IKEA FURNITURE FINDER

By

Tan Meng Sheng

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR INFORMATION SYSTEMS (HONOURS)

INFORMATION SYSTEMS ENGINEERING

Faculty of Information and Communication Technology

(Kampar Campus)

JANUARY 2023

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: IKEA FURNITURE FINDER

Academic Session: JANUARY 2023

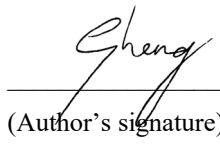
I TAN MENG SHENG

(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,



(Author's signature)

(Supervisor's signature)

Address:
12, JALAN BESAR 2,
TAMAN JOHOR,
81200 JB, JOHOR

DR. AUN YICHIET
Supervisor's name

Date: 28 APRIL 2023

Date: 28 APRIL 2023

DECLARATION OF ORIGINALITY

I declare that this report entitled "**IKEA FURNITURE FINDER**" 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 : 

Name : TAN MENG SHENG

Date : 28 APRIL 2023

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere gratitude to my project supervisor, Dr. Aun YiChiet, for his unwavering support and invaluable guidance throughout this project. His expertise and knowledge in machine learning, modelling, and new technology such as Google Cloud Platform, have been crucial in the successful completion of this project. Dr. Aun has provided me with a bright opportunity to engage in this project, and I am grateful for his trust and belief in my abilities.

I would also like to extend my heartfelt thanks to my supportive friends, Khoo Qing Nian and Khoo Jia Hong, who have been a constant source of motivation and encouragement throughout this journey. Their support and constructive feedback have helped me immensely in overcoming the challenges faced during this project. Additionally, I would like to express my sincere appreciation to my family members for their patience and understanding throughout this entire process.

The Ikea Furniture Finder project has been a significant learning experience for me, and I am grateful for the skills and knowledge that I have gained through this project. The successful completion of this project would not have been possible without the support and contributions of my supervisor, friends, and family members. I look forward to applying the skills and knowledge gained from this project in my future endeavors, and I am excited about the possibilities that lie ahead.

ABSTRACT

The aim of this project was to develop an app that provides users with furniture recommendations based on the category and color of the furniture they are interested in. Deep learning techniques were used to train a TensorFlow model to accurately classify images of furniture. The model was trained on a large dataset of furniture images that were collected and labeled using a PowerShell script for automatic dataset labeling.

A Flask web application was built using this model to predict the category of furniture in images sent by the Android client app. Additionally, a REST API endpoint was implemented in Flask to retrieve random furniture images from Firebase, which were used to display recommendations to users.

To ensure scalability and consistency across environments, the Flask app was deployed on a cloud platform using Docker. User testing was conducted to evaluate the accuracy and usability of the app, and feedback was solicited from users to identify areas for improvement. Overall, the results demonstrate that the Ikea Furniture Finder app is an effective tool for assisting users in finding furniture based on their preferences.

TABLE OF CONTENTS

REPORT STATUS DECLARATION FORM ..... **i**

DECLARATION OF ORIGINALITY ..... **ii**

ACKNOWLEDGEMENTS ..... **iii**

ABSTRACT ..... **iv**

TABLE OF CONTENTS ..... **v**

LIST OF FIGURES ..... **ix**

LIST OF TABLES ..... **xiv**

LIST OF ABBREVIATIONS ..... **xv**

CHAPTER 1: INTRODUCTION **1**

1.1. Problem Statement and Motivation 1

1.2. Objective 3

1.3. Project Scope 4

1.4. Impact, significance and contribution 5

1.5. Report Organization 6

CHAPTER 2: LITERATURE REVIEW **7**

2.1. Review of the Technologies 7

2.1.1. Hardware Platform 7

2.1.2. Software Platform 9

2.1.3. Firmware/OS 16

2.1.4. Database 16

2.1.5. Programming Language 17

2.1.6. Algorithm 18

2.1.7. Software Libraries and Frameworks Used in Machine Learning 19

2.1.8. Summary of the Technologies Review 20

2.2. Previous Works on Deep Learning 22

2.2.1. Gillette 22

2.2.2. Tile 24
2.2.3. Black Rifle Coffee 27
2.2.4. Hello Fresh 30
2.2.5. DavidsTEA 33
2.2.6. Limitation of Previous Studies 36
2.2.7. The Summarized Comparison of Existing Works with Proposed Project 37
CHAPTER 3: SYSTEM METHODOLOGY/ APPROACH 38
3.1. System Design Diagram/Equation 38
3.1.1. System Architecture Diagram 38
3.1.2. Use Case Diagram and Description 39
3.1.3. Activity Diagram 40
3.1.4. CNN Architecture Visualization 42
3.2. Timeline 43
CHAPTER 4: SYSTEM DESIGN 44
4.1. System Block Diagram 44
4.2. System Components Specifications 46
4.3. System Components Interaction Operations 49
CHAPTER 5: SYSTEM IMPLEMENTATION 51
5.1. Hardware Setup 51
5.2. Software Setup 53
5.3. Image Dataset Preparation 54
5.4. Setting and Configuration 63
5.4.1. Anaconda Jupyter Notebook 63
5.4.2. PyCharm 70
5.4.3. Docker 75
5.4.4. Google Cloud Platform 77
5.4.5. Firebase 79

5.4.6. Android Studio 82
5.5. System Operation (with Screenshot) 99
5.5.1. Uploading Docker Container to Google Cloud Run 100
5.5.2. Linking Firebase with Google Cloud Storage Bucket 101
5.5.3. Integrating Google Cloud Platform and Firebase with Android Studio 106
5.6. Implementation Issues and Challenges 110
5.7. Concluding Remark 112
CHAPTER 6: SYSTEM EVALUATION AND DISCUSSION 113
6.1. System Testing and Performance Metrics 113
6.1.1. Results of the h5 model 113
6.2. Testing Setup and Result 115
6.2.1. Validate the h5 model in Anaconda Jupyter Notebook 115
6.2.2. Validate the h5 model in Pycharm 118
6.2.3. Validate the GCP Cloud Run URL in Android Studio 120
6.2.4. Validate the Firebase in Android Studio 122
6.2.5. Validate the User Experience 125
6.3. Project Challenges 126
6.4. Objectives Evaluation 127
6.5. Concluding Remark 128
CHAPTER 7: CONCLUSION 129
7.1. Conclusion 129
7.2. Recommendation 130
REFERENCES 131
BIWEEKLY LOG 133
POSTER 139
PLAGIARISM CHECK RESULT 140
FORM IAD-FM-IAD-005 143

FYP 2 CHECKLIST 144

LIST OF FIGURES

Figure 2.1.1.1 Google Cloud Platform Buckets 8
Figure 2.1.1.2 Google Cloud Run Services 8
Figure 2.1.2.1 Anaconda 2021.11 9
Figure 2.1.2.2 Android Studio Bumblebee 10
Figure 2.1.2.3 Docker 11
Figure 2.1.2.4 Postman send request 12
Figure 2.1.2.5 Google Cloud get request 12
Figure 2.1.2.6 PyCharm main.py 13
Figure 2.1.2.7 PyCharm test.py 13
Figure 2.1.2.8 Google Cloud Run 14
Figure 2.1.2.9 Google Cloud Storage 14
Figure 2.1.2.10 Magick Command 15
Figure 2.2.1.1 Gillette provides the select option 22
Figure 2.2.1.2 Chatbot will recommend the product to clients 22
Figure 2.2.1.3 Chatbots provide links to navigate to related webpage 23
Figure 2.2.1.4 Before large fonts 23
Figure 2.2.1.5 After larger fonts 23
Figure 2.2.2.1 Welcome message of Veronica 24
Figure 2.2.2.2 Click the link will navigate to new webpage about the related product 24
Figure 2.2.2.3 Clients can choose to contact customer care 25
Figure 2.2.2.4 chatbot provide suggestion 25
Figure 2.2.2.5 Tile can send transcript via email 26
Figure 2.2.3.1 Welcome message of Black Rifle Coffee 27
Figure 2.2.3.2 Allow clients to edit subscription 27
Figure 2.2.3.3 Clients can select a question from the option 28
Figure 2.2.3.4 Request for customer service 28
Figure 2.2.3.5 UI while waiting for a real agent 28
Figure 2.2.3.6 Turn to the real agent 29
Figure 2.2.3.7 Coffee Quiz 29
Figure 2.2.4.1 Welcome message 30
Figure 2.2.4.2 User query 31
Figure 2.2.4.3 Chatbot reply 31

