CA400 Functional Specification

Project Title: “Face Mask Detector”

Student names: Mahjabeen Soomro, Andrew Nechifor

Respective student numbers: 17362496, 16415432

Date finished working on the document: 04/12/2020

0. Table of contents

1. Introduction

1.1 Overview

The application to be developed by the team will be named the “Face Mask Detector” app. At the moment, there is no dedicated online service that allows users to verify if they are wearing their mask correctly or not. There are models that can detect if you are wearing a mask, that can detect and recognize your facial features and more but none of them are capable of performing all these functions at once.

Our application aims to give users the ability to create and build a wide array of databases for faces. With the help of our app they can take photos with their phone or they can be uploaded to the application. Our app will be able to use its facial recognition model, trained and tested using a neural network, to identify each unique face, have the capability of detecting whether a face mask is being worn or not and will be able to detect how a face mask is being worn incorrectly, which would be followed by the app notifying the user of how it is being worn incorrectly.

The app will exclusively operate on mobile phones, solely on Android devices. We will be using a large dataset of thousands of images that contain pictures of thousands of faces with both masks worn and masks off so that our app is as accurate as possible. The application will conduct an analysis on this dataset and output the results and statistics from the analysis. An easy-to-use, clear interface that any user can easily navigate will be necessary in order to achieve the main aims of the application.

1.2 Business Context

Within any business setting, the application can function as a system that allows businesses to verify if employees or customers are wearing their mask appropriately. The app can give instructions to the user if they are wearing their mask incorrectly. The app will also be able to be used as a biometric facial scanner for employees, verifying their identity to ensure that they have signed into their workplace.

As well as that, the app can perform multiple analyses on datasets that contain pictures of faces. Being able to output statistics from the dataset like how many images had masks on or off, which had masks worn correctly or incorrectly and the occurrences of each individual mistake when a mask was worn in correctly would be invaluable to a business. They can build up a database and dataset of these images, acquire the statistics from the app’s analysis and make decisions for the health and safety of employees and customers based on this information, given the current global situation.

1.3 Glossary

- API - Application Programming Interface. It allows for data access to a desired system.
- UI - User Interface. This is the means by which the user and the app interact. It allows the user to input the brand and receive / view the data about a brand.
- 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.
- 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.
- IDE - Integrated Development Environment. A software application that provides comprehensive facilities to computer programmers for software development.
- Android Studio - The official IDE for developing applications exclusively for an Android platform.
- XML - Extensible Markup Language.
- Neural Network - A series of algorithms that improves itself over a series of iterations, choosing better input and output data to send with each generation.
- SDK - Software development kit.
- Cloud computing services - Computer system resources like computing power and computer storage that do not involve management by the user of it.
- SQLite - An embedded relational database management system.
- BLOB - Binary large object.

2. General Description

2.1 Product / System Functions

- Users can take a picture of their face with assistance from the app.
- As well as that, they can upload images of a face to the app.
- The facial recognition model will be applied to the face and the data will be categorised and visualised for the user.
- If the app cannot recognise a face (due to conditions like poor lighting, inadequate angles and uncentered positioning), the user will be informed.
- The application will inform you if you are wearing a mask.
- The user will be notified how they are wearing a mask incorrectly if doing so.
- The user can view statistics from datasets on the app.
- The user can upload their datasets to the app.
- All navigability functions of the application should be quick and fluid.

2.2 User Characteristics and Objectives

Describes the features of the user community, including their expected expertise with software systems and the application domain. Explain the objectives and requirements for the system from the user's perspective. It may include a "wish list" of desirable characteristics, along with more feasible solutions that are in line with the business objectives.

Generally, anyone with access to an Android mobile device should be able to intuitively utilize and access the main features of the application. The opening screen should not intimidate users. Keeping the interface clear and easy to understand is the main goal here. The navigation bar that contains the different functions of the app should be straightforward, with sections ranked in order of functional importance. This corresponds to users who just initially want to utilize and see the facial recognition data first and have easy access to it without any extra effort, while on the other hand those who wish to analyse whole datasets and perform more elaborate functions can simply go further through the navigation.

The common user may have accessibility issues like poor eyesight or color blindness. It is vital that the design of the UI satisfies as many users as possible of the application. A screen-reader and screenwriter should be compatible with the app, and the app should have an optimal colour palette and colour contrasts.

2.3 Operational Scenarios

Scenario #1 | User Objective: Take a photo of a face and see the results

Source	**Step**	**Action**
User	1	Open app
Program	2	Display start screen
User	3	Tap “Camera” button
Program	4	Open phone camera with features
User	5	Take app-assisted photo
Program	6	Display loading icon
	7	Display results when finished
User	8	View results

Scenario #2 | User Objective: View training data for facial recognition model.

