Smart Rekognition

Members:

Gan Jaye Jyn (12) 2A1 (Leader)
Zhang Jinrui (33) 2A1
Lau Jie En (9) 2A3
Yap Linxun (32) 2A3

Introduction

Problem Statement

There are people with eyesight impairments who are having difficulty identifying daily objects. They need to rely on people around them to help them identify objects. This is not only inconvenient, sometimes can also pose safety concerns.

Rationale of Project

We want to provide a simple, easy-to-carry solution to help these people. We hope that this solution will reduce their reliance on others, and reduce safety risks.

We want to be able to create an app to recognize objects. Also, we hope to be more proficient in Python Programming, and we want to learn to develop android apps.

Significance of Project

Our solution showcases how we can leverage on the latest technologies, such as Artificial Intelligence (AI), to create easy-to-use solutions that benefit people. It should also be a platform on which we can introduce new features for various purposes.

Scope

We want to limit the scope of our solution to be a smartphone app, which makes use of the phone camera to recognise objects. Language support would be limited to English. We should leverage on the off-the-shelf pre-trained models, rather than developing our own machine learning models.

Literature Review / Case Study / Theoretical Framework / Reference Models

Based on our study, people with eyesight impairment tend to seek medical treatment (such as surgery), buy customized vision aids (such as different lenses for reading and various other tasks) and seek assistance from other people, either physically or online.

There are a few practices that exist today, such as laser vision surgery[1], which reduces or eliminates the need for eyeglasses or contact lenses. A growing cohort of companies, like eSight, offer high-tech visual tools that can enhance vision by making things bigger, brighter and bolder so that they're able to be visible to people with eyesight impairments. Be My Eyes is a free app that connects blind and low-vision people with sighted volunteers and company representatives for visual assistance through a live video call.

As we had the original idea of using Artificial Intelligence to solve our problem, we had also investigated the advancement of AI technologies. We came to realize that there are machine learning and artificial intelligence models to help analyze images, or even videos, and make predictions. Some of these machine learning models are made available via Amazon Web Service[6] and PyTorch[7], and that should allow us to easily develop applications to make use of these models.

The Study & Methodology

Validation of Ideas (Survey)

We did a survey, and we had 34 responses.

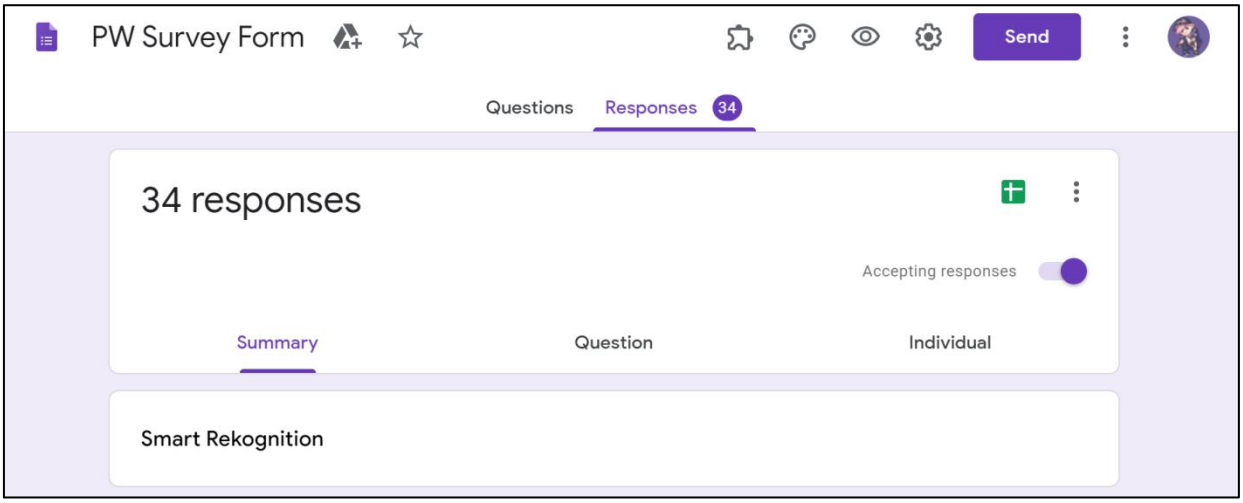

Figure 1: Survey Page

We asked what kind of eyesight impairments the people have, and what kind of device they use. We also asked their opinions on our project, and these are the results:

