CA400 - Final Year Computer Applications Project

Functional Specification - 29/11/20

Jester - controlling Android devices via hand gestures.

Paul Treanor	16347251
Colin Gorman	17354073

Table of contents

1. Introduction

1.1 Overview

Our project is to create a mobile app that uses gesture recognition to allow the user to interact with their mobile device without touching the screen. When within view of the phone's camera, the user will be able to make various hand gestures which the app will identify and respond to. The app will be able to distinguish between different gestures and respond in various ways to these different gestures.

Our initial plan is to implement functionality for gestures that can take photos, and, and start and stop video recording, but this could be expanded upon to add more features that use different gestures. The gesture recognition technology itself could be used in many other applications other than a mobile app, such as a camera for an elderly person to signal for help if they have fallen.

The Android app will be useful for taking pictures and videos with a phone without needing to actually physically interact with the phone. It will be useful for families taking group photos at the beach, athletes recording their technique, or amateur video makers who don't have a dedicated cameraperson. The hands-free nature of the app could be particularly useful during the Covid-19 pandemic, as it allows multiple users to interact with the device without having to touch it.

1.2 Business Context

The project is not being developed with a business goal in mind.

1.3 Glossary

OpenPose - Open source software that can analyse and track key points in images and videos of people.

Key points - In the context of pose estimation software, key points are areas in a person's body that are tracked by the software. For example shoulders, elbows, wrists, and fingertips.

Convolution neural network - a machine learning method that is most commonly applied to analyzing visual imagery.

2. General Description

2.1 Product / System Functions

The core functionality of the system is to allow users to control some aspect of their mobile device using hand gestures.

The system will be implemented as an Android mobile application. When the user enters the app, a stream of images is sent to a server at regular intervals (every second or so) to be processed in real-time. If a person is recognised to be making a known hand gesture, the app will then carry out a function specific to that hand gesture.

The functions the app is capable of, are taking and saving a photo, and starting, stopping and saving a video recording. There will be audio and visual feedback when the app recognises a user's gesture and carries out these functions. An adjustable timer can run between the gesture being given and the camera action being carried out.

While the app is running, a preview of the camera will be displayed on the screen with a simple user interface. This will allow the user to ensure that they are within the frame for any photos or videos, and also ensure that they are in the camera's field of vision so gestures can be recognised.

Alongside this main view, there will be a simple help view that will show the user which gestures are recognised by the app, as well as what functions those gestures trigger. The user can adjust the timer from within the help view.

2.2 User Characteristics and Objectives

The user base we are targeting the app towards is general users. Basic proficiency in using Android devices and installing the app via the Play Store will be required, but the app will be designed with ease of use in mind.

From a user's perspective, the requirements of the system are that a photo is taken, or video recording is started or stopped when the user makes specific gestures in view of the device's camera. Once the photo or video recording is taken the user will be able to access the file as they would any other image or video file on their phone, for example, using their device's file explorer application.

Users will expect a seamless user experience, so the gesture recognition will need to be fast and accurate, and the system will need to be robust.

Users will also expect the app to be secure. In order to recognise gestures, a stream of images of the user will be constantly sent from the phone to a server via an API. These images will need to be encrypted.

Accessibility guidelines will be followed for the system when possible to allow for our app to be usable by as many people as possible.

2.3 Operational Scenarios

Scenario 1: User takes a photo

1. The user opens the Jester application.
2. The user ensures they are within the field of view of the device's camera. This can be done as the camera's view is constantly shown to the user in realtime on the device's display.
3. The user makes a predetermined hand gesture in the view of the camera to make the device take a photo. This might be a thumbs-up gesture, for example.
4. An adjustable timer is triggered, then a photo is taken once the timer has stopped.
5. The user is shown audio-visual feedback from the app as the device's camera takes the photo.
6. The photo is saved to the device.

Scenario 2: User starts and stops video recording

1. The user opens the Jester application.
2. The user ensures they are within the field of view of the device's camera.
3. The user makes a predetermined hand gesture to start video recording in the view of the camera. This might be a peace sign gesture, for example.
4. A timer is triggered, then video recording begins.
5. The user is shown audio-visual feedback from the app as the device's camera begins recording.
6. As long as the device is recording, a red dot is displayed in the corner of the screen.
7. If the device senses that the user has made a predetermined gesture to stop recording the recording will stop.
8. The video is saved to the device.

Scenario 3: User enters 'Help' view

1. The user opens the Jester application.
2. The user selects the "Help" button on the top right corner of the main app view.
3. A new view is opened which contains information on the gestures the app can recognise, the function that each gesture triggers, as well as a setting for changing the camera timer.
4. The user can select the back button to return to the main app view.

Scenario 4: User adjusts camera timer

1. The user opens the Jester application.
2. The user enters the apps help view.
3. The user scrolls down to the settings heading.
4. A camera timer value can be adjusted. The input is an integer from 0-60, representing seconds.
5. The user selects enter on the keypad to apply their change.

6. The user can select the back button to return to the main app view.

2.4 Constraints

Hardware

OpenPose requires a powerful graphics processing unit (GPU) to run. While running on the integrated graphics unit of an Intel processor (like those found on most Windows laptops), it takes roughly a minute for OpenPose to return keypoint data on the single image. In order for our app to be responsive, the image processing will need to be done on a machine with a powerful GPU.

Time

We will be constrained by deadlines while developing the system. Because of this limited amount of time we will be required to focus on the core functionality of the application. We believe that with good prioritisation of tasks and time management we will be able to implement all the systems features within the set time.

Security

The mobile application will be sending images of people from the device's camera across a network API. Because images of people are considered personal information, GDPR will need to be adhered to[1] and ethics approval will be needed. In order to ensure that we are following those guidelines, we will need to keep our users informed of what the app is doing with their data, and the images sent across the API will need to be encrypted.

UX Design

Since our app is aimed at a general audience it should be as user friendly as possible. Extensive user testing will need to be carried out throughout development to ensure that the design is simple and intuitive to use.

The app should be designed with accessibility in mind. Android has extensive accessibility documentation for developers and we will follow their guidelines when possible. Organisations such as the W3C also provide resources we can use to maximise accessibility, usability, and inclusion, such as[2].