Source	**Step**	**Action**
User	1	Tap “Database” icon
Program	2	Display categories of data
User	3	Tap “Faces” category
Program	4	Display faces
User	5	Tap a face
Program	6	Display all images recognized to contain the face
User	7	View results

Scenario #3 | User Objective: Upload a dataset of images

Source	**Step**	**Action**
User	1	Tap “Upload” icon
Program	2	Display interface for uploading images or datasets
User	3	Tap “Dataset” option
	4	Select dataset to be uploaded
Program	5	Highlight and confirm selection
User	6	Tap “Confirm” button
Program	7	Display loading icon
	8	Notify user that dataset is uploaded

2.4 Constraints

Hardware Constraints

The application will be developed using Android Studio and will be coded on a Linux Machine. We aim to create an efficient application that will be tested and then accessed by a user on an Android device.

Security Constraints

Since this application is involved with detecting facial recognition, the user’s facial features will be detected when a camera is pointed which will then allow the features of the application to be performed. The application will also access the user’s gallery when a user wants to upload a picture. This will be sent to the server so an analysis can be performed on it. The user can also upload a dataset to the application which will be thoroughly accessed. These features will be taken into consideration in terms of security, by securely accessing and storing a user's data in a safe environment and respecting the user’s privacy.

User Constraints

We want to create such an application that will be of ease to all users. The functionality of the application must be straightforward and fast. We want it to be simple and user friendly and to do this we must understand a user's need in order to build an efficient application.

Time Constraints

We will use a facial recognition model, trained and tested using a neural network which will perform in conjunction with the user interface. We will have to execute many tests to make sure our model is as accurate as possible. We will need to set up a database server working with the application and the facial recognition model. We aim to present all listed out features of the application to which the user can access whichever feature they desire to. This project must be completed by 5pm Friday 7th May 2021. We aim to complete this project and submit all the relevant documents.

3. Functional Requirements

3.1.1 Frontend And Backend Communication

Description - Regardless of what we develop for the app (the frontend) or what we develop for the backend of the app on Amazon Web Services (AWS), it is all pointless if neither are implemented with the capabilities of communicating with each other properly. These two cannot exist in isolation and depend on working in tandem for functionality.

Criticality - This is the most important aspect of the project. The basis and purpose for every other feature is relying on the ability of the application to perform facial recognition on an image. If this feature is not functional, the operational value of the app is pointless. Essentially, there would be no point using the application beyond its starting screen.

Technical Issues - Since the backend will be stored on a cloud computing service, the user is required to have an internet connection so the application can communicate with AWS.

Dependencies with other requirements - A constant internet connection will be required. As well as that, the project code must be implemented properly to facilitate the communication.

3.1.2 Recognizing A Face With Facial Recognition

Description - When an image of a face is taken or uploaded, or if a dataset is uploaded, the system should be able to perform its neural network trained facial recognition model on any image presented.

Criticality - This is the most important aspect of the project. The basis and purpose for every other feature is relying on the ability of the application to perform facial recognition on an image. If this feature is not functional, the operational value of the app is pointless. Essentially, there would be no point using the application beyond its starting screen.

Technical Issues - The app has to be able to call the facial recognition system located within the backend in order to analyse any amount of images it receives. If one of those aspects is not functional, then the entire application will not work. A steady internet connection is required.

Dependencies with other requirements - The application (the frontend) and the numerous systems that perform the facial recognition (the backend) must be able to work together. If the implementation of this aspect of the project fails, this feature of the application will not function properly. (See FR 3.1.1)

3.1.3 Taking A Photo

Description - Upon opening the app for the first time, the user is presented with a large camera icon feature to take a photo of themself. The user taps the icon and takes them to their phone's camera and are assisted with taking an appropriate image of their face.

Criticality - This is one of the most fundamental and vital features of the application. If this feature does not work, the application will suffer since this means we are lacking a functional facial recognition system and the loss of a way of giving the app data.

Technical Issues - The app should tell the user if there is a face within a centered box of the camera, if they are wearing a mask, if the mask is being worn correctly or not, if the lighting or angle is poor and notify the user clearly. It also requires the user to have a camera on their Android device. A steady internet connection is required.

Dependencies with other requirements - The application (the frontend) and the numerous systems that perform the facial recognition (the backend) must be able to work together. This requirement relies on the facial recognition model being functional as well (See FR 3.1.1 and FR 3.1.2).

3.1.4 User Uploads Image To App

Description - The user should not only be required to take a photo with their camera to use the app. The user will be able to upload a previous image of themself from their phone to the app to access all of the features and functions of the application as well.

Criticality - This is an important feature of the application. Giving the user multiple options and allowing the application multiple methods of acquiring data to test and utilise is significant. As well as that, some users may not have functional cameras on their Android phone.

Technical Issues - The user is required to have enough memory on their phone to have the images they want to upload to the app. The app must be able to receive the image and prepare itself to send it to the database. A steady internet connection is required.

