steps/stages for each sub-project (machine learning and software development) were outlined, a high-level view of the system, the system architecture, and high-level flow chart diagrams were designed. The requirements of the web application and local application were designed with use-case diagrams and the frontend prototypes were designed with JustinMind. The design diagrams had a very significant role in defining the requirements, structure, and end result of the project.

The machine learning side of the design involved "data understanding" which was conducted in the research chapter and "data preparation" in the design section. Data preparation involved planning the extraction of images from a CSV file and defining the directory structure of the data. The machine learning design involved planning the displaying of the data, evaluation of the model and deployment of the model.

6.2.4. System Develoment

The backend of the web application was developed in Java with Spring-Boot and included controller classes, entry points, DAO classes, entity classes, services classes, and authentication classes. The authentication method used was JWT (JSON Web Token) and JDBC was used to access the database.

The frontend was developed in JavaScript with React and the pages included the home page, login, register, dashboard, view test details, view test results and create usability test. The design of the "view test results" page has been redeveloped in the development section. A new widget to display the video timestamps of emotions and tasks has been designed and implemented.

The local application has been developed with the PyQt5 GUI library and the features include a screen recorder, displaying of tasks and questions, displaying camera feed, and predicting the facial expressions with the FER model. Once the usability study is finished the data is uploaded to the web application's backend and the video is uploaded to Vimeo.

Two machine learning models were trained on the FER2013 dataset and trained to classify the 7 basic emotions (including neutral). Model 1 was a variation of the VGG-16 model and both models were CNNs. The accuracy of the first model was on the validation set is 57.7% and the accuracy of the second model was on the validation set is 56%. The functionality that was implemented includes the CSV to image converter, dataset face extractor, and dataset loader.

6.2.5. System Testing & Evaluation

The FER model that has been most successful is Model 1. It is a variation of the VGG-16 model trained on the FER2013 dataset. This model has been used in the production version of the project and evaluated in many ways. Such ways include testing the model on two other datasets, collecting data on the relationship between facial features and specific emotions, occlusion testing, and live camera feed testing. It was discovered that 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 almost 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.

To frustrate the participant and obtain the emotion data a usability test was conducted on Blinkee.com, a website with poor design. As a result of the poor design, some participants have expressed amusement and laughter when testing the website. 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 visible 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.

All system 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, local application, and the FER model. Although there is room for improvement both the web application and local application can be used to conduct usability studies.

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 general 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 displayed on the matrix did not match the accuracy of the model. The issue was with the data loader of the images and to solve this problem a new function was implemented which loaded the images from a directory and classified an image at a time. Once the predictions were made with the new data loader 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 feed 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 libraries with pip only. Another issue encountered was with installing the TensorFlow and Keras libraries and required additional configuration in the code. 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 in the screen recorder was synchronising the time of the recorded video and 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 executed concurrently when necessary. The multi-threading aspect of the local application took a while to figure out and eventually it was discovered that the GUI library, PyQt5 had a threading module which helped solve the problem of the GUI freezing.

Before implementing the video emotion timestamps there were ideas of using a library. However, the libraries did not work at all and some libraries 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 short, same-origin policies prevented JavaScript code from accessing the HTML of an embedded video unless its origin is the same website. The solution was to use the Vimeo API.

Deploying the web application was a challenge that involved using dependencies and plugins to compile the project automatically. The setting up process of these dependencies and plugins was time consuming. 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 with React 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. The issue with the boot-up time and accessing assets was overcome by compiling the local app as a directory instead of a single EXE.

6.4. System Improvements & Future Work

6.4.1. Web Application

Improvements to the web app include more understandable error messages in the frontend when unexpected errors occur in the backend 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 of the page where the video is location when an emotion timestamp is clicked. Other improvements include 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 timeframe 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 a 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 same position it was last moved to as it is currently done with the task window, and allowing the user to change the size of 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 and ensuring everything displays appropriately.

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 moving. 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 automatically not detect emotions when audio is detected which is a better alternative.

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. Additionally, 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 used in an attempt 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 of 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 151. As the text is not very legible, the sprint plan figure has been split into two figures (Figure 152 and Figure 153).