Responsiveness

The core functionality of the app, using gesture recognition to trigger camera functions, will need to be as responsive as possible. If there is a delay between the gesture being recognised and the camera action being triggered, then the convenience of controlling the device without needing to pick it up would be cancelled out by a bad user experience.

Cloud Computing Costs

We will need to use a cloud computing service to have access to a powerful GPU that can run OpenPose.

Amazon Web Services (AWS) offers such services at a per hour cost. We have access to €200 worth of free AWS credits through our GitHub student accounts. This should be enough server time to build our training and testing datasets of OpenPose data for the neural network, and have a functioning server for the demo day. If our free credits run out, we would have to use their pay-as-you-go service, which would not be ideal as students.

We are also looking at other cloud computing services which have similar free credits and pricing plans. It is necessary for us to stay within our allowance of hours on these platforms to not avoid racking up a large bill.

3. Functional Requirements

1. Take Photo Using Gesture

Description:
While the user is in the app and there are no other in-app processes running (such as a camera timer or video recording), the user can trigger the device to take a photograph by making a hand gesture within the camera's field of vision.

Brief Overview of Implementation of the Feature:
The app will access the device's camera API, sending a stream of images back to a server to be processed. Each of these images is first processed by OpenPose, and then by a convolutional neural network which can identify if a person is making a gesture in the image. If the 'take photo' gesture is recognised in any of the images, a response is sent from the server to the device signalling to take a picture. A timer is activated before the photo is taken if the app's timer value is set to greater than 0. The camera then takes a picture and saves it to the device's gallery.

Criticality:
Using gesture recognition to take a photograph is the most core component of the app. It is critical that this process works and it will be our main priority early in the development of the system.

Technical Issues:
1. Convolutional Neural Network with OpenPose for image processing:
    a. Could potentially be difficult to implement.
    b. Will need to be tuned to be accurate.
    c. Requires a large gesture image training set.
    d. Potential performance issues as OpenPose is resource-intensive.
    e. Requires use of a server which we have limited access to.
2. Image stream from device to server for processing:
    a. Images will need to be encrypted between the device and the server.

Dependencies on other Requirements:
1. Dependent on the camera's field of vision being displayed on the screen.

2. Start and Stop Video Recording Using Gestures

Description:
While the user is in the app and there are no other in-app processes running, the user can trigger the device to start video recording by making a specific hand gesture within the camera's field of vision. The video will continue to record until the "stop recording" gesture is made within the field of vision of the camera. The video recorded will be saved to the device.

Brief Overview of Implementation of the Feature:
This feature will be implemented in a similar way to the first requirement, although two separate gestures will need to be recognised this time, one for starting the video recording, and one for stopping it.

Criticality:
This requirement is important to demonstrate the system's ability to differentiate between gestures and take different actions depending on the gesture it recognises.

Technical Issues:
The technical issues with implementing this requirement will be similar to those with the first requirement.

The gestures for the camera functions will need to be distinctive enough from each other so the neural network can accurately classify them.

Dependencies on other requirements:
1. Dependent on the camera's field of vision being displayed on the screen.

3. Photos and Videos Automatically Save to Device

Description:
After a photo or video is taken using the app, the photo or video will automatically be saved to the device.

Brief Overview of Implementation of the Feature:
Saving photos and videos can be implemented with the Android camera API.

Criticality:
This requirement is critical for the app.

Technical Issues:
No difficult technical issues expected.

Dependencies on other requirements:
1. Take photo using gesture
2. Stop and start video recording using gestures

4. Camera Preview Displayed on Screen in Realtime

Description:
While the app is running footage of the camera’s field of vision will be displayed on the device’s screen in real-time. This is a standard feature in camera apps as it allows the user to see what they are taking a photo of.

Brief Overview of Implementation of the Feature:
The camera preview will be implemented using the Android Camera API.

Criticality:
This feature is not critical to the app's functionality, but it is helpful for the user to see if the gesture they are making is within the camera's field of vision.

Technical Issues:
Displaying the camera preview, recording, and sending an image stream back to the server at the same time could be difficult to implement. Those processes would all have to run in parallel.

Dependencies on other requirements:
1. Not dependent on any other requirements.

5. Help Screen can be Viewed

Description:
The app will have a simple help view so the user can familiarise themselves with the gestures the app recognises and learn what functions are triggered by those gestures. The user can also adjust the camera timer via the help screen.

Brief Overview of Implementation of the Feature:
The help view will be implemented as a separate view from the main app view.

Criticality:
A help view is a nice edition of the app and would improve user experience, but it is not a critical feature of the app.

Technical Issues:
No technical issues expected.

Dependencies on other requirements:
1. Not dependent on any other requirements.

6. Audio/Visual Feedback when Photo Action is Triggered

Description:
Like most applications that use a camera, there will be audio and visual feedback when a picture is taken. This will be a shutter sound occupied with the phone screen replicating a camera shutter closing

Brief Overview of Implementation of the Feature:
An MP3 file that will play when the camera is triggered can be stored locally and implemented with the Android Studio SoundPool class.

The shutter animation will be implemented as a bitmap graphic animation played on top of the main camera view.

Criticality:
Audio and visual feedback is critical for users with visual or hearing impairments. It allows the user to receive different methods of feedback to verify that the camera action is triggered.

Technical Issues:
No technical issues expected.

Dependencies on other requirements:

1. Take photo using gesture

7. Feedback to Show when Recording Action is Triggered

Description:
Similar to the photo feedback, an audio cue should sound to signal when the recording starts. Instead of a camera shutter animation, a red dot will appear on the screen as the visual feedback for the recording.

Brief Overview of Implementation of the Feature:
An MP3 file that will play when the camera is triggered can be stored locally and implemented with the Android Studio SoundPool class.

The red dot will be implemented as a bitmap graphic animation played over the top of the main camera view.

Criticality:
Similar to the camera feedback, feedback is also required for the recording process as well.

Technical Issues:
No technical issues expected.