Dependencies with other requirements - The frontend and backend must be able to communicate with each other. The app must be able to perform facial recognition on the image in order to verify that the image contains a face. (See FR 3.1.1 and FR 3.1.2)

3.1.5 User Uploads Dataset To App

Description - The user should be able to send a dataset of images to the application in order to perform analysis on the dataset. This will allow users to analyse hundreds or thousands of images contained within the dataset.

Criticality - This is a crucial aspect of the project. Allowing users to perform an analysis on thousands of images will give the user and application a lot of data to share and interact with. Having the ability to simultaneously perform facial recognition and gather data gives an extra dimension to the application.

Technical Issues - The user must have the memory storage on their phone to be able to store a dataset. The app must be able to receive the dataset and prepare itself to send it to the database. A steady internet connection is required.

Dependencies with other requirements - The frontend and backend must be able to communicate with each other (see FR 3.1.1)

3.1.6 Analyze Dataset Of Images

Description - The programs that run the analysis on the dataset of images must be accurate, effective and well implemented as possible within a reasonable margin of error (+5% / -5%).

Criticality - This is a very important part of the project that gives extra dimensionality and purpose to the project. Without this functional requirement, the application would solely consist of a database and a facial recognition model built by a neural network, which would be insufficient.

Technical Issues - The app must be able to upload new images or datasets. The system that performs the facial recognition will need to be able to identify facial features within the images, which allows the analysis to be performed. Finding data like whose face is being recognized, are they wearing a mask or not and is the mask being worn correctly or not, just would not be possible if the facial recognition model does not work.

Dependencies with other requirements - The frontend and backend must be able to communicate with each other. As well as that, this requirement depends on the app being able to take a photo, upload a photo and upload a dataset (See FR 3.1.1, 3.1.3, 3.1.4 and 3.1.5).

3.1.7 Temporarily Hold Onto Faces

Description - After the application receives a photo taken, an uploaded image or an uploaded dataset, the system must go through any photo and apply the facial recognition model to it. Any faces detected will be temporarily handled by our system and let go when the app's operations have been performed. No data submitted by the user will be saved.

Criticality - This is crucial in the overall operations of the application. The application will need the ability to temporarily hold onto the faces and other information that many features of the application will need in order to perform their functions.

Technical Issues - A stable internet connection is required.

Dependencies with other requirements - The frontend and backend must be able to communicate with each other. As well as that, this requirement depends on the app being able to take a photo, upload a photo and upload a dataset (See FR 3.1.1, 3.1.3, 3.1.4 and 3.1.5).

3.1.8 Sending Faces And Analyzed Data From App To User

Description - The user can view a photo just taken by the camera or uploaded, and can also tap the Start Analysis button to request an analysis of a dataset. The backend must be able to retrieve the faces and data and display the results and findings acquired to the user.

Criticality - It is vital that the user is able to request and see the data and information gathered by the application. Without this functional requirement, the interaction between the app and the user would be completely one-sided.

Technical Issues - The system architecture must be well designed for the app (frontend) to acquire the faces and information found by the backend. A stable internet connection is required.

Dependencies with other requirements - The application must be able to communicate with the backend. The user must be able to upload images or a dataset of images in order to view the data from them within the app. (See FR 3.1.1, 3.1.3, 3.1.4 and 3.1.5).

3.1.9 Optimally Quick Performance

Description - When the user uploads an image, takes a photo or performs an analysis, they must wait through a loading screen in order to continue using the application. To make sure there is a smooth presentation and an ideal user experience, they should be able to perform the tasks they have set to do within a short period of time.

Criticality - If the performance of the app was slow, the functionality and the purpose of the app would still be intact. However, an easy and comfortable user experience is very significant when using any application.

Technical Issues - If the Android device has a slow internet connection, or the systems that run the application and facial recognition are too slow, this will affect the user experience. A steady internet connection is required.

Dependencies with other requirements - The performance of the application is dependent on an efficient development of all of its features. (See all FR's above.)

3.1.10 Application Must Deal With High Traffic Loads

Description - If there are multiple users on the app at the same time, the application is required to handle the large amounts of traffic in the form of calls and requests that would be made.

