ANIME CHARACTER FACE DETECTION AND RECOGNITION

BY

NG HUI CHIN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology
(Kampar Campus)

JAN 2022

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: ANIME CHARACTER FACE DETECTION AND RECOGNITION

Academic Session: JAN 2022

I NG HUI CHIN
(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

[illegible]	[illegible]
(Author's signature)	(Supervisor's signature)

Address:
No.3, Jalan Mahsuri Impian 12,
Taman Mahsuri Impian,
31900 Kampar, Perak.

Ng Hui Fuang
Supervisor's name

Date: 20/04/2022

Date: 22/04/2022

Universiti Tunku Abdul Rahman			
Form Title : **Sample of Submission Sheet for FYP/Dissertation/Thesis**			
Form Number: **FM-IAD-004**	Rev No.: **0**	Effective Date: **21 JUNE 2011**	Page No.: **1 of 1**

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 20/04/2022

SUBMISSION OF FINAL YEAR PROJECT

It is hereby certified that ***NG HUI CHIN*** (ID No: ***18ACB02518***) has completed this final year project entitled "ANIME CHARACTER FACE DETECTION AND RECOGNITION" under the supervision of Dr. Ng Hui Fuang (Supervisor) from the Department of Computer Science, Faculty of Information and Communication.

I understand that University will upload softcopy of my final year project in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

[illegible]

(NG HUI CHIN)

*Delete whichever not applicable

DECLARATION OF ORIGINALITY

I declare that this report entitled "**ANIME CHARACTER FACE DETECTION AND RECOGNITION** " is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature	:	[illegible]
Name	:	NG HUI CHIN
Date	:	20/04/2022

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere thanks and appreciation to my supervisor, Dr. Ng Hui Fuang who has given me various advice and constructive suggestions. Thank him for passing this project that I proposed, so that this idea has a chance to be realised. This is the first time that I have carried out such a huge personal project. There are still shortcomings in many places. I cherish and appreciate this opportunity. Benefited a lot from you. Thank you.

Next, I want to thank my sister, who patiently listen to my complain and helped me along the way. Thank you.

Last but not least, I would like to thank my parents for not scolding me for staying up all night to do projects so that the project can be completed in time. Thank you all.

ABSTRACT

Artificial Intelligence (AI) applications have grabbed headlines with their potential to radically improve human life and even transform the industry. Currently, in the field of pursuing detection and recognition of objects, human face recognition is a well-deserved representative and has reached the highest level. Nevertheless, the use of AI in the field of art is still very few. The only famous ones nowadays are the appraisal of paintings and recognition of painting style. Let alone facial recognition in the field of art, such as the domain of this project, anime character face. There is still no mature technology or well-known application for recognising anime faces. Even the Google Lens, which claims to be able to search for all objects in the world, cannot identify an anime character with half accuracy. Thus, there is an acute need for such a system to detect and recognise anime character faces.

Therefore, in this project, an AI mobile app named "Gease" with the main functionality of detecting and recognising multiple anime character's faces are developed. The mobile app will be the first mobile app specifically for anime character face detection and recognition. Likewise, the recognisable number of characters is up to 205 which are more than doubled compared to most of the existing anime face detection and recognition projects which only involve at most 100 characters. The accuracy achieved 97% and 63.2% Top-1 accuracy for detection and recognition respectively. While the inference time is 31.9ms and 4.8ms per face detection and recognition respectively. Moreover, the FastAPI web framework that supports Python development was utilized in this project. FastAPI allows the app developed to be opened to external connections as a REST API and return the detection and recognition results in JSON form which greatly flexes and expands the value of the app.

Furthermore, this project developed an exquisite and smooth user interface while having multi-platform adaptability. The mobile is a portable PWA which flourishing recently. It accommodated multiple platforms including web browser, Android and iOS with only one set of codes but maintains most of the functions of native apps. The Ionic framework was used in design a cross-platform mobile user interface.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

2D	two-dimensional
3D	three-dimensional
ACGN	Anime, Comic, Game and Novel
AI	Artificial Intelligence
AR	Augmented Reality
ASGI	Asynchronous Server Gateway Interface
CASE	Computer Aided Software Engineering
CORS	Cross-Origin Resource Sharing
CNN	Convolutional Neural Network
CNNs	Convolutional Neural Networks
Faster R-CNN	Faster Region Based Convolutional Neural Networks
HOG	Histogram of Gradients
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
PaaS	platform as a service
PC	Personal Computer
PWA	Progressive Web App
QR	Quick Response
RAD	Rapid Application Development
ROI	Region of Interest Pooling
RPN	Region Proposal Network
SDLC	System Development Life Cycle
SVM	Support Vector Machin
UI	User Interface
WSGI	Web Server Gateway Interface

CHAPTER 1: INTRODUCTION

1.1 Problem Statement and Motivation

1.1.1 Problem Statement

Artificial Intelligence (AI) applications have grabbed headlines with their potential to radically improve human life and even transform the industry. These include the automation of data input, the automation of pipeline engineering, and the assistance of data analysis. AI has unknowingly haunted every corner of human life, aided the shortcomings of human abilities, and assisted humans in achieving their endless goals. Currently, in the field of pursuing detection and recognition of objects, human face recognition is a well-deserved representative and has reached the highest level. It has even become a unique authentic method like a fingerprint. It can not only scan the face to unlock the phone but also scan the face to log in, make payments and else. Even there is also smart video surveillance that uses AI facial recognition to capture wanted criminals hiding among the crowds on the street. Nevertheless, the use of AI in the field of art is still very few. The only famous ones nowadays are the appraisal of paintings and recognition of painting style. Let alone facial recognition in the field of art, such as the domain of this project, anime character face.

Anime characters, as the name suggests, are characters in a certain anime. Figure 1.1.1 shows an example of anime character.

Figure 1.1.1: Chika Fujiwara from *Kaguya-sama: Love Is War*.

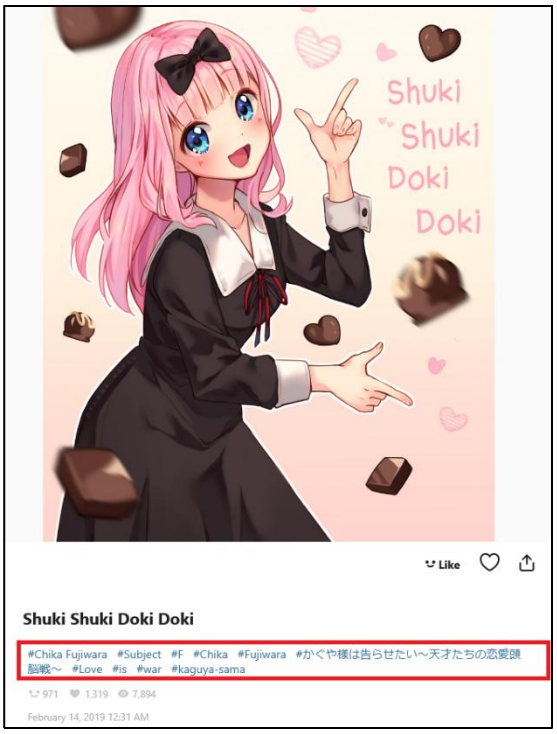
As we all know, animation is a branch of the art field, and animation characters are the artistic expression of real humans. Anime characters usually simplify or exaggerate the characteristics of real humans and the handling of light and shadow in the real world. They are two-dimensional (2D) characters that are different from three-dimensional (3D) humans. Although the similarity is lower than the sketch, overall the anime characters are very similar to human beings. However, there is still no mature technology or well-known application for recognising anime faces. This is virtually an unreasonable thing. Even the Google Lens, which claims to be able to search for all objects in the world, cannot identify an anime character with half accuracy. Thus, there is an acute need for such a system to detect and recognise anime character faces.

1.1.2 Motivation

The inability to identify anime characters has plagued the following groups: video sites, illustration storing sites, anime productions, anime enthusiasts as well as laymen.

In the age when the recommendation algorithms of major video sites are so mature, it should be incredible that the tagging system of video websites still relies on manual work. The sites will categorize the videos through tags marked by the uploader himself. Certainly, the site mentioned here is not a website similar to Netflix or iQIYI which uploads videos from a central server, but a platform like YouTube or Dailymotion which allows users to upload videos freely. Generally, this kind of website relies heavily on the tags of a video. When we take a look at the home page of these websites, we can merely see a variety of video category that is conducted by tags, such as games, mysteries, stocks, animes and so on. It is convenient for users to search for a video and website to carry out related pushes. For example, if the video uploader does not do tags on his video, the website cannot accurately classify it, and can only vaguely classify it according to the title and description. It is facing serious problems in the anime category especially. MAD, a kind of multimedia works in the anime culture, game culture, and fan culture that are composed and integrated of existing anime clips, sounds, games, images, etc. by personal editing and synthesis. It is so obvious that below a MAD, "Who is the character who appears in 3:05?", this kind of comment is almost flooded. There are very few people who give them answers, and the people who answer right are even rarer. The MAD was very well, and it succeeded in getting laymen interested in the anime or character. However, they never get the real name of the character. This is a sad thing, and the anxiety of users has nowhere to be resolved.

Another problem with classification is illustration storing sites, such as Pixiv, DeviantArt, Imgur and so on. These websites that store massive amounts of illustrations urgently need an effective classification system. When the primary classification, which is equivalent to cross-species classification has reached its peak, what remains are the classification above the second level. For example, anime is a primary classification, specific anime is a second classification, and specific anime character is a third classification. For example, Pixiv, an illustration storing and sharing site that specializes in the ACGN, an abbreviation of "Anime, Comic, Game and Novel", the classification above level-2 is extremely important. Pixiv faced with the same problem as YouTube, it mainly relies on the tag system for classification. So when the uploader does not manually tag it, the website cannot classify it accurately. Therefore, when users search for pictures in a certain category, they may miss some untagged quality works. Figure 1.1.2 shows the artwork tags (in red rectangle) manually added by Pixiv uploader, Hyanna-Natsu [1] to help the website categorise. The title of the artwork is "Shuki Shuki Doki Doki", which is totally unrelated to Chika Fujiwara. If without the manual tagging, it cannot be classified accurately.



Shuki Shuki Doki Doki

#Chika Fujiwara #Subject #F #Chika #Fujiwara #かぐや様は告らせたい～天才たちの恋愛頭脳戦～ #Love #is #war #kaguya-sama

971 1,319 7,894

February 14, 2019 12:31 AM

Figure 1.1.2: Manual tags of Pixiv.

These two kinds of websites are imminent for accurate classification, and they even introduced various sets of tagging systems. However, these tagging systems will only increase the workload of the uploader and reduce the user experience. This is an incorrect orbit. Therefore, anime character face detection and recognition is necessary to improve the development as soon as possible.

Apart from that, anime character face detection and recognition is like a boon to anime productions. It may be hard to believe that in an era when copyright protection laws are so advanced, piracy is still rampant on the Internet that no one has jurisdiction, especially cloud storage. Uploading, sharing, just some simple steps then complete the entire grey industry chain. The interests of anime productions need to be safeguarded, and these piracy needs to be punished. Like the popular methods of monitoring pornography in China, the animation industry must also have an anime character face detection and recognition system for the appraisal of videos that shared on clouds. Since manual identification may violate the rights of privacy, AI identification can well avoid this shortcoming.

Furthermore, the most beneficial party of anime character face detection and recognition application is anime enthusiasts. Yes, anime enthusiasts also need it very much, and can even be said to be the most needed group. This is because anime enthusiasts are not encyclopaedia after all although they have watched a lot of animes. They do not all know every character; they may not have seen it, or they may have a deviation in their memory and forget the character. For example, Pixiv stores and shares a large number of fan arts, but due to laziness of the uploaders or other reasons, the illustrations often uploaded without indicating who the characters are drawn, and fans can only glance at it without knowing who the character is. Therefore, anime character face detection and recognition are the same as online shopping image search engines, it can easily help anime enthusiasts to identify an ambiguous character.

On the other hand, anime character face detection and recognition applications also benefit the public. Honestly, many anime characters have similar faces, and a layman usually cannot recognise those characters. For them, anime character face detection and recognition are like a dictionary equipped with a scanner, which able to detect and recognise every character easily.

Moreover, here are some prospects about the future application of anime character face detection and recognition. In fact, anime character face detection and recognition have great potential in the advertising field, although it seems irrelevant. This technology has the potential to replace the Quick Response (QR) code in the advertising field in the future. Imagine that on a colourful and beautiful advertising poster, the existence of a black and white ugly QR code will simply ruin the entire poster. But if one day, the QR code in the advertisement is replaced by cute cartoon characters, a web page can be opened by just recognising the characters with a mobile phone, which will be a boon for the advertising industry. Actually, the current advertising industry is also trying its best to combine Augmented Reality (AR) technology to create more ornamental and attractive advertising effects. If this technology is combined in the future, it will be the icing on the cake, without any disadvantage.

1.2 Objectives

The main objective of this project is to develop a mobile app with the main functionality of **detecting the faces of multiple anime characters** and **recognising** them.

While the sub-objectives are

- To implement anime character face detection and classification using deep learning models with a high Top-1 accuracy and less than 1 second response time.
- To develop a mobile app as the frontend to make engagement with the user such as capture image and show the recognition information.
- To develop a Python backend to detect and recognise the image, and then return the results.

1.3 Project Scope

The deliverable of this project is an AI mobile application (also called a mobile app) with the main functionality of detecting the faces of multiple anime characters and recognising them. The name of the mobile app is “Gease”. Sub functionalities may include the history tracker, app preferences settings and result feedback.

Why mobile app? As we all know, the mobile app is portable and easy to control, and it is not difficult for laymen to get started. In addition, there is also a camera on the phone as an auxiliary, which can scan pictures for recognition from time to time. Moreover, there is no such mobile app released before. Therefore, this project will develop the first mobile app for anime character face detection and recognition. The reasons are mainly as following:

- Mobile app dominate nowadays with a very large market share, and currently there is no such mobile app, thus it is an opportunity.
- The future development of the mobile app has great potential, especially in the field of advertising.

- Mobile app has the potential to replace the immature Google Lens to detect and recognise anime character, therefore seize the market.
- Mobile app is suitable to be a small-scale project that can be developed within a few months as the project engineering time is limited.
- There is a visual deliverable, which is easy to evaluate.

There are three kinds of mobile app, which are native app, hybrid app and Progressive Web App (PWA). A PWA was developed in this project instead. Hybrid app and PWA allowed to use only one set of codes but can still accommodate multiplatform, including web browser, Android and iOS. PWA flourishes in recent years, and major companies have gradually abandoned costly native apps and shifted their positions to PWA. For example, Starbucks, Filpboard and 2048 Game. The User Interface (UI) of the Starbucks PWA [2] is shown in figure below.

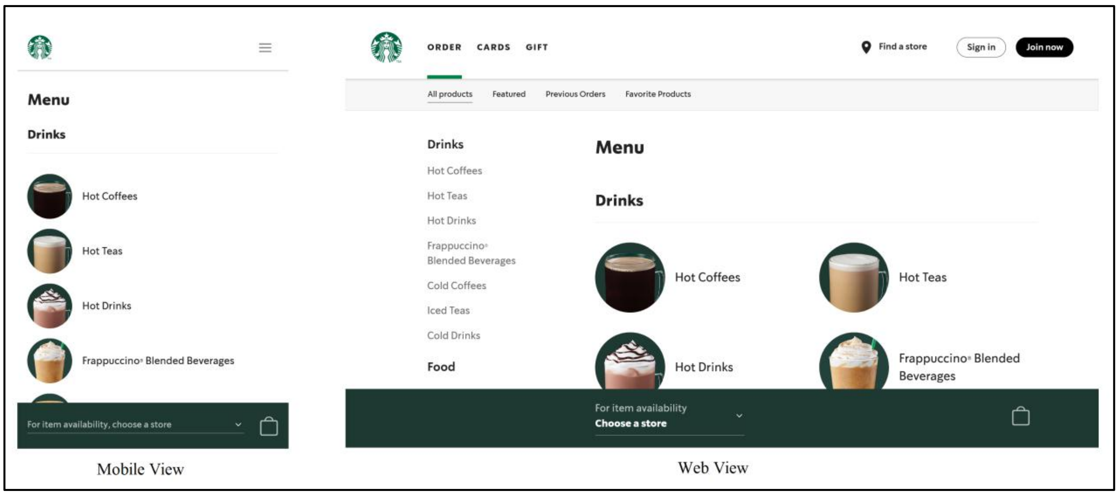


Figure 1.3.1: PWA of Starbucks.

Since there is only one set of codes, PWA is more convenient and cheaper to maintain. Hybrid apps are just like native apps, which are able to access device-specific features and hardware, such as GPS, contacts list and camera. While PWAs have some restrictions on them. It cannot access the contacts list and some other features. However, since the mobile app in this project only use the camera, so neither hybrid apps nor native apps were needed. PWA is enough.

Meanwhile, Additionally, the project has developed a dual-purpose backend that can also act as a RESTful API in addition to the backend for the mobile app. The backend can open to external connections as an API and return the detection and recognition results in JSON form which greatly flexes and expands the value of the app.

Apart from that, in terms of main functionalities realisation, deep learning was utilised. The detection and classification models were trained using the deep learning algorithm —— Convolutional Neural Network (CNN). CNN is a class of neural networks that can simulate how the visual cortex breaks down and analyses visual imagery. It has achieved unprecedented success in the field of computer vision, and it can be said to be very suitable for image-based object detection and classification. Some CNN architectures are planned to try out in model training to find the most suitable model that fulfils both accuracy and speed requirements.

Moreover, from the perspective of user experience, the mobile app was able to detect multiple anime character faces and achieve high accuracy while being fast response. The recognition range involved around 205 characters.

1.4 Impact, significance and contribution

This project developed the first mobile app specifically for anime character face detection and recognition. There is no such app released in the market before. Although Google Lens dominates the current image-based object searching market, it is only designed for the general field. And the anime image search results usually come from reverse image search. Besides, the mobile app has the capability to detect multiple anime faces simultaneously, which is different from most of the existing apps that only can detect one face. The mobile app achieved an accuracy of 97% for face detection and 63.2% for recognition while being able to recognise 205 characters, and accuracy of 87% with 10 characters during experimental, beating most existing apps. Furthermore, the mobile app is a PWA instead of native apps, which means the app can be install and used on any device. There is no platform restriction like native apps. PWA is highly recommended and progressed by Google, and even Google Play store plans to create an area dedicated to PWA to provide high-quality PWA downloads. It is believed that it will not take many years for native apps on the market to be converted into cheap and portable PWAs.

By having this mobile app, anime enthusiasts even laymen able to easily identify anime characters by just capturing an image using a smartphone. They can recognise an anime character anytime at anywhere. For example, when watching MAD on YouTube and browsing fan art on Pixiv. User can just open this mobile app and take a picture to recognise it. The emergence of this application can provide convenience for users of anime-related video and image sites, including the uploaders, who no longer need to force themselves to tag the character involved, and the users no longer need spam the comment area with the questions which destroying the experience of other users.

1.5 Report Organization

This report consisted of six chapters, each chapter was indicated in Table 1.1.

Table 1.1: Report organization.

	Chapter	Description
1	Introduction	This chapter introduced the motivations of this project, including the problem statement, motivation, project scope and objectives, as well as the contributions.
2	Literature Review	This chapter researched and reviewed other people's works or methods that related to anime face detection and recognition.
3	System Methodology/Approach	This chapter described in detail how the project is developed, including the top-down system design diagrams and its implementation details. The methodology, tools, timeline, and implementation issues are also mentioned.
4	System Design	This chapter described in detail how the system is designed. The block diagram includes flowcharts and an architecture diagram. This chapter also discussed in detail the design of the system from the frontend and backend perspectives.
5	System Implementation and Evaluations	This chapter discussed in detail the implementation of the system. From project setup to app deployment.

6	Conclusion and Recommendation	This chapter concluded the project with a summary and personal perspective. Some recommendation for further improvement and the prospect of anime character face detection and recognition was proposed.

CHAPTER 2: LITERATURE REVIEW

2.1 Google Lens by Google LLC [3]

2.1.1 Brief Introduction

Google Lens is an AI image recognition mobile app designed to display relevant information about the objects it recognised through visual analysis based on neural networks. Google Lens has multiple features, such as translate text in real-time, recognise handwritten text, visual search, etc. Its tagline is "Search what you see". Just scan or take a photo, it can recognise the information on the picture and return the result.

Regardless of the text-related features, the focus of discussion here will be on its visual search engine. Different from common image search engines, including Google Image Search, Google Lens does not use the previous king of image search — reverse image search engine — but the leader of modern image recognition technology — neural networks, a machine-learning algorithm [4]. Google did not disclose the detailed technology, but according to Sagar [5], Google Lens uses computer vision, machine learning and Google's Knowledge Graph to realise image recognition. He also point out that CNN are the backbone of many computer vision applications.

Google Lens claims to be able to identify everything in the world, including flora and fauna, fashion items, famous people, etc. So, it stands to reason that anime can also be searched, including the characters. But disappointingly, the recognition results are not ideal. Figure 2.1.1 shows the recognition result of Ayame Yomogawa, the third main character from *Kabaneri of the Iron Fortress*. The recognition result is not related to her at all.

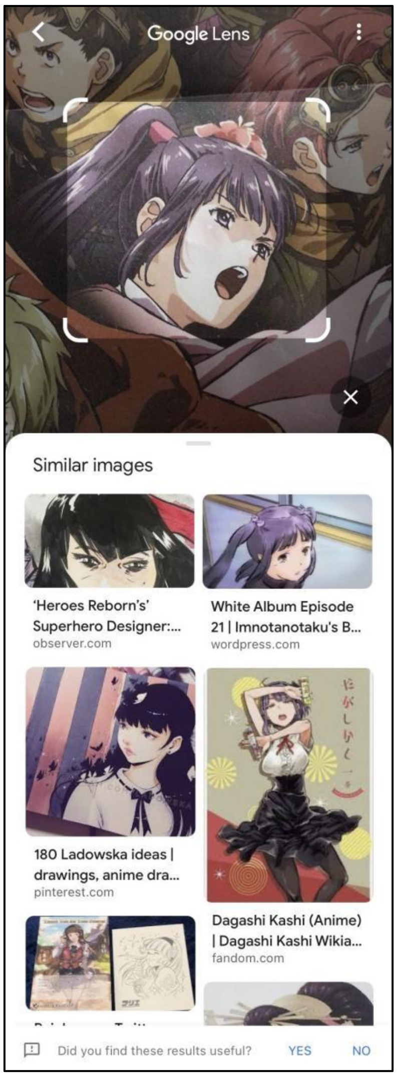

Google Lens
Similar images
'Heroes Reborn's' Superhero Designer:...
observer.com
White Album Episode 21 | Imnotanotaku's B...
wordpress.com
180 Ladowska ideas | drawings, anime dra...
pinterest.com
Dagashi Kashi (Anime) | Dagashi Kashi Wikia...
fandom.com
Did you find these results useful?
YES
NO

Figure 2.1.1: Google Lens's recognition result of Ayame Yomogawa from *Kabaneri of the Iron Fortress*.

2.1.2 Strengths

- A mobile phone application.

  Users can turn on the camera of their mobile phone to scan and recognise objects anytime, anywhere. And will return the result in a specific language according to the device language.

- Provide multiple download versions for different operating systems.

Google Lens can be used on Android and iOS. However, the Android version can be direct downloaded from Google Play Store, while the iOS version is bundled with the Google application as a sub-feature.

- Fast response.

  The image recognition result can be returned in a few seconds. Far surpasses its direct competitors, such as Baidu.

- The flexibility of detection.

  In addition to automatic face detection, it also allows the user to manually trimmed the range.

- Provide relevant web pages in search results.

  Google Lens can be said equivalent to a Google Search with a lens installed, which using pictures instead of text to search. Thus the returned results are almost the same as Google Search, providing links to related web pages, similar images, etc.

2.1.3 Weaknesses and Limitations

- Unsatisfactory performance of anime character recognition.

  Google Lens may be specialized in the general field, so the recognition performance in the field of anime, especially recognition of a specific character is very unsatisfactory. In addition to Figure 2.1.1 above, Figure 2.1.2 is also a poor result. Figure 2.1.2 is Google Lens's recognition result of Subaru Natsuki, the main character of *Re:Zero − Starting Life in Another World*. This anime is very popular and should not be unrecognizable.

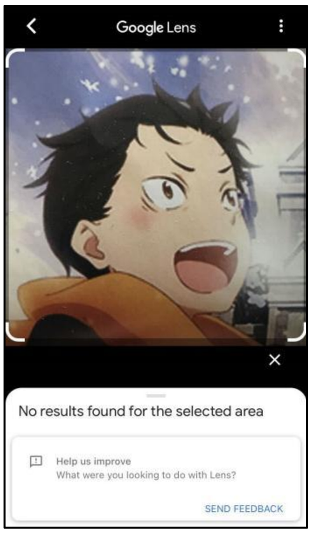

Google Lens
No results found for the selected area
Help us improve
What were you looking to do with Lens?
SEND FEEDBACK

Figure 2.1.2: Google Lens's recognition result of Subaru Natsuki from of *Re:Zero − Starting Life in Another World*.

- Image recognition is affected by the brightness.

  Figure 2.1.3 shows Google Lens's recognition result of Lucy Heartfilia from *Fairy Tail* under normal brightness.

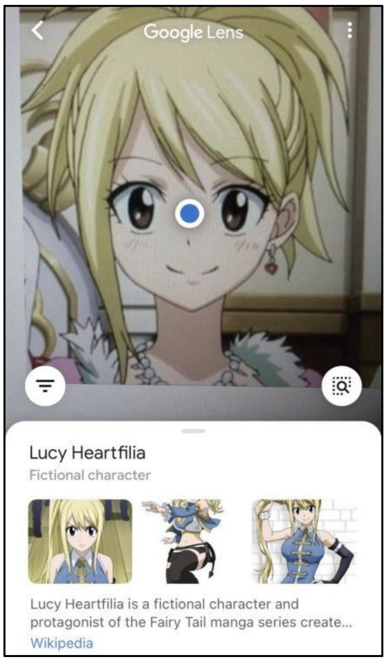

Google Lens
Lucy Heartfilia
Fictional character
Lucy Heartfilia is a fictional character and protagonist of the Fairy Tail manga series create...
Wikipedia

Figure 2.1.3: Google Lens's recognition result of Lucy Heartfilia from *Fairy Tail* under normal brightness.

Figure 2.1.4 shows Google Lens's recognition result of Lucy Heartfilia from *Fairy Tail* under low brightness.

