Project Name: Facial Recognition App

Student 1 Name: Andrew Nechifor

Student 1 ID: 16415432

Student 2 Name: Mahjabeen Soomro

Student 2 ID: 17362496

Project Supervisor: Suzanne Little

Table of Contents

1. Introduction (HEAVY WIP)

1.1 Overview

Provides a brief (half page) overview of the system / product that was developed.

The Facial Recognition App is an application that involves the user opening their camera and being assisted in the act of taking a picture or uploading a picture in order to have a facial recognition model applied to the image. Once the model has analyzed the picture, the results are returned and displayed to the user. The application will inform the user of whether or not they are wearing a mask and if they are wearing the mask correctly or incorrectly. The facial recognition system is developed, trained and tested using a large, public and publicly available online dataset of thousands of images that contains many images of faces with both masks on and masks off. The user will also have the ability to upload their own dataset and have it analysed as well.

1.2 Glossary

Define the **technical** terms used in this document. *Only include those with which the reader may not be familiar.*

User interface - The visual way in which a user and a machine system interact.
Machine Learning - A computer system that can learn and evolve through the use of algorithms and data models to perform tasks.
Facial Recognition - The ability of a computer system to recognize facial features in an image.
Application Program Interface - A set of functions and methods that allows the creation of programs or applications.
Frontend - The part of the app with which the user interacts directly.
Backend - The part of the app that is not directly accessed by the user. This will be responsible for receiving, storing and manipulating data. The results of which will be sent back to the frontend.
Integrated Development Environment - This is a system that provides numerous features to software developers for software development
GDPR - This is a set of laws that sets guidelines for the collection and processing of personal information from people who live within the European Union.

2. Research and Motivations

2.1 Research Performed

We began our research on the 5th of October, 2020. We were initially informed of CA Final Year Project ideas supplied by the CA staff by the module co-ordinator. We browsed many of the project ideas in order to understand what the scope of a project could entail and what different areas could be approached by our project. Many different areas of software engineering were covered by all of the project ideas but we were the most captivated by the ideas that utilized large databases of media like photos and videos.

We began discussing what areas we wanted this project to cover and we came to the agreement that facial recognition was the most fascinating subject for both of us. We knew that facial recognition was a vast and evolving area within the field of software engineering. Facial recognition is regularly used in everyday life in the form of the lock screen on people's phones for example. With its constant presence and continuous developments, the research performed needed to be accurate and straight to the point. The global pandemic caused by the novel coronavirus was constantly relevant in our lives and as such, it was taken into account during the researching process. Many articles, chat forums and social media pages were engaged in the conversation of mask usage and facial recognition systems during the research phase of our project. At the time, we googled "facial recognition" and "facial recognition mask" to see many of these articles and discussions.

The results of our initial research gave us vital information in the form of what facial recognition is defined as and what it is not. As well as that, we learned about the sentiments and ethical concerns that some people shared about facial recognition as a system. We noted these findings as we thought they would be useful for us in the future. We decided to further refine our research strings by googling "facial recognition technology" and "facial recognition works" in order to view examples of facial recognition systems and to inspect how these systems operate in detail. This gave us insights into the widespread use of facial recognition, the applications of this technology and the techniques involved like face analysis and machine learning.

After gathering and collecting the results of our research, we firmly chose to work on a project that was involved in the area of facial recognition and to relate the project to the current global state of affairs. Finally, after a couple iterations we conceptualized the idea of using an app to perform facial recognition on a person to detect whether a person is wearing a mask or not and whether they are wearing correctly or not.

2.2 Motivations

Many factors came into play when deciding what our project was going to be and why we chose this particular idea. Conceptualizing an idea and following through with that idea can be difficult tasks to perform without properly justified motivations. First and foremost, we decided to undertake a facial recognition app as our project because the idea felt very relevant and topical for the current situation taking place around the world. Wearing masks in public is now a global and daily part of most people's routine around the world. Previously, countries in Asia regularly wore masks in public but due to the novel coronavirus, that practice has become global. Being able to recognize if someone is wearing a mask and whether that mask is being worn correctly or not is a task humans perform all the time, but transferring this task to an automatic, autonomous system has other benefits that we were motivated by.

Our app has many potential applications in a global business setting. This was another motivation behind our research and project. This facial recognition system can help keep businesses safer and cleaner by analyzing the faces of the customers that enter an establishment to see if they are wearing a mask properly. As well as that, employees can also use this system to ensure that they are correctly wearing a mask. Ensuring every customer and every employee on the premises has an appropriate mask worn can help guarantee a clean and safe environment for the business. This improves the retail experience for both customers and employees. This is not only limited to retail shops, as this can be used in banks, colleges and libraries to ensure the safety of everyone on the premises.