Figure 2.2.4.4 Menu button to download the chat 31
Figure 2.2.4.5 Chatbot will help clients to download the chat script among the chat 32
Figure 2.2.4.6 Unable to reply to the query 32
Figure 2.2.5.1 Select Language 33
Figure 2.2.5.2 Welcome Message of Davi 33
Figure 2.2.5.3 Support picture reply 34
Figure 2.2.5.4 Able to get transcript after a session 34
Figure 2.2.5.5 Main menu of DAVI 35
Figure 2.2.5.6 Time remaining for log out 35
Figure 3.1.1.1 System Architecture Diagram 38
Figure 3.1.2.1 Use Case Diagram 39
Figure 3.1.3.1 Activity Diagram 40
Figure 3.1.4.1 CNN Architecture Visualization for Model2_Train20 42
Figure 3.1.4.1 Timeline 43
Figure 3.1.4.1 System Block Diagram 44
Figure 3.1.4.1 System Components Interaction Operations 49
Figure 3.1.4.1 Device Manager 51
Figure 3.1.4.2 GeForce GTX 1650 52
Figure 3.1.4.3 Download 52
Figure 3.1.4.4 Finished Install 52
Figure 3.1.4.1 make directory command for 20 categories 54
Figure 3.1.4.2 rename bat 54
Figure 3.1.4.3 Character that need to be replaced 55
Figure 3.1.4.4 Each ID with the series name 55
Figure 3.1.4.5 make directory command 55
Figure 3.1.4.6 Rename command 56
Figure 3.1.4.7 PowerShell Command to retrieve directory 56
Figure 3.1.4.8 All of the directory list is pasted in Excel Sheet 57
Figure 3.1.4.9 Whole dataset quantity 57
Figure 3.1.4.10 Remove the main folder address 58
Figure 3.1.4.11 Copy the same column from A to H column 58
Figure 3.1.4.12 Remove the id after two underscore for opening the file 59
Figure 3.1.4.13 Script for making directory 59
Figure 3.1.4.14 Scripting in PowerShell 60

Figure 3.1.4.15 Folder that made by scripting 60
Figure 3.1.4.16 Script of move the dataset base on each of the file 60
Figure 3.1.4.17 Scripting in PowerShell 60
Figure 3.1.4.18 Move the directory to less layer 61
Figure 3.1.4.19 Final image dataset 61
Figure 3.1.4.20 resize command 62
Figure 3.1.4.21 Resized image 62
Figure 5.4.1.1 Separated file 65
Figure 5.4.1.2 Trained model had been saved 67
Figure 5.4.1.3 Validation for h5 model 69
Figure 5.4.2.1 Test the accuracy by local URL 70
Figure 5.4.3.1 Open command prompt in particular directory 75
Figure 5.4.3.2 Build the docker image in Docker 75
Figure 5.4.3.3 Uploaded Docker Image 76
Figure 5.4.3.4 Create a container 76
Figure 5.4.3.5 List the file in src file 76
Figure 5.4.4.1 Create new project 77
Figure 5.4.4.2 Define project name 77
Figure 5.4.4.3 Created project 78
Figure 5.4.4.4 Create new service for Google Cloud Run 78
Figure 5.4.4.5 Create new service for Cloud Storage 78
Figure 5.4.5.1 Step 1 to create Firebase project 79
Figure 5.4.5.2 Step 2 to create Firebase project 80
Figure 5.4.5.3 Step 3 to create Firebase project 80
Figure 5.4.5.4 Success to create Firebase project 80
Figure 5.4.5.5 Firebase storage is created 81
Figure 5.4.5.6 Furniture set 2 in Firebase 81
Figure 5.4.6.1 Android Studio interface 84
Figure 5.4.6.2 IkeaLogo.png 86
Figure 5.4.6.3 ikeaapp.png 86
Figure 5.4.6.4 background_image_1.png 86
Figure 5.4.6.5 background_image_2.png 86
Figure 5.4.6.6 background_image_3.png 86
Figure 5.4.6.7 background_image_4.png 86

Figure 5.4.6.8 background_image_5.png 86
Figure 5.4.6.9 background_image_6.png 86
Figure 5.4.6.10 ikealaunch.png 86
Figure 5.4.6.11 Splash Screen 87
Figure 5.4.6.12 Background Picture demo 1 87
Figure 5.4.6.13 Background Picture demo 2 87
Figure 5.4.6.14 Picture will change after inserting image 87
Figure 5.4.6.15 The image will fit the original orientation 88
Figure 5.4.6.16 After clicking submit will show transparent grey overlay 88
Figure 5.4.6.17 Will show the type of Furniture and show image from Firebase 88
Figure 5.4.6.18 Can select either image and bigger the image 88
Figure 5.5.1.1 upload the file from Docker Container to Google Cloud 100
Figure 5.5.1.2 Deployed successfully on google cloud 100
Figure 5.5.1.3 Google cloud appeared deployed model 100
Figure 5.5.2.1 Get Started 101
Figure 5.5.2.2 Set up Secure Rules 102
Figure 5.5.2.3 Set up Location 102
Figure 5.5.2.4 Creating bucket 103
Figure 5.5.2.5 Add Bucket to GCP 103
Figure 5.5.2.6 Put related detail to the blank 104
Figure 5.5.2.7 Click create to create 104
Figure 5.5.2.8 Successfully added to GCP 105
Figure 5.5.3.1 Click the Add App 106
Figure 5.5.3.2 Put Ikea furniture finder detail and register 106
Figure 5.5.3.3 Download JSON file and then add config file 107
Figure 5.5.3.4 Add JSON file to Android Studio 107
Figure 6.2.1.1 Prediction Output for Image 117
Figure 6.2.2.1 Test the accuracy by local URL 118
Figure 6.2.2.2 Test the accuracy by GCP Cloud Run URL 119
Figure 6.2.3.1 Result of GCP Cloud Run URL in Android Studio 121
Figure 6.2.3.2 Logs of GCP Cloud Run URL in Android Studio 121
Figure 6.2.4.1 Output of retrieve all JPEG URL Firebase in Android Studio 123
Figure 6.2.4.2 Output of the Firebase in Android Studio 123
Figure 6.2.5.1 Survey Question 1 125

Figure 6.2.5.2 Survey Question 2 125

LIST OF TABLES

Table 2.1.1.1 Specifications of laptop 7
Table 2.1.7.1 Software Libraries and Frameworks Used.....19
Table 2.2.7.1 Summarized Comparison of Existing Works with Proposed Project.....37
Table 3.1.3.1 Software setup table.....53
Table 5.4.1.1 Code and the modules or libraries used for Anaconda Jupyter Notebook.....63
Table 5.4.6.1 Packages and Settings Listed in the dependencies of pubspec.yaml.....82
Table 5.4.6.2 Set up for pubspec.yaml.....85
Table 6.1.1.1 Furniture Model Performance.....114
Table 6.1.1.2 Failed Furniture Model Performance.....114

LIST OF ABBREVIATIONS

RNN	Recurrent Neural Network
AI	Artificial Intelligence
NLP	Natural Language Processing
NLU	Natural-language understanding
MD	Make directory
REN	Rename
CNN	Convolutional Neural Network
CV	Computer Vision
GCP	Google Cloud Platform

CHAPTER 1: INTRODUCTION

1.1. Problem Statement and Motivation

This project is aimed to create an application that allows users to take a picture of furniture or select an image from their gallery, and then receive suggestions for similar furniture pieces. The motivation behind this project is to provide users with a more seamless way to discover furniture and home decor items they may be interested in. Instead of having to manually search for similar items, this app will use machine learning and image recognition techniques to automatically suggest similar items based on the user's input.

To address the challenge of deploying the furniture classification model on a cloud-based platform, this project uses Google Cloud Run to deploy the model and Firebase to store images. Google Cloud Run is a fully managed compute platform that automatically scales containers, while Firebase provides cloud storage for user images. Once the model is deployed on Google Cloud Run, the mobile application can use it to classify user-uploaded images. The output generated by the model is then used to select related images from Firebase to suggest to the user.