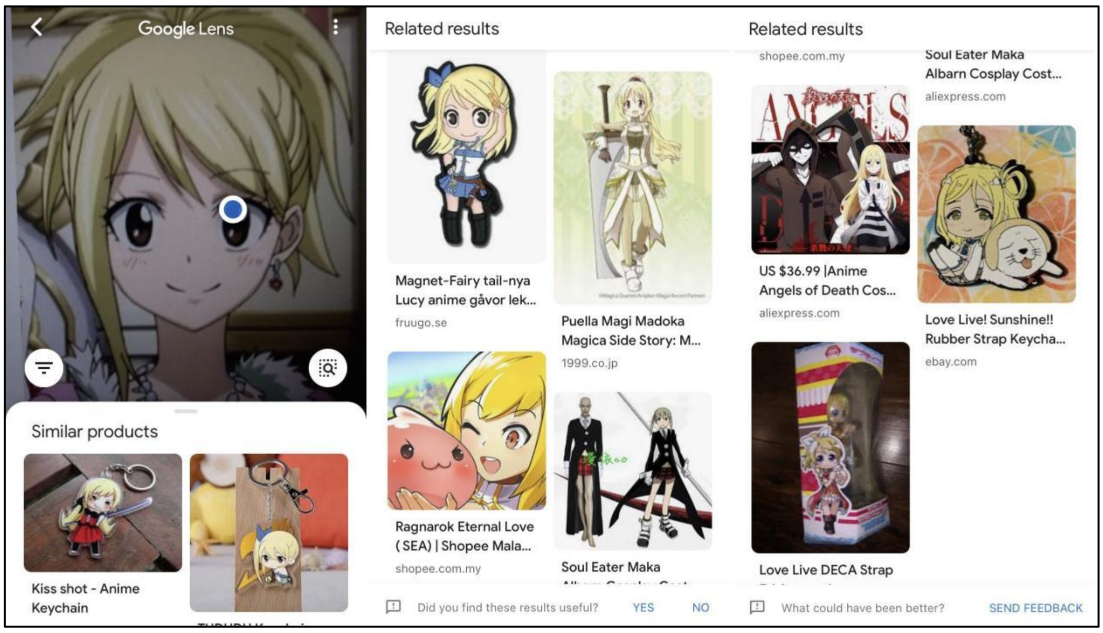

Google Lens
Similar products
Kiss shot - Anime Keychain
Related results
Magnet-Fairy tail-nya Lucy anime gåvor lek...
fruugo.se
Puella Magi Madoka Magica Side Story: M...
1999.co.jp
Ragnarok Eternal Love (SEA) | Shopee Mala...
shopee.com.my
Soul Eater Maka
Did you find these results useful? YES NO
Related results
shopee.com.my
Soul Eater Maka Albarn Cosplay Cost...
aliexpress.com
US $36.99 |Anime Angels of Death Cos...
aliexpress.com
Love Live! Sunshine!! Rubber Strap Keycha...
ebay.com
Love Live DECA Strap
What could have been better? SEND FEEDBACK

Figure 2.1.4: Google Lens's recognition result of Lucy Heartfilia from *Fairy Tail* under low brightness.

- Can only recognise anime but not specific characters.

Figure 2.1.5 shows Google Lens's recognition result of Chika Fujiwara from *Kaguya-sama: Love Is War*. *Kaguya-sama: Love Is War* is the anime that always stays on the popular rankings of many websites. The wrong recognition result is definitely not because the anime is very niche or unpopular. And Chika Fujiwara is the third protagonist of the anime, not an inconspicuous supporting role.

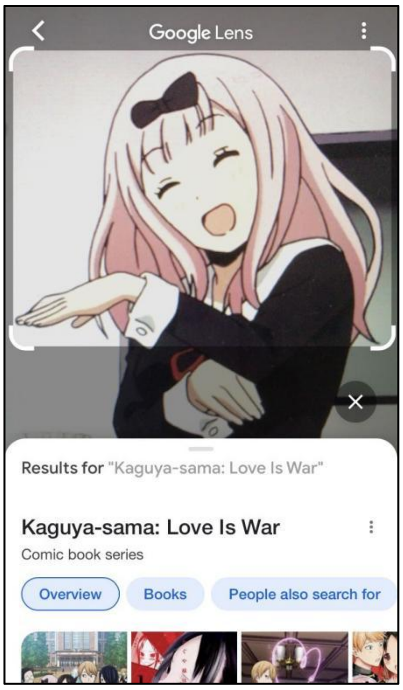


Figure 2.1.5: Google Lens's recognition result of Chika Fujiwara from *Kaguya-sama: Love Is War*.

2.1.4 Recommendation

- Strengthen the recognition in the field of anime. Train the recognition model with more popular anime characters.
- Train the model with pictures of different brightness and sharpness.
- Setup a top-down classification system to recognise characters after recognising anime.

2.2 Abyss by Malusama [6]

2.2.1 Brief Introduction

Abyss is an anime character recognition website created by Malusama on a whim. The website is very simple, there are only three navigations, and the anime character recognition is the only feature. By uploading a picture to the website, the website will return the recognition result. Figure 2.2.1 shows the home page of Abyss.

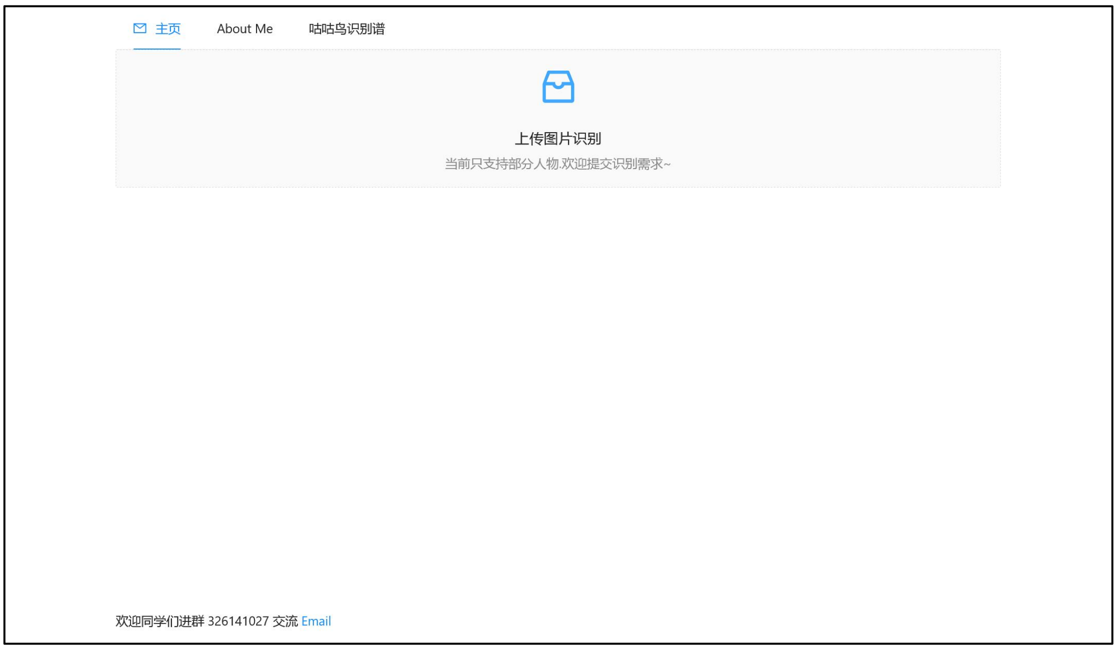

主页
About Me
咕咕鸟识别谱
上传图片识别
当前只支持部分人物,欢迎提交识别需求~
欢迎同学们进群 326141027 交流 Email

Figure 2.2.1: Home page of Abyss.

Malusama said that the origin of Abyss was because he could see a lot of fan arts created by artists every day on Twitter, but he occasionally could not recognise the character in the painting since there was no label or tag. As a result, he started this project. Therefore, his dataset mostly comprises fan arts from Pixiv.

The training model used in Abyss is mobilenet_v2 of Tensorflow.keras， and the accuracy for the validation set reaches 85.236%. However, when it started to be used online, the results were not satisfactory.

2.2.2 Strengths

- Chrome extension

  After installing the plug-in, users can right-click any picture in the browser and select the option to recognise it. After clicking, the web page will direct to Abyss's website and display the recognition result. It solves the inconvenience that users need to upload image personally to the website when recognising.

- Use MobileNets to train the model.

  MobileNets is a small and efficient CNN model developed by Google. It has a compromise between accuracy and latency. It is suitable for mobile and embedded vision applications. It solves the insufficient memory problem faced by other models that are too large. It reduces the model size and improves the calculation speed while maintaining the model performance - accuracy.

2.2.3 Weaknesses and Limitations

- Very few anime characters can be recognized.

  Malusama personally created and labelled the data set from scratch. Each picture in the data set is a high-quality fanart, which is carefully selected from Pixiv, one by one, one keyword by one keyword. After screening the images, he then manually label them. Due to the heavy workload of data collection, so the dataset is very small. The website currently only supports about 100 anime characters.

- Low recognition accuracy.

  To effectively test the recognition accuracy of Abyss, the character was selected from the list of supported anime characters provided by Malusama. Figure 2.2.2 shows the input, Mio Akiyama from *K-ON!*, and Figure 2.2.3 shows the recognition result. Mio Akiyama was misrecognised as Azusa Nakano, which also a character come from the same anime.

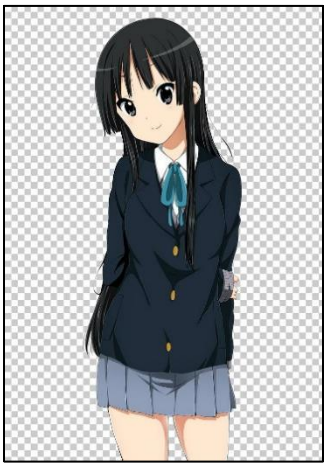
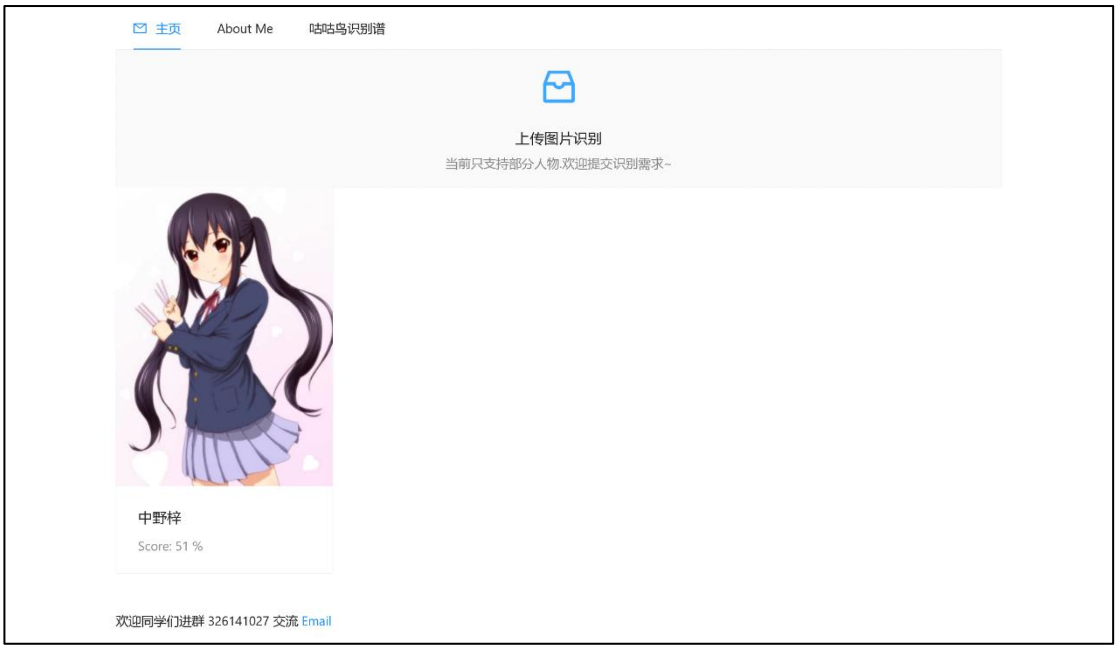


Figure 2.2.2: Mio Akiyama from *K-ON!*.



中野梓
Score: 51 %

欢迎同学们进群 326141027 交流 Email

Figure 2.2.3: Abyss's recognition result of Mio Akiyama.

- No face detection.

  This tool does not conduct face detection to bound a range for recognition but directly uses the entire picture, which becomes the main reasons for the low accuracy.

- Image recognition only by uploading a picture from the device or obtaining a picture URL.

The website does not provide the function of uploading pictures by camera, which leads to cumbersome steps in recognising objects in reality such as poster. Users need to take a photo and store it in the device before uploading it to be recognised.

2.2.4 Recommendations

- Expand the dataset and increase the number of recognisable characters. Increase the training set of each character to improve its recognition accuracy.
- Train a face detection model or use other open-source face detection tools to eliminate noise. Besides, the user should be allowed to trim the picture to enhance flexibility.
- Develop a mobile app or add a real-time camera function to improve user experience and solve tedious recognition steps.

2.3 anime face recognition by Fu, et al. [7]

2.3.1 Brief Introduction

This is a python web app for anime characters classification and recognition. The web app is simple. Just uploads a picture to the web app, the web app will return the recognition result. Figure 2.3.1 shows the home page of the web app.

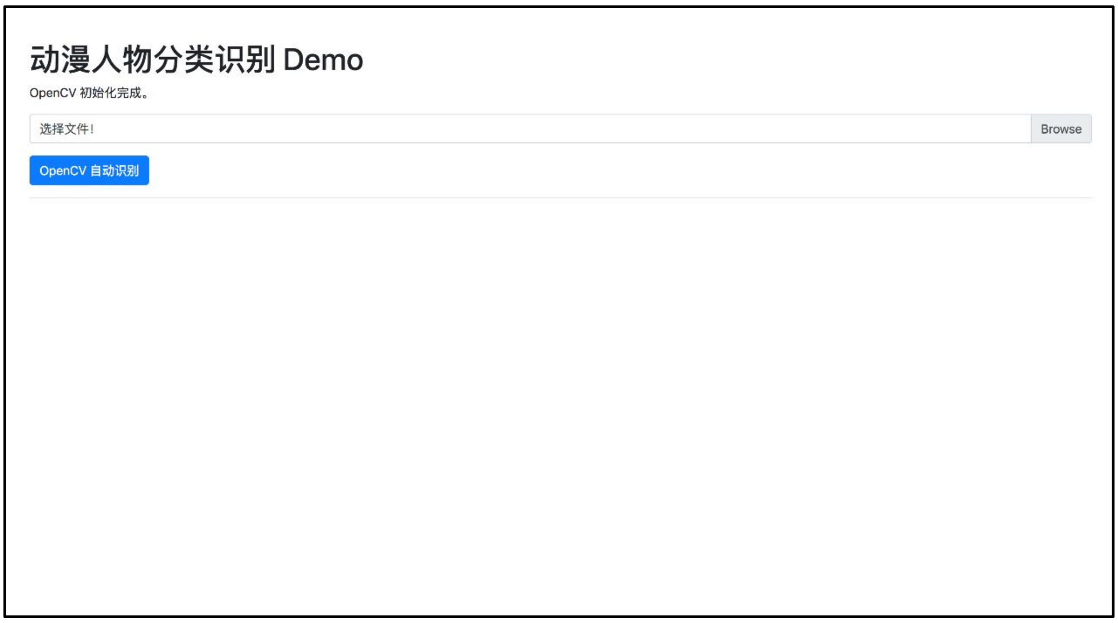


Figure 2.3.1: Home page of web app developed by Fu, et al..

This project uses Tensorflow's Inception-v3 model in training and adopted transfer learning. Transfer learning is an approach in deep learning, which enables the training model of the previous task to be used as the starting point of the next training task. Inception-v3 was proposed by Google and is mainly used for large-scale visual recognition tasks on ImageNet. Inception-v3 is a CNN architecture of the Inception series with many improvements. It can use fewer parameters and a 42-layer deep learning network to achieve similar complexity to VGGNet [8]. Transfer learning allows retraining the final layer of an existing model. Thus, this project borrowed the model as a feature extractor and only trained the final fully connected layer.

Fu, et al also found that data augmentation, including flip, cut, zoom, rotate, affine transformation, all white or all black fill, Gaussian blur, mean blur, median blur, sharpening, effect embossing, contrast changing and Pixels moving for dataset expanding was not suitable for anime characters. This is mainly because the face shape of anime characters is very important, so distortion may cause bad results.

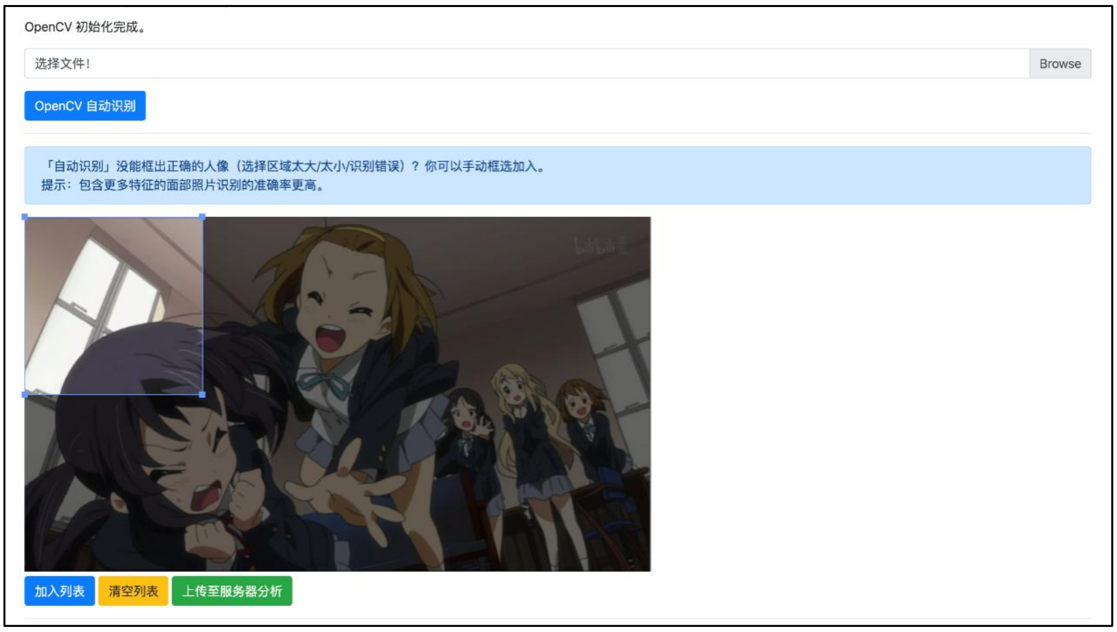
2.3.2 Strengths

- Use transfer learning.

  Transfer Learning allows the final layer to be retrained, which means that the knowledge learned by the model during its original training can be retained and applied to a smaller dataset so that a high degree of accuracy can be obtained without a lot of training and computing power.

- Detection task is done at front-end.

  The web app detects anime face at the front end. So when a picture has multiple faces, it can allow the user to manually crop. Figure 2.3.2 shows the cropping function of the webpage.


OpenCV 初始化完成。

选择文件!

「自动识别」没能框出正确的人像（选择区域太大/太小/识别错误）？你可以手动框选加入。
提示：包含更多特征的面部照片识别的准确率更高。

Figure 2.3.2: Cropping function of the web app developed by Fu, et al.

- No detection model developed.

  Developers did not do the training for face detection but used another open-source tool, which is the OpenCV based anime or manga character face detector that released by nagadomi, the author of the famous anime-style picture super-resolution reconstruction program "waifu2x", in 2011.

- Feedback system

  If the recognition result is wrong, the user can choose to fill in the correct answer into the web app for feedback. After the web app has collected a good amount of feedback, a new round of training can be carried out to improve accuracy.

2.3.3 Weaknesses and Limitations

- There are only seven recognisable anime characters.

  Fu, et al. create the dataset personally from scratch. They use the open-source multimedia processing program ffmpeg to extract frames from the original anime video and then perform data processing such as AI face detection and manual labelling. Due to the heavy workload of data collection, the training dataset is terribly small, with only seven main characters from the same or different anime.

- The image can only be recognised by uploading from the device or obtaining a URL.

  The web app does not have a photo-taking function, which leads to cumbersome steps in recognising objects in reality such as poster. Users need to take a photo and store it on the device before uploading it to be recognised.

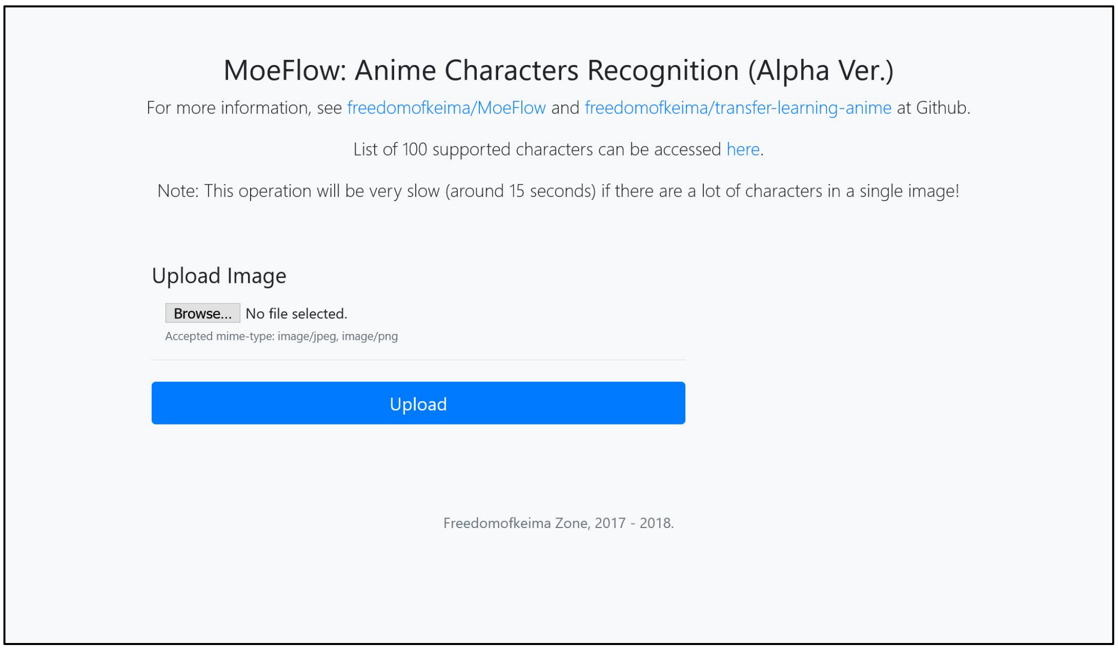
2.3.4 Recommendations

- Expand the dataset and increase the number of recognisable characters. Increase the training set of each character to improve its recognition accuracy.
- Develop a mobile app or add a real-time camera function to improve user experience and solve tedious recognition steps.

2.4 MoeFlow by Setiadi [9]

2.4.1 Brief Introduction

MoeFlow is the Anime Characters Recognition website developed by Setiadi in 2017. The web page is very simple. Users upload pictures from the device for recognition. Figure 2.4.1 shows the main page of MoeFlow.



MoeFlow: Anime Characters Recognition (Alpha Ver.)

For more information, see freedomofkeima/MoeFlow and freedomofkeima/transfer-learning-anime at Github.

List of 100 supported characters can be accessed here.

Note: This operation will be very slow (around 15 seconds) if there are a lot of characters in a single image!

Upload Image

Browse... No file selected.

Accepted mime-type: image/jpeg, image/png

Upload

Freedomofkeima Zone, 2017 - 2018.

Figure 2.4.1: Home page of MoeFlow.

According to the article "Image Recognition for Anime Characters" written by Setiadi [10], he sees this project as a challenge. First of all, although the human face detection and recognition technology today has been well-developed, there is still a huge difference between the 2D face of anime and the 3D face of human beings. OpenCV's face recognition is completely unusable in the 2D art field. Therefore, he used the OpenCV based anime or manga character face detector released by nagadomi to detect the face of anime characters. Besides, he used transfer learning in this challenge as some of the anime characters have very little data since they are new or unpopular. And he chooses Tensorflow's Inception-v3 as the training model.

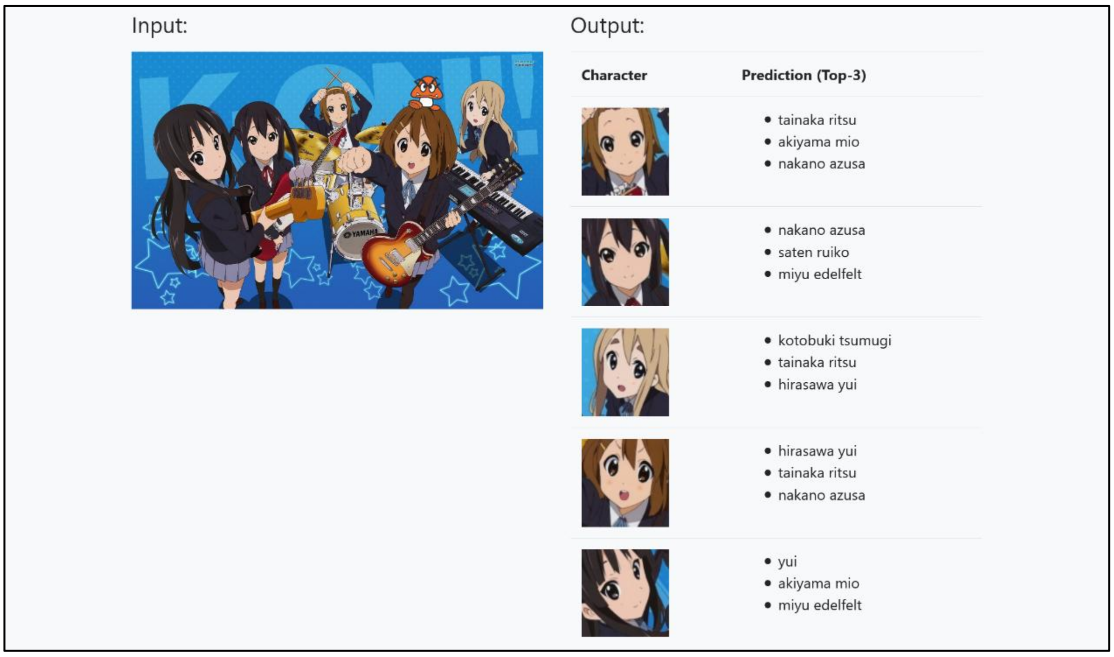
By reading his article and report, it can be inferred that the web app discussed in "2.3 anime face recognition by Fu, et al." is referred by his findings. Except for the final presentation of the website, the training model selection and method for detection are the same. According to Setiadi's [11] report "Transfer Learning for

Anime Characters" on GitHub, he first used three similar-looking characters to train the model and got 83% accuracy on the test set. Later, nagadomi's detector was used, and the accuracy was successfully increased to 90%.

2.4.2 Strengths

- Supports multiple face recognition, and attains a high accuracy rate.

  Figure 2.4.2 shows the recognition result of a role group photo from *K-ON!*. Except for the fifth character, Akiyama Mio was recognised as the second probability, every face in the picture was successfully detected and recognised.



Character	Prediction (Top-3)
	• tainaka ritsu • akiyama mio • nakano azusa
	• nakano azusa • saten ruiko • miyu edelfelt
	• kotobuki tsumugi • tainaka ritsu • hirasawa yui
	• hirasawa yui • tainaka ritsu • nakano azusa
	• yui • akiyama mio • miyu edelfelt

Figure 2.4.2: MoeFlow's recognition result of a role group photo from *K-ON!*.

2.4.3 Weaknesses and Limitations

- Only 100 supported characters.
- The image can only be recognised by uploading from the device or obtaining a URL.

The web app does not have a photo-taking function, which leads to cumbersome steps in recognising pictures in reality. Users need to take a photo and store it on the device before uploading it to be recognised.

- The recognition process is not instant.

  After uploading the picture, it takes 15 seconds for the recognition process. If there are multiple faces in the picture, it will take longer.

2.4.4 Recommendations

- Expand the dataset and increase the number of recognisable characters.
- Develop a mobile app or add a real-time camera function to improve user experience and solve cumbersome recognition steps.
- Use a more responsive model for data training.

2.5 Anime Face Detection and Recognition by Wang and Yan [12]

2.5.1 Brief Introduction

This is a python program for anime face detection and recognition developed by Wang and Yan in 2019. They train the detection model using the Support Vector Machine (SVM) and Histogram of Gradients (HOG) of the images. SVM is a binary classifier. While HOG is a feature descriptor used to extract features from image data, thus often used in computer vision and image processing. The detect region on the image will be highlighted by drawing red rectangles. On the other hand, they use the powerful VGG16, which is a kind of neural networks classification method, to train the recognition model with labelled pictures.

2.5.2 Strengths

- Train both detection and recognition model.