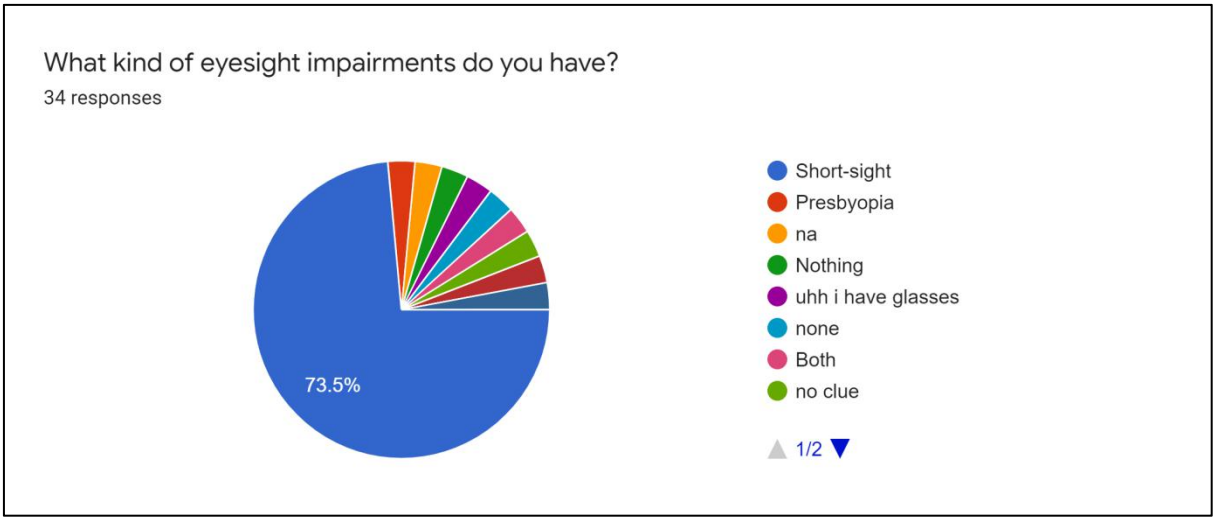

Figure 2: 1st question results

It seems that most people who did our survey have short-sightedness, though there are two of them with presbyopia and one with astigmatism. Additionally, some of them do not have eyesight impairments.

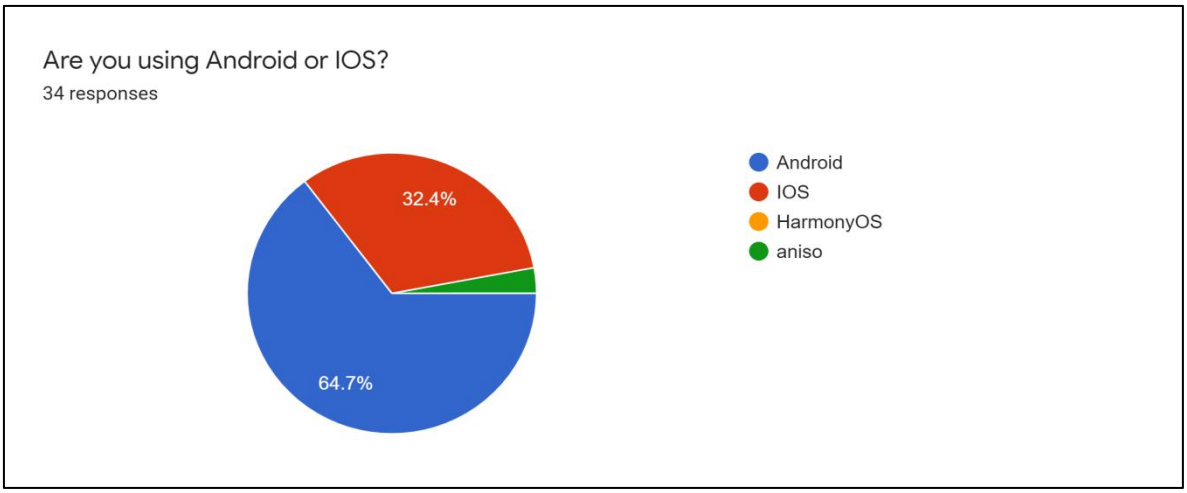

Figure 3: 2nd question results

It is also observed that most people who did our survey use Android, and some of them use iOS.