Lastly, we were motivated by our keen interest in the field of machine learning. We were interested in facing the unique challenges and making use of the opportunities that this area gives to developers. Researching the difficulties in developing a facial recognition model motivated us to work in this area. Performing a balancing act of knowing how much to train a model, how much to test a model, how to verify its accuracy and how to integrate the model into a mobile application seemed appealing to both of us.

3. System Architecture (basically the what of the arch.)

This section describes the high-level overview of the system architecture showing the distribution functions across (potential) system modules. Architectural components that are reused or 3rd party should be highlighted. Unlike the architecture in the Functional Specification - this description must reflect the design components of the system as it is demonstrated.

3.1 Overall System Architecture

The application and complete frontend architecture of the project was completely developed using Android Studio. The user interface and functionality were all fully handled using this IDE. Android Studio utilizes the Java object-oriented programming language which helps with the structure and design of the application and the user interface. Within Android Studio are virtual machines and shortcuts for error handling which makes the creation, modification and evaluation of each element of the architecture and user interaction easier to perform.

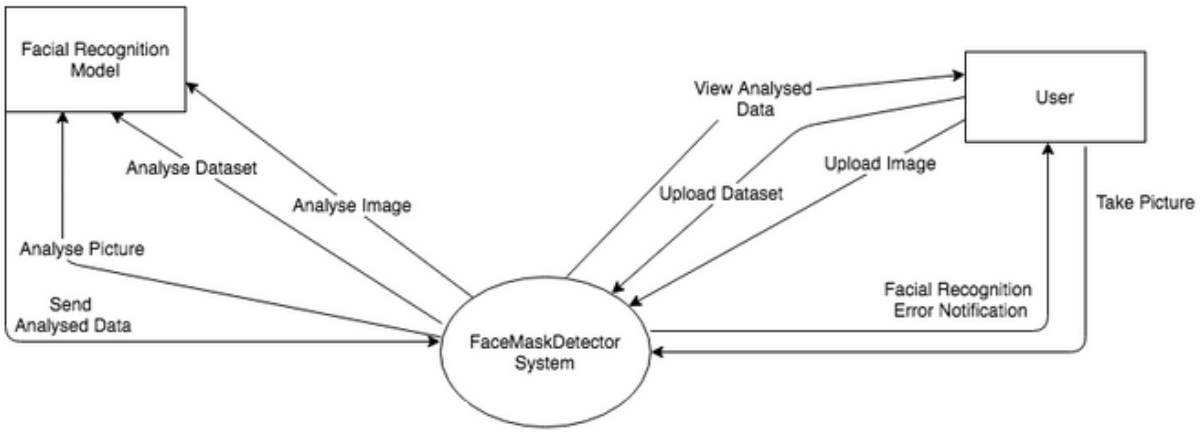
The structure of the user interface was designed to be as simple and understandable as feasibly possible. When the user enters the application for the first time, they enter the home page of the app. They will see **(description of the user interface interactions)**

The design of the backend involves many functions and modules interacting with each other. When the user inputs an image into the facial recognition system, the app has a trained model ready in order to perform the analysis on the picture. When the app performs this analysis, the results are sent to the frontend in order to display the data and any temporary image or data that the backend has processed is automatically deleted.

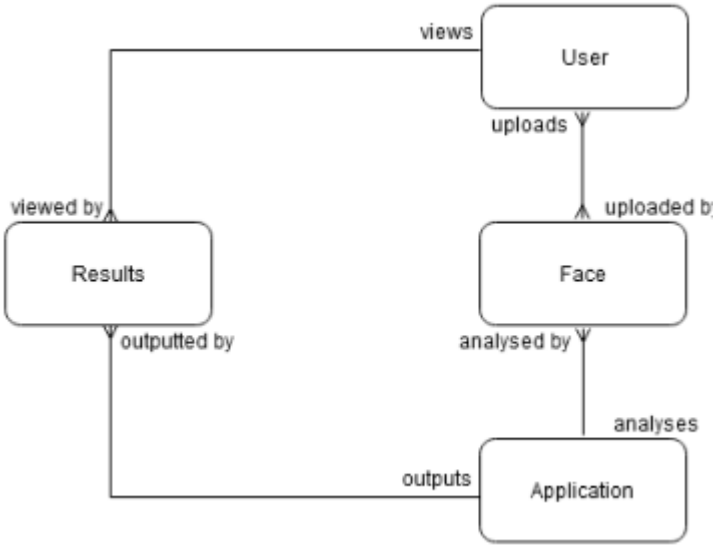
The facial recognition model is trained using an online dataset with thousands of publicly available faces of people with masks both on and off. Bigger sample sizes provide more accurate mean values, provide smaller margins of error and help find outliers that could skew the data that would be caused by a smaller sample size. The model is trained and tested twice with this dataset in order to verify the accuracy of the facial recognition system. The training and testing data ratio is a split between 85 / 15, we felt that this was a fair split considering the large dataset size we have. This was to avoid the issues of overfitting where systems have good performance on the

training data but poor performances on other data. This was also done to avoid underfitting which involves the system having a poor performance on the training data and a poor performance on other data as well.