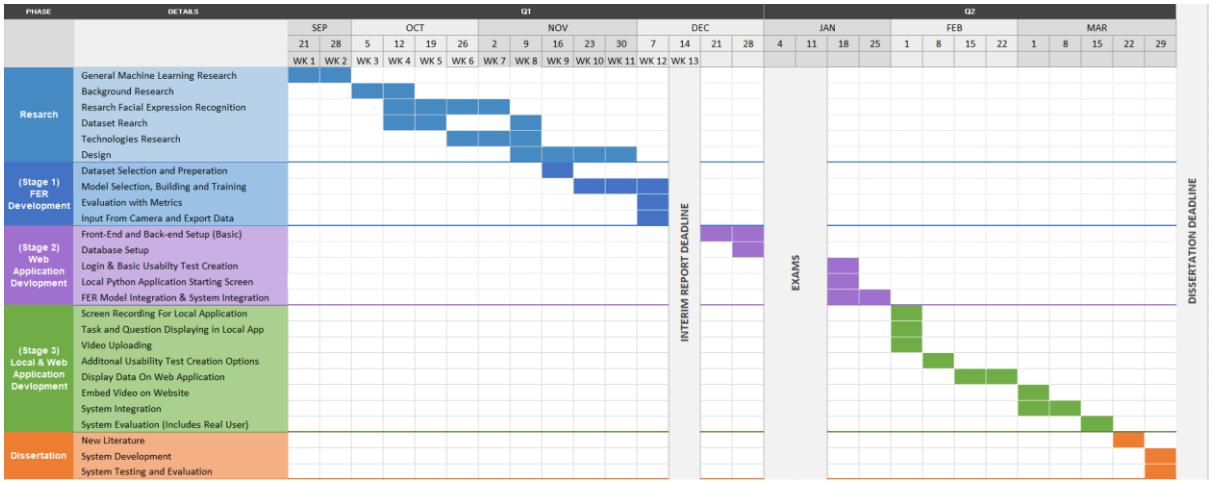


Figure 151 – Sprint Chart Overview (All Stages)

The first two stages are Research and FER Development and illustrated in Figure 152. This sprint chart is the representation of the research and FER development.

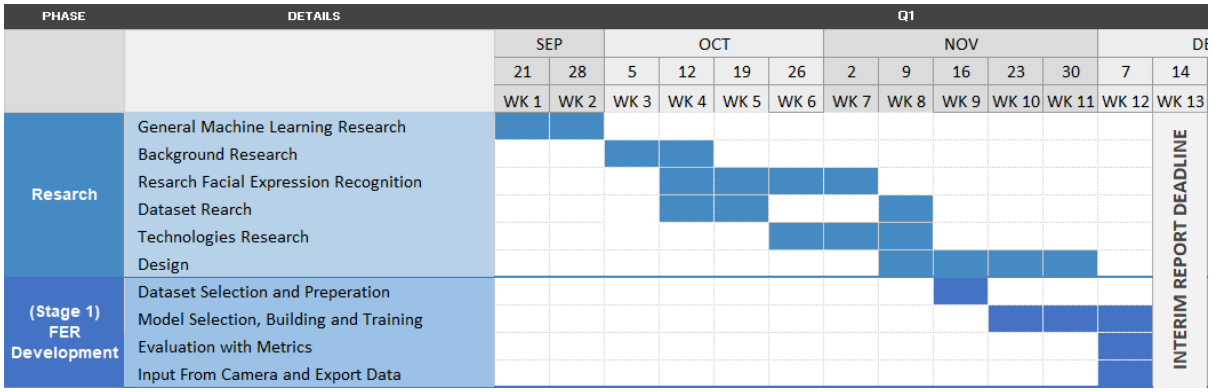


Figure 152 – Sprint Chart First Two Stages (Zoomed in for Readability)

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 153. This sprint plan was designed when the interim report was submitted.