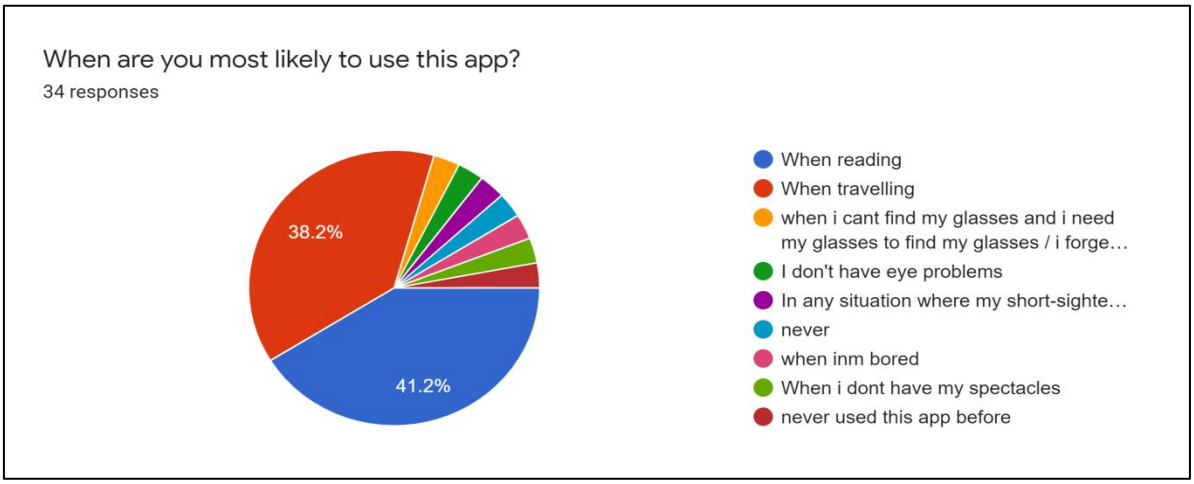

Figure 4: 3rd question results

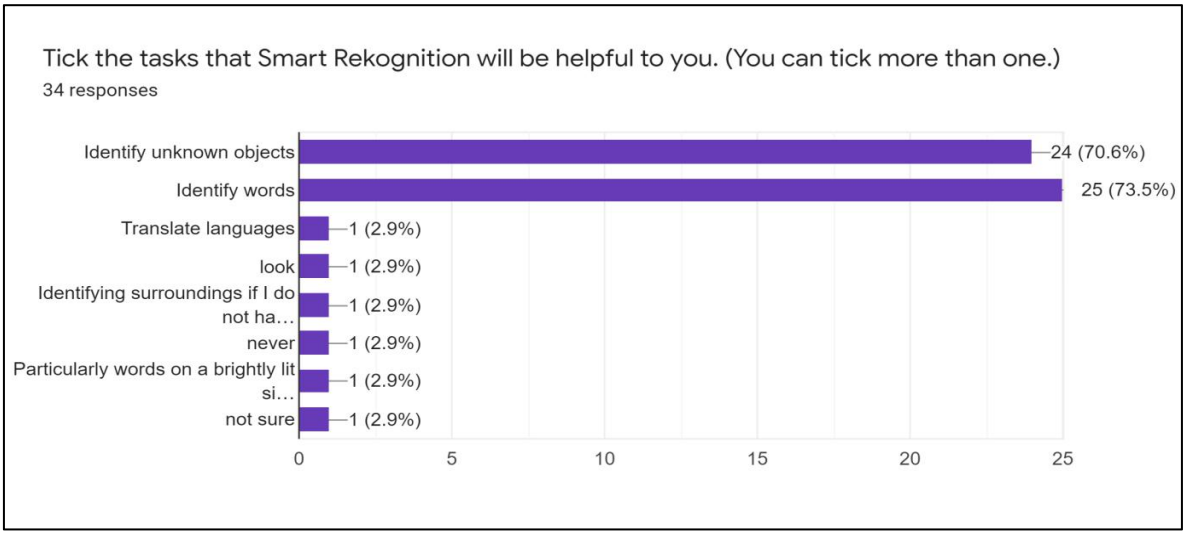

Figure 5: 4th question results

From the survey, we concluded that our solution should be:

- An Android app to recognize texts when they are reading
- An app to recognize unknown objects
- Convenient and easy to use

Feasibility Study:

We identified 2 core technologies that could be used to build our solution, namely Amazon Web Services' Rekognition Service and Kivy. We did a feasibility study on these technologies.
Amazon Web Services' Rekognition Service comes with a pre-trained model that is maintained by Amazon. In this feasibility study, we used Boto3 Library in our Python codes. By following some online tutorials such as the one by PyLenin[10], we managed to demonstrate that we are able to recognize some objects in our test images.
For the development of the Android application itself, we chose Kivy. Kivy is a free and open source Python library for developing mobile apps and other multitouch application software with a native user interface. With help from some online tutorials like the Kivy tutorial series made by Tech With Tim on YouTube[11], we managed to prove Kivy's feasibility.

Project Plan:

Members role and responsibilities:

	Jaye Jyn	Jinrui	Linxun	Jie En
AWS Rekognition with Python	✓		✓	
App coding with Kivy	✓	✓		✓
App Testing	✓	✓	✓	✓

Table 1: Members' Roles

Project timeline:

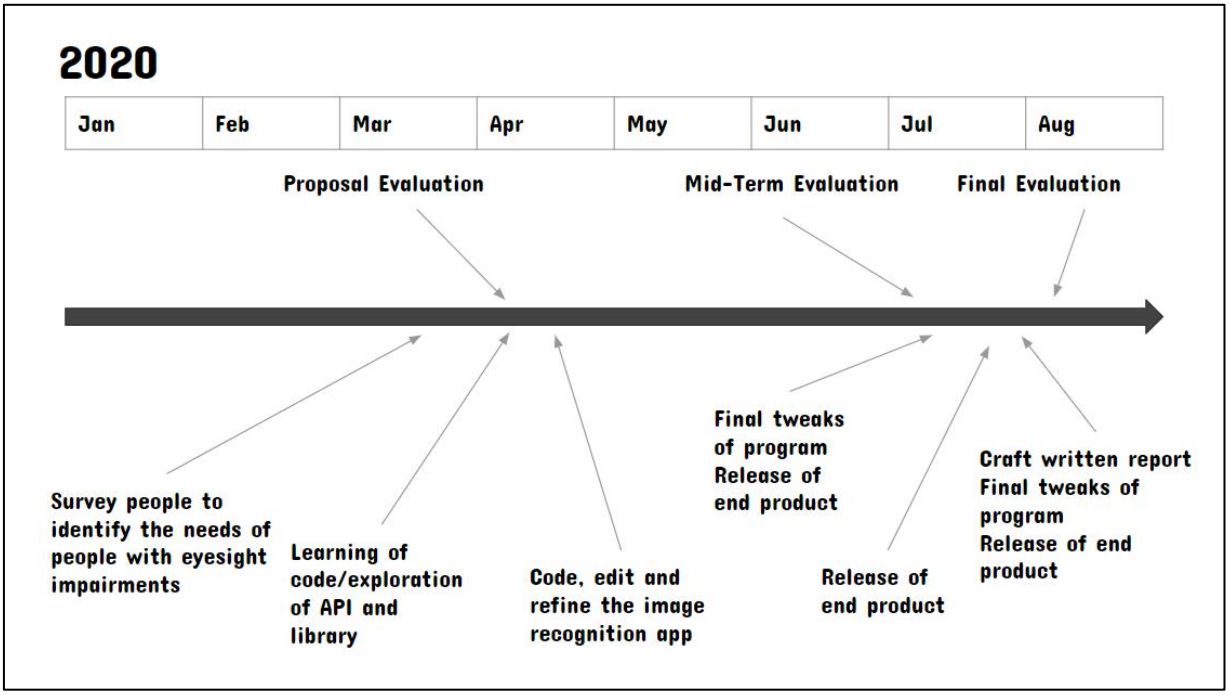

Figure 6: Project Timeline

Outcomes, Analysis & Discussions

Flowchart of the system

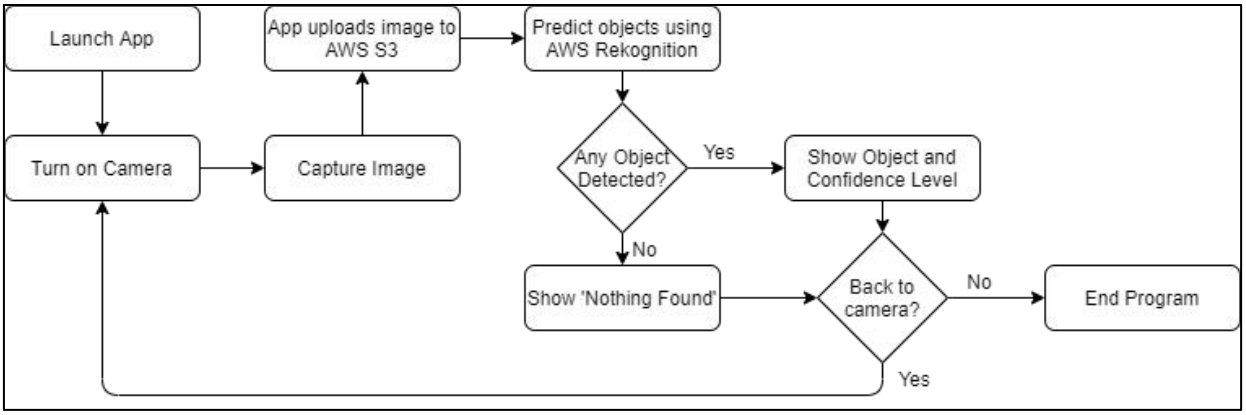

Figure 7: Flowchart of Smart Rekognition Application

Features of the system

Feature: Launching App and Turning On Camera

Figure 8: Smart Rekognition Home Icon

Figure 9: Camera turned on by the App

When clicking on the icon on the smartphone's home screen, the application will launch, and it will turn on the camera and a 'Capture' button.

Feature: Capturing Image

```
def captureImage(self):
    print("captureImage")
    btn = self.ids['btn']
    camera = self.ids['camera']

    btn.disabled = True
    camera.play = False

    print("Capturing image")