2.5.3 Weaknesses and Limitations

- Datasets are of poor quality.

  Most of the data sets come from the Internet, which are public resources published on the Internet, so they are unevenness. Some pictures are too low in pixels, and some pictures are fan work instead of the original design. Figures 2.5.1, 2.5.2 and 2.5.3 are Mio Akiyamas, the sample images retrieved from the dataset.

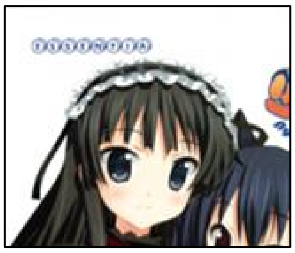


Figure 2.5.1: Mio Akiyama (1), a sample image retrieved from Wang and Yan’s dataset.

Figure 2.5.2: Mio Akiyama (2), a sample image retrieved from Wang and Yan's dataset.

Figure 2.5.3: Mio Akiyama (3), a sample image retrieved from Wang and Yan's dataset.

- Low accuracy.

  SVM is not powerful enough, it is easy to cause underfitting and affect the accuracy rate.

- Recognition speed is slow.

  The size of HOG descriptor is large and numeric, and VGG is too complex. These factors make the calculation slow.

2.5.4 Recommendations

- Use other models for training, such as CNN. MobileNets is recommended to replace SVM as it can increase the calculation speed more obviously. Besides, it would be better to use the Vectorized Binary descriptor or Region of Interest Pooling (ROI) or Faster-RCNN instead of HOG.

2.6 Anime-Face-Detector by Dwiarto W., Lamp and Zhou [13]

2.6.1 Brief Introduction

This is a Faster Region Based Convolutional Neural Networks (Faster R-CNN) based anime face detector. Use 6000 training samples for training and 641 testing samples for testing. The dataset collected using crawlers, which will randomly select images from the top 10 Pixiv daily ranking. Moreover, for convenience, developers did not manually mark faces in the picture but used AI assistance. That is, the famous OpenCV based anime face detector released by nagadomi. The tool helps to mark out the anime faces in the picture and will highlight them with a red box as a visual presentation. The marked images then used as a training sample to train the detection model.

2.6.2 Strengths

- Use Faster R-CNN

  The Faster R-CNN is a detection pipeline that uses the *Region Proposal Network* (RPN) as a region proposal algorithm and Fast R-CNN as a detector network. Faster R-CNN can bring down the calculation time. [14]

2.6.3 Weaknesses and Limitations

- AI marked dataset.

  This project uses the OpenCV based anime face detector released by nagadomi to mark the dataset samples. Since it is machine marking, inaccurate marking may occur during the period, which will influence the training effect.

2.6.4 Recommendations

- Manually marking the dataset to ensure 100% correct marking.

2.7 Critical Remarks of previous works

Table 2.1: Summary of the strengths and limitations of the existing system and comparison to the app "Gease" developed in this project.

	2.1 Google Lens by Google LLC	**2.2 Abyss by Malusama**	**2.3 anime face recognition by Fu, et al.**	**2.4 MoeFlow by Setiadi**	**2.5 Anime Face Detection and Recognition by Wang and Yan**	**2.6 Anime-Face-Detector by Dwiarto W., Lamp and Zhou**	**Gease develop-ed in this project**
Mobile App?	Yes	No	No	No	No	No	Yes
Training model	CNNs	Mobile Nets	Transfer Learning	Transfer Learning	SVM, HOG, VGG16	Faster-RCNN	Transfer Learning
Face detection?	Yes	No	Yes	Yes	Yes	Yes	Yes
Train a face detection model?	Yes	No	No	No	Yes	Yes	Yes
Flexibility of detection	High	No	Moderate	High	No	N/A	Yes
Recognition performance	Moderate	Low	Moderate	High	Low	N/A	High
Recognisable anime characters	N/A	100	7	100	10	N/A	205

CHAPTER 3: SYSTEM METHODOLOGY/APPROACH

3.1 Methodology and General Work Procedures

In this project, phased development methodology, which a system development methodology under Rapid Application Development (RAD) was used. This is because the project has a short lead time and development period, and needs to get a useful system to users in a short time. Therefore, a structured design that requires complete and lengthy paper documentation is unsuitable. Agile development is also excluded since the developer of this project is not an expert team and does not have professional knowledge as well as execution ability. A fully completed analysis and design document is required, they should not be simplified or eliminated to avoid development midway mistakes.

The System Development Life Cycle (SDLC) of this project is as shown in Figure 3.1.1, designed according to phased development methodology.

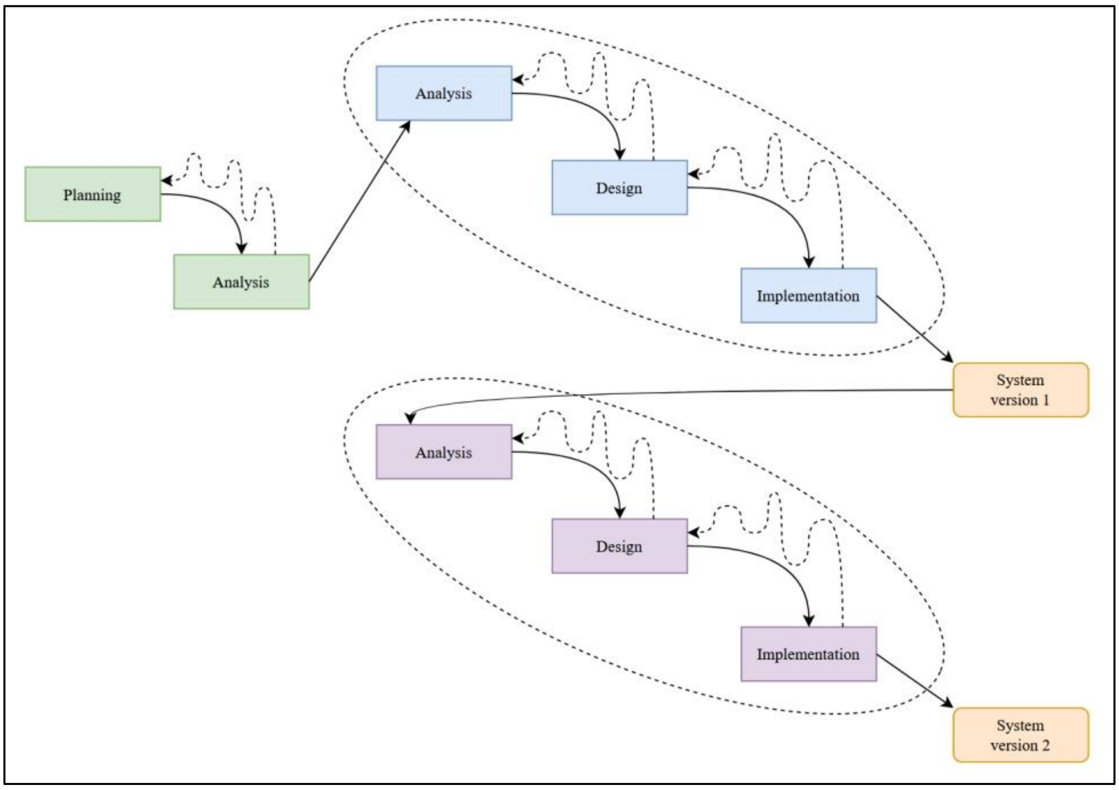


Figure 3.1.1: SDLC of the project.

The planning and analysis stages at the most beginning were completed during Jan 2021. A preliminary proposal was the final deliverable for these stages. Next, the project continued the remaining phase and inaugurate the first version of the system during May 2021 trimester. During this period, the analysis was conducted again to enhance the system requirements and the studies on the existing system. The design phase was started concurrently. A complete and concise design that fulfils the requirements should be emerged, including physical design, architectural design, interface design, database and file design, program design, etc. After that, a preliminary program with the major functions, which is detection and recognition was developed during the implementation phase. Users can begin to work with intentionally incomplete systems.

Furthermore, the second phase was carried out during Jan 2022 trimester. Additional analysis was performed based on previous requirements and combined with new ideas to enhance the system even more. The new ideas may come from the users' feedback that collected from the first phase. Then, the design and implementation phase were updated with new requirements. And eventually, a fully functional and complete system developed.

3.2 Tools to Use

a) CASE Tools

CASE stands for Computer Aided Software Engineering. CASE tools are a set of **software** application programs that used to assist the SDLC activities. The CASE tools that will be used in this project are the followings:

- Diagram tools
  - Draw.io

    It is a quite satisfactory drawing tool and is very user friendly. It can be used to draw some simple diagrams such as a flowchart.

  - Venngage

It provides a lot of Infographics templates, including Gantt chart which can help in constructing the timeline of the project.

- Prototyping Tools
  - Balsamiq

    It provides user-friendly tools to sketch wireframes for the user interface of system.

- Programming Tools
  - Visual Studio Code

    It is a free source code editor provided by Microsoft that can be used for development in any programming language such as Python, Java as well as CSS and HTML markup. With Visual Studio Code, the tasks such as developing, analysing, debugging, testing, collaborating, and deploying can be done.

  - Visual Studio 2019

    It is an Integrated Development Environment (IDE) provided by Microsoft. Its features are similar to Visual Studio Code, one of the differences is it is user friendly to work in a special environment such as Anaconda environment.

  - Anaconda

    It is a distribution of the Python and R programming languages for scientific computing such as data science and machine learning. It is the toolkit that equips developers to work with thousands of open-source packages and libraries, including Keras, PyTorch, and OpenCV. It can create an isolated Python virtual environment to keep the dependencies such as the libraries, that are required by the different projects. Besides, Jupyter Notebook can also be accessed with the anaconda provided runtime. In short, it is suitable for this project to build and train machine learning models even server site development.

- Google Colab

  Google Colab is a cloud platform that allows developers to write Python codes just like Jupyter Notebook. It provides GPU runtime with a limitation of memory. The normal runtime is also limited to 12 hours.

- Web Development Tools
  - Ionic

    It is an open-source mobile app framework for hybrid app and PWA development. It provides beautiful pre-designed UI components that are adapted to multiple platforms. It allows developers to shift their apps as a PWA with a single code base.

b) Hardwares

Table 3.1 and 3.2 shows the specifications of the computer and mobile phone that will be used in this project respectively. The computer is used throughout the project development, implementation and testing process while the mobile phone is only used for testing purposes.

Table 3.1: Device specifications for computer.

	Specifications
Model Name	Huawei MateBook 13
Operating System	Windows 10 Home
System Type	64-bit operating system, x64-based processor
Graphics Processor	Radeon™ Vega 8 Graphics
Processor	AMD Ryzen™ 5 3500U
RAM	16GB LPDDR4 2133MHz
HDD Capacity	512GB PCIe SSD

Table 3.2: Device specifications for mobile phone.

	Specifications
Model Name	Huawei P20
Operating System	Android 8.1 (Oreo), upgradable to Android 9.0 (Pie), EMUI 9.1
Chipset	Kirin 970 (10 nm)
Graphics Processor	Mali-G72 MP12
Processor	Octa-core (4x2.4 GHz Cortex-A73 & 4x1.8 GHz Cortex-A53)
RAM	4GB
Storage	128GB
Main Camera	DUAL 12 MP, f/1.8, 27mm (wide), 1/2.3", 1.55µm, PDAF, Laser AF, OIS 20 MP B/W, f/1.6, 27mm (wide), 1/2.7", PDAF
Selfie Camera	SINGLE 24 MP, f/2.0, 26mm (wide), 1/2.8", 0.9µm

3.3 User Requirements

Main Functionalities: Detection and Recognition

- ✓ As a user, I want to capture an image so that I can recognise an anime character anytime anywhere.
- ✓ As a user, I want to upload an image by selecting from the gallery so that I can recognise a character through an old image captured before.
- ✓ As a user, I want to view the recognising results in a list so that I can know who is he/she.
- ✓ As a user, I want to do a quick search with the results by clicking the links so that I can be redirected to the website.

Sub Functionalities: History tracker, Feedback and Personalise Settings

- ✓ As a user, I want to give feedback on the recognition results so that the app can know there was a wrong classification occurred.
- ✓ As a user, I want to trace my recognition history so that I can refer back to them again.
- ✓ As a user, I want to get the recognisable character list so that I know who are able to be recognized in the app.
- ✓ As a user, I want to change the app settings so that I can personalise the app preferences.

3.4 Implementation Issues and Challenges

- Hardware limitation

In this project, hardware limitations became the biggest problem in implementation. To train a CNN model, GPU can accelerate the training speed. However, since an AMD computer equipped with Radeon™ Vega 8 Graphics was used in this project, PyTorch and Tensorflow's Keras library was not able to use as both of them are based on NVIDIA's CUDA. If without the utilisation of GPU, the CNN model training will be extremely slow.

Google Colab may solve this problem as the GPU available are among NVIDIA's K80s, T4s, P4s and P100s. Regrettably, the GPU runtime provided has a limitation of usage, the runtime will be disconnected after ran out of usage. Thus, the classes was reduced to 205 to cope with this situation.

- Keras model failed to applied

To solve the problem that Tensorflow's Keras was not able to use, another backend for Keras called PlaidML, provided by Intel was preferred. PlaidML allows developers to utilize their Intel and AMD GPU the same way as Nvidia GPU. However, when applying the trained Keras model to FastAPI app development, the model was raised with errors due to the essential usage of Tensorflow's Keras in some of the utils. While PyTorch runs successfully by adding map_location='cpu' arguments when loading the model. This argument will make the model run using CPU instead of GPU. So even though PyTorch does not support AMD GPU, it can run on CPU to solve the application problem.

Therefore, a new recognition model was retrained using PyTorch and the Keras model trained in the first phase was abandoned.

- Interclass similarity

Anime characters typically do not have uniqueness, similar characters can be seen everywhere. Therefore, how to successfully identify them is very challenging. The accuracy of the model is likely to be unsatisfactory because of this.

- Inconsistency of dataset

In the selected data set, a variety of images are included. For example, images that are different in painting styles, different in angles, and different in brightness. These factors will affect the model training. Other characters similar to these abnormal images are likely to be misclassified. Moreover, the number of samples per class in the database is extremely uneven. The gap between the rich and the poor will make the classification accuracy for rare character drops.

3.5 Timeline

The project timeline is separate into two parts, which is FYP 1 and FYP 2. The total project period is 24 weeks, 12 weeks for each. All the main tasks and milestones are noted using Gantt chart in Figure 3.4.1 and 3.4.2. While Table 3.3 and 3.4 show in details of the milestone and its key deliverable.

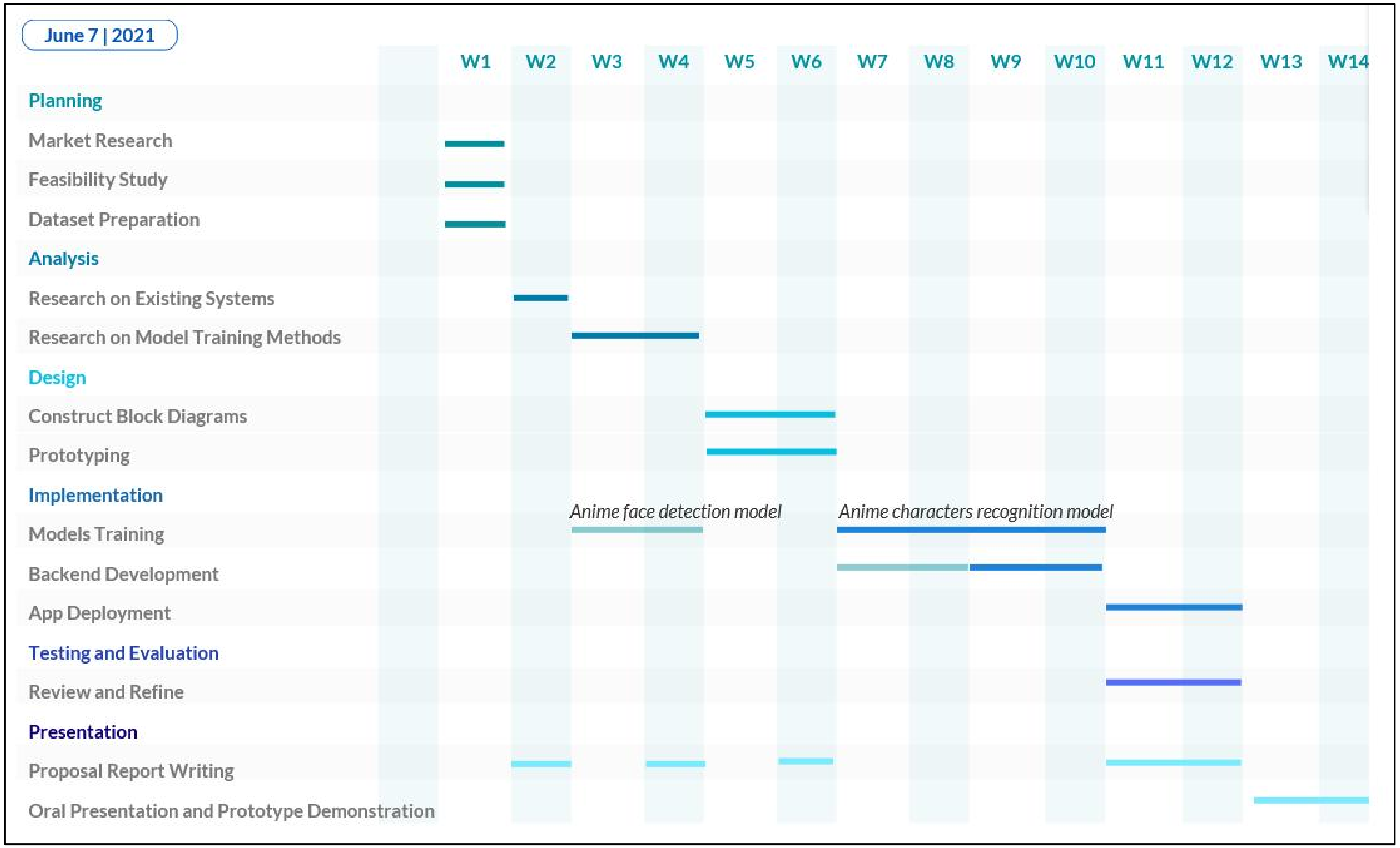


Figure 3.4.1: Gantt chart for FYP 1 period.

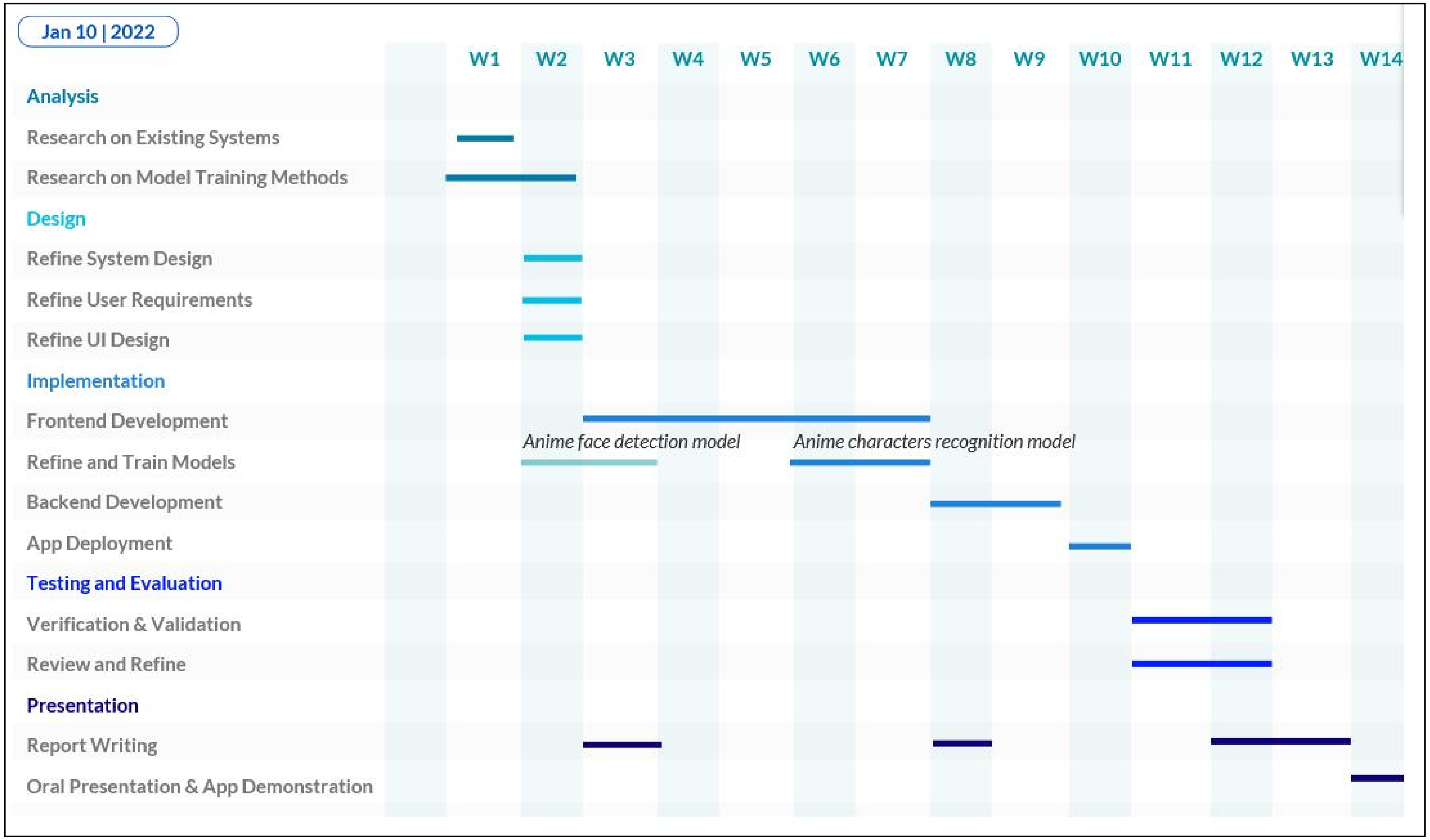


Figure 3.4.2: Gantt chart for FYP 2 period.

Table 3.3: Milestone of FYP 1 and its key deliverable.

#	MILESTONE	KEY DELIVERABLE	
1	**Week 2**		
	Complete the dataset preparation.	1)	A nice dataset.
	Complete the research on existing systems.		
2	**Week 4**		
	Complete the research on model training methods.		
3	**Week 6**		
	Complete preliminary system design.	1)	Block diagrams.
	Complete the app prototyping.	2)	Site map, wireframes and Mock-Ups
4	**Week 8**		
	Complete the model training for anime face detection.	1)	A YOLOv5 based anime face detector.
5	**Week 10**		
	Complete the model training for anime characters recognition.	1)	An anime recognition model.
6	**Week 12**		
	Complete the preliminary backend development.	1)	A developed backend.
	Complete the proposal report writing and submission.	2)	Final proposal report.
7	**Week 14**		
	Complete the oral presentation and prototype demonstration.	1)	Presentation slides.
		2)	An app demo.

Table 3.4: Milestone of FYP 2 and its key deliverable.

#	MILESTONE	KEY DELIVERABLE	
1	**Week 2**		
	Complete the research on existing systems and model training methods.	1)	Refined block diagrams.
	Complete the refinement of system design.	2)	Refined site map, wireframes and Mock-Ups
2	**Week 3**		
	Complete the final model training for anime face detection.	1)	A final YOLOv5 based anime face detector.
3	**Week 7**		
	Complete the final model training for anime characters recognition.	1)	A final anime recognition model.
	Complete the front end development.	2)	A fully developed frontend.
4	**Week 9**		
	Complete the backend development.	1)	A fully developed backend.
5	**Week 10**		
	Complete the app deployment.	1)	A deployed app that host on cloud.
6	**Week 12**		
	Complete system testing and evaluations.	1)	A deployed app that ready to be installed by everyone.
7	**Week 13**		
	Complete the final report writing and submission.	1)	Final report.
7	**Week 14**		
	Complete the oral presentation and final app demonstration.	1)	Presentation slides.
		2)	App demonstrations.

CHAPTER 4: SYSTEM DESIGN

4.1 System Block Diagram

4.1.1 General System Flow

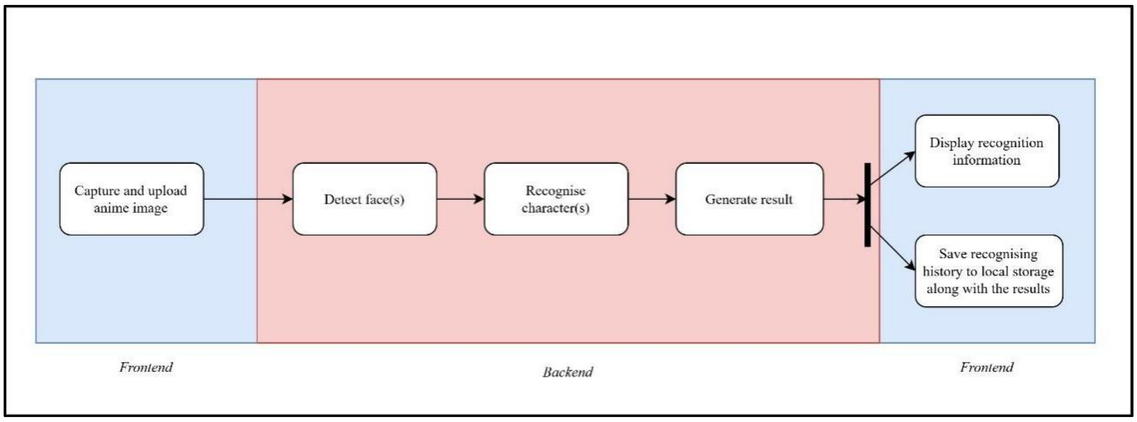

Capture and upload anime image
Detect face(s)
Recognise character(s)
Generate result
Display recognition information
Save recognising history to local storage along with the results
Frontend
Backend
Frontend

Figure 4.1.1: The general system flow.

The general flow of the system is very simple. First, users can engage with the system through the mobile app's frontend. Users can either capture an anime image using the camera or choose a picture from the device and upload it to the app. Next, the app's backend will detect the face(s) that appeared on the image and recognises them accordingly. The backend will then find match case from the database and generate result. Lastly, the computation result will be returned to the frontend and displayed to the users. The recognising history will then be saved to the device local storage along with the results.