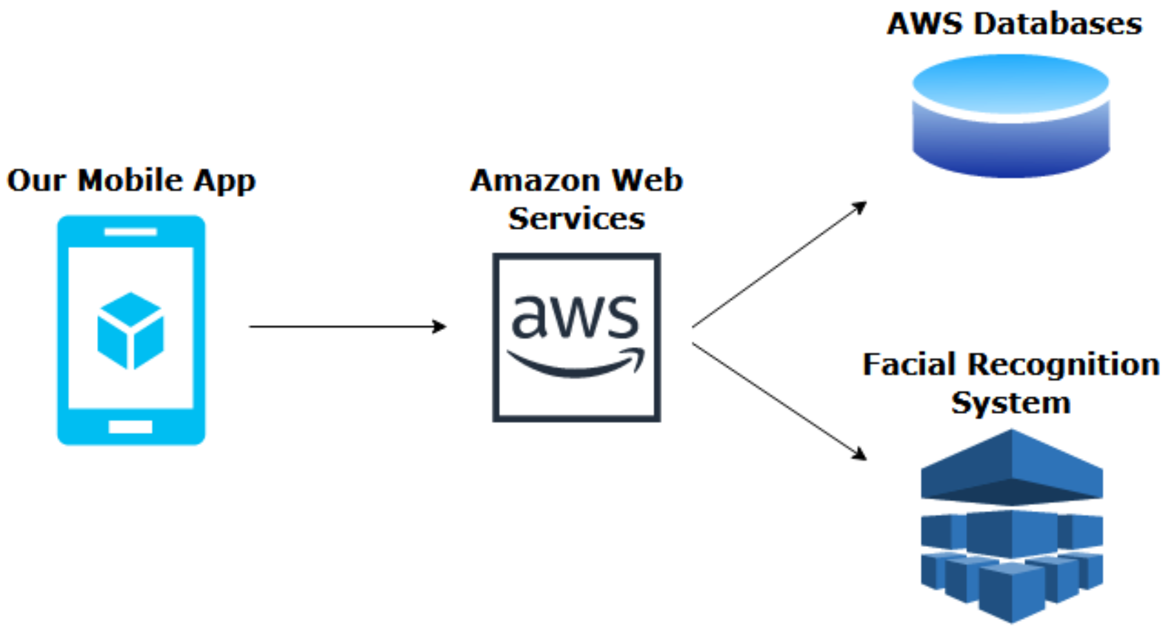
Criticality - Even if only one person could access the application at any given time, the app would still be fully operational and fulfilling its aim. However, this is not the intended purpose for the app and multiple users should be able to use the app at once.

Technical Issues - If the server cannot deal with the load that multiple users at the same time might bring, then the app must be created with only one person at a time allowed to use it. A stable internet connection is required.

Dependencies with other requirements - The efficiency and effectiveness of the code structure, as well as how well the code is optimised for all features, will affect whether the app can deal with high traffic loads. (See all FR's above.)

4. System Architecture

4.1 General System Architecture of the Project



Mobile App

The application will be designed and developed using Android Studio. Google has continuously worked to improve and refine this software platform and has now become one of the most popular app development platforms for good reason.

Amazon Web Services

The backend of the application will be located on this cloud computing service. This results in the project having a serverless structure model. As a team, we will not be required to handle the management and maintenance of any servers. This will allow for high expandability and high availability if necessary for the project.

AWS Databases

This is the main tool we will use in order to hold onto the data that we will train and test our facial recognition system with for our application.

Facial Recognition System

The facial recognition model will be built using an open source machine learning platform called Tensorflow. The model will be trained and tested with datasets containing thousands of images with faces that have masks on and masks off, as well as datasets that have masks worn incorrectly.