    timestr = time.strftime("%Y%m%d_%H%M%S")
    current_dir = os.path.abspath(os.path.dirname(__file__))
    print("Current dir=" + current_dir)
    photo = "IMG_{}.png".format(timestr)
    camera.export_to_png(current_dir + "/" + photo)
    image_files = os.listdir(current_dir)
    for file in image_files:
        print(file)

    # photo = image_files[-1]
    # self.show_image(photo)

    # camera.export_to_png("IMG_{}.png".format(timestr))
    print("Captured photo " + photo)

    global captured_image
    captured_image = photo

    self.rekognize_image(current_dir, photo)

    btn.disabled = False
    camera.play = True

    sm.current = 'image'
```

Figure 10: Code for Capturing Image

This code is basically the code for the 'Capture' button. When we press the button, the application captures the image of the phone camera.

Feature: Upload of image to AWS S3 and Predicting Objects using AWS Rekognition

```
def rekognize_image(self, directory, photo):

    # with open(photo, 'rb') as source_image:
    #    source_bytes = source_image.read()
    response0 = s3Client.upload_file(directory + "/" + photo, 'smart-rekognition-project', photo)

    #To be removed
    #photo = 'Paramore3.jpg'

    response1 = client.detect_labels(
        Image={
            'S3Object': {
                'Bucket': 'smart-rekognition-project',
                'Name': photo
            }
        },
        MaxLabels=2,
        MinConfidence=90)
    response6 = client.detect_text(
            Image={
                'S3Object': {
                    'Bucket': 'smart-rekognition-project',
                    'Name': photo,
                }
            }
        )
```

Figure 11: Code for Uploading Image to S3 Bucket and Predicting Object

This is the code for uploading the image to S3 Bucket and predicting the object. The code uploads the captured image to an Amazon S3 Bucket. Then, Boto3 is called to predict the objects and show the responses.