One of the key challenges of this project faced in this project was building a high-quality deep learning model. In the past decade, researchers have proposed and demonstrated various approaches for detecting and tracking moving objects. Although object detection has significantly improved with the development of deep convolutional neural networks (CNNs), accurately estimating the object's position remains a major challenge. Object detection involves identifying real-world objects like table, chair, sofa, and more from images. Once objects are detected, the goal of object classification is to distinguish them from the background and assign them to their respective categories. This is achieved by focusing on the most distinguishing features of objects, which are used to accurately predict their category [1]–[3].

The project aims to provide a seamless user experience, where users can easily upload images of furniture items and receive suggestions for related furniture items based on the model's classification output. This project demonstrates the potential of machine learning and image recognition techniques to streamline the furniture shopping process for users. By building a deep learning model and deploying it on Google Cloud Run, this project able to provide accurate and real-time suggestions to users, while also leveraging the power and scalability of cloud computing. The Flutter mobile application provided a user-friendly interface for submitting images and receiving suggestions, and Firebase provided a reliable and secure way to store and manage user data. Overall, this project serves as a proof-of-concept for using machine learning in e-commerce applications and provides a foundation for future work in this area.

1.2. Objective

The goal of this research is to develop an Ikea Furniture Finder. Although consumers have their own needs and inquiries, they seek product recommendations as alternatives to out-of-stock items. The project's main objectives are as follows:

- To develop PowerShell scripting for automatic dataset labelling.
- To develop a machine learning pipeline that can recommend similar furniture based on category and colour.
    - To develop a machine learning model to accurately classify images of furniture based on their category.
    - To evaluate the accuracy and usability of the app by conducting user testing and soliciting feedback to identify areas for improvement.
- To implement the Cloud-Based Deployment and Efficient Image Storage for Ikea Furniture Finder
    - To deploy the ML model and app on a cloud-based platform, such as Google Cloud Run, to allow for easy scaling and management.
    - To implement image storage and retrieval using Firebase, allowing the app to efficiently manage and access a large library of furniture images.
- To create a mobile application that allows users to take photos of furniture or select images from their device's gallery and receive furniture recommendations based on the ML model's classification results.

1.3. Project Scope

The title of this project is Ikea Furniture Finder, this project aims to develop a mobile application using Flutter that allows users to capture or upload images of furniture items and receive suggestions for similar products based on category. The application will incorporate a deep learning model (CNN Model) trained on a labelled dataset of furniture images and deployed on a cloud-based platform such as Google Cloud Run. Firebase will be used for image storage and retrieval to efficiently manage and access a large library of furniture images. The project scope will involve developing PowerShell scripting for automatic dataset labelling, designing, and implementing a machine learning pipeline for recommendation, creating a Flutter-based mobile application, and deploying the application and model on the cloud. The success of the project will be evaluated based on the accuracy of image recognition, the efficiency of image retrieval, and the overall usability of the mobile application.

1.4. Impact, significance and contribution

The impact of this project is significant as it addresses the need for an efficient and effective way to search for furniture items. With the Ikea Furniture Finder app, users can easily find similar furniture items based on category and color, saving them time and effort. This app can be especially useful for those who are not familiar with Ikea's products or are unable to find what they need in stores. Additionally, the app's ability to suggest alternative items can help to increase customer satisfaction and loyalty, leading to potential business growth for Ikea.

Furthermore, this project contributes to the development of machine learning and mobile app technology. The use of machine learning algorithms in the app's backend allows for accurate and efficient furniture item recommendation, while the use of mobile app technology in the front-end provides users with a convenient and accessible way to search for and discover furniture items. The project also showcases the potential of cloud-based platforms such as Google Cloud Run and Firebase in the deployment and management of machine learning models and image databases. As such, this project has the potential to inspire further research and development in these fields.

1.5. Report Organization

The report is organized into seven chapters, each with a specific focus. Chapter 1 provides an introduction to the project, including the problem statement, objective, project scope, and significance of the study. Chapter 2 is a literature review that examines the relevant technologies and previous works on deep learning.

Chapter 3 outlines the system methodology and approach, including the system design diagram and timeline. Chapter 4 focuses on the system design, including the system block diagram, system components specifications, and system components interaction operations.

Chapter 5 discusses the system implementation, covering the hardware and software setup, image dataset preparation, system setting and configuration, and system operation. This chapter also addresses implementation issues and challenges and concludes with a remark.

Chapter 6 delves into system evaluation and discussion, including system testing and performance metrics, testing setup and results, project challenges, objectives evaluation, and concluding remarks.

Finally, Chapter 7 concludes the report with a summary of the findings, recommendations for future research, and overall conclusions.

CHAPTER 2: LITERATURE REVIEW

2.1. Review of the Technologies

2.1.1. Hardware Platform

- **Acer Nitro AN515-56**

Acer Nitro AN515-56 laptop runs on Windows 10 Home, a 64-bit operating system, which is upgradable. The processor is an 11th Gen Intel® Core™ i5-11300H quad-core processor that operates at a speed of 3.10 GHz. The laptop uses an NVIDIA® GeForce® GTX 1650 graphics controller with 4 GB of dedicated GDDR6 graphics memory[4].

Table 2.1.1.1 Specifications of laptop

Description	Specifications
Model	Acer Nitro AN515-56
Processor	Intel® Core i5-11300H
Operating System	Windows 10
Graphic	NVIDIA GeForce GTX 1650 4GB GDDR6
Memory	8GB 3200Mhz DDR4 RAM
Storage	512GB SATA SSD

- **NVIDIA GeForce GTX 1650 graphics processing unit (GPU)**

The hardware platform used for training the machine learning model is an NVIDIA GeForce GTX 1650 graphics processing unit (GPU). The GPU is based on the Turing architecture and has a boost clock range of 1245 - 1560 MHz. It has 896 CUDA cores, 56 texture mapping units (TMUs), and 32 render output units (ROPs). The GPU also has 4 GB of GDDR5 memory with a bus width of 128 bits and a memory speed of 8 Gbps. The NVIDIA GeForce GTX 1650 is a mid-range GPU that is suitable for machine learning tasks and is widely used in the field. It provides good performance for the price and can handle most deep learning tasks efficiently[5].

- **Google Cloud Platform**

In this project, the machine learning model will be deployed on Google Cloud Run, a fully managed platform for containerized applications. The trained model will be stored in a Google Bucket, which provides scalable storage for unstructured data. Based on [6], both Google Cloud Run and Google Bucket are hosted on Google Cloud Platform servers located in the United States. The choice of Google Cloud Platform was based on several factors, including its reliability, scalability, and cost-effectiveness. By leveraging the power of cloud computing, this project can take advantage of the high-performance infrastructure and resources provided by Google Cloud Platform, while also providing easy deployment and management of the machine learning model.