Dependencies on other requirements:

1. Start and stop video recording using gesture

8. Timer Occurs Between Gesture Recognition and Camera Action

Description:
Add a timer function to tell the user when the photo or recording will start to ensure the user is in position and ready. A countdown animation will display the seconds remaining before the photo is taken or video begins recording.

Brief Overview of Implementation of the Feature:
The timer will be implemented as a variable read by the timer method. This method would delay when the photo is taken or when the video begins recording.
The animation will be implemented using bitmap graphics of digits, with different digits displayed depending on the time left before the image is taken.

Criticality:
Not critical to the app, but this feature will lead to a more positive user experience.

Technical Issues:
Displaying the time remaining on top of the screen by showing images of digits could be challenging.

Dependencies on other requirements:
1. Take photo using gesture
2. Start and stop video recording using gesture

9. Timer can be Adjusted

Description:
Allow the user to change the time length of the timer to add more flexibility. The timer length can be adjusted from the help view.

Brief Overview of Implementation of the Feature:
The user can adjust the timer to an integer between 0-60, which represents seconds. This changes the time variable in the timer function which is run before the photo is taken of the video begins recording.

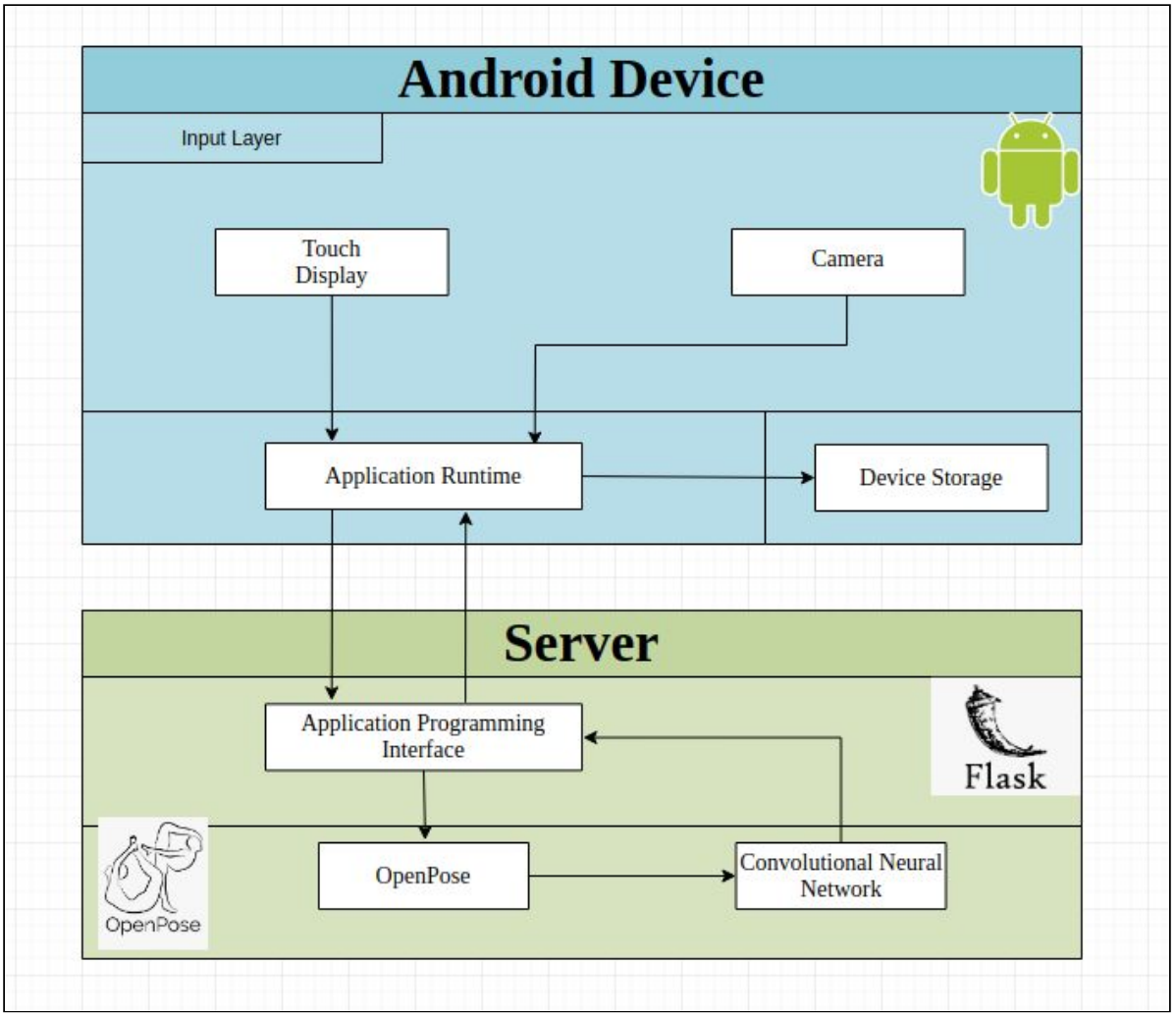
Criticality:
This feature is not critical, but improves the user experience of the app.

Technical Issues:
No technical issues expected.

Dependencies on other requirements:
1. Timer occurs between gesture and camera action.
2. Take a photo using a gesture.
3. Stop and start video recording using gestures.

4. System Architecture

System Architecture Diagram:



This high-level system architecture diagram shows the relationship between the two main components of the system, the Android app and the server.