Feature: Showing Objects and Confidence Level

```
print(response1, "\n--------------------------------------------------")
global resp
for key, value in response1.items():
    if key == 'Labels':
        if len(value) == 0:
            resp = 'Nothing found\n'
        else:
            objs = ''
            for obj in value:
                objs += obj['Name']
                objs += ' (' + str(obj['Confidence']) + ')\n'
                # print(obj.Confidence)
            resp = objs

print(response6, "\n--------------------------------------------------")
for key, value in response6.items():
    if key == 'TextDetections':
        if len(value) == 0:
            resp = resp + 'No Text Detected\n'
        else:
            textresp = ''
            for texts in value:
                textresp += texts['DetectedText']
                textresp += ' (' + str(texts['Confidence']) + ')\n'
            resp = resp + textresp
```

Figure 12: Code for showing objects and confidence level

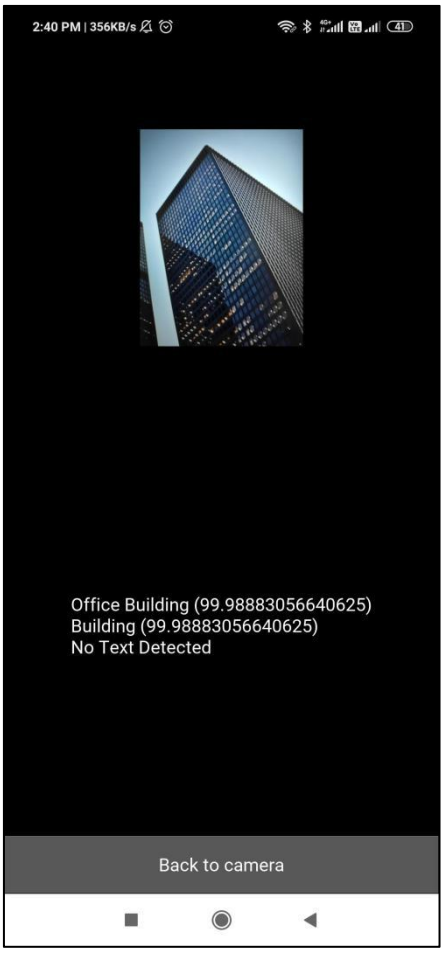

Figure 13: Showing Objects and Confidence Level

This app extracts the objects and their confidence levels and displays it on screen.

Exporting to APK File

The next step was to code the Smart Rekognition application itself. We combined our knowledge on AWS Rekognition and Kivy to write the code above. It took numerous test runs and debugging, and the result is a functional Smart Rekognition application.

```
jayejyn@LAPTOP-RIUGF5SK: ~/JJKivyApp
jayejyn@LAPTOP-RIUGF5SK:~$ cd JJKivyApp/
jayejyn@LAPTOP-RIUGF5SK:~/JJKivyApp$ ls
'[HS]'   bin   buildozer.spec   credentials.csv   main.py   main.py.bak   main.py2.bakk   main.py3.bak
jayejyn@LAPTOP-RIUGF5SK:~/JJKivyApp$
```

Figure 14: Ubuntu Windows Subsystem

Of course, after finishing the application, we have to export it to an APK file, so it is compatible for Android phones. Therefore, we used Buildozer on Ubuntu Windows Subsystem to do so. We moved the APK file to our phones and installed it afterwards.

Architecture

In our architecture, we have a client (Smart Rekognition App that runs on Android smartphones) and server (S3 and Rekognition services running on AWS) components. The client app can be installed onto multiple devices, all of which connect and use the same AWS services. As AWS is scalable, it can serve thousands of users at the same time.

Smart Rekognition app makes use of the camera on the smartphone to capture the images. It uses the Boto3 client, which is an AWS SDK running in Python, to connect to the AWS Services. On the Cloud, there is a pre-designated S3 bucket, which is used to store the images that are to be analyzed. The Rekognition Service is responsible for recognizing objects in the images. It contains a Machine Learning model that is capable of recognizing thousands of different objects.