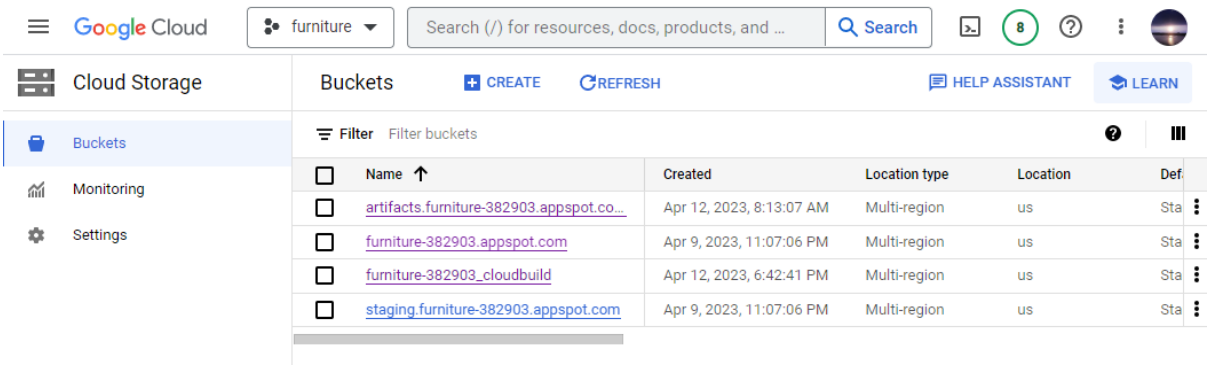


Figure 2.1.1.1 Google Cloud Platform Buckets

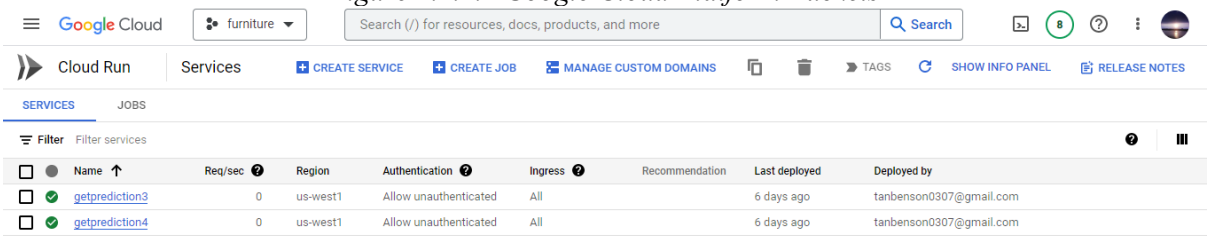


Figure 2.1.1.2 Google Cloud Run Services

2.1.2. Software Platform

- **Anaconda 2021.11**

Anaconda 2021.11 is a popular Python distribution that includes a wide range of scientific computing and data science libraries. It provides an easy-to-use interface for managing packages and environments, allowing users to easily switch between different versions of Python and different package configurations. With Anaconda, Ikea Furniture Finder set up a development environment for their Python projects and use various tools for data analysis, machine learning, and other tasks. Anaconda provides users with access to a vast array of libraries and tools for data analysis, including NumPy, Pandas, Matplotlib, Scikit-learn, and TensorFlow, which are commonly used for building machine learning models. Overall, Anaconda is a valuable tool for data scientists, researchers, and developers working with Python.

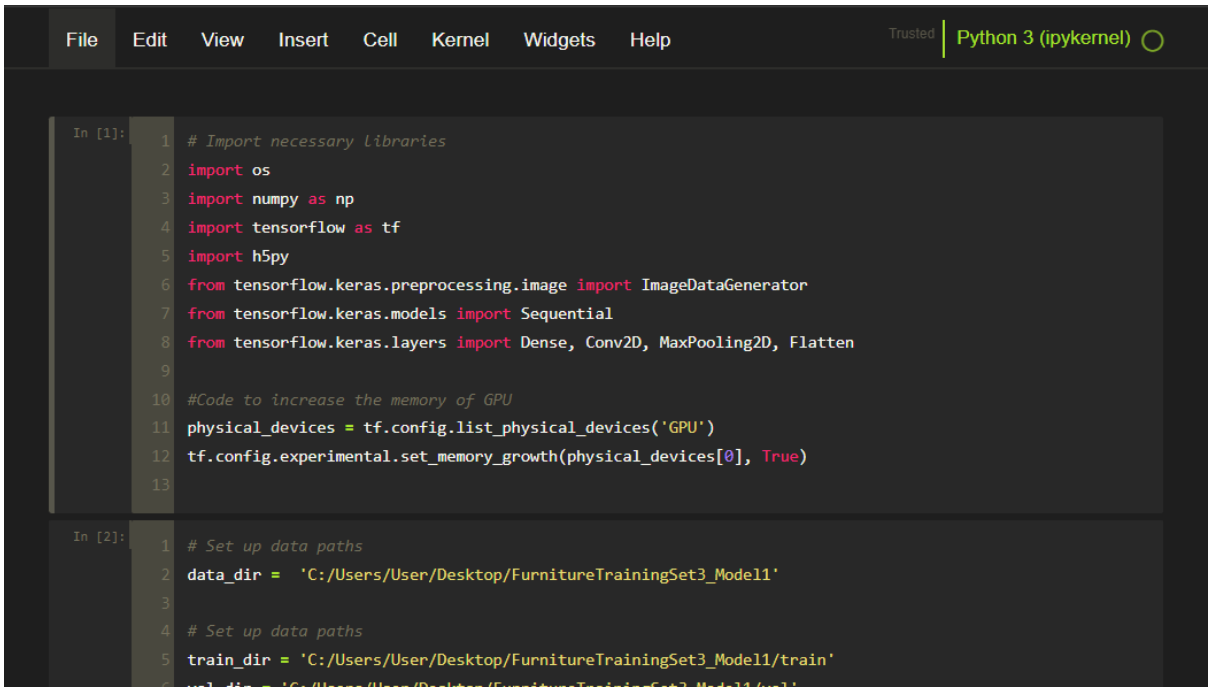


Figure 2.1.2.1 Anaconda 2021.11

- **Android Studio Bumblebee | 2021.1.1**

Android Studio Bumblebee | 2021.1.1 is a popular integrated development environment (IDE) for Android app development. It provides a comprehensive set of tools and features for developing, debugging, and testing Android apps. Android Studio includes an intelligent code editor, a visual layout editor, a powerful emulator, and various tools for profiling and analyzing app performance. With Android Studio, developers can easily create high-quality Android apps that can run on a wide range of devices. The backend then sends a request to the specified URL (https://getprediction3-biptoq2fwq-uw.a.run.app) to receive a JSON string containing the predicted class label ('prediction') for the uploaded image. The predicted class label is then used to specify the Firebase Storage location for the uploaded image, which is stored using the FirebaseStorage.instance and a reference to the specified folder.

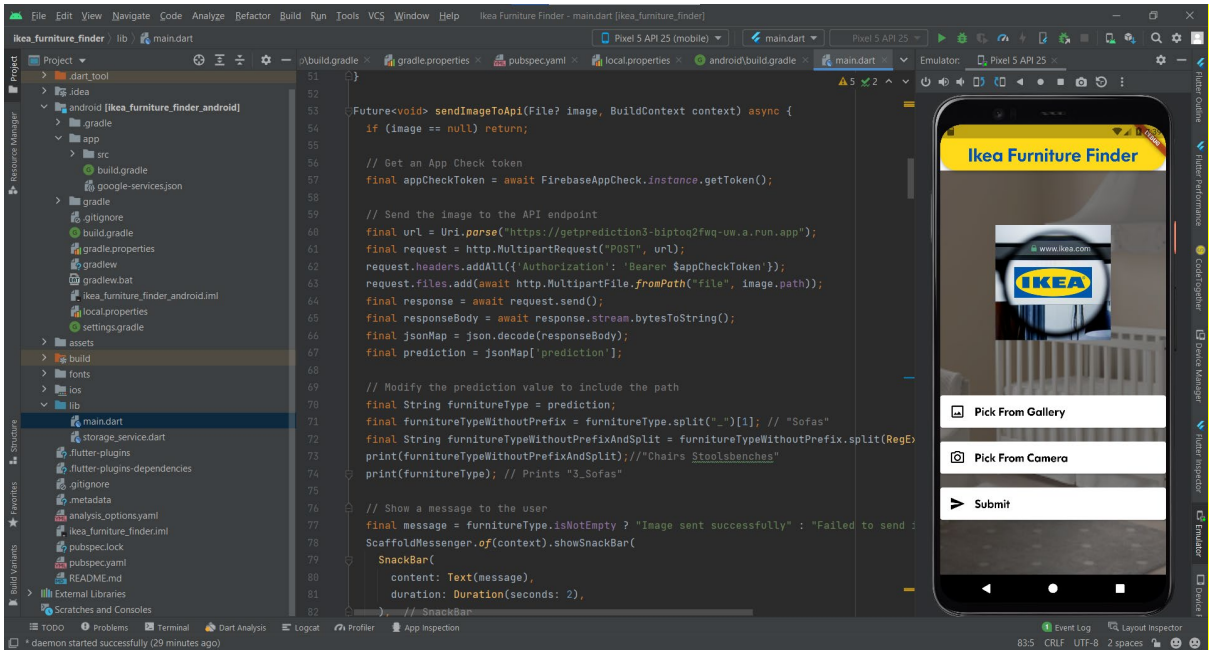


Figure 2.1.2.2 Android Studio Bumblebee

- **Docker 4.18.0**

Docker 4.18.0 is a platform that allows developers to create, deploy, and run applications in a containerized environment. Docker provides an easy way to package and distribute applications as containers, which are lightweight, portable, and isolated from the host system. With Docker, this project set up development and production environments, manage dependencies, and deploy applications to any platform or cloud provider. Containers are lightweight, portable, and self-contained environments that allow applications and their dependencies to run consistently across different platforms and environments. Docker provides a way to package and distribute applications in a standardized and efficient manner, making it easier for developers to deploy their applications in a variety of settings. This project used Docker to create a container that runs your Flask web application, which serves a trained TensorFlow model for image classification. By using Docker, you can ensure that your application runs consistently and reliably, regardless of the underlying system or environment.