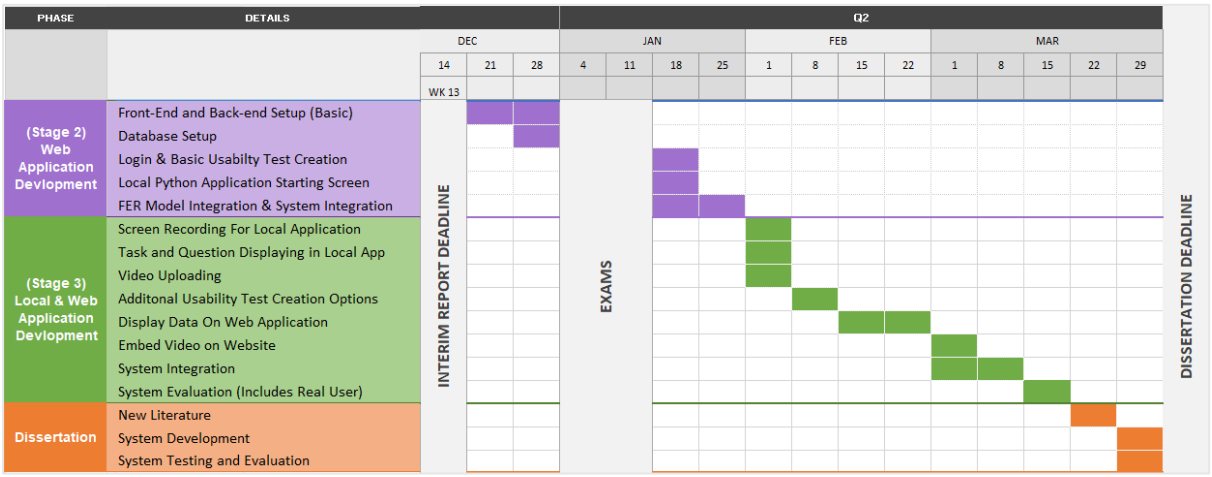


Figure 153 – Sprint Chart Last Three Stages (Zoomed in for Readability)

Plan Followed

The development that occurred from January until April is displayed in Figure 154.

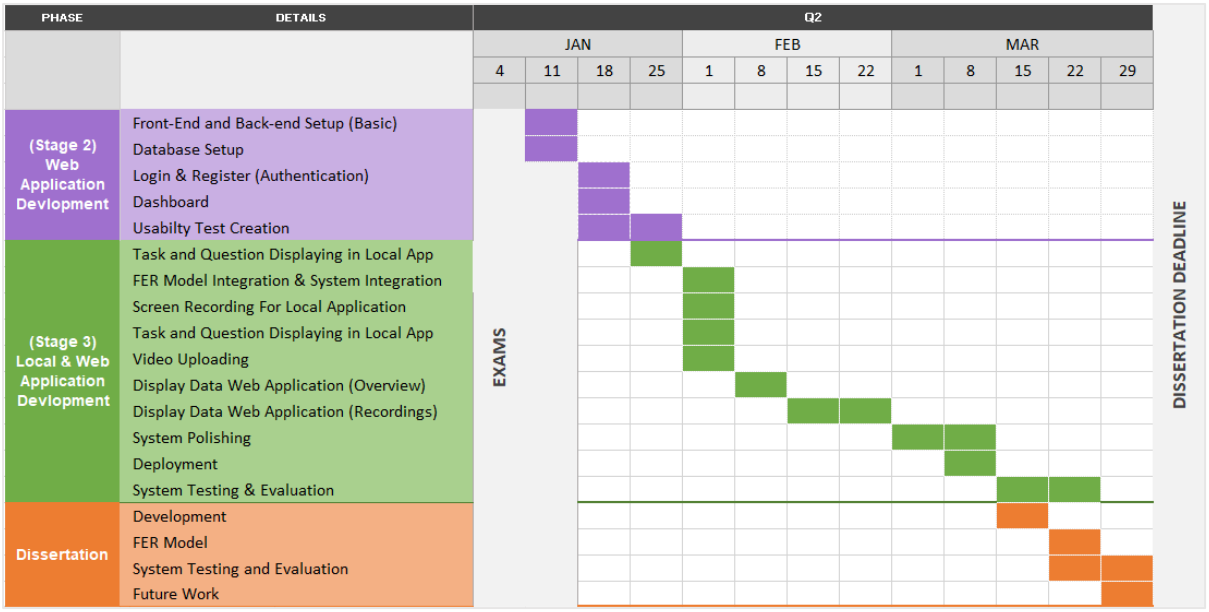


Figure 154 – Sprint Plan Followed

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 155 and Figure 156. 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: • 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 • 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 • 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 • 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 155 - Project Development Schedule (Part 1)