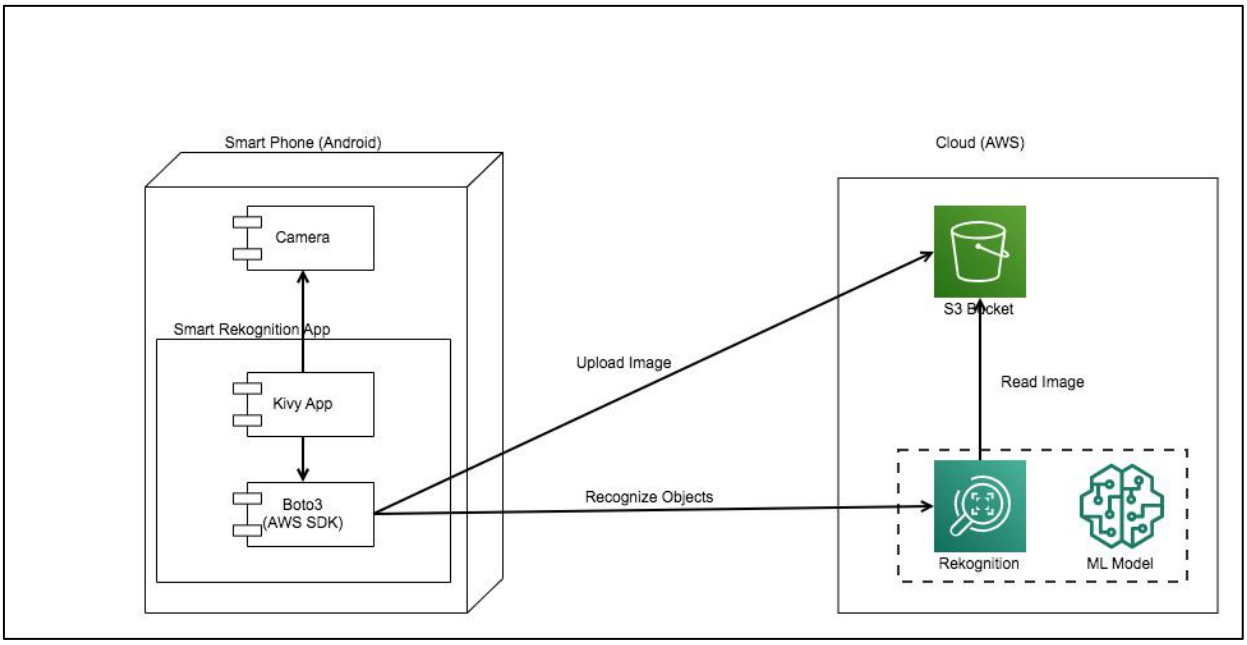

Figure 15: Architecture Diagram

Testing of Smart Rekognition

App Launches	✓
Connects to Phone Camera	✓
Able to Take Pictures	✓

Sends the Pictures to S3 Bucket	✓
Predicts Objects Using AWS Rekognition	✓
Displays Correct Results	✓
Able to Run Another Recognition	✓

Table 2: Checklist of functions

We made a checklist of the app functions. We verified that we have successfully delivered all the functions in the checklist.

	Able to Recognize	Not Able to Recognize
People	✓	
Car	✓	
Buildings	✓	
Book	✓	
Standing Fan	✓	

Table 3: Checklist of Object Recognition

We tested the app on objects such as people, cars, buildings, books and standing fans, and the app is able to recognize these objects accurately.

Implications and Recommendations

Opportunities for Future Improvements

Due to the restrictions in schedule and resources, we have to limit the scope of this project. However, we have also identified several opportunities for future improvements as follows:

- As this is an Android only app, in future it can be developed for iOS too, so more people can get access to this app.
- This app only recognizes images, and it could be improved to be able to recognize objects in live video from the phone's camera.
- After recognizing the objects, it displays the output on the screen. To make it more convenient for users, there could be a voice output, so that users just need to listen, and do not need to look at the screen to identify objects.
- This app only supports English. It could be programmed to support multiple languages in the future, so it can appeal to non English speaking audiences.
- There are some things that this app cannot recognize. It would take some time for the AWS Rekognition API to train more models so that this app could recognize more things.
- We could decorate the User Interface of this app to make everything more appealing to the user.