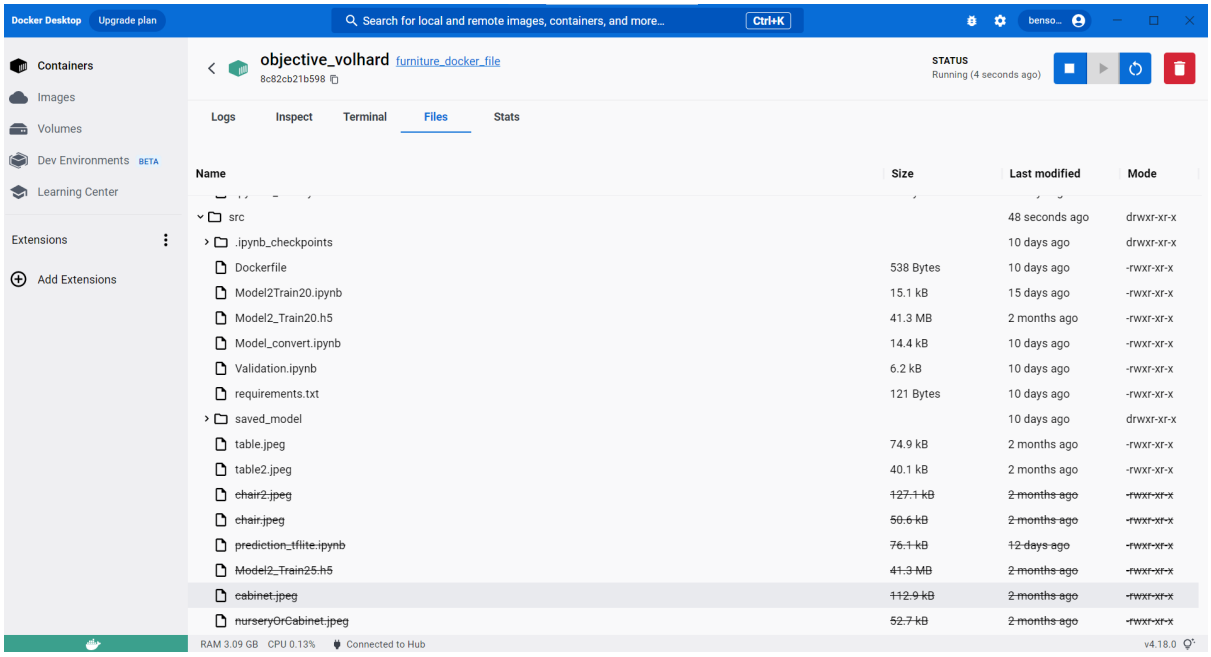


Figure 2.1.2.3 Docker

- **Postman 10.12.0**

Postman 10.12.0 is a popular API testing tool that allows users to send and receive HTTP requests and responses. It provides an easy-to-use interface for creating, testing, and managing APIs, making it a popular choice among developers and testers. With Postman, this project can create collections of API requests, and automate tests to ensure that APIs are working as expected. In this project, Postman is used to test the availability of a web server by sending an HTTP POST request to the server and checking the response. This is a common use case for Postman, as it allows developers to quickly test and debug APIs before deploying them to production.

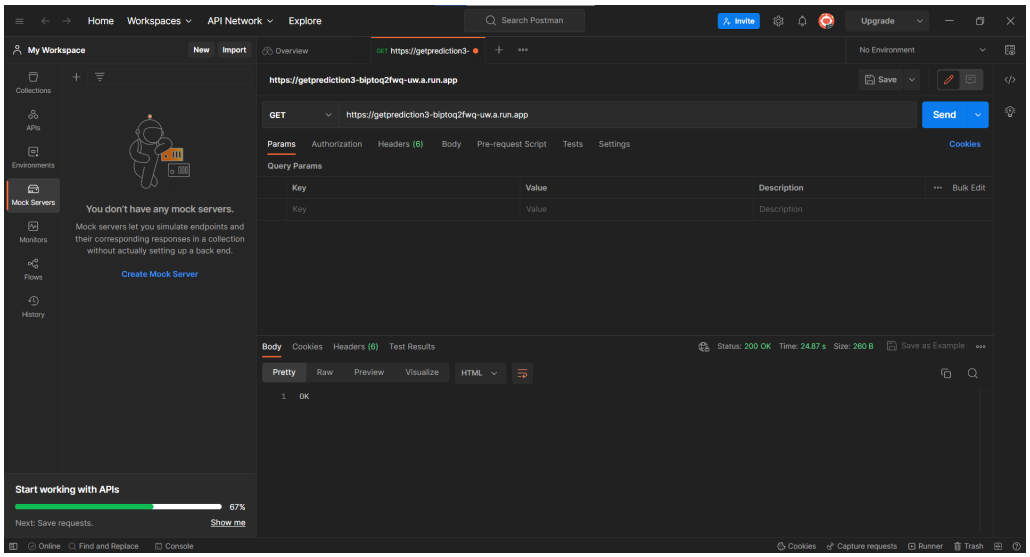


Figure 2.1.2.4 Postman send request

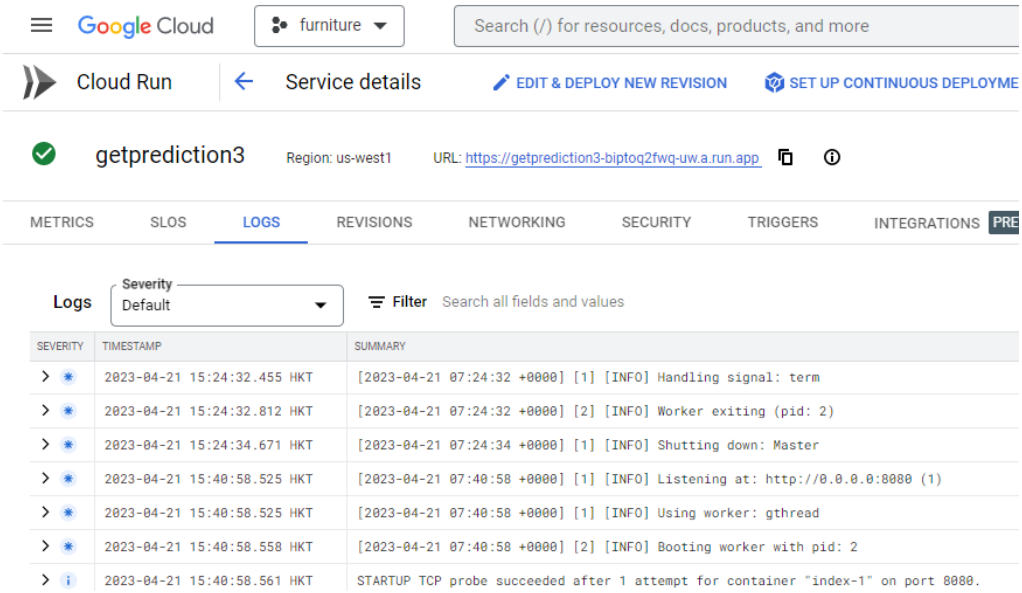


Figure 2.1.2.5 Google Cloud get request

- **PyCharm Community Edition 2023.1**

PyCharm Community Edition 2023.1 is an integrated development environment (IDE) for Python programming that is used to create files such as .dockerignore, Dockerfile, README.md, requirements.txt, and main.py. The main.py file contains a Python script for a Flask web application that serves a trained TensorFlow model for image classification. This script defines a REST API endpoint that accepts image files and returns the predicted class label for the image, using the loaded Keras model file. The Flask web application uses the Flask library to create a web server and define the REST API endpoint, and it also includes a dictionary to map integer class labels to human-readable class names. Additionally, test.py is a Python script that sends an HTTP POST request to the web server running at a specified URL, and then prints the response to the console using the 'json' method.