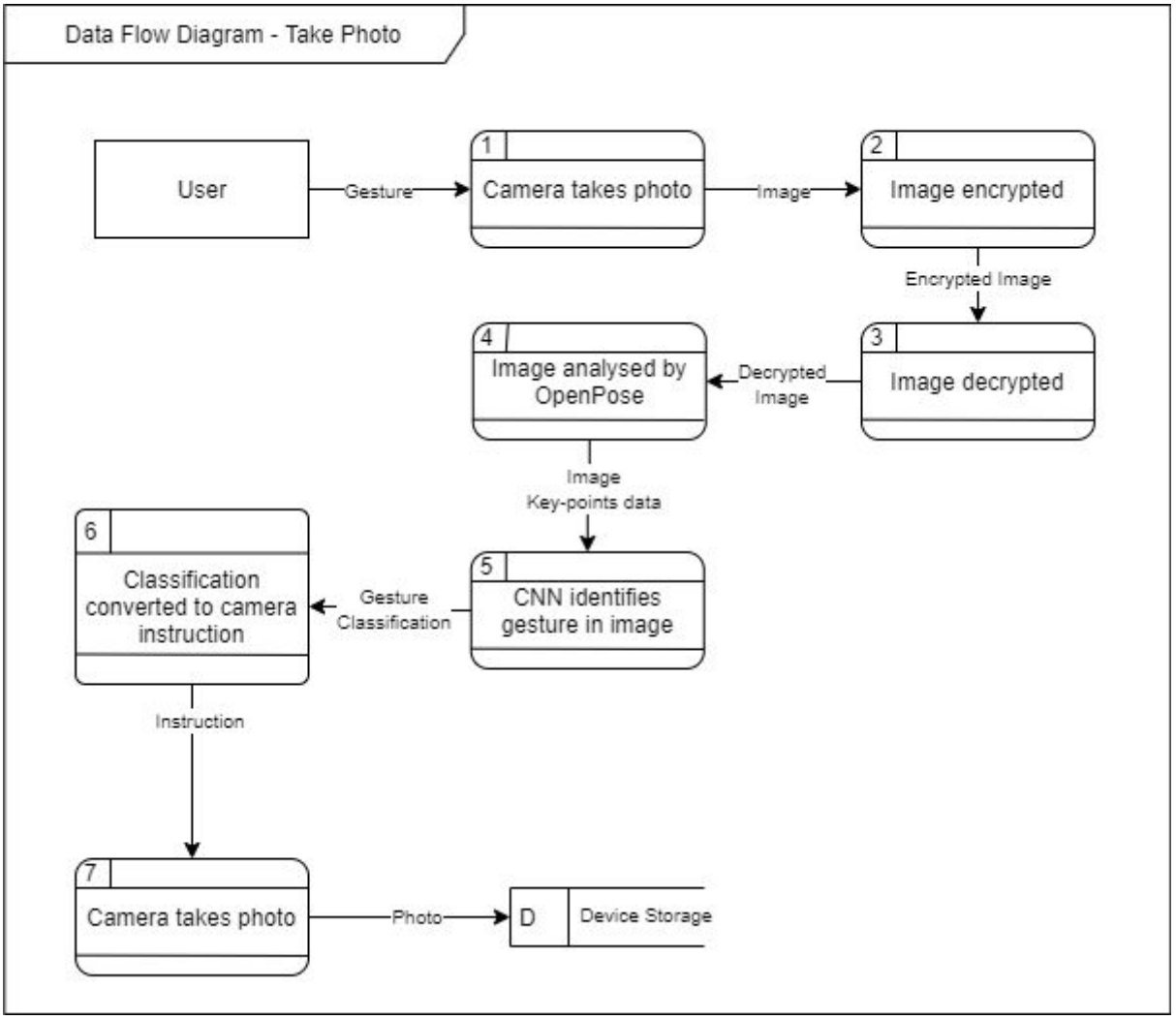
At the top level, there is the application's user interface on the mobile device. Here the user can interact with the app via the device's camera or touch screen. The touch display and camera connects to the application runtime which communicates with the API on the server level of the system. The Android Application also interacts with the device's local storage, where it saves photos and videos taken with the app.

The server will be hosted on a cloud computing service such as AWS and it will be where the image processing is done. The server and smartphone will communicate via an API. This API could be implemented as a REST API using a framework such as Python Flask, or as a WebSocket API.

Images are sent from the device's camera to the server at regular intervals. This image stream is processed in the server by OpenPose and a neural network which can detect and classify hand gestures.

5. High-Level Design

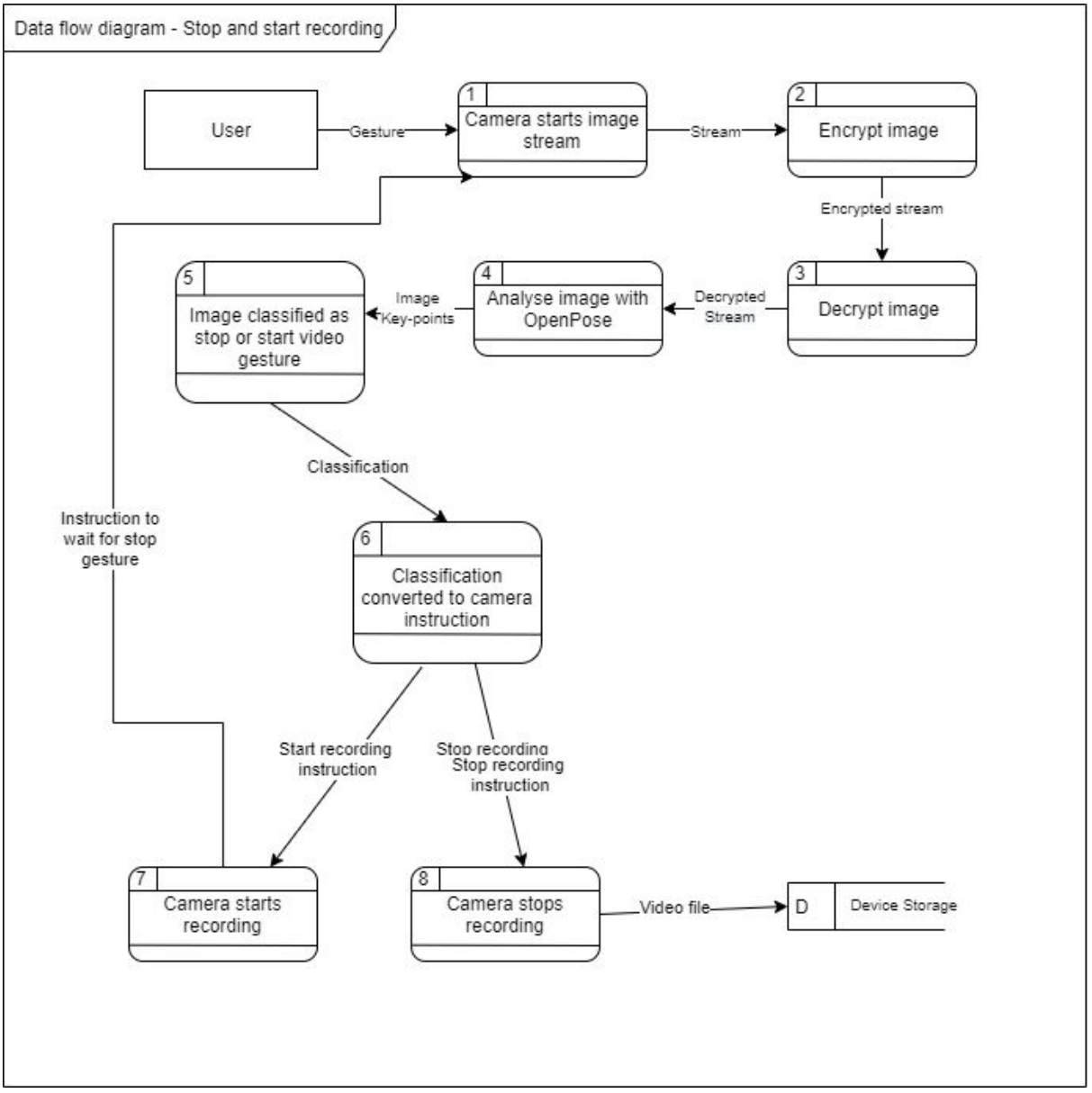
Data Flow Diagram - Use gesture to take photograph:



This data flow diagram shows what processes will be used to make the phone take a picture when the user makes a certain gesture.

The user provides the system with a gesture. A picture of this gesture is taken as part of a stream of images taken at regular intervals. The image is then encrypted and sent to the server. Once the image arrives at the server it is decrypted and analysed by OpenPose. A CNN that can classify the output of OpenPose as gestures will classify the gesture as the "photo gesture". This classification is converted to an instruction for the camera to take a photograph. This photo is then stored in the device's local storage.

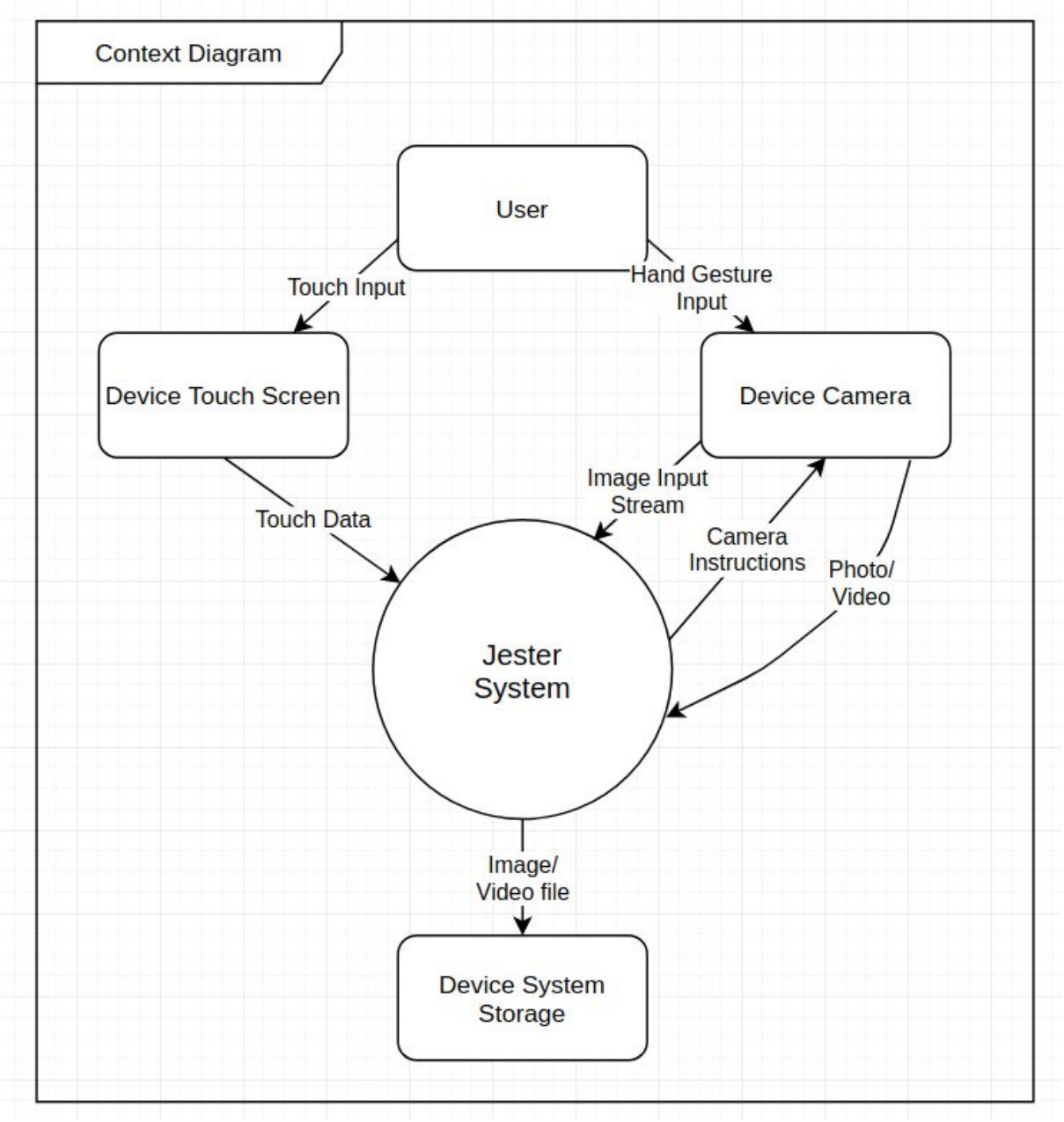
Data Flow Diagram - Use gestures to start and stop video recording:



This data flow diagram shows how a series of gestures lead to the camera to start and then stop recording.