Conclusion

Jaye Jyn's Point of View:

After working on this project, I realized how far technology has come because of how easy it was working on this project. Initially, I thought we had to write the code and train the models all by ourselves. But, thankfully we soon discovered this API called Amazon Web Services Rekognition API. The people at Amazon already previously worked on training the models, and this allowed us to develop this Android app that can recognize many daily objects more easily. Throughout this project, it was an enjoyable experience working with Jinrui, Linxun and Jie En. Our progress was smooth, and often we got to hang out and know each other better, that was until COVID-19 hit us, then we had to do everything online. It was definitely harder, mainly because of all the technical difficulties we faced during our meetings. But in the end, we still managed to develop this app so we are happy about it.
Overall, working on this project is a fun experience, and I got to learn more about programming compared to my peers in Infocomm class. But for subsequent years I would still like to try out different categories instead of continuing to do Infocomm.

Jie En's Point of View:

After working on this project, I realized how hard it was for the blind to go on with their daily life. Therefore, it is important that we create such a device to help the blind go through their daily lives. In addition, it was quite a new and eventful experience for me this year. Due to the Covid-19 outbreak, we were not able to meet together after school. However, this does not affect us as we are still able to still communicate with each other due to the wonders of technology. Still, it was a different experience compared to discussing face to face as there were a lot of technical difficulties, contributing to a huge amount of time being wasted. In conclusion, working on this project is an enjoyable experience as I got to learn more knowledge on python.

Jinrui's Point of View:

This project has been a great learning experience for me as I discovered how apps work. This project was meant to be a lot more eventful but due to covid-19, some things did not go as planned. The worst setback that we had was the fact that we could not meet up and code together, so we tried using google meet but it was not very efficient. This resulted in Jaye Jyn doing most of the coding, which really just meant less fun for us. I would have liked it if we could have done the coding together. One thing that we would like to improve on our app is that we would like to add a function where the users could enter in their own "recognisable objects",as the AI we used is unable to recognise absolutely everything. Another problem is that since it is an app, people

are still required to see the text that says what the object was, meaning that completely blind people would not be able to use the app very well. The overall aesthetics and looks were also kind of plain, so if truly needed we could have also improved on that. All in all I believe I enjoyed doing this project, and I just hope that the next one will not be disrupted by some world-wide virus.

Linxun's Point of View:

This project has given me a great learning experience. I learned how hard it is to build up an app from scratch and more about python through this project. There are many things about python that we were not taught in class. Although due to the Covid-19 and the new timetable, we cannot meet up frequently and code together, it is still a fun experience. Besides, we are able to discuss our progress using Whatsapp. However, it is still a different feeling compared to face to face meeting as there may be technology difficulties causing waste of time. But, we managed to finish this project in the end. Overall, I feel that it is fun working on this project as I got to learn more about python and app development. I just hope that next year, our project will not be disrupted.

Bibliography

1. Mandal, Dr. Ananya. “Treatment of Visual Impairment.” *News*, 5 June 2019, www.news-medical.net/health/Treatment-of-visual-impairment.aspx.
2. Team, All About Vision Editorial. “Helping Someone With Low Vision.” *All About Vision*, All About Vision, 11 July 2020, www.allaboutvision.com/lowvision/helping.htm.
3. *Be My Eyes - See the World Together*, www.bemyeyes.com/.
4. *Content - Health Encyclopedia - University of Rochester Medical Center*, www.urmc.rochester.edu/encyclopedia/content.aspx?ContentTypeID=1.
5. Andrew Zaleski, special to CNBC.com. “Amazing Electronic Glasses Help the Legally Blind See, but They Are Costly.” *CNBC*, CNBC, 20 Aug. 2018, www.cnbc.com/2017/09/20/these-amazing-electronic-glasses-help-the-legally-blind-see.html.
6. *https://aws.amazon.com/rekognition/?blog-cards.sort-by=item.additionalFields.createdDate&blog-cards.sort-order=desc*
7. *PyTorch*, pytorch.org/?utm_source=Google.
8. Hunt, Marsha. “Free.” *Amazon*, Plume, 1994, aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank.
9. *Kivy (Framework)*. 6 June 2020, en.wikipedia.org/wiki/Kivy_(framework).
10. PyLenin. “Image Recognition with AWS and Python | Tutorial 1 | The Setup | AWS Rekognition | Boto3” YouTube video, 4:52. February 24, 2019. https://www.youtube.com/watch?v=Jtr0gyM9rCI
11. Tech With Tim. “Kivy Tutorial #1 - How to Create Mobile Apps With Python” YouTube video, 11:05. February 3, 2019. https://www.youtube.com/watch?v=bMHK6NDVlCM