Testing the App

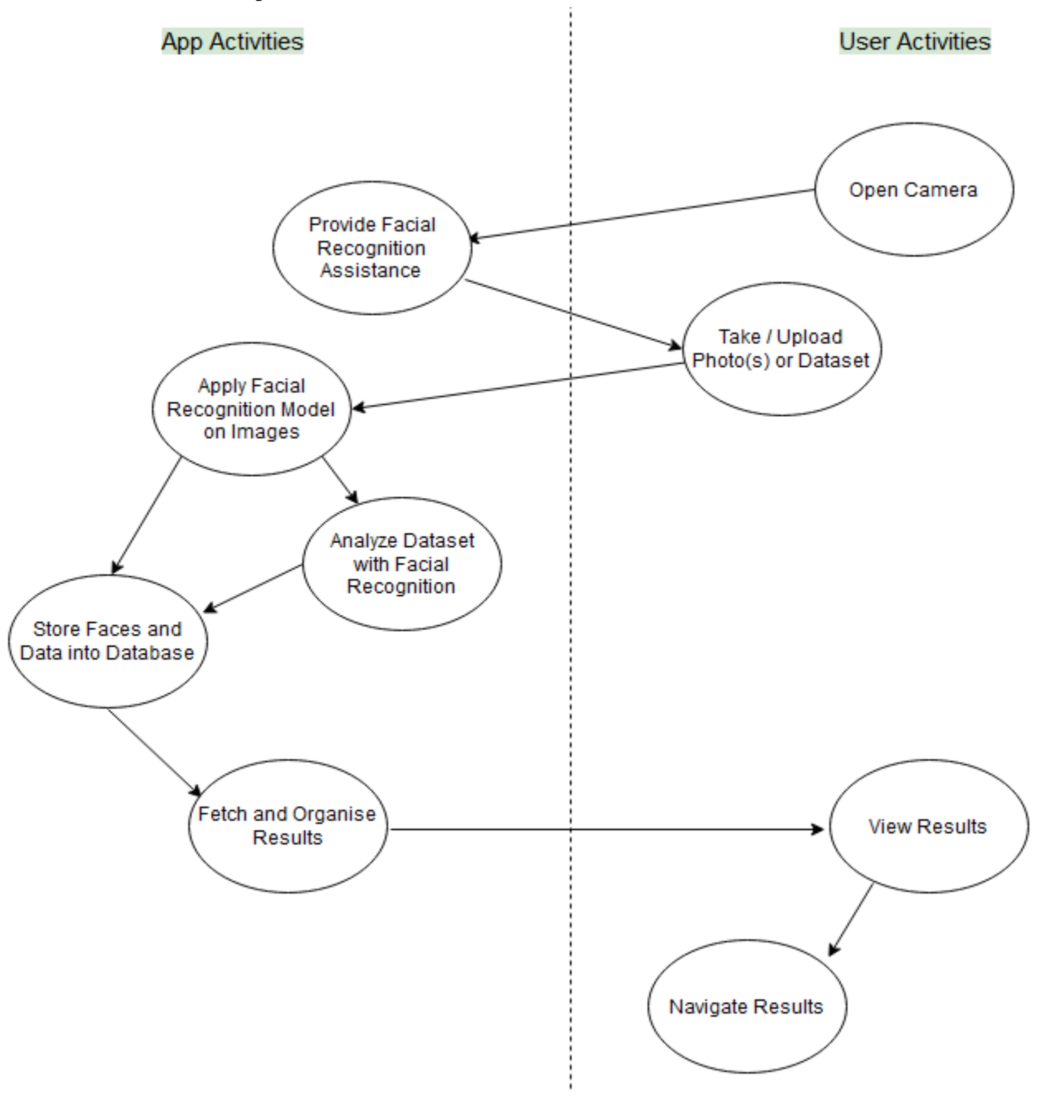
While coding the app on Android Studio, to check the update on the user interface, we will run tests on the application's functionality on an Android device manually by connecting it to the linux machine. To check the constant performance of the app, we will test it using Logcat on Android Studio which displays any programming errors and makes debugging easier in a sense. To check if the app connects to the server, we will send data from the app when it is manually connected to the machine and see if it is being collected by the server.

Testing the Facial Recognition System

The training and testing data from the datasets will have a split ratio of 80% to 20% respectively. The dataset will be classified properly and the model will create predictions based on the classification of the images in order to facilitate the accuracy tests of the model. Any testing performed on the facial recognition system will be performed using the pytest framework. Unit testing like this will help us in debugging the code during every iteration while developing the code.