The camera is continuously sending images back to the server to be processed at regular intervals. When a ‘start recording’ gesture is recognised, an instruction is sent to the camera to start recording and the system waits for a stop recording gesture. When an image with the stop recording gesture is identified, the camera stops recording and saves the video.

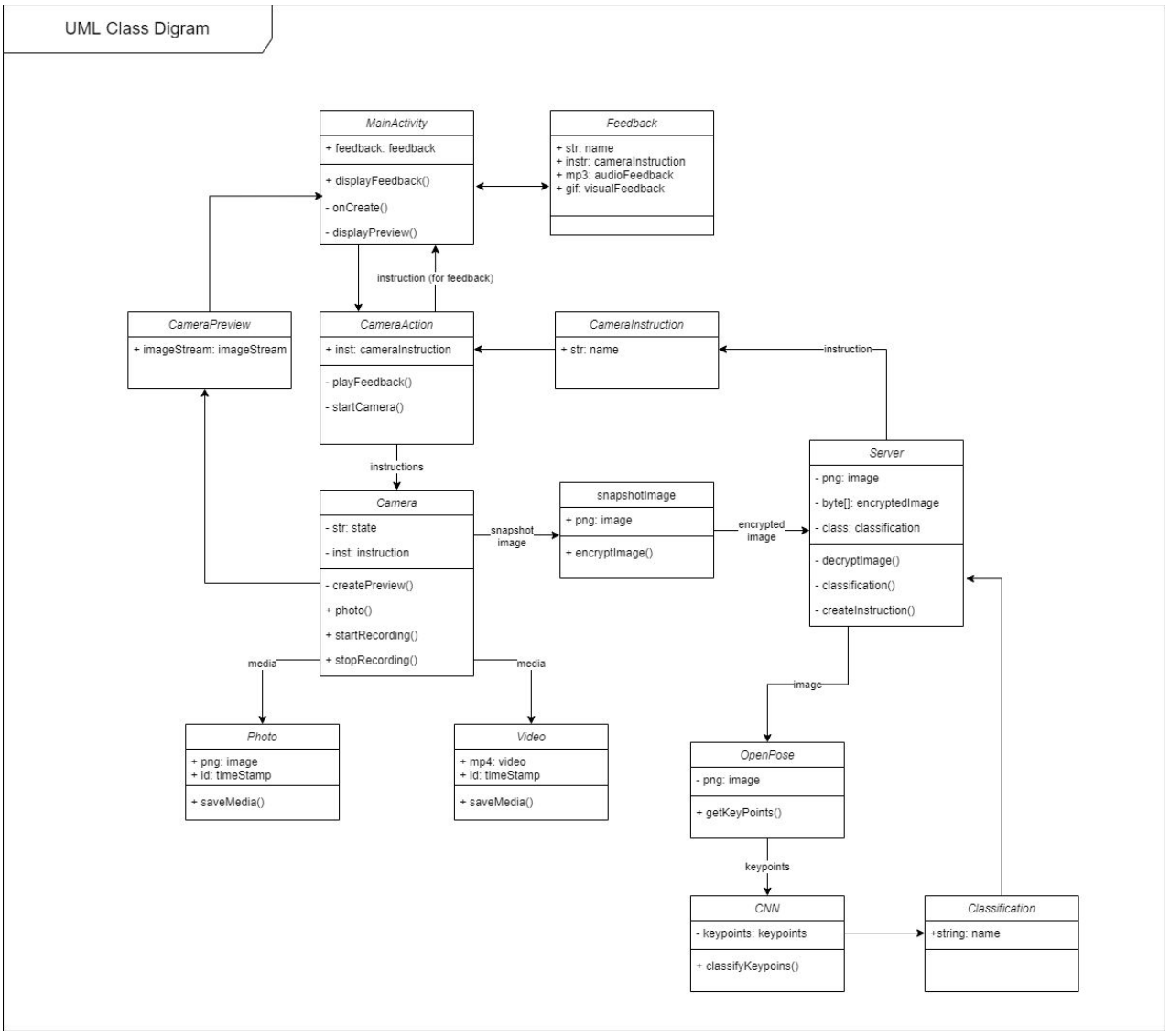
Context Diagram:



The context diagram shows how our system will interact with the user and the Android device.

The user interacts with the app via the touch screen and device camera. The camera sends the system images and videos, and receives instructions from the system. The output of the system is the photo or video taken by the camera, which is saved in the device's local storage.

UML Class Diagram:



The classes, attributes, and methods in this diagram show a rough idea of how our system will be implemented and how the various objects within the system will interact with each other.

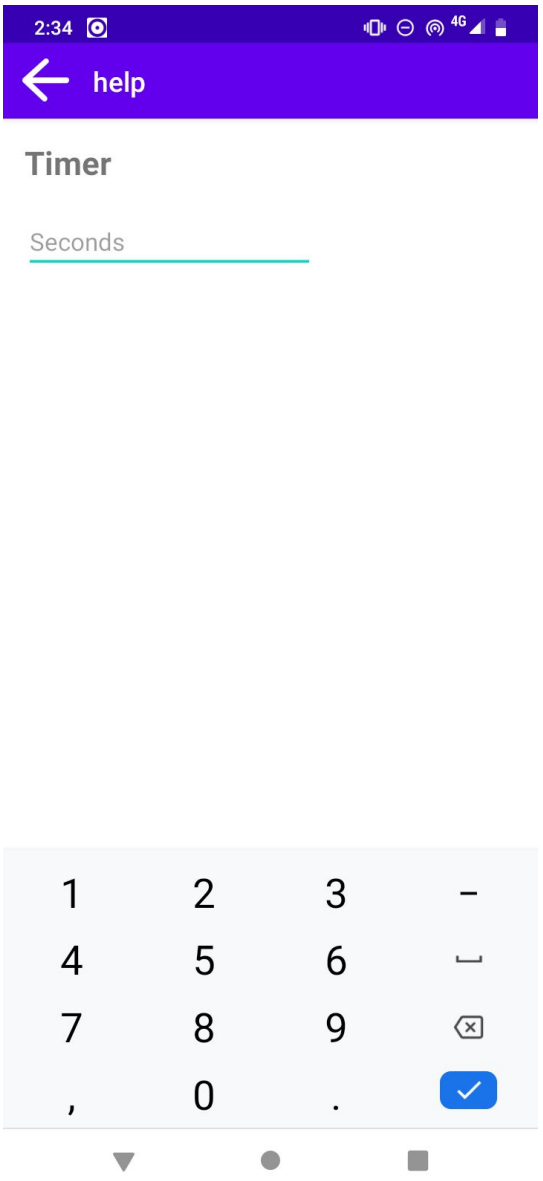
The app is displayed by the MainActivity class, which creates the CameraAction and Camera classes when the app is opened. The Camera class creates a CameraPreview instance, which is displayed on the screen by MainActivity.