4.1.2 Detection Flow

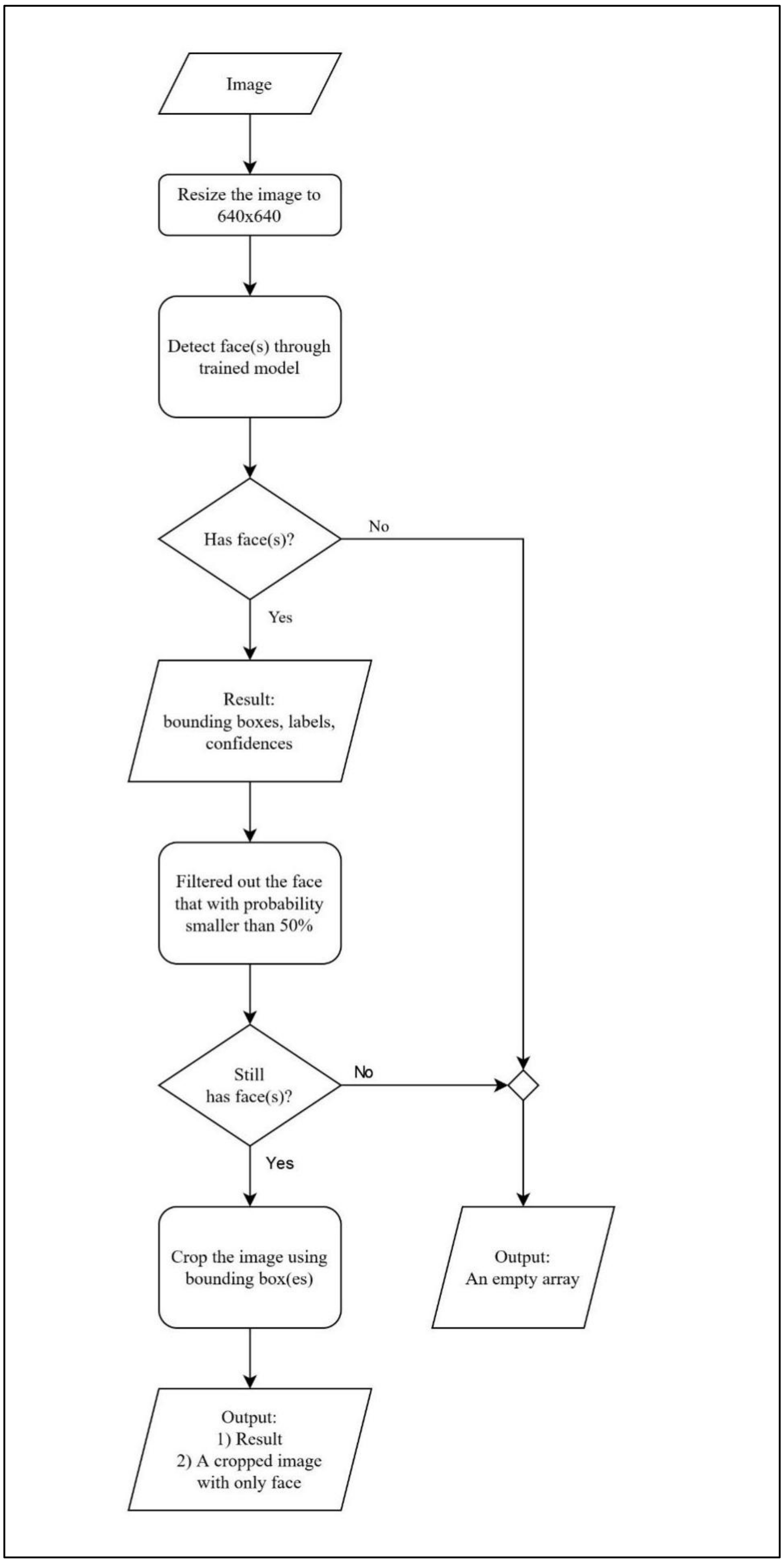


Image
Resize the image to 640x640
Detect face(s) through trained model
Has face(s)?
No
Yes
Result: bounding boxes, labels, confidences
Filtered out the face that with probability smaller than 50%
Still has face(s)?
No
Yes
Crop the image using bounding box(es)
Output: An empty array
Output: 1) Result 2) A cropped image with only face

Figure 4.1.2: Flowcharts of anime face detection.

4.1.3 Recognition Flow

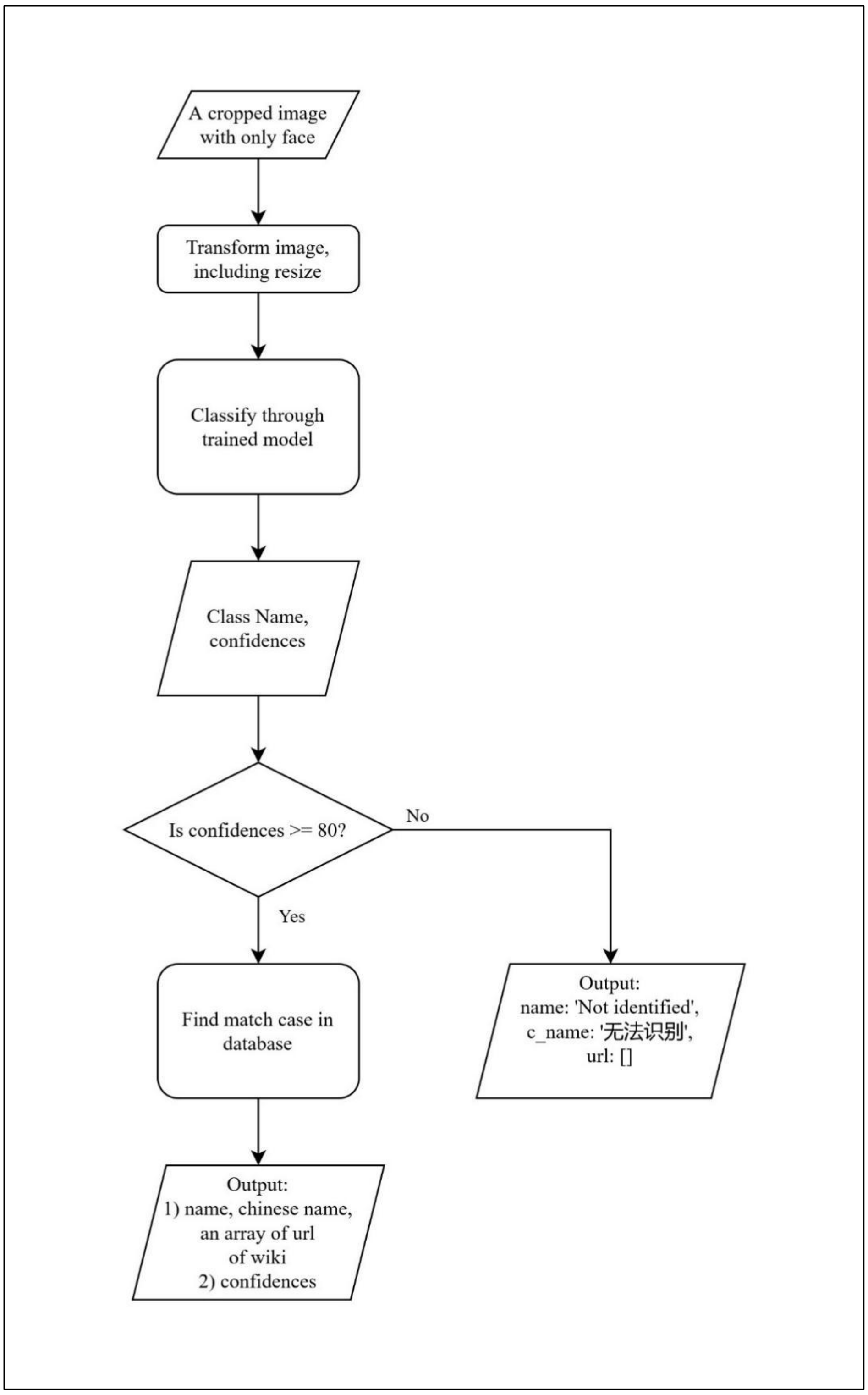

A cropped image with only face
Transform image, including resize
Classify through trained model
Class Name, confidences
Is confidences >= 80?
No
Yes
Find match case in database
Output: 1) name, chinese name, an array of url of wiki 2) confidences
Output: name: 'Not identified', c_name: '无法识别', url: []

Figure 4.1.3: Flowcharts of recognition.

4.1.4 System Architecture Diagram

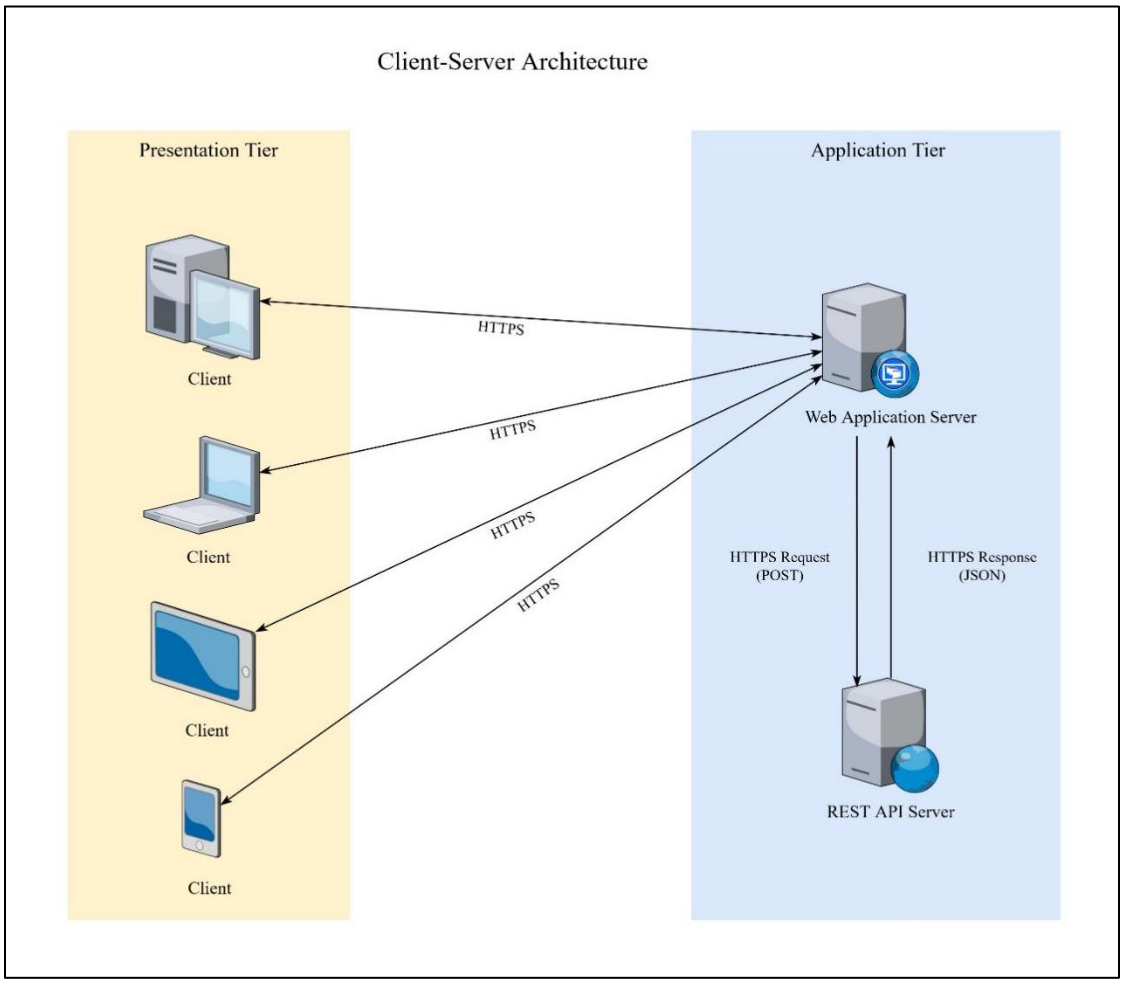


Figure 4.1.4: Client-server architecture diagram.

The app developed in this project adopts a client-server architecture. There are two servers involved in hosting the frontend and backend. From the above figure, the web application hosted the frontend, which is the web app. While the REST API server hosted the backend, which is the application that performs detection and recognition.

As a PWA, the app can be browsed using any platform from any device. The client makes an HTTPS request to the web application server to access the app. When the web application server receives an image to be recognised, the server will make an HTTPS POST request to the REST API server to perform detection and recognition remotely. The REST API server will return an HTTPS response with JSON results back to the web application server. The web application server displays the result to the client as mentioned '4.1.1 General System Flow'.

4.2 Frontend

4.2.1 Site Map and Wireframes

Figure 4.2.1 shows the site map of the app. The app adopts tab-based navigation. The three tabs are individual Tab components that can retain their state until refreshed. The “Search Page” is supposed to be the home page for the app, which will be the initial tab when the user enters the app. This page handles the capturing of images for recognition purposes. “Results Detail Page” is supposed to show the recognition results return by the REST API server. The “Feedback Page” allows users to do feedback when the results were wrong. Besides, the “Settings Page” allows users to change any settings about the app. For example, language, privacy and theme preferences. Moreover, users are able to view their recognition histories via the “History Page”. On this page, a list of recognition history will be displayed. Users can click the item and navigate to the “Result Page”. To ease the maintenance of source codes, it will share the “Results Detail Page” together with the “Search Page”. In addition, figures 4.2.2 to 4.2.5 show the wireframes of the mobile app. The private mode in “Settings Page” means that the recognition history would not be saved, and the language can change to either English or Chinese so that the results will return in the preferred language. The “Recognisable List Page” will display the list of anime characters that can be recognized.

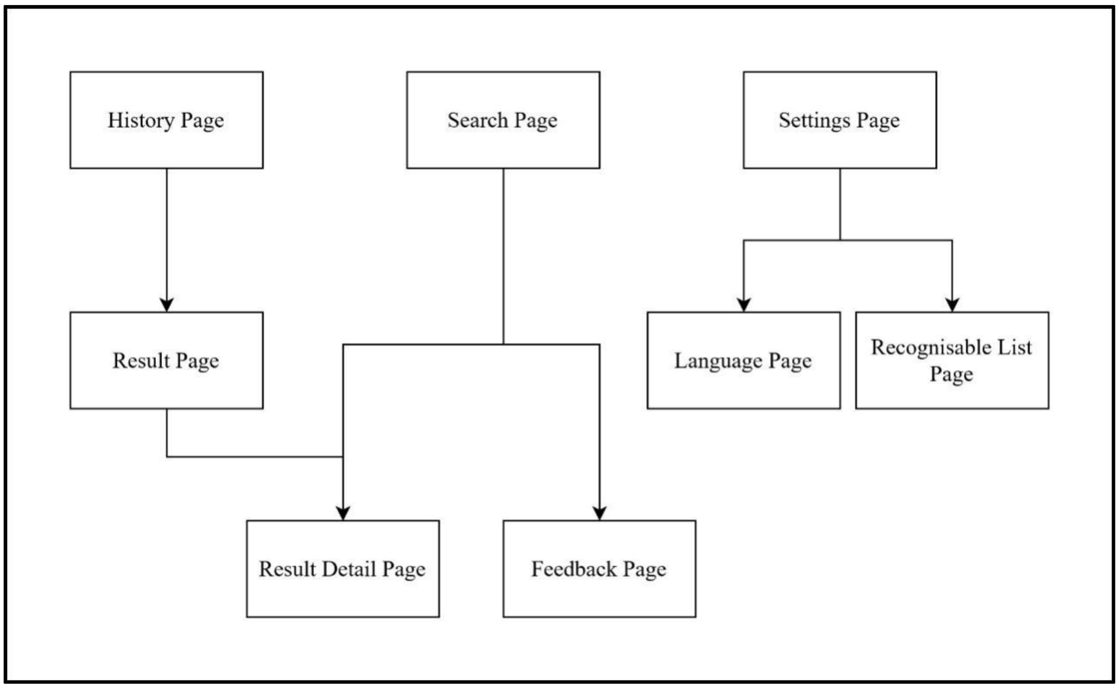


Figure 4.2.1: Site map.

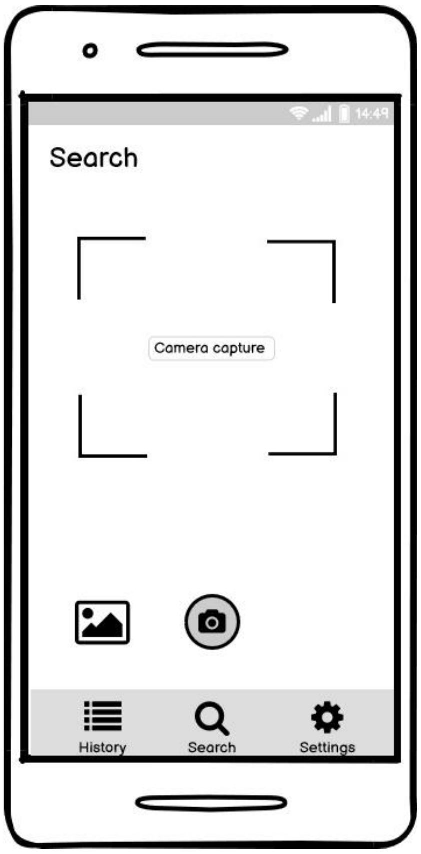


Figure 4.2.2: Wireframe of Search Page.

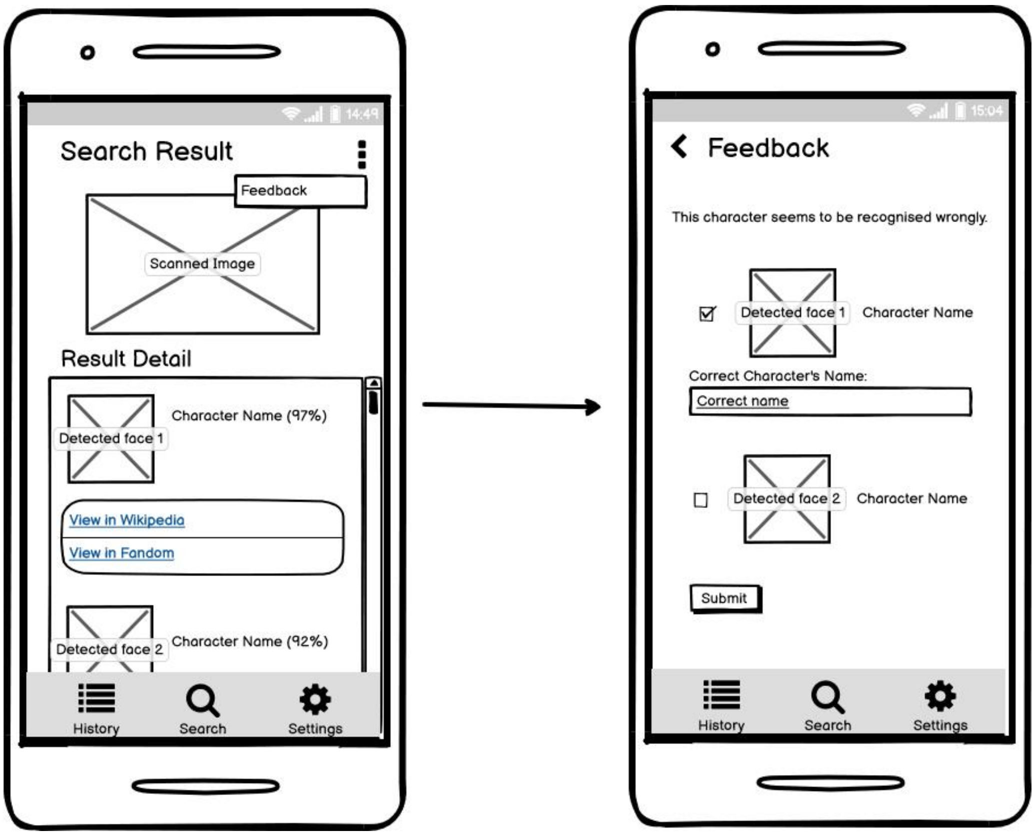


Figure 4.2.3: Wireframe of Search Result Page and Feedback Page.

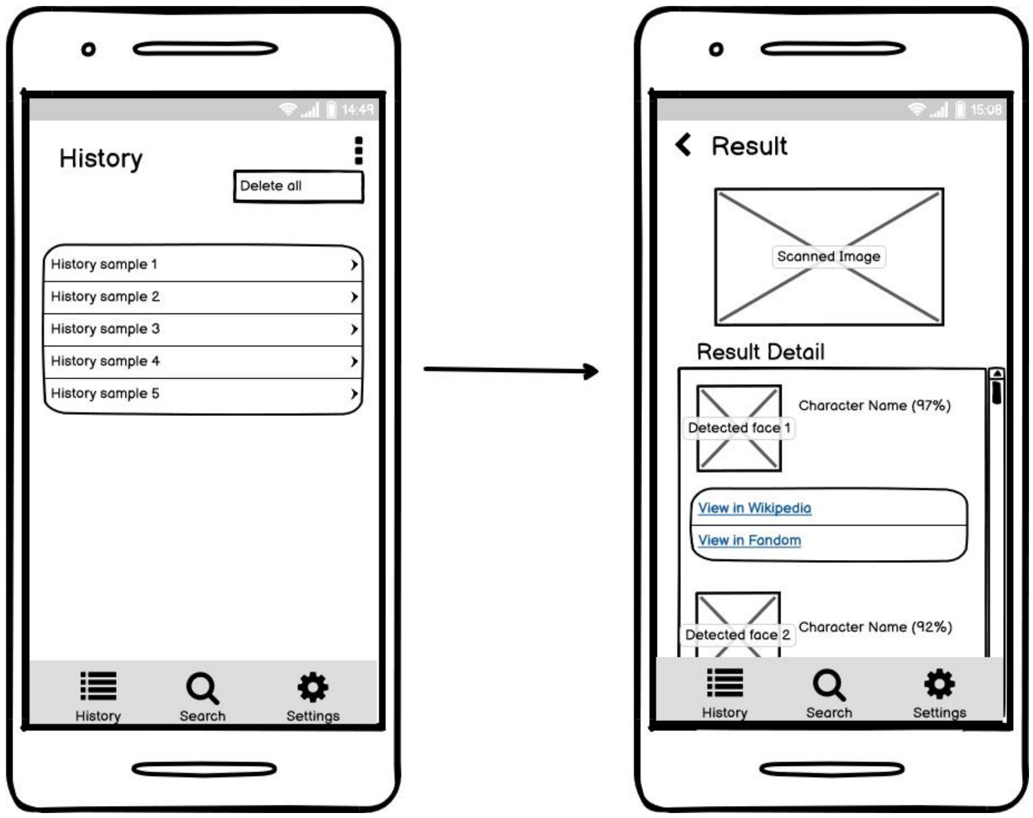


Figure 4.2.4: Wireframe of History Page and its Result Page.

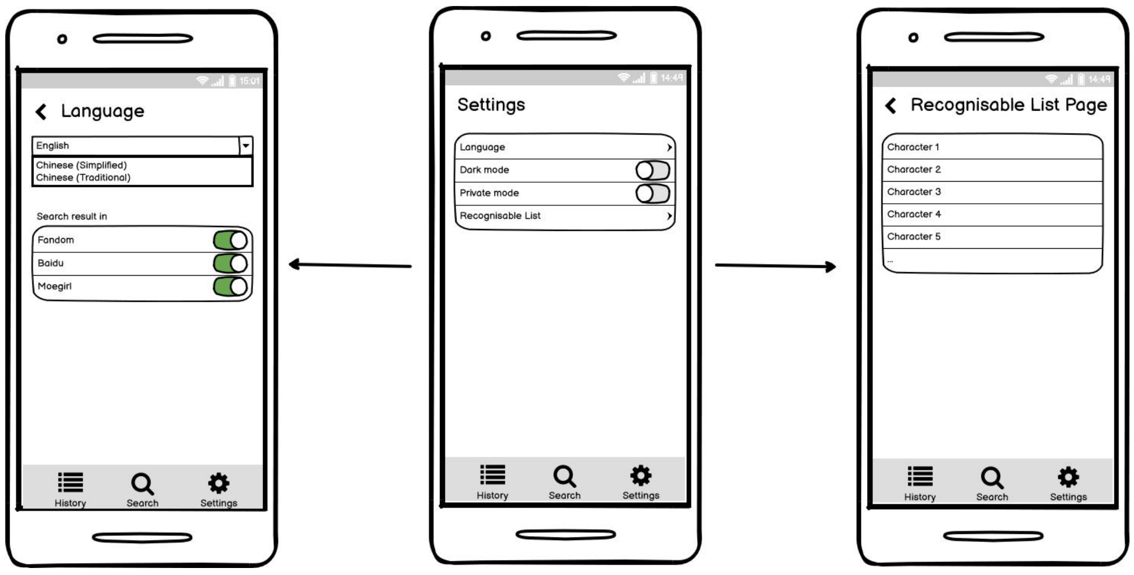


Figure 4.2.5: Wireframe of Settings Page, Language Page and Recognisable List Page.

4.2.2 Ionic Framework and React Hook

Ionic Framework was utilised in this project. The Ionic Framework provide a wide range of beautiful mobile UI components, such as the toggle, floating action button and action sheet. Moreover, the framework provides convenience to shift a typical web app into PWA, greatly saving the developer's effort and time. It supports Vue, Angular and React. Angular was excluded from the consideration because it is more suitable for writing complex business programs. While React is closer to JavaScript than Vue, just right for beginners with a JavaScript foundation.

To take pictures using the device's camera, Capacitor Camera API was utilised. A React hook named usePhotoTaking() was created. The hook contains function takePhoto(), loadSearchHistory(), deleteRecord(), callAPI() and saveResult() in which the last two functions were within the takePhoto(). The hook can be imported to be used when scripting the web pages. The job of each function is as Table 4.1.

Table 4.1: Functions of usePhotoTaking hook.

Function Name	**Function**
takePhoto()	Access device's camera to take picture. Access device's photo gallery or filesystem to upload an image.
loadSearchHistory()	Load history from local storage and store as a constant state.
deleteRecord()	Access to local storage and delete a record from history.
callAPI()	Make an HTTPS POST to the backend to get detection and recognition results.
saveResult()	Save result into local storage just after calling callAPI().

4.2.3 Data Storage

The storage method used in this project is Ionic Storage, which is an easy way to store key/value pairs and JavaScript Object Notation (JSON) objects. When running on the web or as a PWA, Storage will attempt to use IndexedDB, WebSQL, and local storage, in that order. As the app will only store recognition results as history, there is no need to configure a relational or non-relational database. The recognition result will be stored in the local storage as a JSON object. The key is named "history" , and its value will be a list of recognition results. Every time a detection and recognition activity perform, the old list of history will be retrieved and appended to the new JSON object to the list.

The structure of the recognition results history list is as follows

```
"history": [
       {
              id: <datetime>,
              json_results: <a list of returned recognition results from
              backend>,
              plotted_img: <a base64 image string>
       },
       {
              id: <datetime>,
              json_results: <a list of returned recognition results from
              backend>,
              plotted_img: <a base64 image string>
       },
       {
       .
       .
       .
       }
         ]
```

4.2.4 Data Type

Three data types were defined to easily manage the recognition results returned by the backend as an object.

```
type apiReturnJSON = {
  id: string;
  plotted_img: string;
  json_results: recResult[];
}
```

```
type recResult = {
  bbox: number[];
  cropped_img: string;
  rec_info: recInfo;
  rec_confidence: number;
}
```

```
type recInfo = {
  name: string;
  c_name: string;
  url: [];
}
```

4.2.5 Deployment

The app deployment adopted cloud computing method. The frontend was deployed using firebase. Firebase hosting offers many benefits for PWAs, including fast response times that thanks to CDNs, HTTPS enabled by default, and support for HTTP2 push.

4.2.6 Art Design

Logo

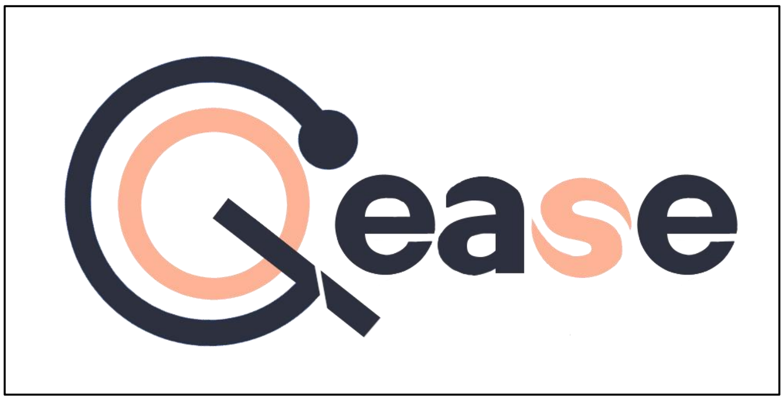


Figure 4.2.6: Logo of the app.

Icon

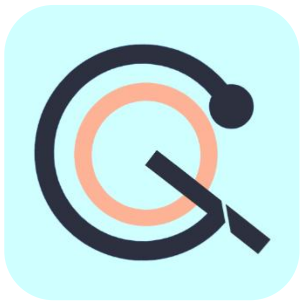


Figure 4.2.7: Icon of the app.