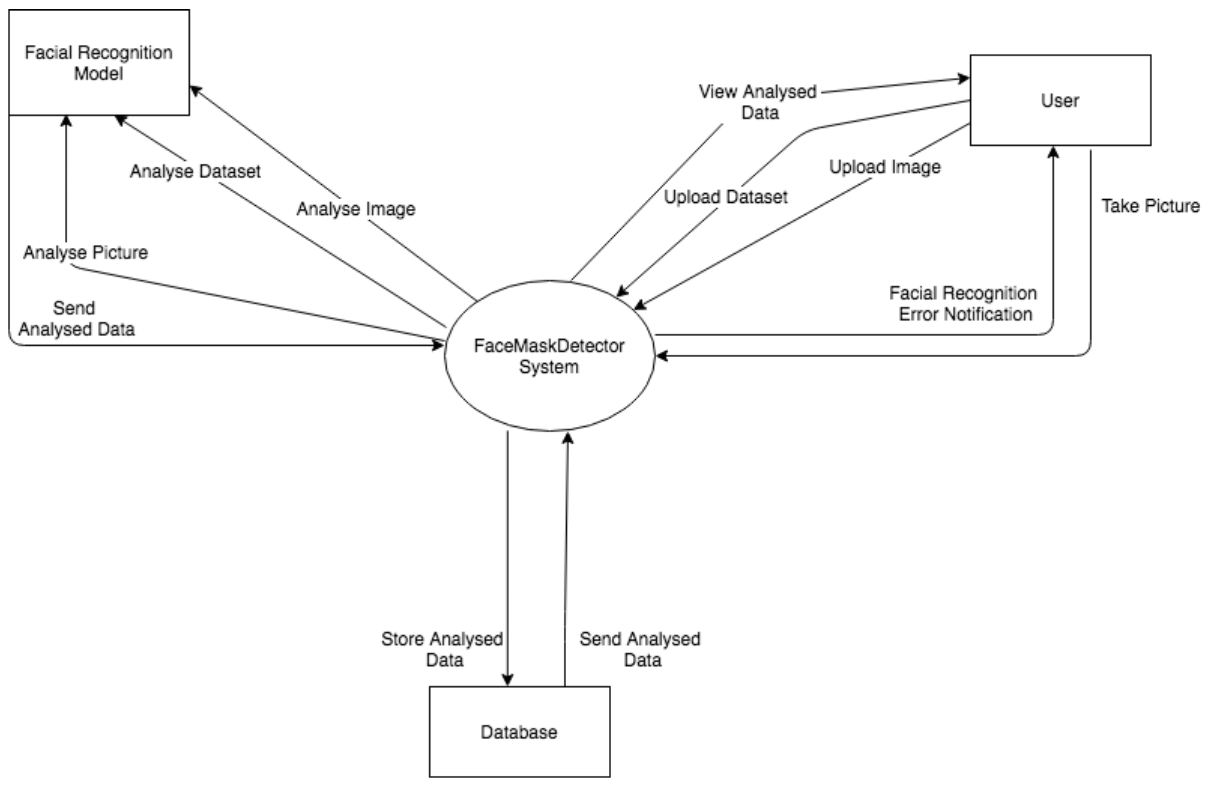
5. High-Level Design

5.1 Business Activity Model



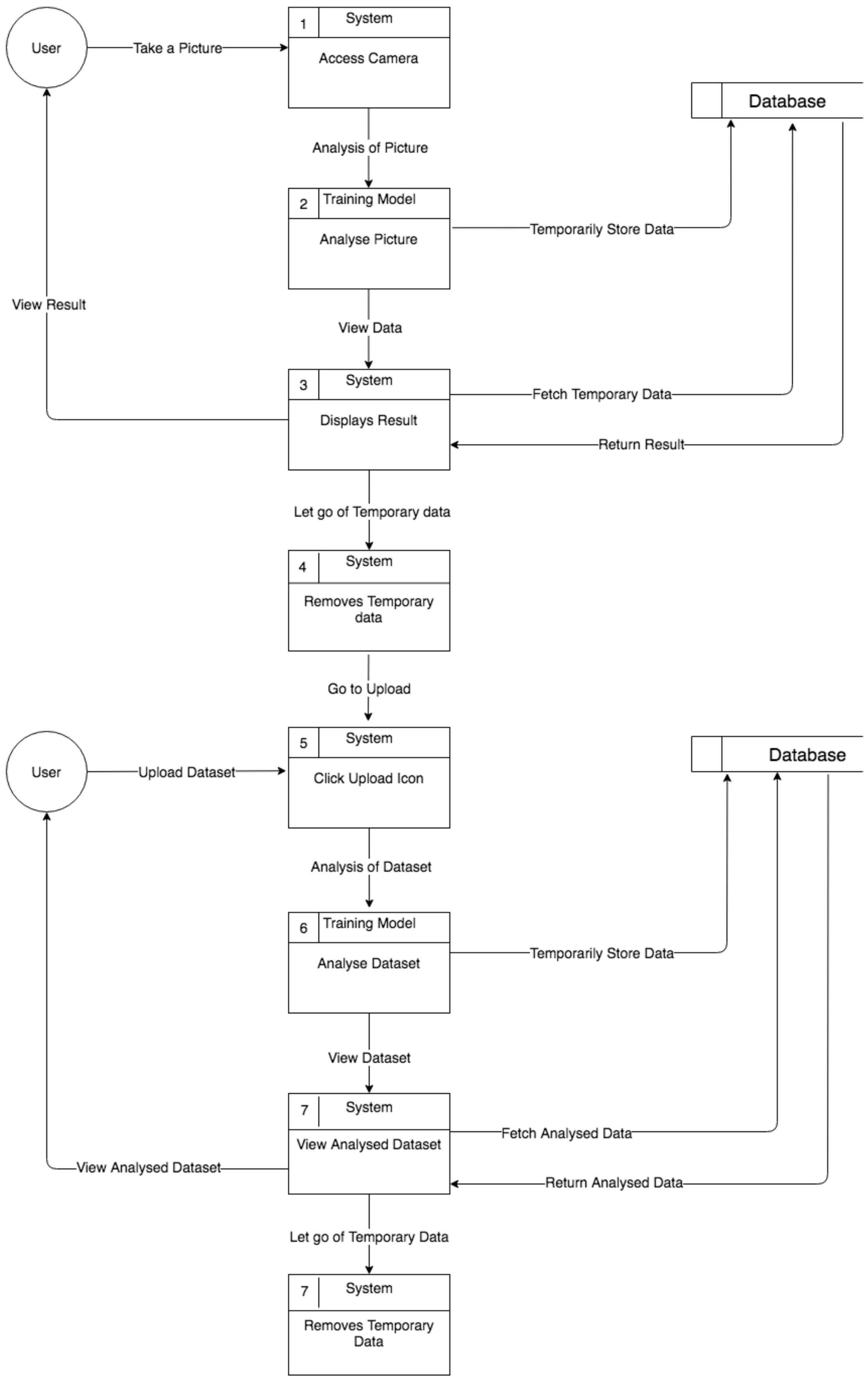
App Activities
User Activities
Open Camera
Provide Facial Recognition Assistance
Take / Upload Photo(s) or Dataset
Apply Facial Recognition Model on Images
Analyze Dataset with Facial Recognition
Store Faces and Data into Database
Fetch and Organise Results
View Results
Navigate Results