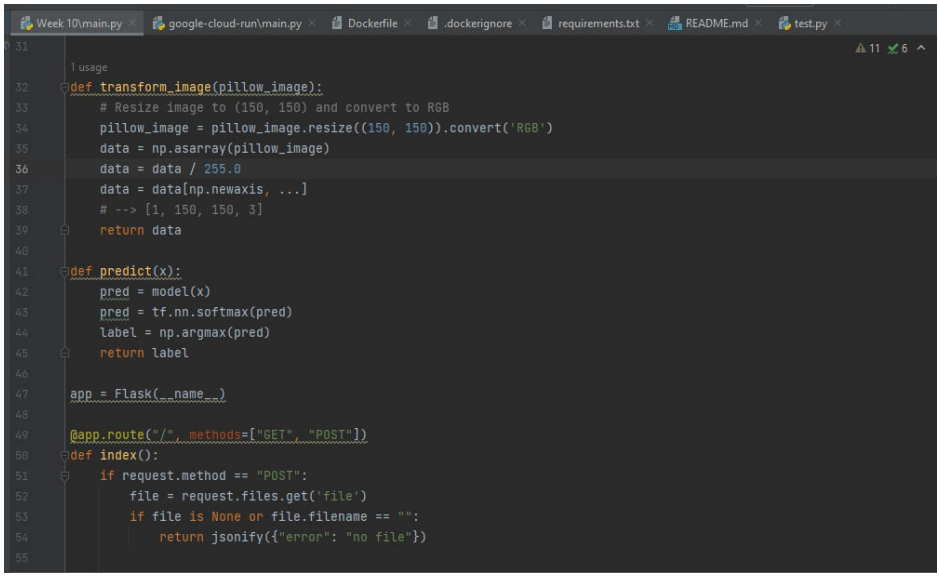


Figure 2.1.2.6 PyCharm main.py

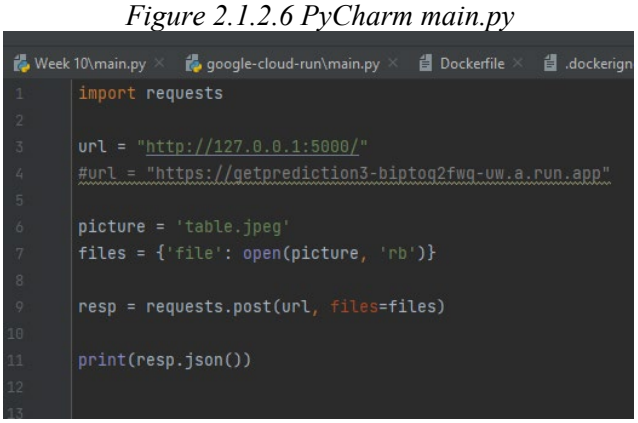


Figure 2.1.2.7 PyCharm test.py

- **Google Cloud Run**

Google Cloud Run is a serverless compute platform that allows this project to run stateless containers in a fully managed environment. It supports containers built from popular languages such as Java, Python, Go, and Node.js, and provides automatic scaling and infrastructure management. In this case, the Flask web application running on the Docker container is deployed to Google Cloud Run to provide the REST API endpoint for image classification prediction.

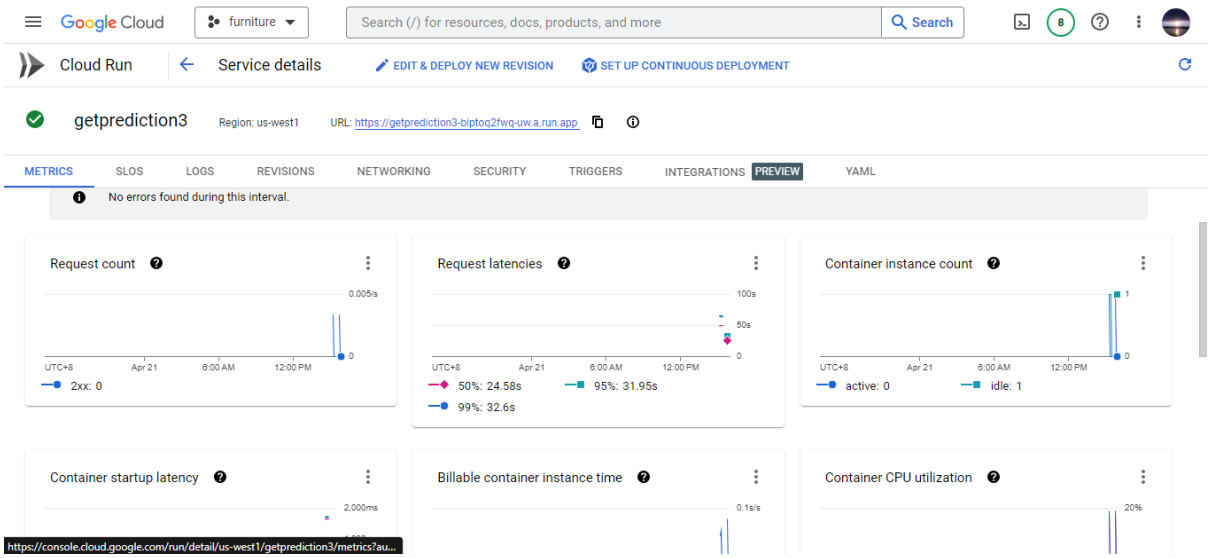


Figure 2.1.2.8 Google Cloud Run

- **Google Cloud Storage**

Google Cloud Storage is a cloud object storage service that allows users to store and retrieve data from Google's infrastructure. It provides a simple and cost-effective way to store data, and supports various types of data including images, videos, and documents. In this case, the trained TensorFlow model file and the uploaded images are stored in a Google Cloud Storage bucket for easy access by the Flask web application running on Google Cloud Run.

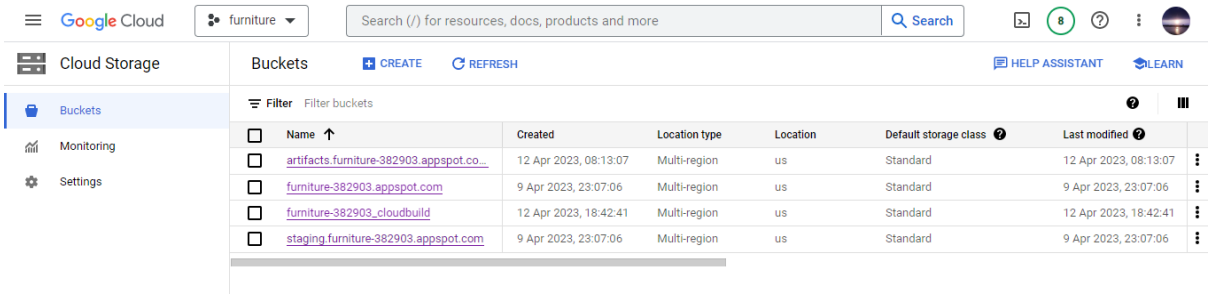


Figure 2.1.2.9 Google Cloud Storage

- **Magick**

Magick is a powerful command-line tool that allows users to quickly and efficiently modify image pixels using code. With Magick, users can perform a wide range of image processing tasks, such as resizing, cropping, and applying various effects and filters.It supports a wide range of image file formats, including popular formats such as JPEG, PNG, and TIFF.

One of the main advantages of using Magick is its flexibility and ease of use. With a few lines of code, image dataset can perform complex image processing operations on large sets of images quickly and efficiently. The tool offers a wide range of options and parameters, allowing users to fine-tune their image processing tasks to achieve the desired results.