4.3 Backend

4.3.1 Model Training Method

Transfer learning will be adapted in this project. Transfer learning is a technique under deep learning in which the model was initialised with the pre-trained weights. This means the weights would not be trained from random initialisation that takes time and memory. It is very useful to train a model with a small dataset. Besides, transfer learning allows

- retraining the final layer of a pre-trained model as a feature extractor.
- fine-tuning the top layers. The backbone of the model will be freeze. Only the top layers will be trained. By fine-tuning the top layers, the model can be fit to the dataset faster while consuming less time and memory because the parameters need to be trained was lesser.
- retrained the whole model. The model will start training with the pre-trained weights.

However, the thing that has to be aware of is, the final layer should be refined to adapt to current class numbers.

Detection Model

For detection, YoloV5 will be utilised. YOLOv5 is a family of CNN object detection architectures and the models were trained on the COCO dataset. YOLO family is able to achieve a higher speed than other object detection architectures such as R-CNN family. According to Dwivedi [15], when both YOLOv5 and Faster R-CNN models running on NVIDIA 1080 Ti, the inference speed of YOLOv5 achieve 52.8 frames per second while Faster R-CNN only 21.7 frames per second. It is faster than R-CNN. Thus, YOLOv5 was given priority. YOLOv5 is the fifth generation of the YOLO family. It abandoned the use of Darknet that was used by the previous generation and adopted PyTorch instead. YOLOv5 provides four pre-trained models as shown in Figure 4.31. A custom model will be trained base on the YOLOv5s as it is the fastest model.

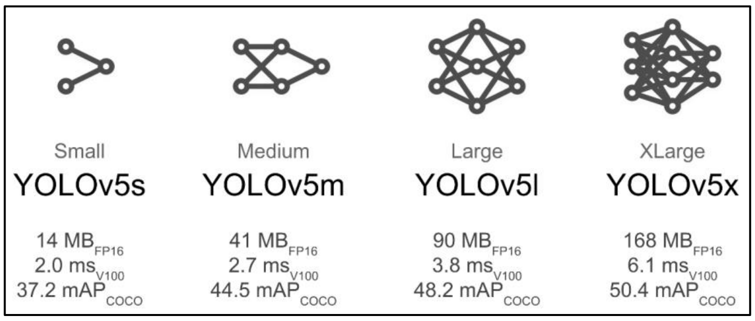


Figure 4.3.1: Types of YOLOv5 model.

Moreover, for recognition, the Torchvision library will be utilised in this project. As shown in Table 4.2, Torchvision provides a wide range of deep learning models alongside their pre-trained weights. **ResNet152, EfficientNetB0** and **EfficientNetB7** were enrolled in the experiment to find out the most suitable model that fulfils both accuracy and speed requirements because they have the higher Top-1 Accuracy than others. Top-1 accuracy would be the more important aspect than Top-5 accuracy because the system of this project will only give one output (class label) to users instead of the top 5 classes predicted.

Table 4.2: Available models by Torchvision [16].

Model	Top-1 Accuracy	Top-5 Accuracy
AlexNet	56.522	79.066
VGG-11	69.020	88.628
VGG-13	69.928	89.246
VGG-16	71.592	90.382
VGG-19	72.376	90.876
ResNet-18	69.758	89.078
ResNet-34	73.314	91.420
ResNet-50	76.130	92.862
ResNet-101	77.374	93.546
ResNet-152	78.312	94.046
SqueezeNet 1.0	58.092	80.420

SqueezeNet 1.1	58.178	80.624
Densenet-121	74.434	91.972
Densenet-169	75.600	92.806
Densenet-201	76.896	93.370
Densenet-161	77.138	93.560
Inception v3	77.294	93.450
GoogleNet	69.778	89.530
MobileNet V2	71.878	90.286
MobileNet V3 Large	74.042	91.340
MobileNet V3 Small	67.668	87.402
ResNeXt-50-32x4d	77.618	93.698
ResNeXt-101-32x8d	79.312	94.526
Wide ResNet-50-2	78.468	94.086
Wide ResNet-101-2	78.848	94.284
MNASNet 1.0	73.456	91.510
MNASNet 0.5	67.734	87.490
EfficientNet-B0	77.692	93.532
EfficientNet-B1	78.642	94.186
EfficientNet-B2	80.608	95.310
EfficientNet-B3	82.008	96.054
EfficientNet-B4	83.384	96.594
EfficientNet-B5	83.444	96.628
EfficientNet-B6	84.008	96.916
EfficientNet-B7	84.122	96.908

4.3.2 FastAPI Framework

After training the models, the models will cooperate with the web framework and be deployed to the backend server to realise the main functions of the mobile app. FastAPI, a modern and fast web framework will be used in this project instead of Django and Flask. Django is a more powerful and complex Python web framework. In this project, it is sumptuous to develop the server with Django. Meanwhile, Flask is a micro Python web framework. Micro framework means that Flask aims to keep its weight and simplicity but is still highly scalable. However, FastAPI was selected for this project. Actually, there is no big difference between the two of them, both are light and flexible. But as the name in FastAPI, it is much faster than Flask because it is built over Asynchronous Server Gateway Interface (ASGI) instead of Web Server Gateway Interface (WSGI) that the Flask is built.

Another advantage of using FastAPI is that the app developed can be opened to external connections as an API. The API can return detection and recognition results in JSON form. This greatly flexes and expands the value of the app. In this project, the API will return the below JSON as the detection and recognition result.

```
{
      “id”: <datetime>,
      “json_results”: {
                  "class_name": <string>,
                  "bbox": [],
                  "det_confidence": <float>,
                  rec_info": {
                        "name": <string>,
                        "chinese_name": <string>,
                        "url":[]
                  },
                  "rec_confidence": <float>
      },
      “plotted_img”: <a base64 image string>
}
```

4.3.3 Dependencies

The following dependencies were required to install in the environment.

For web framework:

- fastapi
- uvicorn
- aiofiles
- python-multipart
- requests

Basic dependencies:

- Cython
- matplotlib>=3.2.2
- numpy>=1.18.5
- opencv-python>=4.1.2

- Pillow
- PyYAML>=5.3.1
- scipy>=1.4.1
- torch>=1.7.0
- tqdm>=4.41.0
- pandas

For PyTorch models:

- torchvision>=0.8.1

4.3.4 Deployment

The app deployment adopted the cloud computing method. The backend was hosted in AWS EC2 virtual machine. The specification of the machine is shown in Table 4.3.

Table 4.3: Device specifications of AWS EC2 virtual machine.

	Specifications
Instance type	T2.micro (1 vCPU, 1GB Memory)
Platform	Ubuntu (Linux/UNIX)
Firewall	Allows ports :
Storage Volume Size	30GB

Port range	**Type**	**Protocol**	**Source**
80	HTTP	TCP	::/0
443	HTTPS	TCP	::/0
443	HTTPS	TCP	0.0.0.0/0
22	SSH	TCP	0.0.0.0/0
80	HTTP	TCP	0.0.0.0/0

CHAPTER 5: SYSTEM IMPLEMENTATION AND EVALUATION

5.1 Frontend

5.1.1 Project Setup

Install the Ionic CLI and start a React Project. There are three types of default templates (Tabs, List and Menu) that can be chosen from. In this project, the Tabs template was chosen to build a mobile app with a bottom navigation bar. Next, the project can be accessed using Microsoft Visual Studio Code for further development.

```
> npm install -g @ionic/cli
> ionic start gease tabs --type=react
```

Where “gease” is the project name.

5.1.2 Environment Setup

Node.js is required to interact with the Ionic ecosystem. Install Node.js from its official website.

Install the Capacitor Camera and Ionic PWA elements.

```
> npm install @capacitor/camera
> npm install @ionic/pwa-elements
```

5.1.3 Functionalities Development

The main and sub functions were developed with a well user interface. The main functions include picture taking and uploading as well as recognition result viewing. The sub functions include history viewing, recognition result feedback, recognisable character list viewing and preference settings.

Picture Taking

Users have two ways to scan a picture, one is to click on the home screen "GET STARTED" button, and another was click on the bottom right floating action button on the result page. This is because the user would not get to the home page after scanning the first picture until the app is refreshed or restarted.

There are different approaches to uploading a picture from the frontend according to the different devices. For PC, the image can either photo taken by device camera or selected from file system. For iOS, the image can be uploaded by photo-taking using the device camera, selected from the photo library and select from the file system.

Home Page

The default home page is the Search Tab with a welcome screen of the app logo.

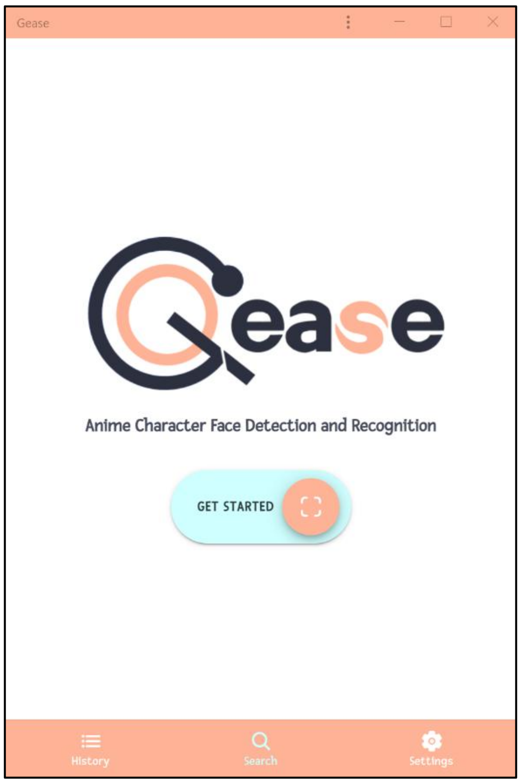


Figure 5.1.1: Home page (Search page).

Search Page with Result Details

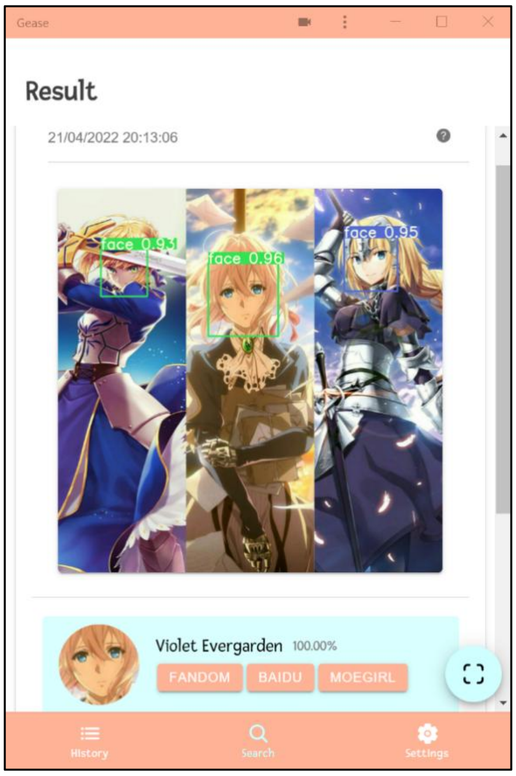


Figure 5.1.2: Search page with result details.

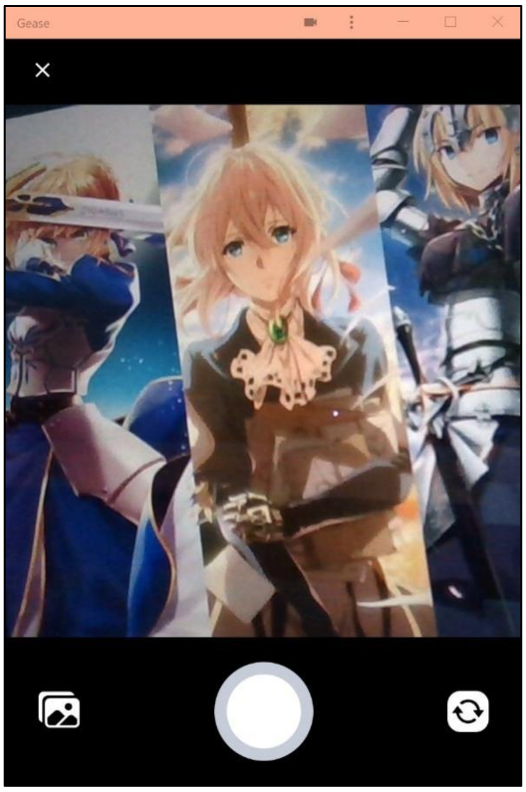


Figure 5.1.3: Capacitor Camera.

History Viewing

Users can view the history recognition result. Each of the recognition results is saved to the local storage and will be retrieved to display to users. The history results were displayed in a recycled view. Users can view the result details by clicking the card. Each record was named with the DateTime of recognising.

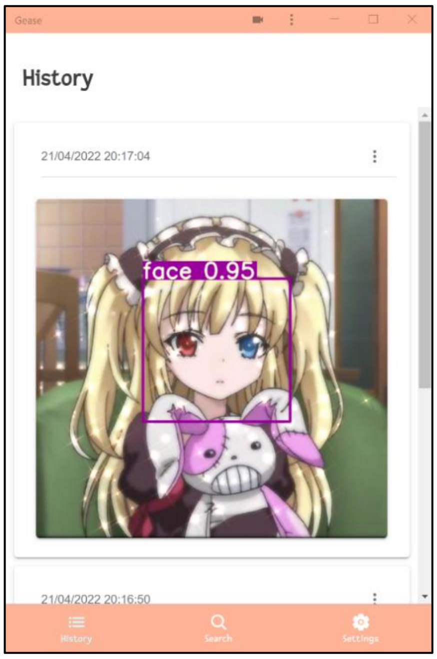


Figure 5.1.4: History page.

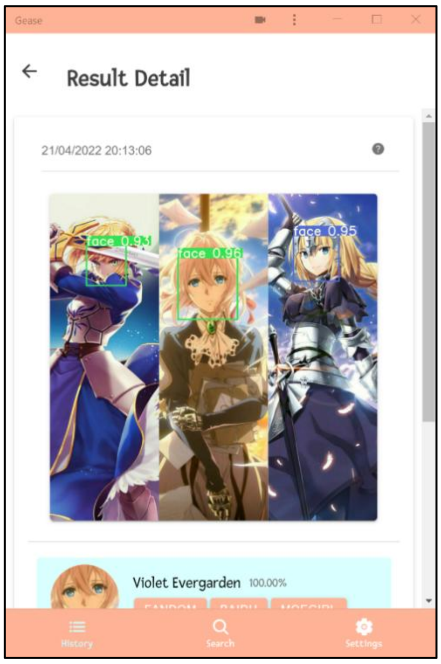


Figure 5.1.5: Result page with result details.

Feedback

Users can click on the top right question marks round icon on the result details page and enter the feedback page. Users can select the character that was wrong recognised and enter the correct name for her, then click "SEND FEEDBACK" for submission. The feedback form submission was handled using the HeroTofu platform. The submission can be viewed in the HeroTofu dashboard.

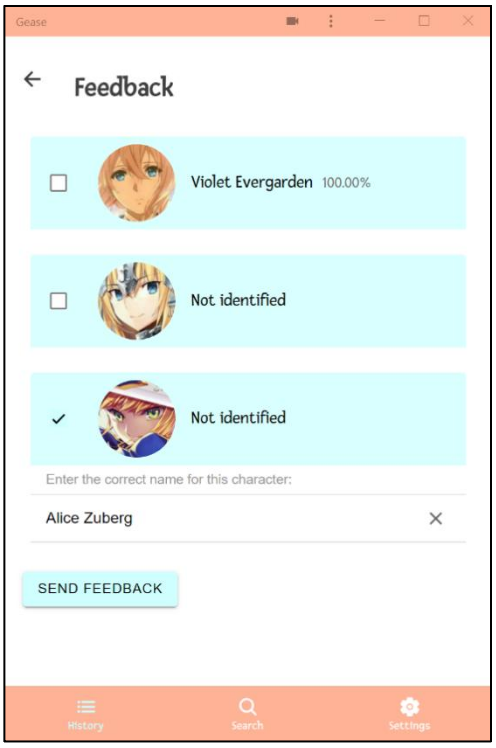


Figure 5.1.6: Feedback page.

Preference Settings

Users can set their preference for language, theme mode and privacy. For language, there are two choices to be choose, which are English and Chinese (Simplified). Users can active one of them or both. The recognition result will return according to the settings. For theme mode, the theme of the app will change to dark mode if users toggle the "Dark Mode" on the settings page or automatically change to dark mode if running on a mobile phone with a dark theme. For privacy, the recognition activity can be hidden and not stored to the database if users toggle the "Private Mode" on the settings page.

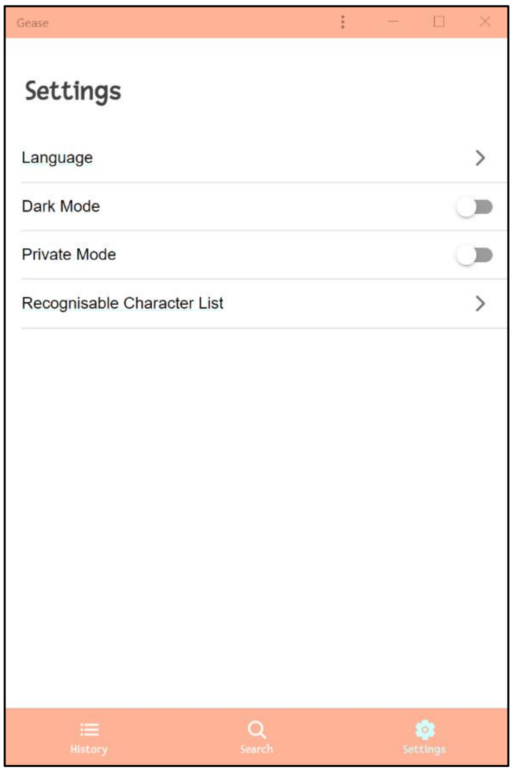


Figure 5.1.7: Settings page.

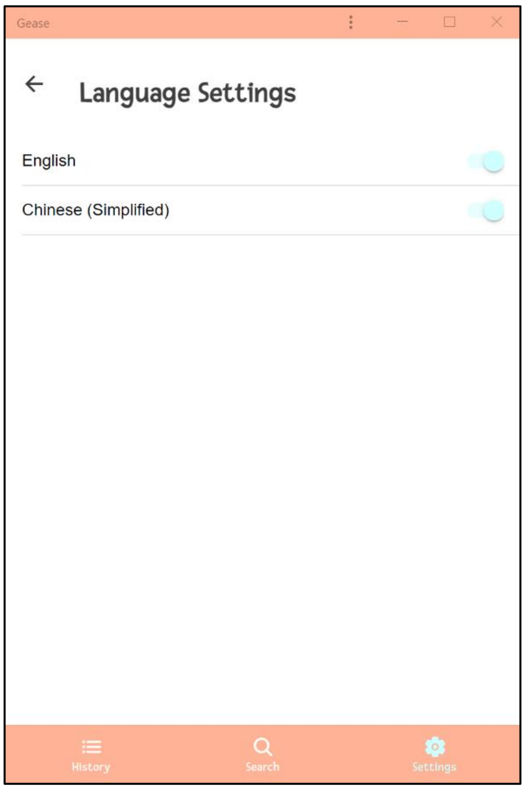


Figure 5.1.8: Language settings page.

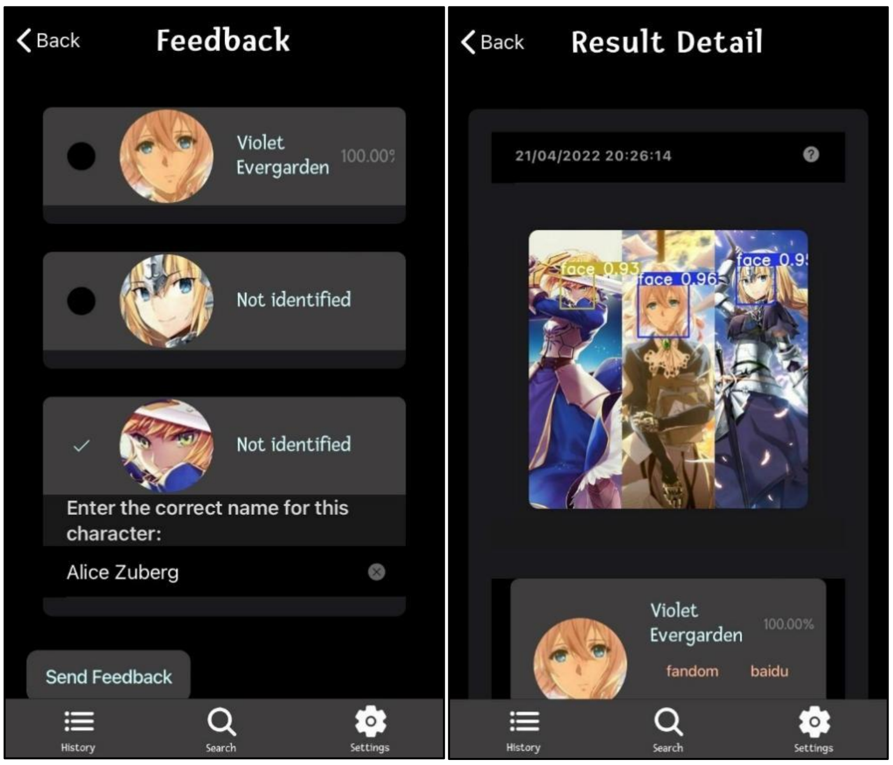
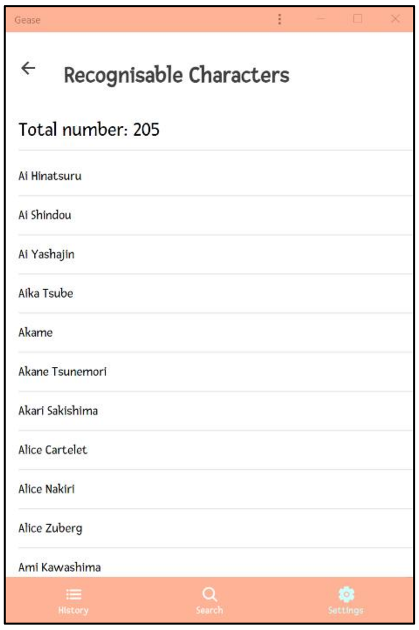


Figure 5.1.9: Result page and Feedback page in iOS dark theme.

Recognisable character list viewing

The recognisable character list can be retrieved from the backend using HTTPS GET. Since the list is get dynamically, any changes to the list when expanding the recognition model with more characters will no need to modify the frontend.


Gease

Recognisable Characters

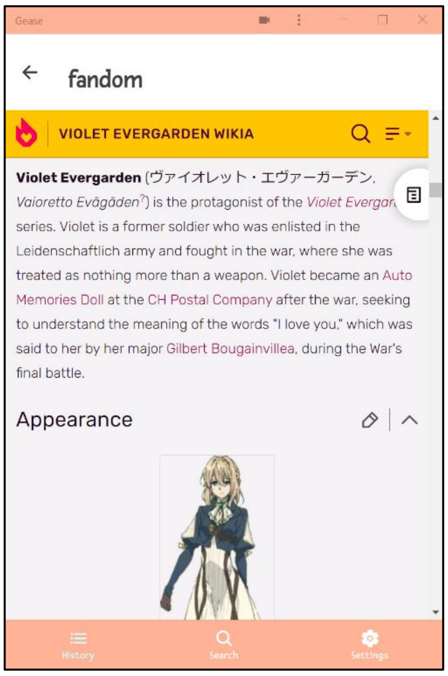
Total number: 205

Ai Hinatsuru
Ai Shindou
Ai Yashajin
Aika Tsube
Akame
Akane Tsunemori
Akari Sakishima
Alice Cartelet
Alice Nakiri
Alice Zuberg
Ami Kawashima

Figure 5.1.10: Recognisable character list page.

View Character's from ACGN Encyclopaedia

The recognition result returned by the backend included the URL of various ACGN encyclopaedias such as Fandom, Baidu and Moegirl. Users can click the button on the Result page and directing to the particular encyclopaedia page.



Violet Evergarden (ヴァイオレット・エヴァーガーデン, *Vaioretto Evāgāden*?) is the protagonist of the *Violet Evergar*
series. Violet is a former soldier who was enlisted in the Leidenschaftlich army and fought in the war, where she was treated as nothing more than a weapon. Violet became an Auto Memories Doll at the CH Postal Company after the war, seeking to understand the meaning of the words "I love you," which was said to her by her major Gilbert Bougainvillea, during the War's final battle.

Figure 5.1.11: View Violet Evergarden's information from Fandom.

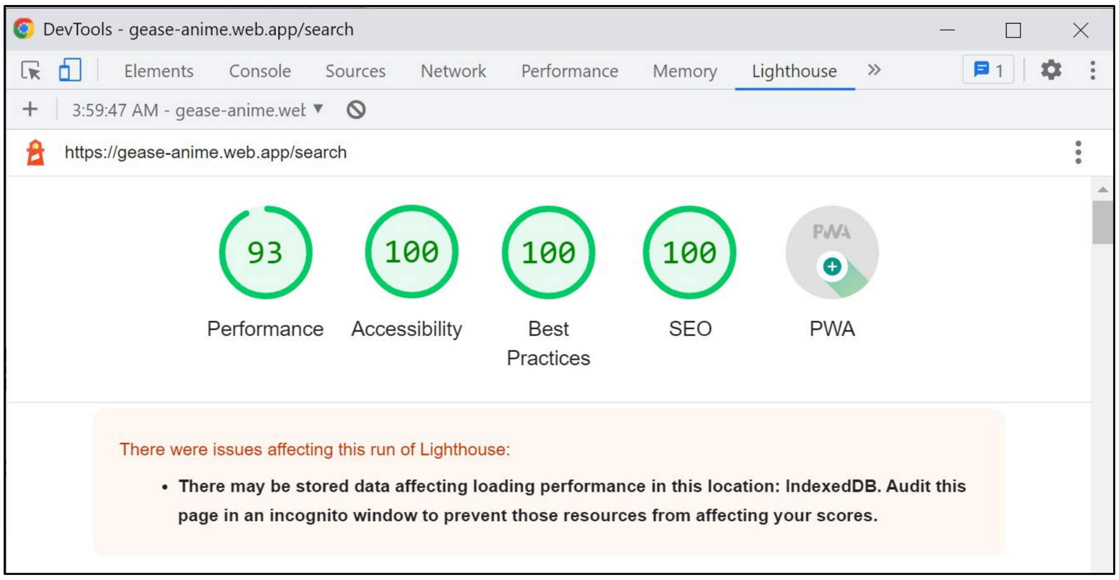
5.1.4 Shift to PWA

The app should be able to install on any platform as PWA by registering the service worker. According to [17], a service worker is a type of web worker. It is essentially a JavaScript file that runs separately from the main browser thread, intercepting network requests, caching or retrieving resources from the cache, and delivering push messages. To shift a web app to PWA, the service worker must be registered. Fortunately, Ionic Framework provides the service worker and is easy to enable.

Besides, a manifest JSON file is also essential for PWA. The manifest file must contain the below information:

```
{
  "short_name": "Gease",
  "name": "Gease",
  "icons": [
    {
      "src": "assets/icon/icon.png",
      "type": "image/png",
      "sizes": "512x512",
      "purpose": "any"
    }
  ],
  "start_url": "index.html",
  "display": "standalone",
  "theme_color": "#ffffff",
  "background_color": "#ffffff"
}
```

A performance report of the PWA can be generated using Google Developer's Lighthouse. By referring to the report, any enhancement required can be identified.