The image steam produces a SnapshotImage object, which is encrypted and sent to the server class. From here the image is processed by the OpenPose and CNN classes to create a Classification object. This object can be converted to a CameraInstruction, which tells the camera what action to take and tells the MainActivity what audio/visual feedback to display.

6. User Interface Mock-ups

Home view:

Help view (adjusting timer):



Description:

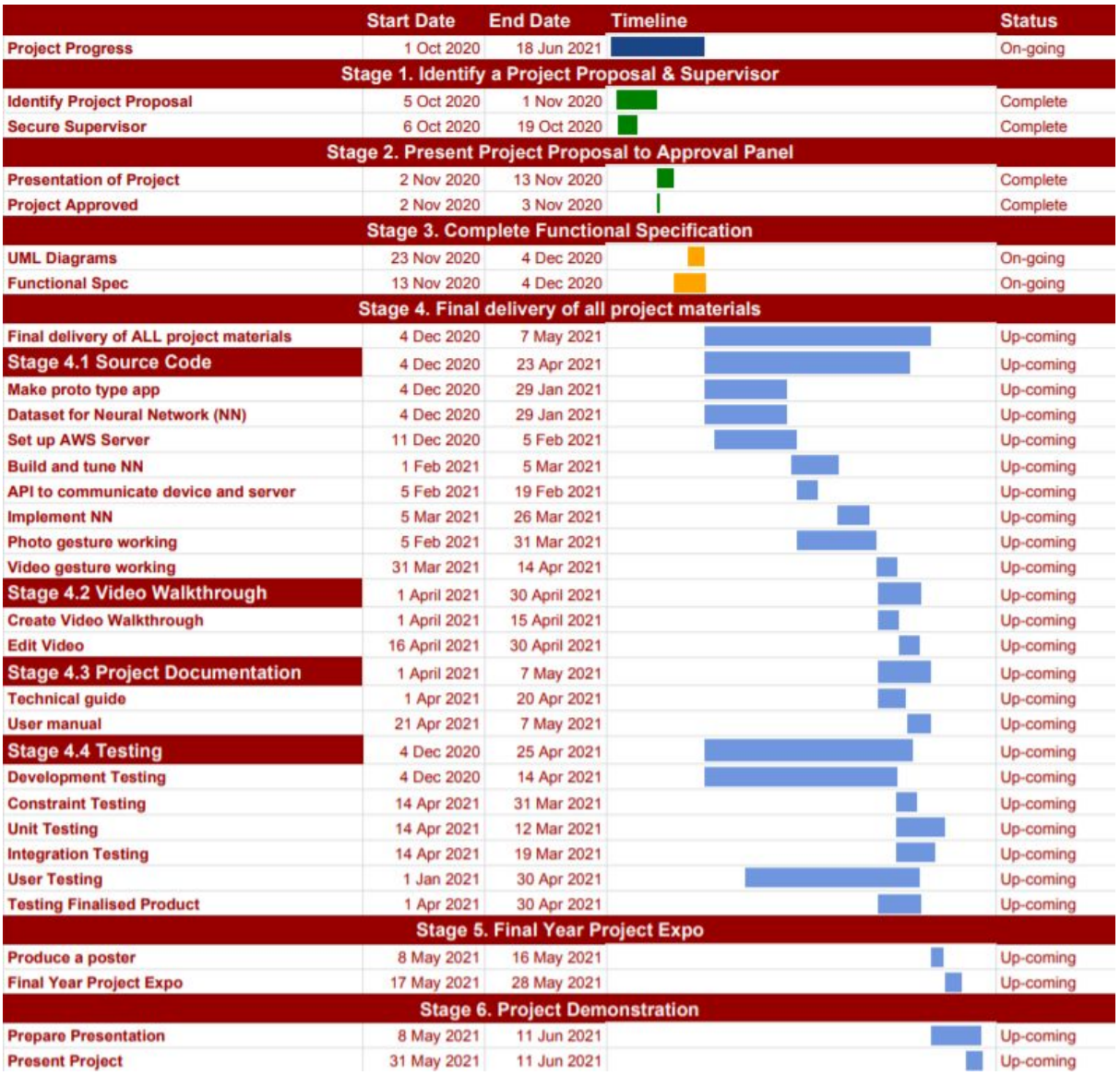
These user interface mockups show the two main views in the app, the camera view and the help view.

Photos can be taken from the home view, where the camera's preview is displayed. When a recognised gesture is detected in this view a photo or video is taken.

In the second mockup image, the user is adjusting the camera's timer. This setting will be adjusted at the bottom of the help view.