FEB & MAR	WK 5 WK 6	25 -> 12	System Polishing Web App: • 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 o Create project o Delete project o Create Test o Delete Test o Change Test Status • Make changes to emotions bars (add spacing) • Give context o What is a project o Pre-test, Test, Task o What is a status o What is a reference code o Text Question/Multiple-choice question o 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 156 - Project Development Schedule (Part 2)

The exact report development schedule that was followed is shown in Figure 157.

			Report
MAR	WK 8	15	• Draw updated DB schema • Draw Backend class diagram • Explain Back-end authentication • Explain Entry points and classes
MAR	WK 8	16	• Draw Front-end diagram • Explain how the pages and components work
MAR	WK 8	18 -> 19	• Get screenshots of all the web app and local app • Explain local app functionality • Explain recording functionality
MAR	WK 8	20 -> 22	• FER (Fix Metrics) • JAFFE & FacesDB Comparisons • Discuss the FER progress
MAR	WK 9	23	• Discuss Gestalt Laws (design/development/where relevant) • Discuss Colours and Emotions • Draw page relationship diagram
MAR	WK 9	25	• Test Local App (Participant 1)
MAR	WK 9	26	• 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	• Test web app with participants
MAR	WK 9	28	• Evaluate machine learning (occlusion etc.) • Local App Automated Testing
MAR	WK 10	29	• 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 157 - 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. Schmidt T, Schlindwein M, Lichtner K, Wolff C. Investigating the Relationship Between Emotion Recognition Software and Usability Metrics. i-com [Internet]. 2020 Aug 26 [cited 2021 Apr 4];19(2):139–51. Available from: https://www.degruyter.com/document/doi/10.1515/icom-2020-0009/html

9. 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

10. 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

11. 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

12. 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

13. 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

14. 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

15. 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

16. 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

17. DeepAI. ReLu [Internet]. DeepAI. DeepAI; 2019 [cited 2020 Dec 14]. Available from: https://deepai.org/machine-learning-glossary-and-terms/relu

18. 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

19. 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

20. VGG16 - Convolutional Network for Classification and Detection [Internet]. Neurohive.io. 2018 [cited 2020 Dec 14]. Available from: https://neurohive.io/en/popular-networks/vgg16/

21. ImageNet [Internet]. Image-net.org. 2017 [cited 2020 Dec 14]. Available from: http://image-net.org/index

22. 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-ilsvlc-2014-image-classification-d02355543a11

23. ImageNet [Internet]. Image-net.org. 2016 [cited 2020 Dec 14]. Available from: http://image-net.org/about-overview

24. 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

25. 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

26. 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

27. 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/

28. Aybüke Yalçıner. Bag of Visual Words(BoVW) - Aybüke Yalçıner - Medium [Internet]. Medium. Medium; 2019 [cited 2020 Dec 14]. Available from: https://medium.com/@aybukeyalcinerr/bag-of-visual-words-bovw-db9500331b2f

29. 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/

30. 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

31. 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

32. Rohit Verma. fer2013 [Internet]. Kaggle.com. 2013 [cited 2020 Dec 14]. Available from: https://www.kaggle.com/deadskull7/fer2013

33. AffectNet – Mohammad H. Mahoor, PhD [Internet]. Mohammadmahoor.com. 2017 [cited 2020 Dec 14]. Available from: http://mohammadmahoor.com/affectnet/

34. Home - FacesDB | VISGRAF [Internet]. Impa.br. 2012 [cited 2020 Dec 14]. Available from: http://app.visgraf.impa.br/database/faces

35. Google facial expression comparison dataset [Internet]. Google Research. 2018 [cited 2020 Dec 14]. Available from: https://research.google/tools/datasets/google-facial-expression/

36. leonshting. leonshting/fec-dataset-loader [Internet]. GitHub. 2020 [cited 2020 Dec 14]. Available from: https://github.com/leonshting/fec-dataset-loader