There were issues affecting this run of Lighthouse:

- There may be stored data affecting loading performance in this location: IndexedDB. Audit this page in an incognito window to prevent those resources from affecting your scores.

Figure 5.1.12: Performance report generated by Google Lighthouse.

5.1.5 Debugging

The app can be previewed and debugged by running the ionic server in localhost, using port 8100. To run the server, open the project directory, and type the below code in the command prompt:

```
> ionic serve
```

The app will be hosted at http://localhost:8100/. Brows http://localhost:8000/ using the web browser to access the app. By using the Ionic server, any changes made within the source code can be viewed instantly.

5.1.6 App Deployment

The PWA was deployed using Firebase. Install the firebase tools and create a project named "gease-anime". The project name will be the domain name after deployment. To deploy the PWA to firebase, first build the project and then deploy it by using the command below.

```
> ionic build -prod
> firebase deploy
```

If every configuration is done successfully, the PWA should now be able to access over HTTPS by **https://gease-anime.web.app/.**

5.1.7 PWA Installation

For a Personal Computer (PC), the PWA can be installed by clicking the install button provided by Google Chrome. For the Android platform, it can be installed via "Install app" option menu. While for iOS, it can be installed using the "Add to Home Screen" function.

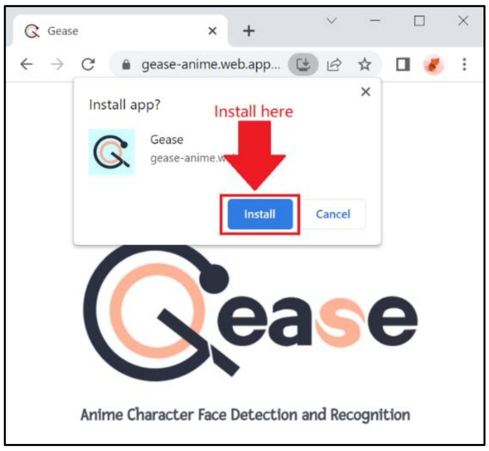


Figure 5.1.13: Install Gease PWA on PC's Google Chrome.

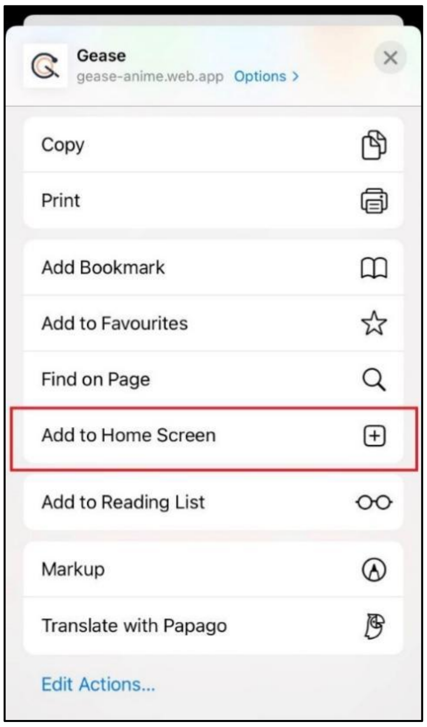


Figure 5.1.14: Install Gease PWA on iOS.

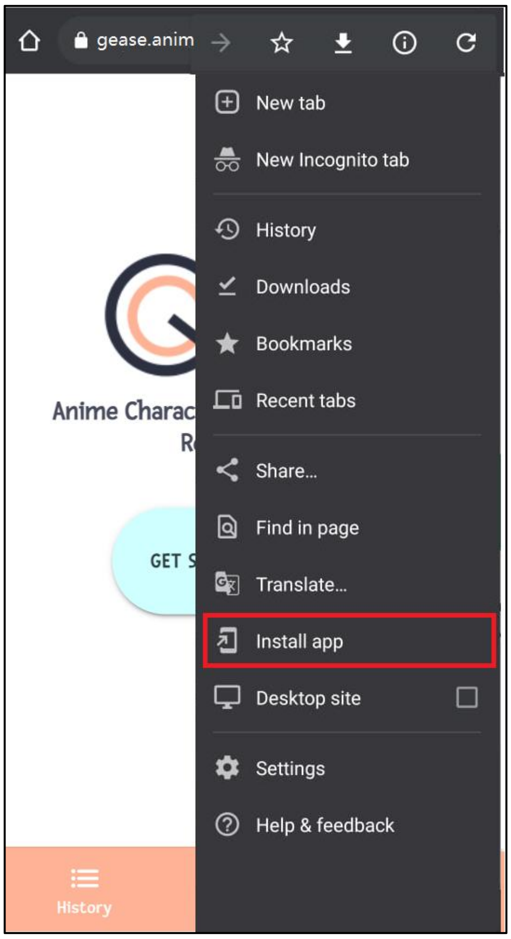


Figure 5.1.15: Install Gease PWA on Android.

5.2 Backend

5.2.1 Detection Model

YOLOv5s was adopted in this project to train an anime face detection model. The dataset used was the open-source provided by Dwiarto W., Lamp and Zhou [13] as used in their anime-face-detector project. The model training was carried out in Google Colab, it cannot support too large datasets hence 1500 samples were chosen instead of the full dataset, which has about 6000 samples. The dataset was in high resolution, most of them are about 1000x1000 pixels. To fit the YOLOv5 model, the images were resized to 640x640 pixels. YOLOv5 had a GitHub repository and provided train.py and detect.py to ease the process of custom model training. The image transformation can be done together with the training process using the codes below

```
!python train.py --img 640 --cfg yolov5s.yaml --hyp hyp.scratch.yaml --
batch 32 --epochs 100 --data anime_face_data_1500.yaml --
weights yolov5s.pt --workers 24 --name yolo_anime_face_det
```

where

--img is the size of an image to be resized,
--data anime_face_data_1500.yaml is the face annotation in YOLOv5 preferred yaml form,
--weights yolov5s.pt is the pre-trained weights,
and the model was trained in 32 batches and 100 epochs.

Figure 5.2.1 shows one example of a training batch.

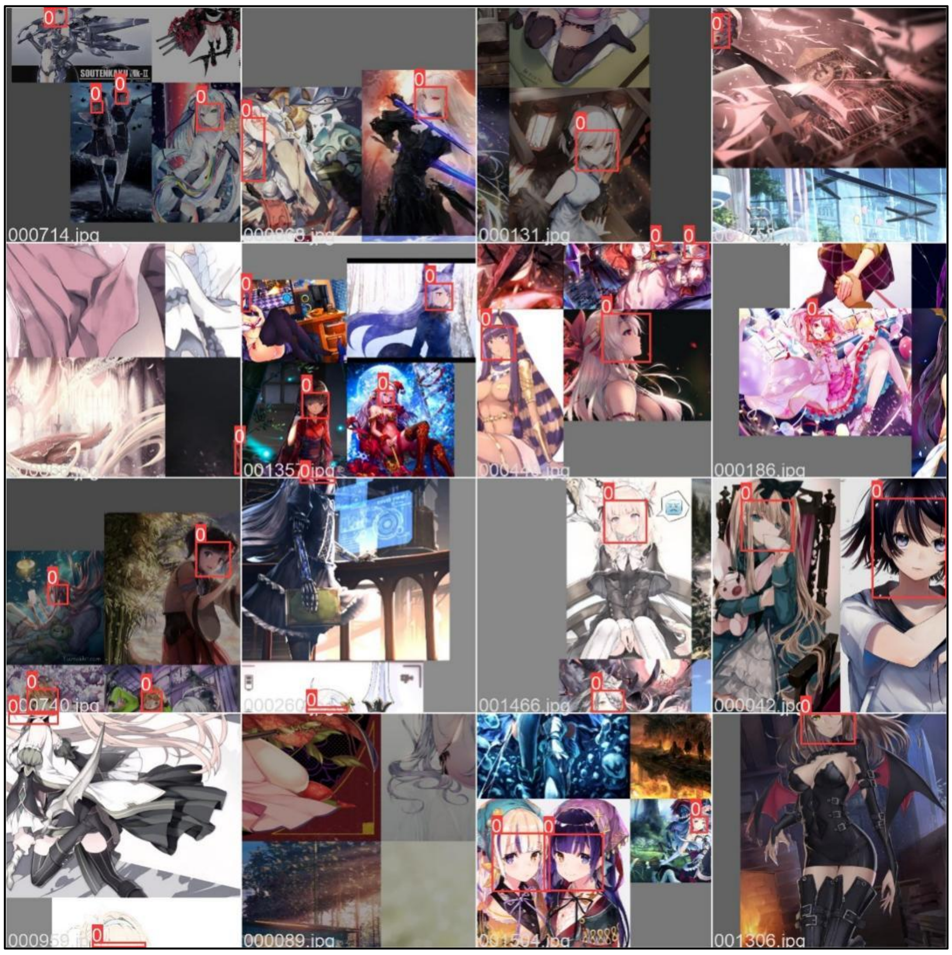


Figure 5.2.1: Training batch 1.

The training performance was then logged to the Weights & Biases site and can be browsed. Surprisingly, although only 1500 training samples were used, the performance was quite good. From Figure 5.2.2 and 5.2.3, they show the model achieve a **test accuracy of as high as 97%** and a precision-recall rate as high as 98.9%. Only 0.3% of the test set produced false negatives.

When running in Google Colab's NVIDIA T4s GPU, it achieved 0.6ms per pre-process, 29.7ms inference, 1.6ms non-maximum suppression per image. Total 31ms per face detection. The inference time is very good.

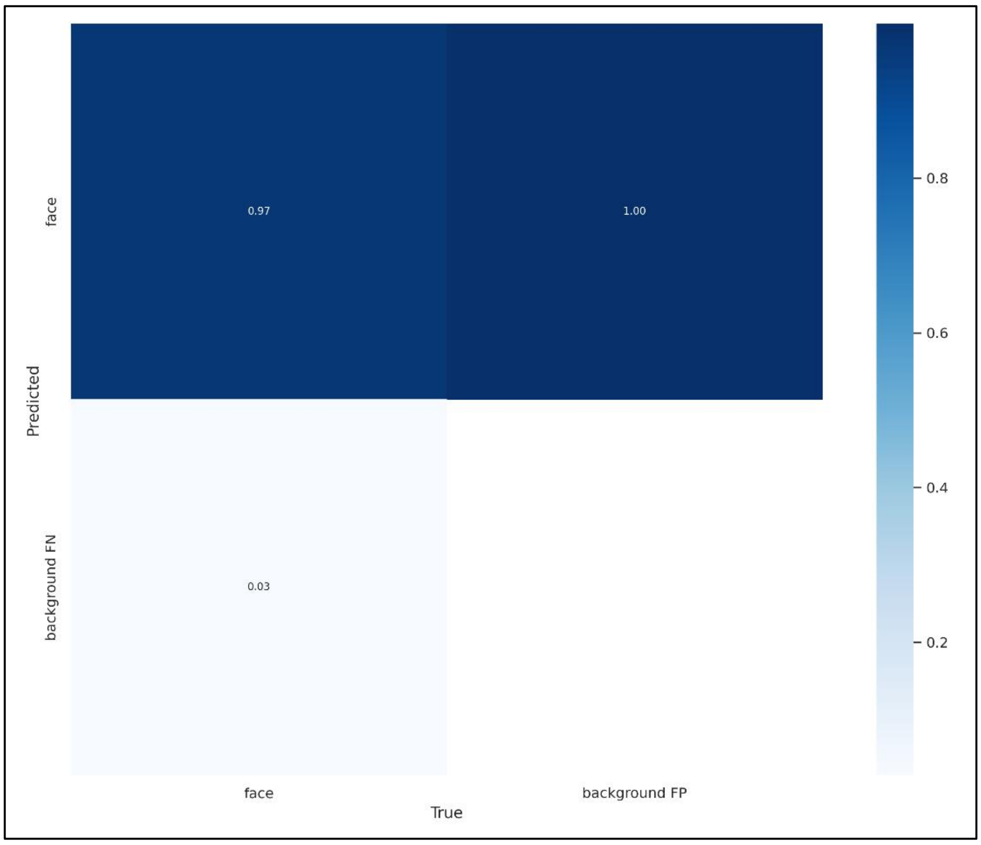


Figure 5.2.2: Confusion matrix for YOLOv5 anime face detection model.

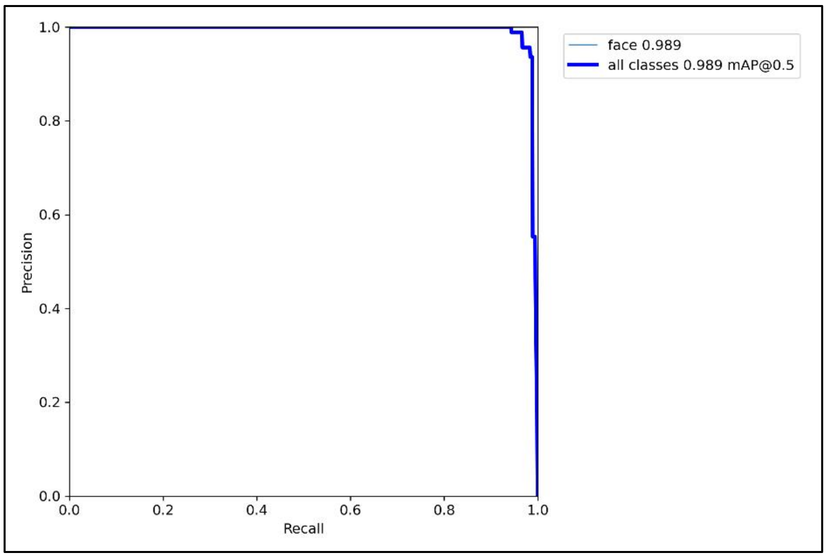


Figure 5.2.3: Precision-recall curve for YOLOv5 anime face detection model.

Below are some examples of testing results

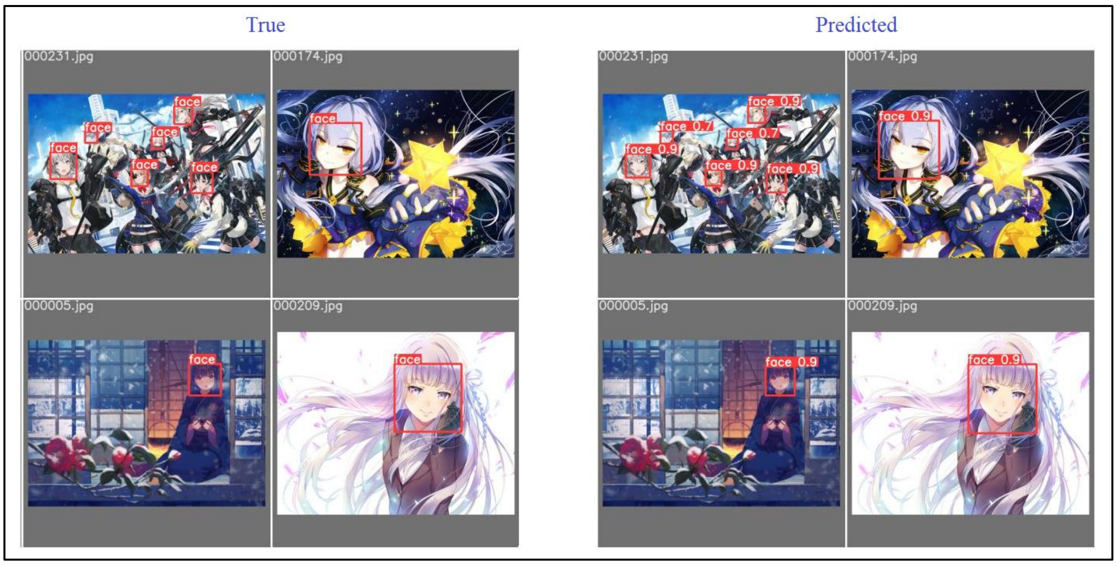


Figure 5.2.4: True positives examples.

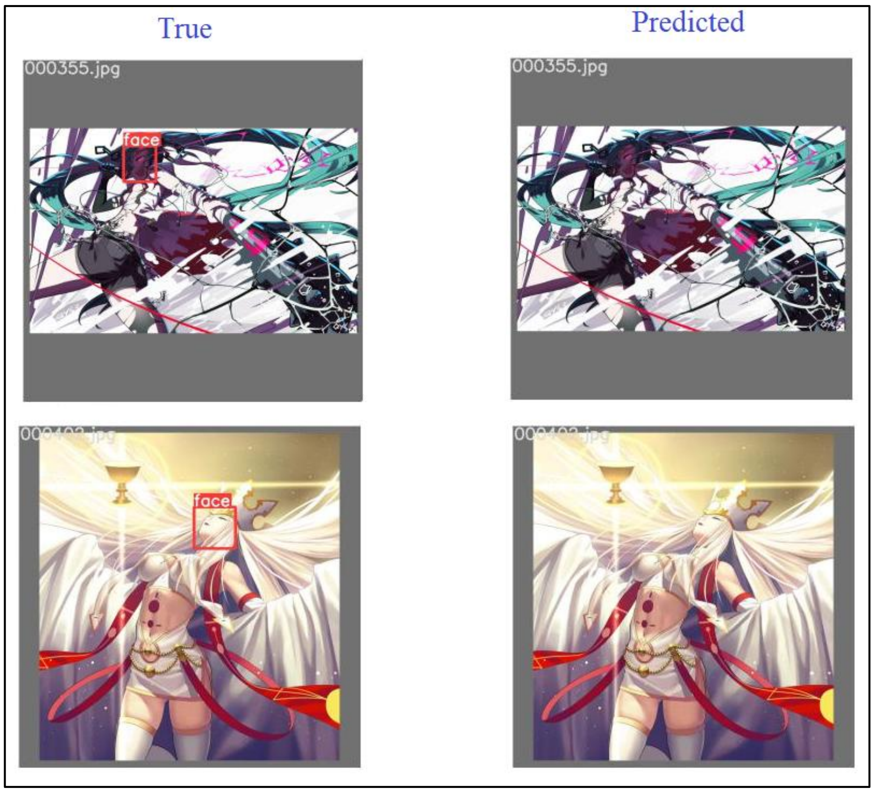


Figure 5.2.5: False negative examples.

Obviously, there is no serious false detection occurred. The false negatives were produced because of the unusual face of the test sample. One is a red face that is different from the normal skin tone, and another is a face looking up without any facial features showing. These misjudgment problems would not be the drawbacks to the project, because a face with an odd angle and no facial features cannot be recognised commonly.

5.2.2 Recognition Model

An experiment was carried out in this project to compare the adaptability of **ResNet152, EfficientNetB0** and **EfficientNetB7** with the dataset. There were 10 classes chosen from the full dataset from [18] to be enrolled in the experiment. The classes that been chosen was extremely similar to each other, which have the same blond hair. Figure 5.2.6 shows the example of each class and their labels.

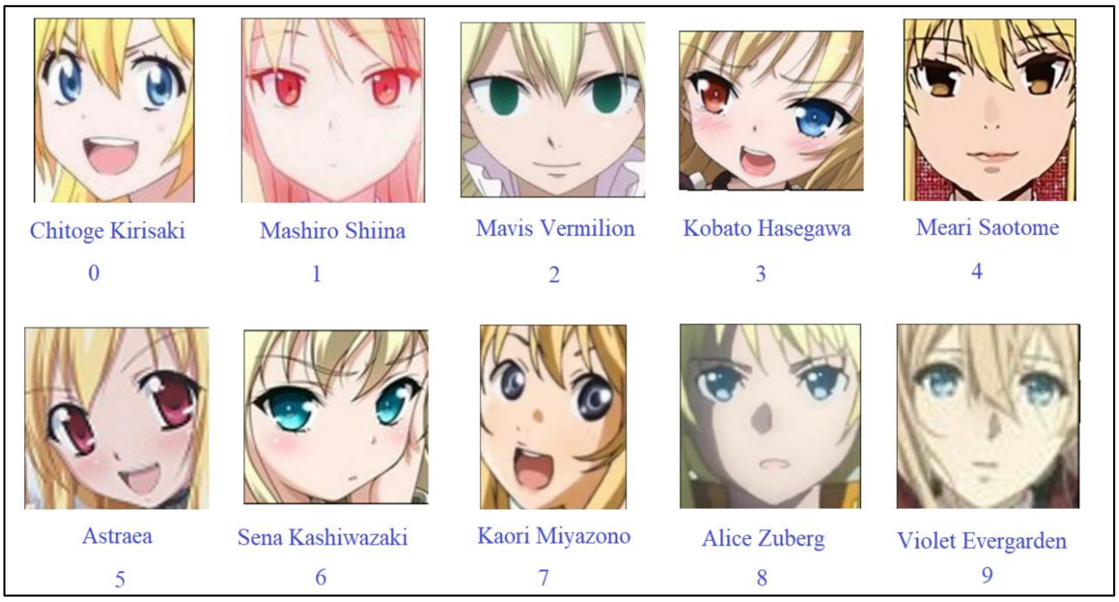


Figure 5.2.6: 10 classes selected to be enrolled in the experiment.

The purpose of this choice is that if such extremely similar classes can be well recognised, the remaining classes can logically be well recognised too. Therefore, in this experiment, the classes that were very different did not choose. For example,

Figure 5.2.7: Example of remaining classes.

Step 1: Data preprocessing

There is usually noise in the full image, such as the background. Only the face is of interest to the classification process. Thus, the image went through some preprocessing as shown in figure 5.2.8.

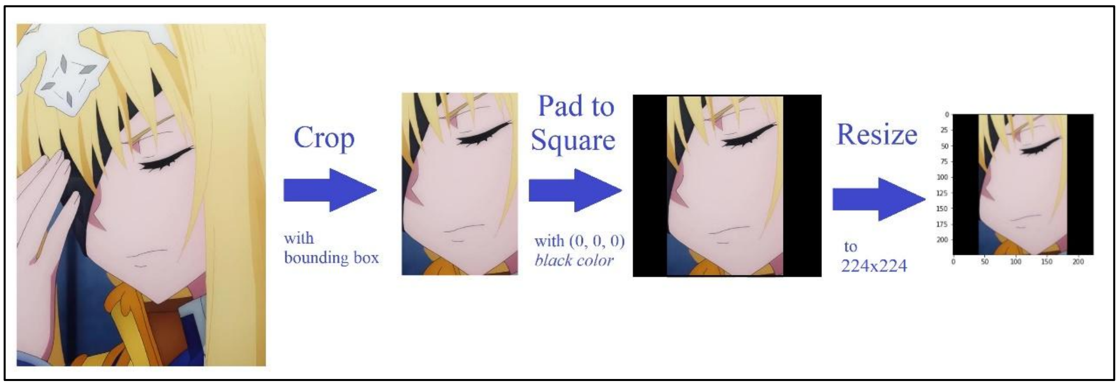


Figure 5.2.8: Process flows of data transformation.

Firstly, a full image needs to be cropped according to the bounding box annotation provided along with the dataset.

The annotation is looked likes below

```
Alice Zuberg/personai_icartoonface_rectrain_02601_0000260.jpg    243    314    743    846
```

The last four integers will be the bounding box values, which are the left upper corner, right upper corner, left bottom corner and right bottom corner. A helper function called extract_bbox was defined to extract them.

```
def extract_bbox(img_path):
  train_det = open("anime_dataset_10_rectrain_det.txt","r")
  for i, line in enumerate(train_det.readlines()):
    if (line.find(img_path)== 0):
      bbox_start_index = line.index('\t')+1
      bbox_raw = line[bbox_start_index:].replace('\n','')
      bbox = bbox_raw.split('\t')
      return int(bbox[0]), int(bbox[1]), int(bbox[2]), int(bbox[3])
```

After extracting the bounding box, it was implemented to the crop function. After cropping, the image was padded with (0, 0, 0) which is the black colour to expand it as a square. This is to avoid the image shrinks when resizing because the anime face is 2D, the face shape with actual aspect ratio is very important. Fu, et al. [7] had done data augmentation including flip, cut, zoom, rotate, affine transformation, all white or all black fill, Gaussian blur, mean blur, median blur, sharpening, effect embossing, contrast changing and Pixels moving to the image to expanding the training samples but the results were unsatisfactory. Therefore, as a lesson from previous works, data augmentation will not be used in this project.

Next, the image was resized to 224x224 pixels. Therefore, the input size will be (224, 224, 3), which the 3 was the number of channels. The channels will be RGB of the image. Finally, the data was converted to tensor and goes through normalization with mean and standard deviation.

The dataset was randomly split into train and validation set with a ratio of 8:2 using torch.utils.data.random_split function. The code is as shown below:

```
trainset, valset = random_split(dataset, [train_size, val_size])
```

where the training and validation set sizes were 730 and 182 respectively.

Step 2: Model Building

There were three models being built in this experiment.

i. ResNet152_model
ii. EfficientNetB7_model
iii. EfficientNetB0_model

Firstly, a base model will be built first by setting up its initial weights and bias that pre-trained with the ImageNet dataset. All models were being modified to adopt transfer learning. Some of the layers were frozen to allow retraining of the final layer and fine-tuning of the top layers.

For ResNet152, the model was being modified by

```
    if finetune_type == 'fixed':
        # ... freeze all layers ...
        parameters = list(model.parameters())
        for param in parameters[:-2]:
            param.requires_grad = False
        num_ftrs = model.fc.in_features
```

```
          model.fc = nn.Linear(num_ftrs, num_classes)

      elif finetune_type == 'toplayers':
          # ... freeze the bottom few layers ...
          for name, param in model.named_parameters():
            if not any(name.startswith(ext) for ext in ['layer4', 'fc']):
              param.requires_grad = False
          num_ftrs = model.fc.in_features
          model.fc = nn.Linear(num_ftrs, num_classes)
```

For EfficientNetB7, the model was being modified by

```
if finetune_type == 'fixed':
          # ... freeze all layers ...
          for name, param in model.named_parameters():
            if not any(name.startswith(ext) for ext in ['classifier[1]']):
              param.requires_grad = False
          model.classifier[1] = nn.Linear(in_features=2560, out_features=nu
m_classes)

elif finetune_type == 'toplayers':
          # ... freeze the bottom few layers ...
          for name, param in model.named_parameters():
            if not any(name.startswith(ext) for ext in ['features.7', 'feat
ures.8', 'classifier.1']):
              param.requires_grad = False
          model.classifier[1] = nn.Linear(in_features=2560, out_features=nu
m_classes)
```

The modification to EfficientNetB0 is the same as EfficientNetB7, the only difference is the in_features is 1280 as the convolution performs in EfficientNetB0 is lesser than in EfficientNetB7. Or simply put, EfficientNetB7 is deeper than EfficientNetB0.

The setting of param.requires_grad=True makes the model able to retrain the particular layers. To fine-tune the top layers, RestNet152 will only unlock the top layer - layer 4 – and the final fully connected layer so that these layers are retrainable. While for the EfficientNet, the last three layers, which are features 7 and 8 as well as the final classifier were unlocked. Meanwhile, only the last layer needs to be unlocked if the model is going to retrain the classifier only.

Step 3: Models Training and Validation

The training hyperparameters were set as batch_size=32 for training set, batch_size=16 for validation set, epochs=100 and learn_rate=0.001. The optimizer used is Adam and the loss function used is CrossEntropyLoss. The model will go through training and validation in each epoch. When a higher validation accuracy was found in an epoch, the weights trained will temporarily store as the best weights. After 100 epochs, the model will fit to the best weights by code bellow

```
    model.load_state_dict(best_model_weights)
```

Table 5.1: Traning and validation accuracy of all models.