5.2 Context Diagram



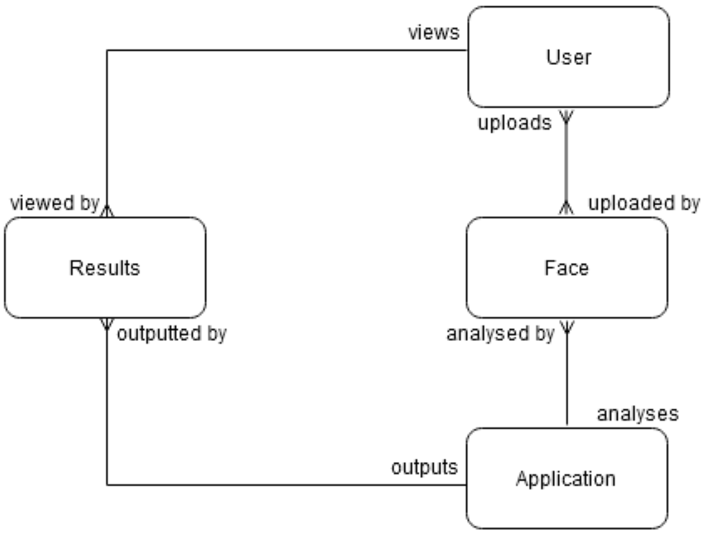
This Context Diagram gives an overview of the whole system and how it interacts with its external entities. For example, the User interacts with the system by taking a picture and if the picture is not detecting a face, a notification will be displayed.

5.3 Data Flow Diagram

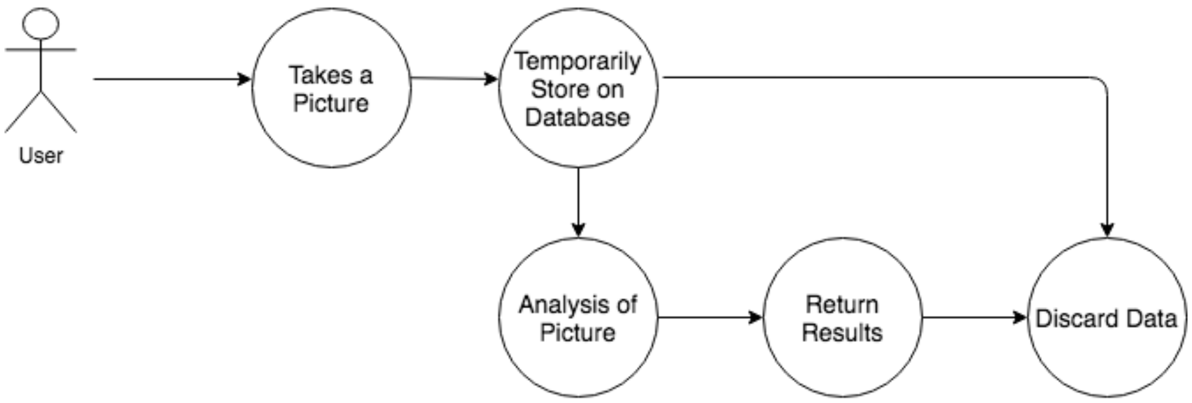


This Data Flow Diagram shows the flow of the data through the interaction between the system. It shows where and how the data is retrieved and stored. For example, it shows the data flow between a User who wants to take a picture and in response view the Result. In comparison, a User could want to upload a dataset to the app and wish to be returned an analysis of the dataset.

5.4 Logical Data Structure



5.5 Use Case Diagram



User

- Takes a Picture: User will take a picture of themselves by accessing the camera on the app.
- Temporarily Store on Database: This data will be stored temporarily on the database or if not needed it will be discarded.
- Analysis of Picture: Training model will view the picture and perform an analysis on it.
- Return Results: User will view the analysed picture, as a result.
- Discard Data: No data will be held on to permanently and will be discarded after the user has viewed the results.

5.6 Use Case Description

Use Case 1	User takes a photo of their face to see the result.
Goal in Context	User is able to access the camera on the app and take a picture of their face returning whether they are wearing a mask and if worn correctly or not.
Scope and Level	System scope, Database level
Preconditions	User has good lighting in order for the system to detect the face, the user can be wearing a mask or not for the system to detect and perform analysis. User must have stable internet condition on their device.
Success End Condition	System has detected users' faces, performed analysis and returns the result.
Failed End Condition	System cannot detect a user's face, cannot detect whether a user is even wearing a mask or not and cannot differentiate between being worn properly.
Primary, **Secondary Actors**	User, App Training Model, Database
Trigger	User downloads the app and clicks on the camera icon in the app.
Description **Step** **1** **2** **3**	**Action** User installs the app on their mobile phone. User opens the camera on the app to take a picture. User is returned with the result of their picture.
Extensions **Step** **1a** **2a** **3a**	**Branching Action** User must have an android phone in order to download this application. User clicks on the camera icon and takes a picture of their face. Once a user is returned with their result, the temporary data is let go and no data is actually stored.