37. 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

38. Emotions in context [Internet]. Uoc.edu. 2019 [cited 2020 Dec 14]. Available from: http://sunai.uoc.edu/emotic/

39. 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/

40. 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/

41. Actionable UX insights for better digital experiences [Internet]. Userzoom.com. 2020 [cited 2020 Dec 14]. Available from: https://www.userzoom.com/

42. UserTesting: The Human Insight Platform [Internet]. UserTesting. 2020 [cited 2020 Dec 14]. Available from: https://www.usertesting.com/

43. Online User Testing Tool | Loop11 [Internet]. Loop11.com. 2020 [cited 2020 Dec 14]. Available from: https://www.loop11.com/

44. Welcome to Python.org [Internet]. Python.org. Python.org; 2020 [cited 2020 Dec 14]. Available from: https://www.python.org/

45. Python Features - javatpoint [Internet]. www.javatpoint.com. 2011 [cited 2020 Dec 14]. Available from: https://www.javatpoint.com/python-features

46. The Web framework for perfectionists with deadlines | Django [Internet]. Djangoproject.com. 2020 [cited 2020 Dec 14]. Available from: https://www.djangoproject.com/

47. Java.com. 2020 [cited 2020 Dec 14]. Available from: https://www.java.com/en/

48. 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/

49. 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

50. Node.js. Node.js [Internet]. Node.js. 2020 [cited 2020 Dec 14]. Available from: https://nodejs.org/en/

51. AngularJS — Superheroic JavaScript MVW Framework [Internet]. Angularjs.org. 2010 [cited 2020 Dec 14]. Available from: https://angularjs.org/

52. React – A JavaScript library for building user interfaces [Internet]. Reactjs.org. 2020 [cited 2020 Dec 14]. Available from: https://reactjs.org/

53. NumPy [Internet]. Numpy.org. 2019 [cited 2020 Dec 14]. Available from: https://numpy.org/

54. pandas - Python Data Analysis Library [Internet]. Pydata.org. 2020 [cited 2020 Dec 14]. Available from: https://pandas.pydata.org/

55. Matplotlib: Python plotting — Matplotlib 3.3.3 documentation [Internet]. Matplotlib.org. 2012 [cited 2020 Dec 14]. Available from: https://matplotlib.org/

56. 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/

57. Introduction to TensorFlow [Internet]. TensorFlow. 2020 [cited 2020 Dec 14]. Available from: https://www.tensorflow.org/learn

58. 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/

59. Keras Team. Keras: The Python deep learning API [Internet]. Keras.io. 2020 [cited 2020 Dec 14]. Available from: https://keras.io/

60. Welcome to fastai | fastai [Internet]. Fast.ai. 2020 [cited 2020 Dec 14]. Available from: https://docs.fast.ai/

61. 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

62. Home - OpenCV [Internet]. OpenCV. 2020 [cited 2020 Dec 14]. Available from: https://opencv.org/

63. 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/

64. StreamingVideoProvider [Internet]. StreamingVideoProvider. 2020 [cited 2020 Dec 14]. Available from: https://www.streamingvideoprovider.com/

65. Vimeo [Internet]. Vimeo.com. 2020 [cited 2020 Dec 14]. Available from: https://vimeo.com/

66. YouTube Data API | Google Developers [Internet]. Google Developers. 2020 [cited 2020 Dec 14]. Available from: https://developers.google.com/youtube/v3

67. Apache Tomcat Project. Apache Tomcat® - Welcome! [Internet]. Apache.org. 2020 [cited 2020 Dec 14]. Available from: http://tomcat.apache.org/

68. 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/

69. Firebase [Internet]. Firebase. 2020 [cited 2020 Dec 14]. Available from: https://firebase.google.com/

70. MySQL [Internet]. Mysql.com. 2020 [cited 2020 Dec 15]. Available from: https://www.mysql.com/

71. 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

72. Atlassian. What is Scrum? [Internet]. Atlassian. 2018 [cited 2020 Dec 14]. Available from: https://www.atlassian.com/agile/scrum