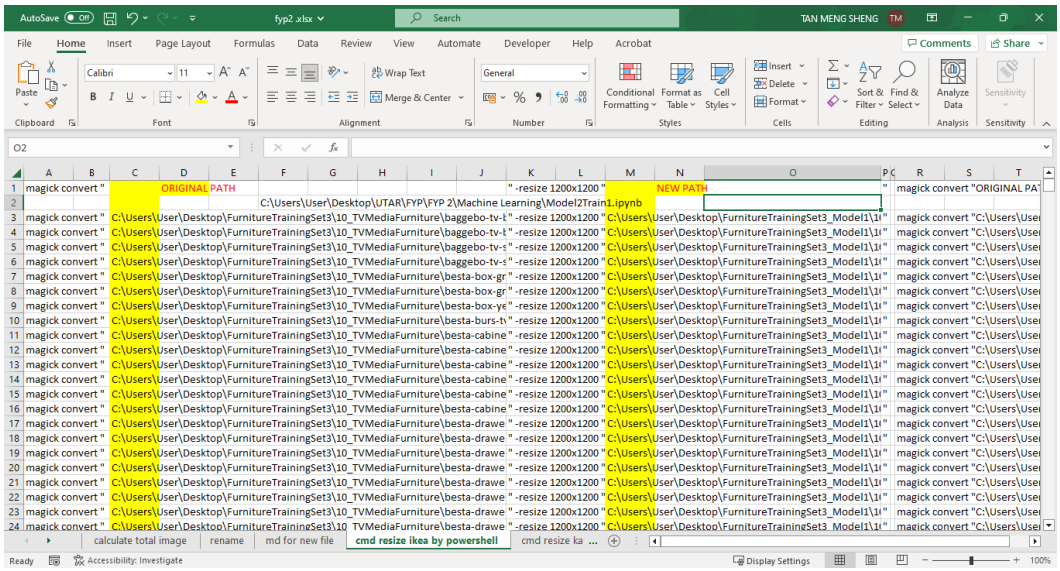


Figure 2.1.2.10 Magick Command

2.1.3. Firmware/OS

- **Windows 10**

Windows 10 is a Microsoft operating system (OS) that was released in 2015 as a successor to Windows 8. It is designed to work on a variety of devices, including desktops, laptops, tablets, and smartphones. Windows 10 is a 64-bit operating system and has several features, including a virtual assistant called Cortana, a new web browser called Microsoft Edge, and support for multiple desktops. It also has an app store where users can download and install various applications. Windows 10 is known for its user-friendly interface, improved performance, and enhanced security features. It supports various input methods, including touch, keyboard, and mouse, making it suitable for a range of devices and use cases[7].

2.1.4. Database

- **Real-Time NoSQL Database**

Ikea furniture finder app uses Firebase to store the images that are required to respond to the user's request. Firebase is a cloud-based platform that provides various services for mobile and web applications, including real-time databases, file storage, and authentication services. In this case, the app could use Firebase Storage to store the images and Firebase Cloud Functions to process the user's request and retrieve the appropriate image to respond to the request. Firebase provides a scalable and reliable infrastructure for storing and serving files and data, making it a suitable solution for the Ikea furniture finder app.

2.1.5. Programming Language

- **Python**

Python is a popular high-level programming language that is widely used in various fields such as scientific computing, data analysis, web development, and machine learning. It is known for its simplicity, ease of use, and powerful libraries that provide ready-to-use functions and modules for a wide range of tasks. Python is an interpreted language, which means that code can be executed directly without the need for compilation. With its extensive support for data structures and object-oriented programming, Python is a versatile language that can be used for many applications, including developing machine learning models like the h5 model used in this project.

- **Flutter**

Flutter is an open-source mobile application development framework created by Google. It uses the Dart programming language and allows developers to build native mobile applications for both Android and iOS platforms. Flutter is known for its fast development cycle, high-performance, and customizable widgets that can be easily customized to match the design of any application. With its built-in widgets, Flutter allows developers to create beautiful and responsive UIs quickly. In the case of the Ikea Furniture Finder project, Flutter was used to develop the mobile application that allows users to search for furniture items by taking a picture of a similar item.

2.1.6. Algorithm

- **Convolutional Neural Network (CNN)**

The artificial neural network system was a pioneering work in the mid-twentieth century and has since gained widespread use due to its good parallel processing, distributed information storage, self-organizing, and self-learning capabilities. Convolutional neural networks have been widely adopted in various fields, particularly in information processing, pattern recognition, intelligent control, and system modelling. The supervised learning algorithm of convolutional neural networks involves using learning samples and backpropagation methods to repeatedly adjust the deviation of the network, which allows for an increasingly close approximation between the output structure and the expected vector[8].

This script is used to train a convolutional neural network (CNN) for image classification using TensorFlow and Keras. The process involves importing necessary libraries, including TensorFlow, NumPy, and h5py, setting up data paths for the training and validation datasets, and defining data generators for data augmentation. The CNN model includes five convolutional layers with 32, 64, 128, 256 and 512 filters, respectively. Each convolutional layer is followed by a max pooling layer with a (2, 2) pool size. The flattened output of the last max pooling layer is connected to a fully connected layer with 512 neurons and a ReLU activation function. The output layer has a softmax activation function for multi-class classification.

The data augmentation performed on the training dataset includes rescaling, shearing, zooming, and flipping the images. The training process is run for 50 epochs with a batch size of 16, 32 and 64. After training more than 30 models, the best model with the highest accuracy and lowest loss is selected and saved as an .h5 file with the name 'Model2_Train20.h5'. This .h5 model is then used as the prediction model to classify new images into their respective categories.

2.1.7. Software Libraries and Frameworks Used in Machine Learning

Table 2.1.7.1 Software Libraries and Frameworks Used

Software Libraries and Frameworks	Version	Description
PyTorch	1.13.1	a popular open-source machine learning framework for building and training neural networks.
cuDNN	8.8.1	a GPU-accelerated library of primitives for deep neural networks.
CUDA	11.6	a parallel computing platform and programming model that allows software to access the power of NVIDIA GPUs.
TensorFlow	2.8.0	another popular open-source machine learning framework for building and training neural networks.
Python	3.9.7	a high-level programming language widely used in scientific computing and machine learning.
NumPy	1.21.5	a fundamental package for scientific computing with Python, providing support for large, multi-dimensional arrays and matrices.
Keras	2.8.0	a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.
Flask	2.2.3	a lightweight web framework for Python used for building web applications.
Gunicorn	20.1.0	a Python WSGI HTTP Server.
Pillow	9.0	a Python Imaging Library (PIL) fork used for image processing.

2.1.8. Summary of the Technologies Review

Ikea Furniture Finder is a client-server model, where the client is an Android application, and the server is a Flask web application. The Android application provides a user-friendly interface for users to upload an image of a piece of furniture and receive a prediction of its type. The Flask web application receives the image from the Android application, processes the image using a pre-trained TensorFlow model, and returns the predicted class label to the Android application.

The Android application is built using Android Studio, which provides a visual editor for creating user interfaces and a code editor for writing the application logic. The application uses a REST API to communicate with the Flask web application. The Android application sends an HTTP POST request to the Flask web application with the image file as a form-data parameter. The Flask web application processes the image and returns the predicted class label as a JSON response. The Flask web application will then use the Firebase API to retrieve the requested number of random images and return them as a JSON response to the Android application. The Flask web application now includes a new REST API endpoint for retrieving random images from Firebase. The Android application can send a request to this endpoint to retrieve a specified number of random images.

The Flask web application is built using Python and the Flask web framework. It uses a pre-trained TensorFlow model for image classification, which is loaded into memory when the Flask application starts. The Flask application exposes a REST API endpoint that accepts HTTP POST requests containing image files. When a request is received, the Flask application processes the image using the pre-trained TensorFlow model and returns the predicted class label as a JSON response.

The system architecture also includes a Docker container that packages the Flask web application and its dependencies. The Docker container is built using a Dockerfile that specifies the dependencies and configuration for the Flask application. The Docker container is then deployed to a cloud platform, such as Google Cloud Platform, where it can be accessed by the Android application.

Overall, Ikea Furniture Finder's system architecture is a simple and efficient solution for image classification that leverages pre-trained machine learning models and cloud computing. The client-server model allows for easy scalability and flexibility, while the use of Docker ensures that the application runs consistently across different environments.

2.2. Previous Works on Deep Learning

2.2.1. Gillette

Functionalities and features

Gillette leaves no stone left when it comes to extremely attractive, supportive, and brand-aware digital assistants. The bot is a seamless addition to the Gillette website in terms of appearance, displaying a consistent style and voice. Account information, order status, product highlights, and FAQs are just a few of the easy-to-find choices available through the bot [9].

Gillette will recommend the product to clients (refer to Figure 2.2.1.2), provide links to navigate to the related webpage (refer to Figure 2.2.1.3).