Use Case 2	User views training data for the recognition model.
Goal in Context	User taps Database icon, taps Faces category and taps a face to be returned with results of all images that recognise that face.
Scope and Level	System Scope, Database Level
Preconditions	Training data must be stored on the database.
Success End Condition	User is returned with a result of images that contains the face searched for.
Failed End Condition	User cannot even tap on a face to be returned with any resulting images.
Primary, **Secondary Actors**	User, App Database
Trigger	User clicks on a face in the Face category.
Description **Step** 1 2 3	**Action** User taps Database icon. User taps Faces category and taps on a chosen face. User is returned with images that contain the chosen face.
Extensions **Step** 3a	**Branching Action** System accesses the database in response to return training data on the app for the user to view.

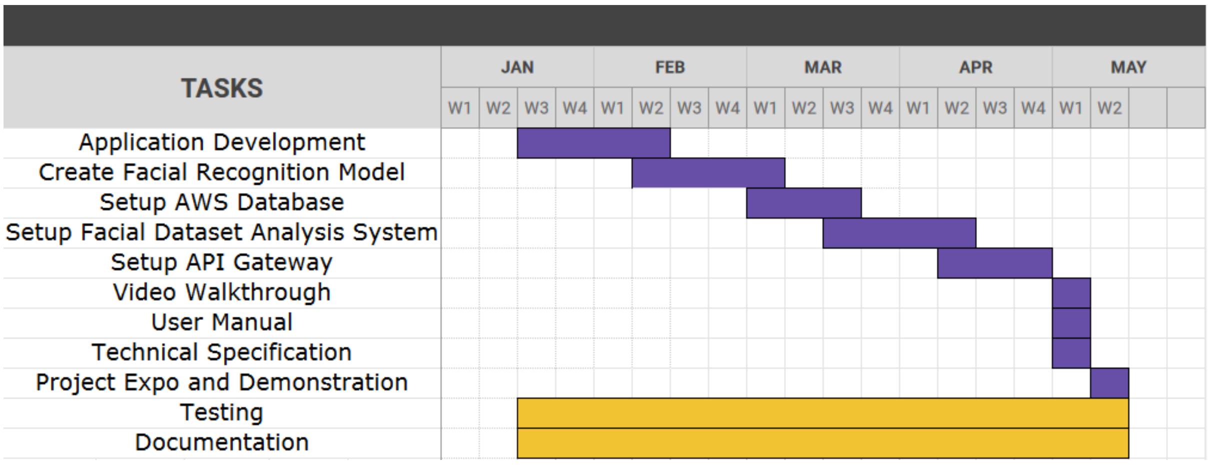
Use Case 3	User uploads a dataset of images.
Goal in Context	User uploads a dataset and is returned with an analysed dataset.
Scope and Level	System scope, Database Level
Preconditions	User must have storage on their phone to even have a dataset to upload. User must have a stable internet connection on their device.
Success End Condition	User is able to upload the dataset on the app and is returned with an analyzed dataset and can see the results.
Failed End Condition	User is not successful in uploading the dataset on the app and no analysis takes place.
Primary, **Secondary Actors**	User, App Training Model, Database
Trigger	User clicks onto the upload icon and clicks onto the dataset section.
Description Step **1** **2** **3** **4**	**Action** User clicks the dataset option on the app and selects the dataset to be uploaded. Dataset uploaded to the database temporarily. Analysis of the dataset takes place. User is returned with the analysed dataset on the app.
Extensions Step **1a** **3a** **4a**	**Branching Action** User can either upload an image or a dataset but the user selects the dataset section. Training model analyses the dataset uploaded by the user. Once a user is returned with an analysed dataset, the temporary dataset is let go from the database.

These are the following Use Case Descriptions. These use cases are derived from the operational scenario and they indicate what are the conditions of such a scenario occurring, what the goal of the scenario is, a detailed description through extensions on how the scenario works and who are the primary and secondary actors in the scenario.

6. Preliminary Schedule

6.1 Gantt Charts

This section provides a visual plan of the project timeline through the use of a Gantt Chart as seen below, with the project tentative start dates and end dates.



7. Appendices

7.1 Appendices

Tool used to build the diagrams: https://www.lucidchart.com/ , https://www.draw.io/

Web Host Service - https://aws.amazon.com/

Building an Android App - https://developer.android.com

Storing image is BLOB in SQLite database -
https://stackoverflow.com/questions/9357668/how-to-store-image-in-sqlite-database

Facial Recognition Model - https://www.tensorflow.org/

Dataset collection - https://www.kaggle.com/andrewmvd/face-mask-detection