73. Scrum methodology [Internet]. Digital.ai. 2020 [cited 2020 Dec 14]. Available from: https://digital.ai/glossary/scrum-methodology

74. 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

75. 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/

76. 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/

77. Videojs-markers [Internet]. Sampingchuang.com. 2020 [cited 2020 Dec 14]. Available from: http://sampingchuang.com/videojs-markers

78. Karibasappa G C (KB. Disadvantages of Hibernate | Javainsimpleway [Internet]. Javainsimpleway.com. 2016 [cited 2021 Apr 4]. Available from: http://javainsimpleway.com/disadvantages-of-hibernate/

79. bezkoder. React JWT Authentication (without Redux) example - BezKoder [Internet]. BezKoder. 2019 [cited 2021 Apr 4]. Available from: https://bezkoder.com/react-jwt-auth/

80. Koniaris G. How to securely store JWT tokens. [Internet]. DEV Community. DEV Community; 2020 [cited 2021 Apr 4]. Available from: https://dev.to/gkoniaris/how-to-securely-store-jwt-tokens-51cf

81. Krissanawat Kaewsanmuang. Build a full-featured Modal dialog Form with React - Bits and Pieces [Internet]. Medium. Bits and Pieces; 2019 [cited 2021 Apr 4]. Available from: https://blog.bitsrc.io/build-a-full-featured-modal-dialog-form-with-react-651dcef6c571?gi=b0b3d1b40d67

82. Gestalt Laws of Perceptual Organization and Our Perception of the World [Internet]. Verywell Mind. 2021 [cited 2021 Apr 4]. Available from: https://www.verywellmind.com/gestalt-laws-of-perceptual-organization-2795835

83. Gohar Kadar, Navit. Diagnostic colours of emotions. Usydeduau [Internet]. 2008 [cited 2021 Apr 4]; Available from: https://ses.library.usyd.edu.au/handle/2123/2298

84. Maven Repository: com.github.eirslett» frontend-maven-plugin [Internet]. Mvnrepository.com. 2013 [cited 2021 Apr 4]. Available from: https://mvnrepository.com/artifact/com.github.eirslett/frontend-maven-plugin

85. Valencia A. Apache Maven AntRun Plugin – Introduction [Internet]. Apache.org. 2013 [cited 2021 Apr 4]. Available from: https://maven.apache.org/plugins/maven-antrun-plugin/

86. Alexey Timanovskiy. Heroku buildpack to support deployment from subdirectory [Internet]. Medium. Medium; 2017 [cited 2021 Apr 4]. Available from: https://medium.com/@timanovsky/heroku-buildpack-to-support-deployment-from-subdirectory-e743c2c838dd

87. pydeveloperashish. pydeveloperashish/Facial-Expressions-Recognition [Internet]. GitHub. 2020 [cited 2020 Dec 16]. Available from: https://github.com/pydeveloperashish/Facial-Expressions-Recognition

88. 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

89. neha01. neha01/Realtime-Emotion-Detection [Internet]. GitHub. 2020 [cited 2020 Dec 16]. Available from: https://github.com/neha01/Realtime-Emotion-Detection

90. 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/

91. 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/

92. 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

93. McGehee J. John McGehee Consulting [Internet]. John McGehee Consulting. 2011 [cited 2021 Apr 4]. Available from: http://johnnado.com/pyqt-qtest-example/

94. unittest — Unit testing framework — Python 3.9.3 documentation [Internet]. Python.org. 2021 [cited 2021 Apr 4]. Available from: https://docs.python.org/3/library/unittest.html

95. 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/

96. Marija Milosavljević. How Big Should a Font be on a Site? - Rules of... [Internet]. W3 Lab. W3 Lab; 2019 [cited 2021 Apr 4]. Available from: https://w3-lab.com/how-big-should-a-font-be-on-a-site-rules-of-typography/#:~:text=16px%20is%20the%20minimum%20when,frustrated%20and%20leave%20your%20website.