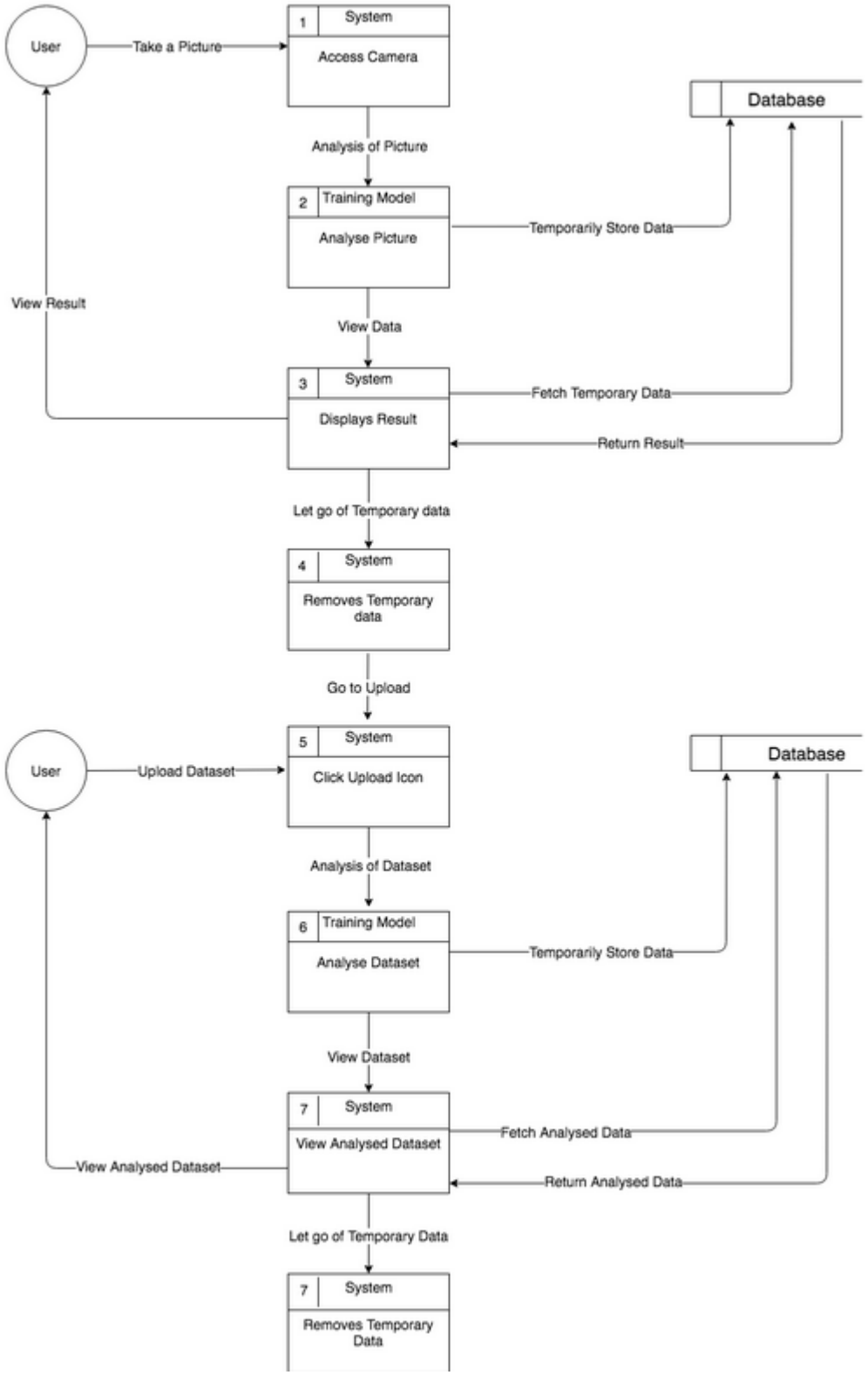
Context Diagram



Logical Data Structure



Data Flow Diagram



3.2 3rd Party / Reused Architecture

4. Implementation (Basically the why of the architecture)

This section should set out the high-level design of the system. It should include code from the system, explanations, justifications and clear reasoning for how and why your system has been designed as such. Unlike the design in the Functional Specification - this description must reflect the design of the system as it is demonstrated.