Model	Validation Accuracy		Inference time	
	Retrain last layer	Fine-tune top layers	Retrain last layer	Fine-tune top layers
RestNet152	0.8076	0.8296	3.50319s	3.42565s
EfficientNetB7	0.8351	0.8956	4.81741s	5.14702s
EfficientNetB0	0.8076	0.8736	1.24481s	1.30743s

From Table 5.1, we can see that all models with top layers fine-tuning have a better performance than last layer retraining only. Furthermore, the EfficientNetB7 has the best validation accuracy, followed by EfficientNetB0 and RestNet152. However, the EfficientNetB0 takes the least inference time, which is less than a quadruple of EfficientNetB7. The inference time for 184 images was 1.30743 seconds, average 7.1ms per image. It was a very surprising result! Therefore, from the inference performance, EfficientNetB0 won.

Step 4: Model Evaluations

Since the dataset used in this experiment was too small, the validation set was used as the test set. As shown in Figure 5.2.9, the class distribution of the dataset used in this experiment was seriously unbalanced. The highest was 264 but the lowest was only 17. Looking at the complete dataset, this gap between the rich and the poor will become even more serious. Therefore, the samples randomly assigned to the test set will also be extremely uneven.

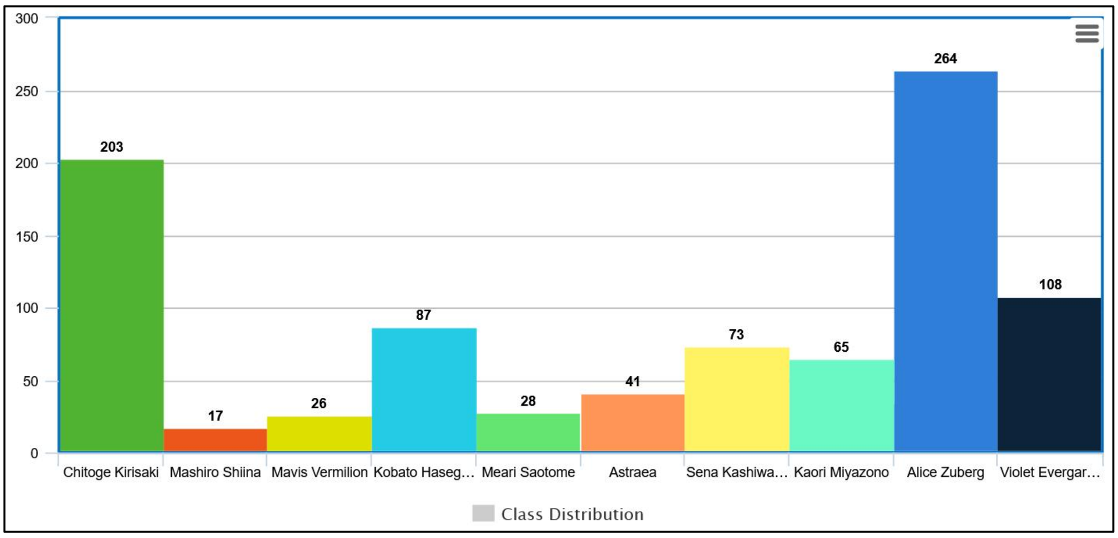


Figure 5.2.9: Class distribution of dataset used in this experiment.

- **Speed**

When running in Google Colab's NVIDIA T4s GPU, the inference time for 184 images was 1.30743 seconds, average 7.1ms per image. The performance in speed is excellent.

- **Accuracy**

It can be seen from Figure 5.2.10 that the prediction performance of Mashiro Shina, who had the fewest samples, is not ideal. All of the output was FNs. While Alice Zuberg with the highest number of samples gives an accuracy of 96% with only 2 FNs and 2 FPs. From a general perspective, the lesser the samples, the lower the accuracy. This gives a lesson that the number of dataset samples is really important. and 3 FPs.

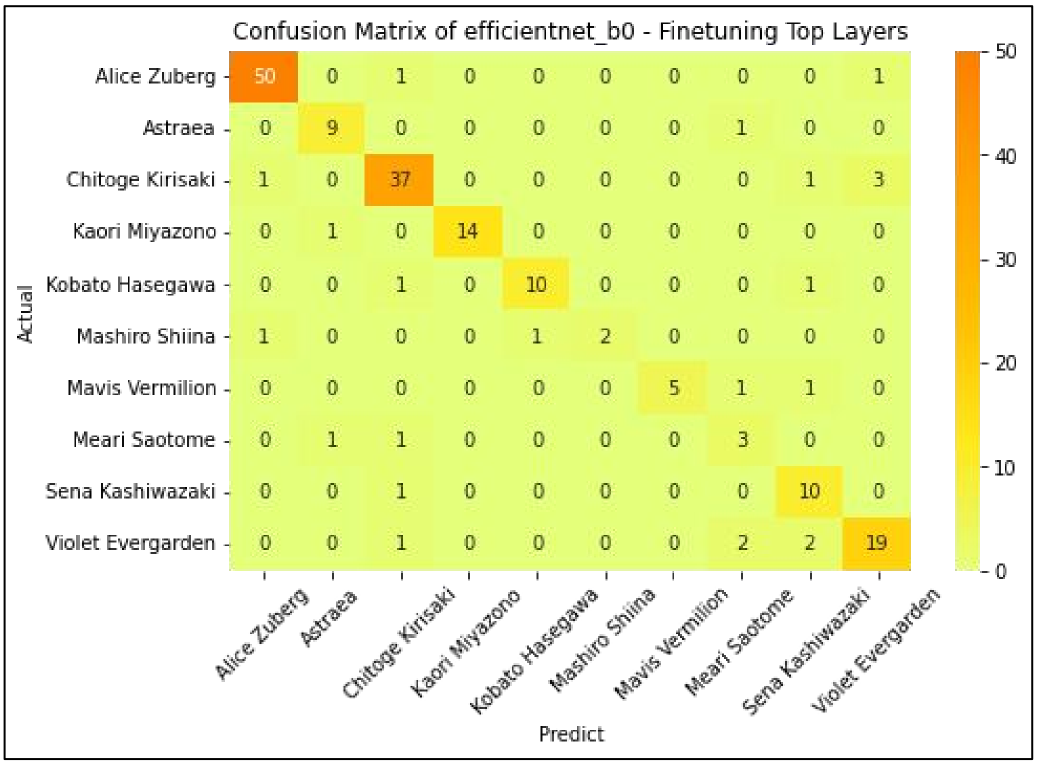


Figure 5.2.10: Confusion matrix of EfficientNetB0_model with fine-tuning top layers.

5.2.3 Retrain Recognition Model with real dataset

After the experiment done in the last section, we can conclude that **EfficientNetB0** is the best model to be used as the recognition base model. Step 1 until step 3 in section 5.2.2 were repeated with the real dataset and EfficientNetB0 model. There were 205 classes chosen from the full dataset from [18] as the real dataset in this project. The classes chosen are also the recognisable characters by the app developed in this project. The classes that have been chosen are all female and "born" before 2020 March. The dataset was spitted into a training and validation set with a number of samples of 13239 and 3310 respectively. Figure 5.2.11 shows some samples from different classes and their labels.

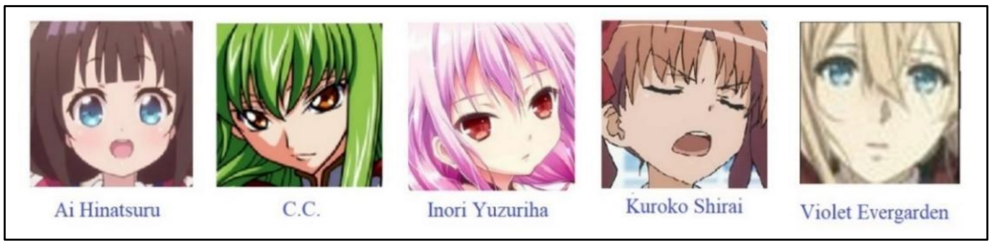


Figure 5.2.11: 5 example classes in real dataset.

As shown in Figure 5.2.12, the class distribution of the dataset was seriously unbalanced. The highest was Honoka Kosaka with a number of 382 samples but the lowest was Shion Sonozaki with only 15.

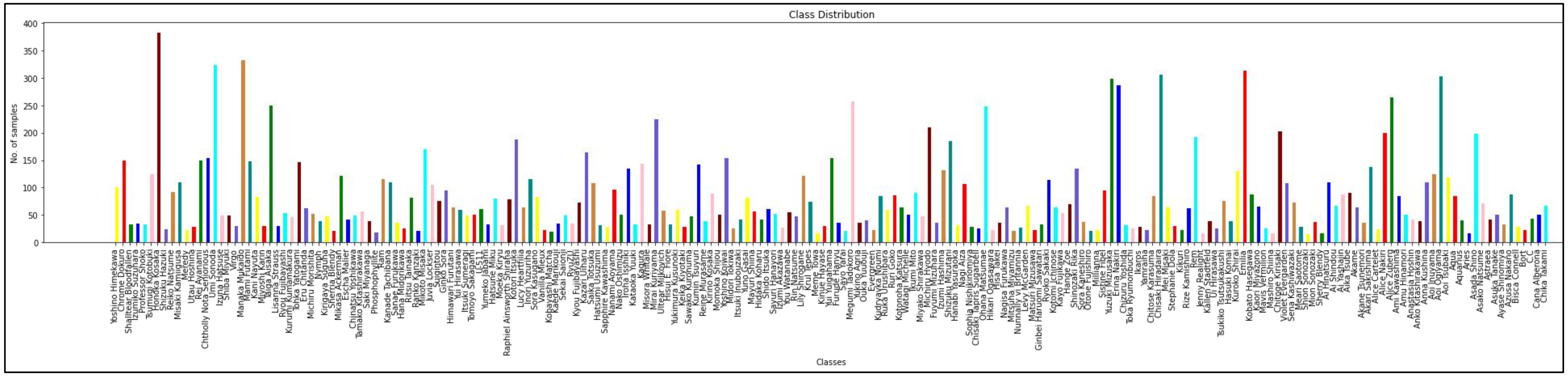


Figure 5.2.12: Class distribution of real dataset.

- **Accuracy**

Due to the increase of classes, the validation accuracy drops to 0.6319 which consider still a good performance. The accuracy can be improved by adding more samples to the dataset and balance the number of samples in each class.

- **Speed**

When running in Google Colab's NVIDIA T4s GPU, The inference time for 3310 images was 15.76132 seconds, average 4.8ms per face recognition, improve from the experiment model which has a inference time of 7.1ms per image. The excellent inference time still retain.

5.2.4 Environment Setup

A virtual environment named "env_anime" specified for FastAPI development has been created using Anaconda. This virtual environment was installed with the dependencies that required for the backend development.

For web framework:

- fastapi
- uvicorn
- aiofiles
- python-multipart
- requests

Basic dependencies:

- Cython
- matplotlib>=3.2.2
- numpy>=1.18.5
- opencv-python>=4.1.2
- Pillow
- PyYAML>=5.3.1
- scipy>=1.4.1

- torch>=1.7.0
- tqdm>=4.41.0
- pandas

For PyTorch models:

- torchvision>=0.8.1

The anaconda environment is then applied in Microsoft Visual Studio 2019 to ensure the that the workspace is works under a machine environment with all dependencies installed.

5.2.5 FastAPI Development

The trained detection and recognition models collaborated with the FastAPI framework to develop a comprehensive backend.

The file structure of the FastAPI app project is

```
Project folder
	|- anime_205_info.txt
	|- app.py
	|- classes-205.pt
	|- best.pt
	|- efficientnet_b0_model.pth
	|- requirements.txt
```

Where the anime_205_info.txt contain the all the information of recognisable character. For example,

```
{"name": "Alice Zuberg", "c_name": "爱丽丝·滋贝鲁库", "url":
[{"fandom":"https://swordartonline.fandom.com/wiki/Alice_Zuberg"},
{"baidu":"https://baike.baidu.com/item/%E7%88%B1%E4%B8%BD%E4%B8%9D%C2%B7%E6%BB%
8B%E8%B4%9D%E9%B2%81%E5%BA%93"},
{"moegirl":"https://zh.moegirl.org.cn/index.php?title=%E7%88%B1%E4%B8%BD%E4%B8%
9D%C2%B7%E6%BB%8B%E8%B4%9D%E9%B2%81%E5%BA%93&variant=zh-
hans&mobileaction=toggle_view_desktop"}]}
```

The app.py is the python file contain all the source code, adopting FastAPI framewok.

The classes-250.pt is a file that contains a list of recognisable characters' names, which is also the class name.

The best.pt is the best weights for the YOLOv5 detection model trained in section 5.2.1. While the efficientnet_b0_model.pth is the best weights for the EfficientNetB0 recognition model that is trained in section 5.2.3. These weights will be loaded to the app.py during the model definition parts.

Model Definitions

```
##############################################
#------------ Model Definition ---------------
##############################################

# Detection model
img_size = 640
det_model = torch.hub.load('ultralytics/yolov5', 'custom', path='best.pt')

# Recognition model
model = models.efficientnet_b0(pretrained=True)
model.classifier[1] = nn.Linear(in_features=1280, out_features=10)
model_name = 'efficientnet_b0_model.pth'
model.load_state_dict(torch.load(model_name,
map_location=torch.device('cpu')))
model.eval()

# Recognition classes
classes = torch.load('classes-205.pt')

# For Image Preprocessing(transformation)
transform = T.Compose([
    SquarePad(),
    T.Resize(224),
    T.ToTensor(),
    T.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
```

Routes

After model definition, the routes defined.

There are three routes in total, /detect_via_app/, /detect_via_api/ and /list/. First two routs implement detection and recognition in a same logic but receives different inputs and gives different ouputs.

i. /detect_via_app/

This route is the function to handle HTTPS POST from frontend.

- It receives a dictionary, which is {"img": <a base64 image data>}.
- It returns a JSON results of running YOLOv5 on the uploaded image. A images with bboxes drawn are base64 encoded and returned inside the JSON response too.

```
@app.post("/detect_via_app/")
async def detect_via_app(request: Request, json: dict) -> dict:

      # Processing input: decode image from base64data
      …

      # Detection
      json_results = …

      # Recognition:
       # 1. crop image accroding to bbox, then add to json_results
       # 2. recognising
       # 3. get info base on rec_name
       # 4. add rec_info & rec_confidence to json_results
      …
      json_results.append(…)

      # Plot bbox on img
      plotted_img  = …

      # Naming result with current datetime
      dt_string = …

      return   {'id':   dt_string,   'plotted_img':   plotted_img,
'json_results': json_results}
```

ii. /detect_via_api/

This route is the function to handle other API connection, either use HTTP POST or HTTPS POST.

- It receives a list of image files as essential input and a boolean value of download_image parameter that includes base64 encoded image(s) with bbox's drawn in the json response as optional input.

- It returns the JSON results of running model on the uploaded image. If download_image parameter is True, images with bboxes drawn are base64 encoded and returned inside the json response.

```
@app.post("/detect_via_api/")
async  def  detect_via_api(request:  Request,  file_list:  List[UploadFile]  =
File(...), download_image: Optional[bool] = Form(False)):


       # Processing input
       …

       # Detection
       json_results = …

       if download_image:
              # Recognition
              …
              json_results.append(…)


              # Plot bbox on img
              …
              json_results.append(…)

       else:
              # Recognition
              …
              json_results.append(…)

       return {'json_results': json_results}
```

iii. /detect_via_api/

This route accept HTTP GET request. The function within this route will Get the list of recognisable anime characters and return the list.

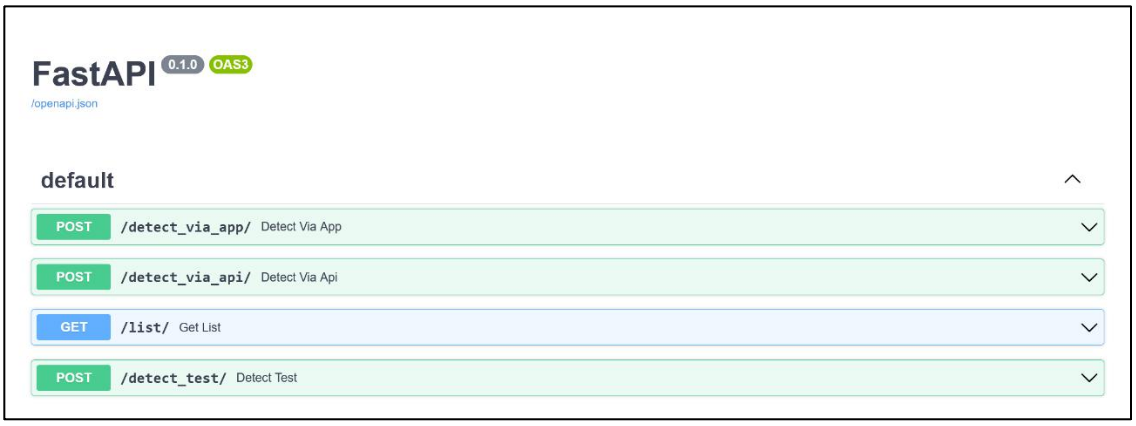
```
@app.get("/list/")
def get_list():
       '''
       Get list of recognisable anime characters
       '''

       return {'total': len(classes), 'list': classes}
```

5.2.6 Debugging

The API can be debugged by running the Uvicorn server in localhost, using port 8000. To run the server, open the project directory, and type the below code in the command prompt:

```
> uvicorn app:app -reload
```

Next, start the server and the FastAPI app will be hosted at http://localhost:8000/. We can access the API on the web browser using the URL http://localhost:8000/docs#. Figure 5.2.13 shows the page generated by Swagger UI.


FastAPI 0.1.0 OAS3
/openapi.json

default

POST /detect_via_app/ Detect Via App

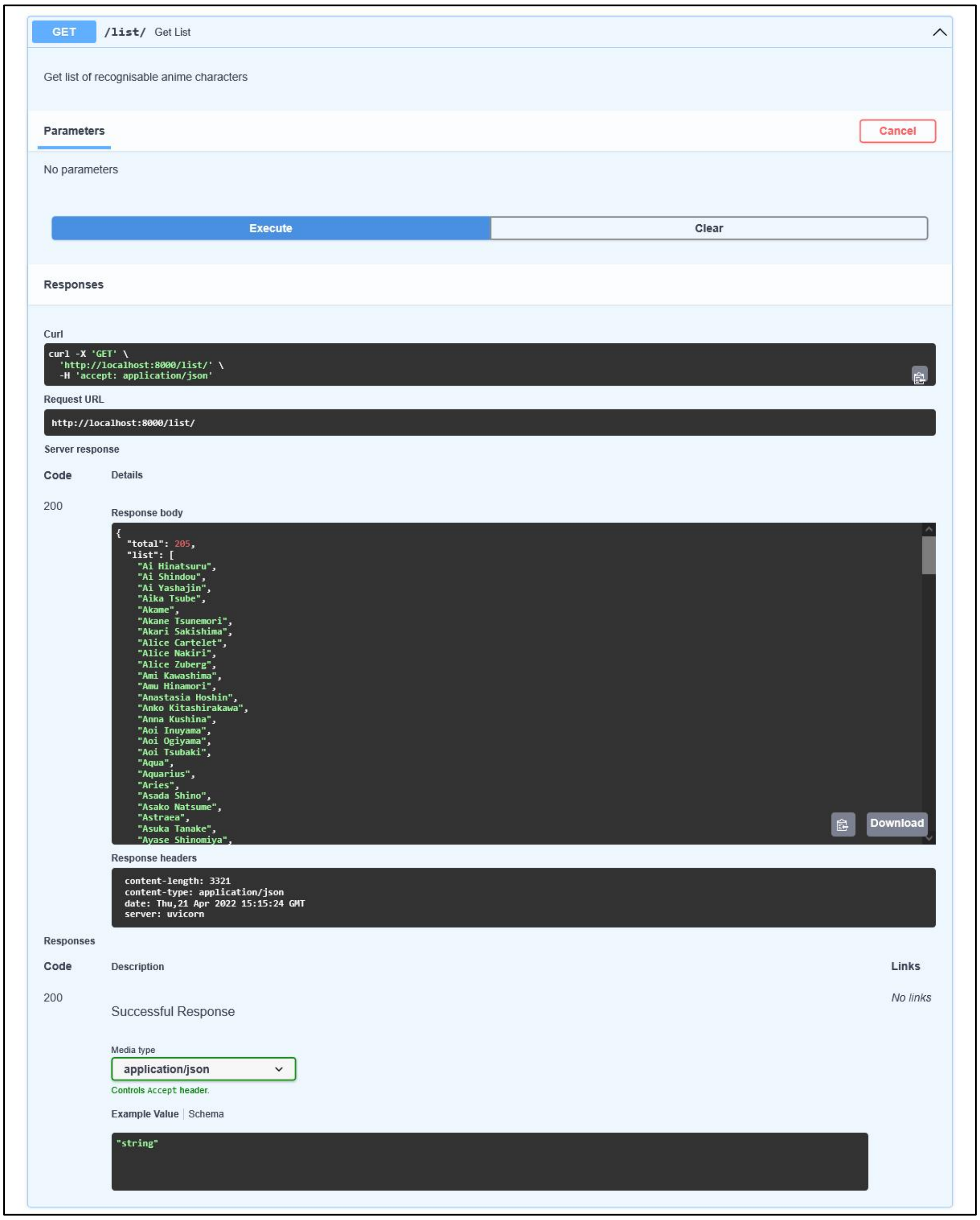
POST /detect_via_api/ Detect Via Api

GET /list/ Get List

POST /detect_test/ Detect Test

Figure 5.2.13: Swagger UI of the app.

All the routes can be access through the Swagger UI. For example,



Get list of recognisable anime characters

No parameters

Curl

```
curl -X 'GET' \
  'http://localhost:8000/list/' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8000/list/
```

Server response

Code | Details

200

Response body

```
{
  "total": 205,
  "list": [
    "Ai Hinatsuru",
    "Ai Shindou",
    "Ai Yashajin",
    "Aika Tsube",
    "Akame",
    "Akane Tsunemori",
    "Akari Sakishima",
    "Alice Cartelet",
    "Alice Nakiri",
    "Alice Zuberg",
    "Ami Kawashima",
    "Amu Hinamori",
    "Anastasia Hoshin",
    "Anko Kitashirakawa",
    "Anna Kushina",
    "Aoi Inuyama",
    "Aoi Ogiyama",
    "Aoi Tsubaki",
    "Aqua",
    "Aquarius",
    "Aries",
    "Asada Shino",
    "Asako Natsume",
    "Astraea",
    "Asuka Tanake",
    "Ayase Shinomiya",
```

Response headers

```
content-length: 3321
content-type: application/json
date: Thu,21 Apr 2022 15:15:24 GMT
server: uvicorn
```

Responses

Code | Description | Links

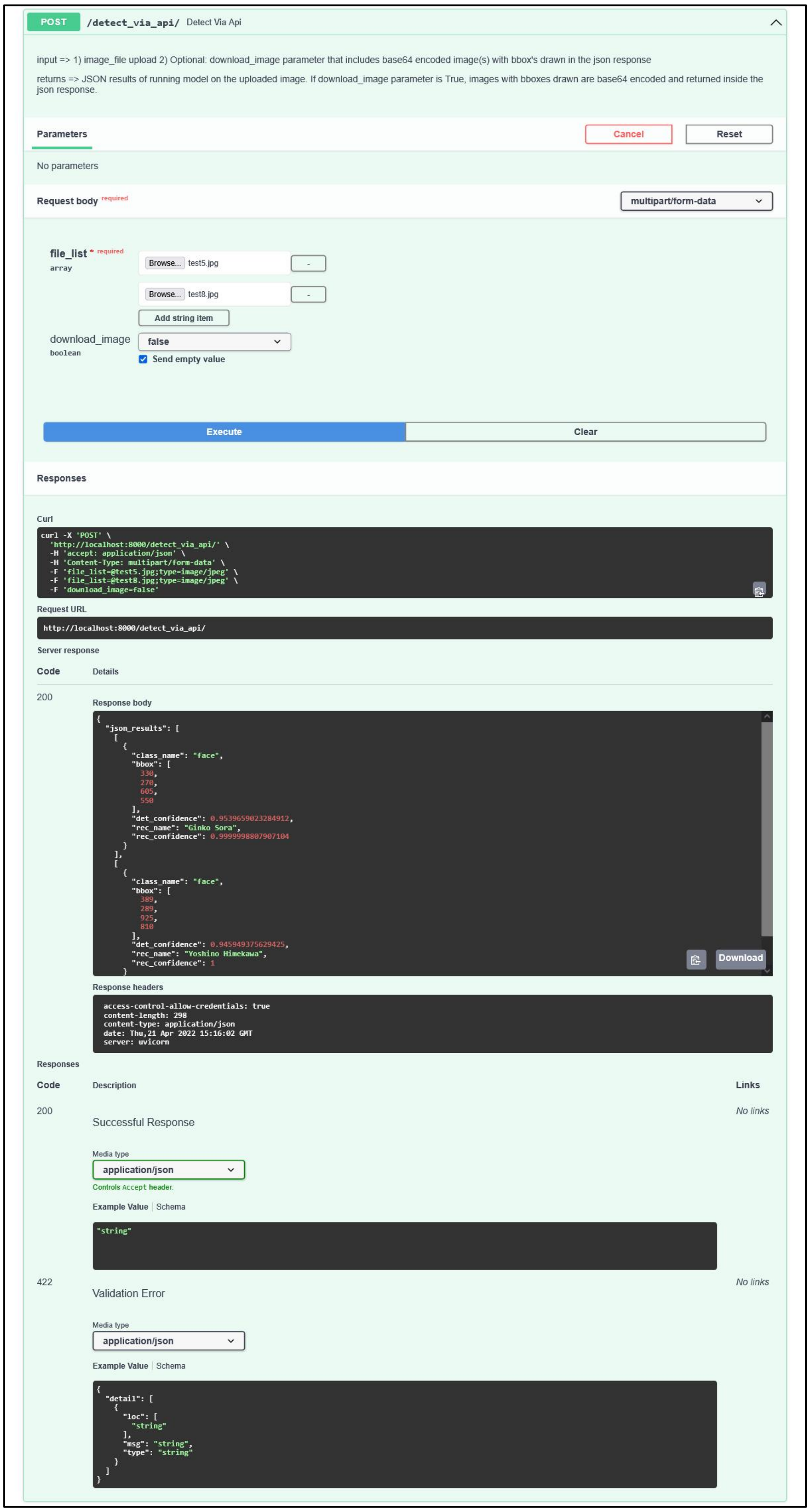
200 | Successful Response | No links

Media type: application/json

Controls Accept header.

```
"string"
```

Figure 5.2.14: Access /list/ in Swagger UI.



/detect_via_api/ Detect Via Api

input => 1) image_file upload 2) Optional: download_image parameter that includes base64 encoded image(s) with bbox's drawn in the json response

returns => JSON results of running model on the uploaded image. If download_image parameter is True, images with bboxes drawn are base64 encoded and returned inside the json response.

Parameters

No parameters

Request body required: multipart/form-data

file_list * required (array): test5.jpg, test8.jpg

download_image (boolean): false

☑ Send empty value

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8000/detect_via_api/' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file_list=@test5.jpg;type=image/jpeg' \
  -F 'file_list=@test8.jpg;type=image/jpeg' \
  -F 'download_image=false'
```

Request URL

```
http://localhost:8000/detect_via_api/
```

Server response

Code: 200

Response body

```
{
  "json_results": [
    [
      {
        "class_name": "face",
        "bbox": [
          330,
          270,
          605,
          550
        ],
        "det_confidence": 0.9539659023284912,
        "rec_name": "Ginko Sora",
        "rec_confidence": 0.9999998807907104
      }
    ],
    [
      {
        "class_name": "face",
        "bbox": [
          389,
          289,
          925,
          810
        ],
        "det_confidence": 0.945949375629425,
        "rec_name": "Yoshino Himekawa",
        "rec_confidence": 1
      }
```

Response headers

```
access-control-allow-credentials: true
content-length: 298
content-type: application/json
date: Thu,21 Apr 2022 15:16:02 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	No links
422	Validation Error	No links

200 Successful Response — Media type: application/json (Controls Accept header.) — Example Value | Schema

```
"string"
```

422 Validation Error — Media type: application/json — Example Value | Schema

```
{
  "detail": [
    {
      "loc": [
        "string"
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

Figure 5.2.15: Access /detect_via_api/ in Swagger UI.

5.2.7 API Deployment

After the backend was tested on the localhost, it was deployed to AWS EC2 virtual machine. However, there is some pre-processing step before proceeding to the server configuration procedure.

CORS Settings

Firstly, Cross-Origin Resource Sharing (CORS) need to be enabled as shown in below code snippet in the FastAPI app.