7. Preliminary Schedule

GANTT Chart:

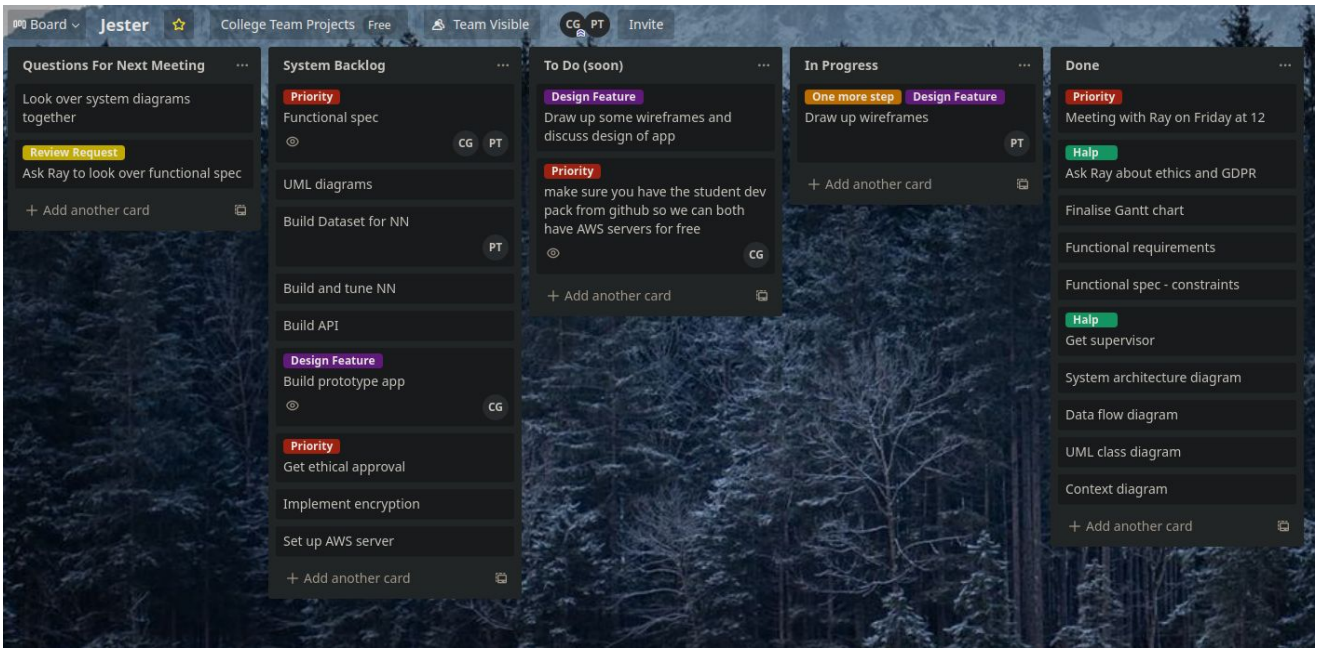


	Start Date	End Date	Status
Project Progress	1 Oct 2020	18 Jun 2021	On-going
Stage 1. Identify a Project Proposal & Supervisor			
Identify Project Proposal	5 Oct 2020	1 Nov 2020	Complete
Secure Supervisor	6 Oct 2020	19 Oct 2020	Complete
Stage 2. Present Project Proposal to Approval Panel			
Presentation of Project	2 Nov 2020	13 Nov 2020	Complete
Project Approved	2 Nov 2020	3 Nov 2020	Complete
Stage 3. Complete Functional Specification			
UML Diagrams	23 Nov 2020	4 Dec 2020	On-going
Functional Spec	13 Nov 2020	4 Dec 2020	On-going
Stage 4. Final delivery of all project materials			
Final delivery of ALL project materials	4 Dec 2020	7 May 2021	Up-coming
Stage 4.1 Source Code	4 Dec 2020	23 Apr 2021	Up-coming
Make proto type app	4 Dec 2020	29 Jan 2021	Up-coming
Dataset for Neural Network (NN)	4 Dec 2020	29 Jan 2021	Up-coming
Set up AWS Server	11 Dec 2020	5 Feb 2021	Up-coming
Build and tune NN	1 Feb 2021	5 Mar 2021	Up-coming
API to communicate device and server	5 Feb 2021	19 Feb 2021	Up-coming
Implement NN	5 Mar 2021	26 Mar 2021	Up-coming
Photo gesture working	5 Feb 2021	31 Mar 2021	Up-coming
Video gesture working	31 Mar 2021	14 Apr 2021	Up-coming
Stage 4.2 Video Walkthrough	1 April 2021	30 April 2021	Up-coming
Create Video Walkthrough	1 April 2021	15 April 2021	Up-coming
Edit Video	16 April 2021	30 April 2021	Up-coming
Stage 4.3 Project Documentation	1 April 2021	7 May 2021	Up-coming
Technical guide	1 Apr 2021	20 Apr 2021	Up-coming
User manual	21 Apr 2021	7 May 2021	Up-coming
Stage 4.4 Testing	4 Dec 2020	25 Apr 2021	Up-coming
Development Testing	4 Dec 2020	14 Apr 2021	Up-coming
Constraint Testing	14 Apr 2021	31 Mar 2021	Up-coming
Unit Testing	14 Apr 2021	12 Mar 2021	Up-coming
Integration Testing	14 Apr 2021	19 Mar 2021	Up-coming
User Testing	1 Jan 2021	30 Apr 2021	Up-coming
Testing Finalised Product	1 Apr 2021	30 Apr 2021	Up-coming
Stage 5. Final Year Project Expo			
Produce a poster	8 May 2021	16 May 2021	Up-coming
Final Year Project Expo	17 May 2021	28 May 2021	Up-coming
Stage 6. Project Demonstration			
Prepare Presentation	8 May 2021	11 Jun 2021	Up-coming
Present Project	31 May 2021	11 Jun 2021	Up-coming

This GANTT Chart shows our targets for when we want to complete each aspect of the project. As time moves on the progress bars will change colour in respect to the task's due date. Green is *completed*, yellow is *in progress*, light blue is *up-coming*, red is *due to start* and dark red is *behind schedule*.

Some of the dates of deadlines are our best guesses as they have not been confirmed. While none of the dates on this chart are set in stone, we will do our best to commit to this schedule.

Trello Board (screenshot):



We are using a Trello board in conjunction with our Gantt chart.

With the Trello Board, we can easily see the tasks on our current to-do list. It is easy to add notes and label tasks with marks such as “priority”. There is also a list for questions to ask our supervisor, a list for tasks that are pending and also a list for tasks marked done. We have used Trello for past group projects and found it both easy to use and useful to quickly assign tasks to one another and to help us keep on top of things.

8. Appendices

References:

1. **Information on GDPR**

https://gdpr-info.eu/

2. **The W3C web accessibility guidelines.**

https://www.w3.org/WAI/fundamentals/accessibility-usability-inclusion/