97. NoCoffee [Internet]. chrome.google.com. Available from: https://chrome.google.com/webstore/detail/nocoffee/jjeeggmbnhckmgdhmgdckeigabjfbddl

98. Chi C. The Beginner's Guide to Usability Testing [+ Sample Questions] [Internet]. Hubspot.com. 2019 [cited 2021 Apr 4]. Available from: https://blog.hubspot.com/marketing/usability-testing

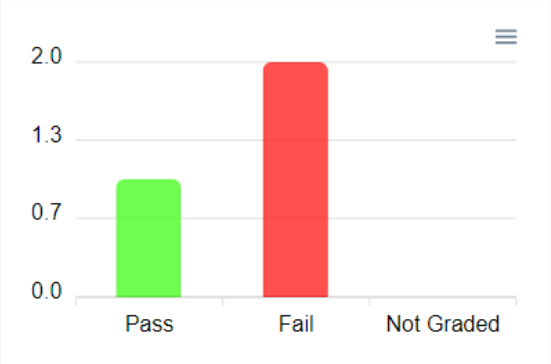
99. Breaking Bad: 21 Bad Website Examples (Upd: 2021) - Weblium Blog [Internet]. Weblium Blog. 2021 [cited 2021 Apr 4]. Available from: https://weblium.com/blog/21-bad-website-examples-of-2018/

Appendix

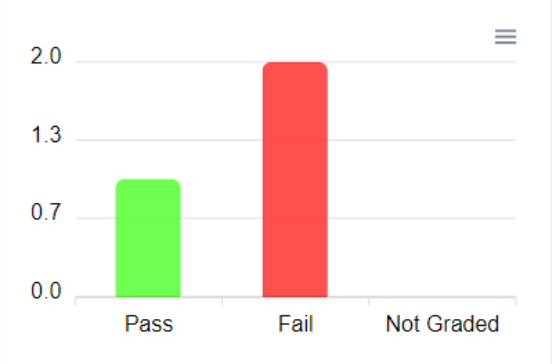
Appendix 1: Blinkee Usability Study Data

Individual Task Performance

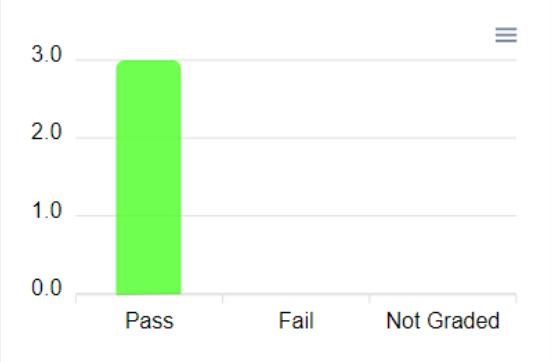
[2] Find necklace glowsticks for a Patrick's day party.



[6] Browse Easter items

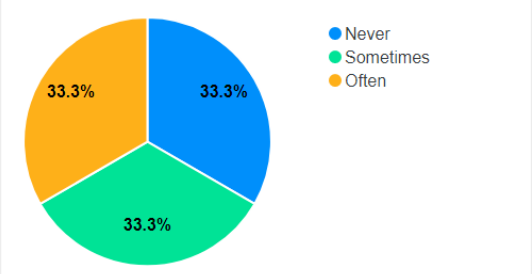


[9] View items in cart

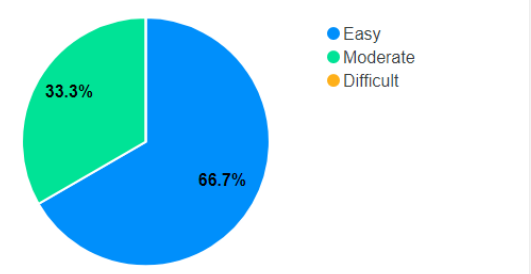


Multiple-Choice Question Answers

[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 wesite 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 sugested 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 wasnt in obvious places. I would be angry if i couldnt find the thing i wanted.
Curious of where the iteam was, concentrated
Happy when I couldnt 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 wasnt where i expected but it only took a second to figure out so it wasnt 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 looks bad.
Cleaner logos, they are very bright, in a way thats not bag but I prefer a cleaner esthetic. 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:

1. 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*

2. Which expression do you feel was best predicted?

Answer: *Happy was predicted best*

3. Which expression do you feel was worst predicted?

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