```
origins = [
    "https://gease-anime.web.app",
    "https://gease-anime.web.app/search",
]

app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

SSL Licence

Furthermore, since the frontend was deployed to Google Firebase with a domain of https://gease-anime.web.app/, so it can only make a request to a remote backend using HTTPS only. If the request is made using HTTP, the browser will block the connection and state that the connection is not secure. Therefore, the backend must be opened to an HTTPS connection. SSL certificates are what enable the web app to move from HTTP to HTTPS.

To get an SSL certificate, the server must be registered to a domain name. Freenom.com is used in this project and a free domain name https://gease-api.ml was obtained. After that, a free SSL certificate was created using ZeroSSL.com. Some verification steps were taken to verify the ownership of the domain.

1 Sign in to the Freenom account.

2 Navigate to the section where DNS records are managed.

3 Add the following CNAME record:

- Name - <a specified name>
- Point To - <a specified url>
- TTL - 3600

4 Save the CNAME record and click "Verity" in ZeroSSL.

5 If verified successfully, the certificate can be downloaded from the website. The package includes certificate.crt, ca_bundle.crt and private.key.

Server Setup

Next, the following steps were taken during the backend deployment in AWS EC2.

Step 1: Upload the project folder to GitHub.

Upload the project to GitHub to ease the project version management. The SSL certificate was uploaded together.

Step 2: Connect the machine using SSH from command prompt.

```
> ssh -i "key.pem" <Public DNS of EC2 instance>
```

Step 3: Git clone the repository to the remote machine.

Step 4: Install dependencies

```
$ cd [to the project directory cloned]
$ sudo apt-get update
$ sudo apt install python3-pip
$ pip3 install -r requirements.txt
```

Step 5: Configure server

NGINX was used to host the backend. Install NGINX:

```
$ sudo apt install nginx
```

Merge the certificate:

```
$ cat certificate.crt ca_bundle.crt >> certificate.crt
```

Move the .crt certificate and private.key to NGINX directory to ease the management:

```
$ cd /etc/nginx
$ sudo mkdir ssl
$ mv /home/project-directory/certificate.crt /home/etc/nginx/ssl/
$ mv /home/project-directory/private.key /home/etc/nginx/ssl/
```

Configure the server:

```
$ cd /etc/nginx/sites-enabled/
$ sudo nano fastapi_nginx
```

Paste the following content into fastapi_ngix. The listen to 443 ssl make the HTTPS connection was enabled in this NGINX server.

```
server {

    listen              443 ssl;


    ssl                 on;
    ssl_certificate     /etc/nginx/ssl/certificate.crt;
    ssl_certificate_key /etc/nginx/ssl/private.key;


    server_name  gease-api.ml;
    access_log   /var/log/nginx/nginx.vhost.access.log;
    error_log    /var/log/nginx/nginx.vhost.error.log;
    location     / {
        proxy_pass http://127.0.0.1:8000;
    }

}
```

After configuration, restart the server:

```
$ sudo service nginx restart
```

Cd to the project directory and run the Uvicorn server.

```
$ cd [project directory]
$ python3 -m uvicorn app:app
```

Screen is required to retain the server running after the SSH connection closes. The last step can be changed to.

```
$ sudo apt install screen
$ cd [project directory]
$ screen
$ python3 -m uvicorn app:app
$ ctrl+a d
```

If every configuration is done successfully, the REST API should now be able to access over HTTPS by **https://gease-api.ml/.**

CHAPTER 6: CONCLUSION AND RECOMMENDATION

6.1 Conclusion

In conclusion, this project developed a mobile app specifically for anime character face detection and recognition. There is currently no mobile app specifically for anime character face detection and recognition on the market. Even a website only supports about a hundred characters and has a single function. The lack of such a mobile will bring great troubles to animation enthusiasts and even outsiders. For anime enthusiasts, it is extremely painful when they see a character but cannot remember who they are and cannot get answers from other channels. From the perspective of a layman, all anime characters are in the same shape, and it is difficult to identify them correctly. Therefore, this mobile app will benefits anime enthusiasts even laymen to easily identify anime characters by just capturing an image using a smartphone. In addition, they can also upload the image via the device photo gallery. The mobile app makes them able to recognise an anime character anytime at anywhere. This not only relieves the pain of anime enthusiasts but also solves the doubts of laymen.

The frontend and backend of the system are completely independent. Any changes on one side will not affect another side. For the frontend, the Ionic framework was utilised to build a cross-platform mobile responsive UI. Functionalities includes results and history viewing as well as app preferences settings. For the backend, it developed using FastAPI web framework. The backend provides the main functionalities of detecting multiples anime face simultaneously from a base64 image and return the corresponding recognising results. The use of FastAPI framework in backend development brings a lot of pros that greatly flexes and expands the value of the app. The app developed can be opened to external connections as a RESTful API and return the detection and recognition results in JSON form.

The detection model used the YOLOv5 architecture to train while the recognition used EfficientNet B0. The number of recognisable characters is up to 205 which are more than doubled compared to most of the existing anime face detection and recognition projects which only involve at most 100 characters. The accuracy achieved 97% and 63.2% Top-1 accuracy for detection and recognition respectively. While for the inference time is 31.9ms and 4.8ms per face detection and recognition respectively.

6.2 Novelty

The most magnificent novel idea of this project is that it will develop the first mobile app specifically for anime character face detection and recognition. There is no such application released before. In addition, the mobile app can detect multiple anime faces simultaneously, which is vary from most of the existing web apps that only can detect one face. Likewise, the recognisable characters have also more than doubled compared to most of the existing anime face detection and recognition projects which only involve at most 100 characters. The accuracy and response time of recognising anime characters will also be significantly improved compared with existing systems.

Furthermore, this project will develop a more beautiful and smoother user interface while having multi-platform adaptability. Apart from that, the mobile app developed in this project is a PWA, which varies from traditional native and hybrid apps that seize the market today. The PWA is flourishing now and highly recommended and progressed by Google. PWA has great potential in the future and it is worth investing in now. The PWA accommodates multiple platforms including web browser, Android and iOS with only one set of codes but maintains most of the functions of native apps. Moreover, the use of FastAPI framework in backend development brings a lot of pros that greatly flexes and expands the value of the app. The app developed can be opened to external connections as a RESTful API and return the detection and recognition results in JSON form.

6.3 Recommendation

Here are some prospects for the future application of anime character face detection and recognition. In fact, anime character face detection and recognition has great potential in the advertising field, although it seems irrelevant. This technology has the potential to replace the Quick Response (QR) code in the advertising field in the future. Imagine that on a colourful and beautiful advertising poster, the existence of a black and white ugly QR code will simply ruin the entire poster. But if one day, the QR code in the advertisement is replaced by cute cartoon characters, a web page can be opened by just recognising the characters with a mobile phone, it will be wonderful.

By doing so, there is no need to train a recognition model with a very large scale dataset since there are millions of anime characters worldwide and can be said hard to include all of them. But if using anime character face detection and recognition in advertising, only a range of characters will be included in the database and will definitely increase the accuracy.

Actually, the current advertising industry is also trying its best to combine AR technology to create more ornamental and attractive advertising effects. If both of these techniques can be combined in the future, it will be the icing on the cake, without any disadvantage.

REFERENCES

[1] Hyanna-Natsu. “Shuki Shuki Doki Doki.” Feb. 14, 2019. Pixiv. https://www.pixiv.net/en/artworks/73168122 (accessed: Mar. 12, 2021).

[2] Starbucks Coffee Company. *Starbucks*. (2021). Accessed: Aug. 26, 2021. [PWA]. Available: https://app.starbucks.com/menu

[3] Google LLC. *Google Lens*. 30 November 2020. (2017). Accessed: Mar. 12, 2021. [Android app]. Available: https://play.google.com/store/apps/details?id=com.google.ar.lens&referrer=utm_source%3Dlanding%26utm_medium%3Ddownload

[4] Laukkonen, J. “What Is Google Lens?” Lifewire. https://www.lifewire.com/google-lens-4153383 (accessed: Mar. 12, 2021).

[5] Sagar, R. “These Machine Learning Techniques Make Google Lens A Success.” Analytics India Magazine. https://analyticsindiamag.com/these-machine-learning-techniques-make-google-lens-a-success (accessed: Mar. 12, 2021).

[6] Malusama. Abyss. https://abyss.malu.moe/#/ (accessed: Mar. 10, 2021).

[7] Fu, J.W., Lee, C.Y., Tao, K.Y. and Zhang, YT. *anime face recognition.* September 30. (2020). Accessed: Mar. 12, 2021. [GitHub repository]. Available: https://github.com/taoky/anime-face-recognition

[8] Milton-Barker, A. “Inception V3 Deep Convolutional Architecture For Classifying Acute Myeloid/Lymphoblastic Leukemia.” Intel.com. https://software.intel.com/content/www/us/en/develop/articles/inception-v3-deep-convolutional-architecture-for-classifying-acute-myeloidlymphoblastic.html (accessed: Mar. 12, 2021)

[9] Setiadi, I. *MoeFlow: Anime Characters Recognition.* 2017. (Alpha Ver.). Accessed: Mar. 14, 2021. [online]. Available: https://freedomofkeima.com/moeflow/

[10] Setiadi, I., “Image Recognition for Anime Characters.” 2017. Accessed: March 14, 2021. [online] Available: https://medium.com/@freedomofkeima/image-recognition-for-anime-characters-e85860dbc4c

[11] Setiadi, I. *Transfer Learning for Anime Characters.* Nov. 18, 2017. (2017). Accessed: Mar. 14, 2021. [GitHub repository]. Available: https://github.com/freedomofkeima/transfer-learning-anime

[12] Wang, Z. H. and Yan, Z.C. *Anime Face Detection and Recognition.* Apr. 3, 2019. (2020). Accessed: Mar. 12, 2021. [GitHub repository]. Available: https://github.com/jonathonyan/Anime-Face-Detection-and-Recognition

[13] Dwiarto W., N., Lamp and Zhou, X. B. *Anime-Face-Detector.* Feb. 2, 2018. (2021). Accessed: Mar. 12, 2021. [GitHub repository]. Available: https://github.com/qhgz2013/anime-face-detector

[14] Ananth, S. “Faster R-CNN for object detection.” Medium. https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46 (accessed: Mar. 15, 2021).

[15] Dwivedi, P. “YOLOv5 compared to Faster RCNN. Who wins?.” Medium. https://towardsdatascience.com/yolov5-compared-to-faster-rcnn-who-wins-a771cd6c9fb4 (accessed: Aug. 26, 2021).

[16] “Models and pre-trained weights.” PyTorch.org. https://pytorch.org/vision/stable/models.html (accessed: Apr 18, 2022).

[17] “Introduction to Service Worker”. Google Developers. https://developers.google.com/web/ilt/pwa/introduction-to-service-worker#what_is_a_service_worker (accessed: Apr. 18, 2022).

[18] Zheng, Y., Zhao, Y. F., Ren, M. Y., Yan, H., Lu, X. J., Liu, J. H. and Li, J. “*Cartoon Face Recognition: A Benchmark Dataset.*” 2019. Accessed: Mar. 12, 2021. [pdf] Available: https://arxiv.org/pdf/1907.13394.pdf

WEEKLY LOG

FINAL YEAR PROJECT BIWEEKLY REPORT
(Project II)

Trimester, Year: 3, 3	**Study week no.:** 1, 2
Student Name & ID: Ng Hui Chin (18ACB02518)	
Supervisor: Dr. Ng Hui Fuang	
Project Title: Anime Character Face Detection and Recognition	

1. WORK DONE

- Start frontend development.
  - Including research on Ionic Framework.
  - Done pages setup and routing.

2. WORK TO BE DONE

- Connect the front and back ends.
- Identify backend problems.

3. PROBLEMS ENCOUNTERED

- Backend encounter problem. The recognition model raises errors. Need to further identify the problems.

4. SELF EVALUATION OF THE PROGRESS

- Remain 30% as the backend encounters problems.

[illegible]
Supervisor's signature

[illegible]
Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: 3, 3	**Study week no.:** 3, 4
Student Name & ID: Ng Hui Chin (18ACB02518)	
Supervisor: Dr. Ng Hui Fuang	
Project Title: Anime Character Face Detection and Recognition	

1. WORK DONE

- Identify backend problems.
  - The recognition model trained with Keras was abandoned.
- Retrain recognition model using PyTorch.

2. WORK TO BE DONE

- Update backend with the latest recognition model and data preprocessing flow.
  - The data preprocessing flow is different from the Keras model.

3. PROBLEMS ENCOUNTERED

- This project has been going on for a long time, many API was updated such as PyTorch. This makes the backend raise errors many times and requires a lot of time to fix it.
- Eventually, I reconfigure the virtual environment in anaconda using the up-to-date PyTorch version but manually modify a file to solve the problem.

4. SELF EVALUATION OF THE PROGRESS

- 40%

[illegible]

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: 3, 3	**Study week no.:** 5, 6
Student Name & ID: Ng Hui Chin (18ACB02518)	
Supervisor: Dr. Ng Hui Fuang	
Project Title: Anime Character Face Detection and Recognition	

1. WORK DONE

- Update backend with PyTorch recognition model.
- Successfully deploy backend to server.
  - o Include server configuration.

2. WORK TO BE DONE

- Connect front and back ends.

3. PROBLEMS ENCOUNTERED

- Heroku and AWS Lambda were not suitable since the model size is exceed the limit.

4. SELF EVALUATION OF THE PROGRESS

- 60%

[illegible]

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: 3, 3	**Study week no.:** 7, 8
Student Name & ID: Ng Hui Chin (18ACB02518)	
Supervisor: Dr. Ng Hui Fuang	
Project Title: Anime Character Face Detection and Recognition	

1. WORK DONE

- Connect front and back ends using REST API.
- POST request and response.

2. WORK TO BE DONE

- Frontend storage configuration
  - to store the returned results from the backend.

3. PROBLEMS ENCOUNTERED

- Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at https://gease-api.ml/detect_via_app/. (Reason: CORS header 'Access-Control-Allow-Origin' missing).
- Solve it by enabling CORS in the backe

4. SELF EVALUATION OF THE PROGRESS

- 70%

[illegible]

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: 3, 3	**Study week no.:** 9, 10
Student Name & ID: Ng Hui Chin (18ACB02518)	
Supervisor: Dr. Ng Hui Fuang	
Project Title: Anime Character Face Detection and Recognition	

1. WORK DONE

- All frontend main functionalities development, such as scan picture and view detection and recognition result.
- Parts of frontend sub functionalities development, such as view history.
- Enhancement of the result returned by the backend, such as reorganising and removing redundant information.
- Add a new backend function, that is, get a recognisable anime character list.

2. WORK TO BE DONE

- The remain sub functionalities of the frontend, such as preference settings and feedback function.

3. PROBLEMS ENCOUNTERED

- Learning a new framework (Ionic React) takes time.

4. SELF EVALUATION OF THE PROGRESS

- 80%

[illegible]
Supervisor's signature

[illegible]
Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT
(Project II)

Trimester, Year: 3, 3	**Study week no.:** 11, 12
Student Name & ID: Ng Hui Chin (18ACB02518)	
Supervisor: Dr. Ng Hui Fuang	
Project Title: Anime Character Face Detection and Recognition	

1. WORK DONE
- Deploy frontend.
- Main functionalities testing.
- All sub functionalities development.

2. WORK TO BE DONE
- Sub functionalities testing.
- Report writing.

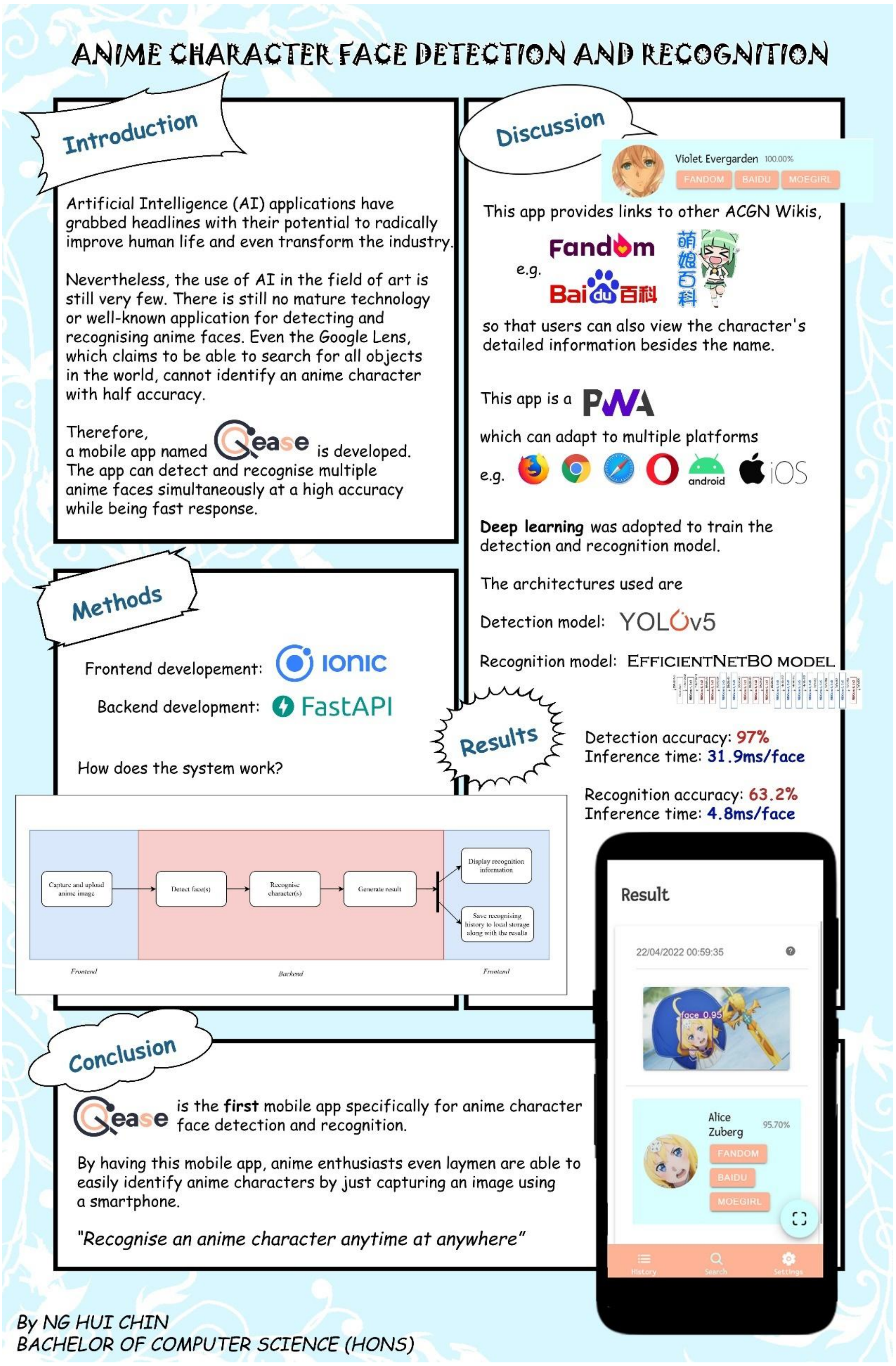
3. PROBLEMS ENCOUNTERED
- The frontend is hosted using firebase, so it is using HTTPS for a secure connection. However, the backend is using HTTP. The connection between frontend and backend was blocked due to an insecure connection.
    - Solved by installing SSL certificate to backend's server and changing connection protocol to HTTPS.

4. SELF EVALUATION OF THE PROGRESS
- 90%

[illegible]
Supervisor's signature

[illegible]
Student's signature

POSTER



ANIME CHARACTER FACE DETECTION AND RECOGNITION

Introduction

Artificial Intelligence (AI) applications have grabbed headlines with their potential to radically improve human life and even transform the industry.

Nevertheless, the use of AI in the field of art is still very few. There is still no mature technology or well-known application for detecting and recognising anime faces. Even the Google Lens, which claims to be able to search for all objects in the world, cannot identify an anime character with half accuracy.

Therefore,
a mobile app named Qease is developed.
The app can detect and recognise multiple anime faces simultaneously at a high accuracy while being fast response.

Methods

Frontend developement: ionic

Backend development: FastAPI

How does the system work?

Capture and upload anime image
Detect face(s)
Recognise character(s)
Generate result
Display recognition information
Save recognising history to local storage along with the results
Frontend
Backend
Frontend

Discussion

This app provides links to other ACGN Wikis,

e.g. Fandom, Baidu百科, 萌娘百科

so that users can also view the character's detailed information besides the name.

This app is a PWA

which can adapt to multiple platforms

e.g. android iOS

Deep learning was adopted to train the detection and recognition model.

The architectures used are

Detection model: YOLOv5

Recognition model: EFFICIENTNETB0 MODEL

Results

Detection accuracy: **97%**
Inference time: **31.9ms/face**

Recognition accuracy: **63.2%**
Inference time: **4.8ms/face**

Conclusion

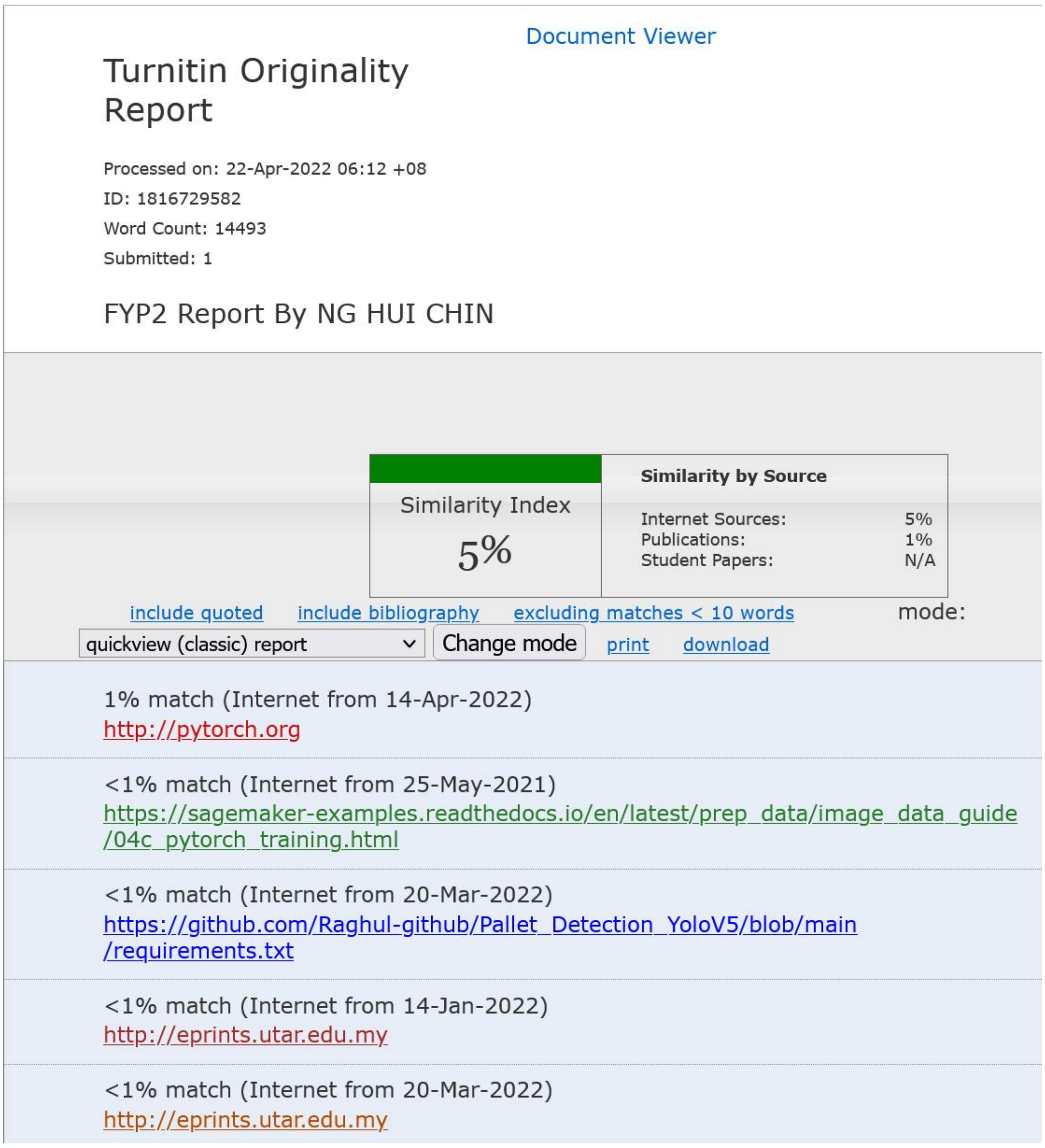
Qease is the **first** mobile app specifically for anime character face detection and recognition.

By having this mobile app, anime enthusiasts even laymen are able to easily identify anime characters by just capturing an image using a smartphone.

"Recognise an anime character anytime at anywhere"

By NG HUI CHIN
BACHELOR OF COMPUTER SCIENCE (HONS)

PLAGIARISM CHECK RESULT



Document Viewer

Turnitin Originality Report

Processed on: 22-Apr-2022 06:12 +08
ID: 1816729582
Word Count: 14493
Submitted: 1

FYP2 Report By NG HUI CHIN

Similarity Index
5%

Similarity by Source

Internet Sources:	5%
Publications:	1%
Student Papers:	N/A

include quoted include bibliography excluding matches < 10 words mode: quickview (classic) report Change mode print download

1% match (Internet from 14-Apr-2022)
http://pytorch.org

<1% match (Internet from 25-May-2021)
https://sagemaker-examples.readthedocs.io/en/latest/prep_data/image_data_guide/04c_pytorch_training.html

<1% match (Internet from 20-Mar-2022)
https://github.com/Raghul-github/Pallet_Detection_YoloV5/blob/main/requirements.txt

<1% match (Internet from 14-Jan-2022)
http://eprints.utar.edu.my

<1% match (Internet from 20-Mar-2022)
http://eprints.utar.edu.my

Form Title: Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date:	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Ng Hui Chin
ID Number(s)	1802518
Programme / Course	Bachelor of Computer Science (HONOURS)
Title of Final Year Project	Anime Character Face Detection and Recognition

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceed the limits approved by UTAR)**
Overall similarity index: 5 % **Similarity by source** Internet Sources: 5 % Publications: 1 % Student Papers: N/A %	
Number of individual sources listed of more than 3% similarity: -	
Parameters of originality required, and limits approved by UTAR are as Follows: **(i) Overall similarity index is 20% and below, and** **(ii) Matching of individual sources listed must be less than 3% each, and** **(iii) Matching texts in continuous block must not exceed 8 words** ***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***	

Note: Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

[illegible]

Signature of Supervisor	Signature of Co-Supervisor
Name: Ng Hui Fuang	Name:
Date: 22/04/2022	Date:

FYP 2 CHECKLIST

UNIVERSITI TUNKU ABDUL RAHMAN
FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	18ACB02518
Student Name	Ng Hui Chin
Supervisor Name	Dr. Ng Hui Fuang

TICK (√)	DOCUMENT ITEMS Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
	Front Plastic Cover (for hardcopy)
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
-	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
-	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

[illegible]
(Signature of Student)
Date: 22/04/2022