4.1 Implementation Choices

4.2 Sample Code

5. Problems and Resolution

This section should include a description of any major problems encountered during the design and implementation of the system and the actions that were taken to resolve them.

5.1 Facial Analysis Issues and Solutions

The largest issues we faced during the implementation of the facial recognition model was dealing with images that had faces in poor lighting or faces that were not taking directly straight at the camera but from other angles instead. The changing of the angle at which you view a face moves the points that the model uses in order to recognize facial features within the image and as such, can make it harder to detect if there is a face or if there is a mask being worn on a face. As well as that, images that contained poor lighting often obscured facial features that made facial detection difficult as a result. This problem is compounded by how the facial recognition system detects whether a face is wearing a mask or not. If there are certain features missing within the image, but the overall shape of the face can be recognized, the system classifies the face within the image to be wearing a mask. So this causes some images where the face is in poor lighting or if the image is taken at a different angle to be classified as a mask wearing image. This is not easily solvable as we found in our research. The best solution we found was to increase the amount of training performed and to increase the size of the training data. While our accuracy scores did improve, this may be a discrete occurrence of slight overfitting as well so this must be taken into account.

5.2 Application Issues and Solutions

The application was created, modified and evaluated using Android Studio as the IDE. We faced many bugs and issues during the development of the app. When attempting to set the name of the application in the virtual machine, the renaming would not occur. We googled this and this is a well documented issue in Android Studio and the fix provided to work around this issue involved clicking the Run button again to re-deploy the application.

5.3 Backend Issues and Solutions

The largest issues faced by the backend of the system was multi-faceted and must be explained with great detail. Initially, the frontend was intended to operate through constant communication with a cloud computing service that contained all of the backend algorithms and processes necessary to perform facial recognition. The cloud computing service would conduct all of the operations necessary with its available resources and return the results back to the app. As we found out during the project, free and online cloud computing services do not provide its users with the amount of

resources necessary to perform intensive machine learning algorithms and facial recognition on images. These machine learning algorithms are very computationally expensive and require a lot of resources that the free tiers of every single online cloud computing service provider do not offer. Each and every online cloud computing service offers different prices and resource limits depending on what payment plan is chosen. After going through each possible service provider and not finding any that were suitable for this task at a reasonable cost, we decided to perform these backend services locally through systems like Flask and Django. However, these web frameworks continuously proved to be incompatible with Android Studio's virtual machines and emulators, with well documented and numerous issues appearing online as well. We found a solution to all of these issues, in order to perform our facial recognition we needed to have the facial recognition system on the application itself. While we are aware that this would slow down our application, we felt this was the simplest and most appropriate option given the position we were in.

6. Testing

Testing documentation, incl results

Every project is different and therefore the testing strategy for projects should involve as a primary objective the confirmation that the quality of the software is sufficient. Some further info in relation to this is provided below.

Testing Type. The type of testing that is desirable may depend very much on the application / project. Some heavy server-side projects may require a greater focus on unit and integration

tests with perhaps less focus on system level tests (though system tests would still be required to evaluate that the system operates as intended). In contrast, projects with relatively high levels of user interaction via a GUI may require higher levels of GUI testing, perhaps through the use of system tests (though unit and integration testing may also be required in order to evaluate that the individual units and components operate as intended). In general, it is beneficial to demonstrate a knowledge of and ability to utilise testing frameworks, e.g. unit testing using pyunit, for the purpose of unit and integration testing.

Ad Hoc testing. It is normal for developers to conduct ad hoc testing as they build software however, more formal testing should also be conducted, which includes specific test cases which have corresponding expected test results that are examined by the execution of the test cases and comparison of expected and actual results.

Where to document system testing (incl test cases and execution results)? These can be located in the Technical Guide (under the Results) or if preferred by individual students, this can also be achieved by having separate testing documentation (i.e. a System Test document). Having a separate system test document may be more appropriate in cases where there is a heavy dependency on system testing to establish quality and therefore, large numbers of system tests have been established and executed.

Integration Testing. Integration tests may be implemented within the code, but they may also be implemented separately using drivers, stubs and component-to-component testing. Where implemented in code, the tests will have expected results and if they do not pass, the error will be flagged. Where implemented in other ways, integration test details may be expected, i.e. the amount of integration test documentation very much depends on the type of integration testing that has been performed. In some cases, it may be appropriate to bundle integration tests with system tests.

Testing strategy. The testing strategy should be highlighted to examiners as part of your demo: i.e. I used unit testing in these parts and for the following reasons; I used formal system testing as well, with a focus on the following aspects of the project... etc. Examiners will seek to establish that testing has been conducted in a robust fashion that produces good quality software (and inevitably evidence that this has been conducted is required).

7. Future Work