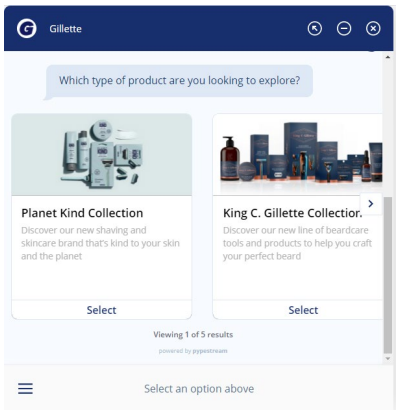


Figure 2.2.1.1 Gillette provides the select option

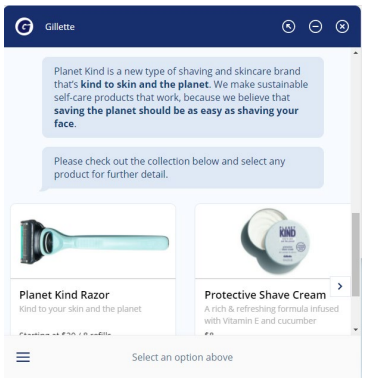


Figure 2.2.1.2 Chatbot will recommend the product to clients

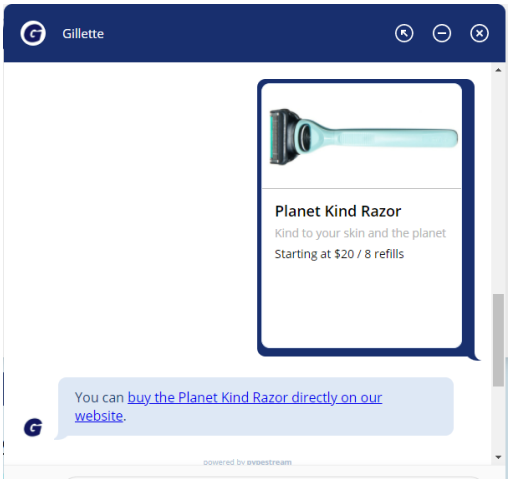


Figure 2.2.1.3 Chatbots provide links to navigate to related webpage

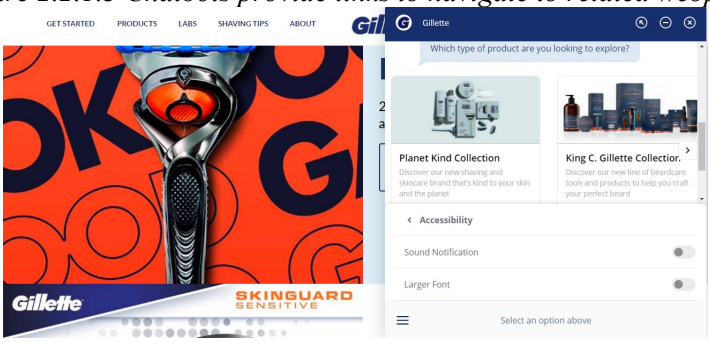


Figure 2.2.1.4 Before large fonts

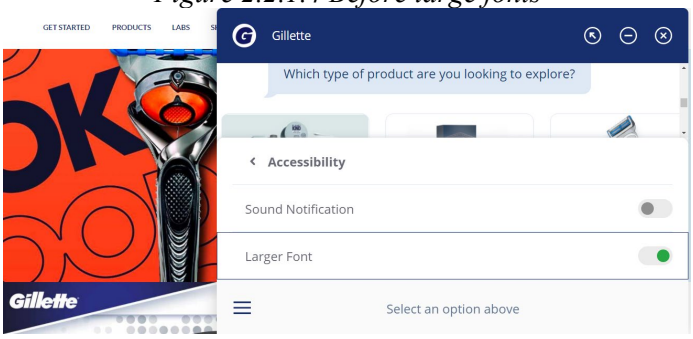


Figure 2.2.1.5 After larger fonts

Strengths

One of the strengths of Gillette is that from color schemes to the **usage of multiple content kinds** such as GIFs, cards, and buttons, conversational UI is engaging. It also **provides larger fonts and sound notifications** so that clients could see the reply clearer and could get the notification in the background (refer to Figure 2.2.1.4 and Figure 2.2.1.5).

2.2.2. Tile

Functionalities and features:

Veronica is Tile's support bot, is exclusively available on the site's product support page. Veronica's mission is completely obvious: she is **more concerned with the technical setup and debugging of the Tile device** than with product discovery or sales.

Veronica provides an option let clients to choose the question, it can understand general words, for example, “hi” (refer to Figure 2.2.2.4), then Veronica will reply to clients [10].

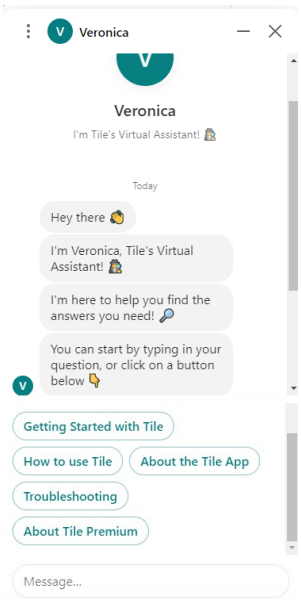


Figure 2.2.2.1 Welcome message of Veronica

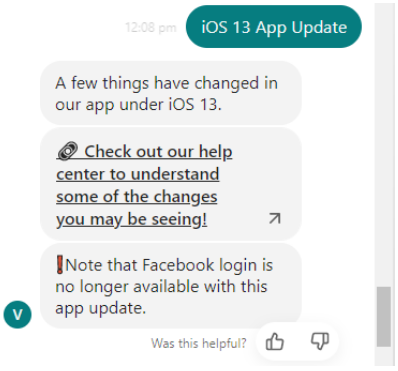


Figure 2.2.2.2 Click the link will navigate to new webpage about the related product

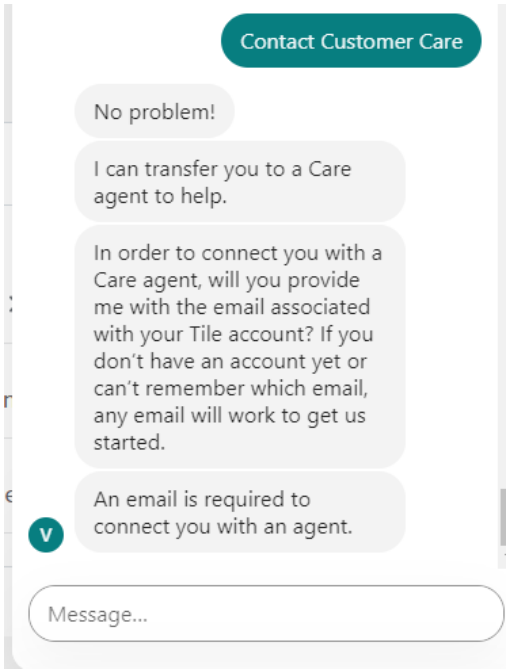


Figure 2.2.2.3 Clients can choose to contact customer care

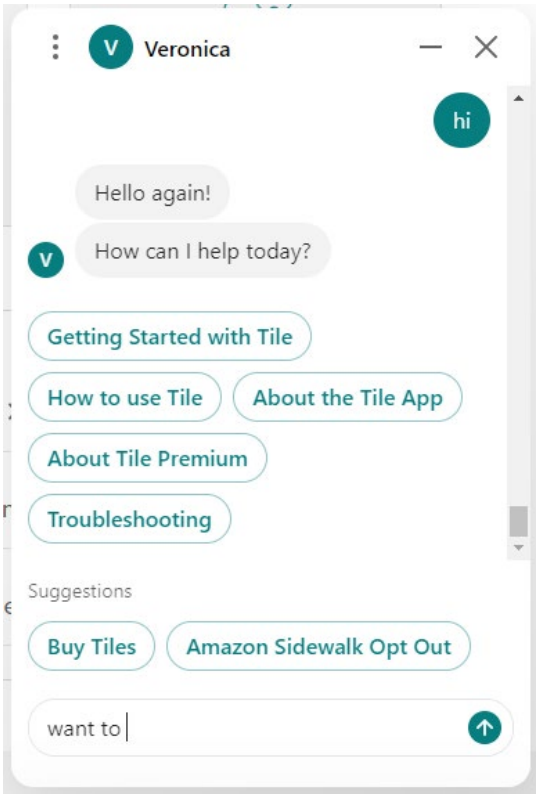


Figure 2.2.2.4 chatbot provide suggestion

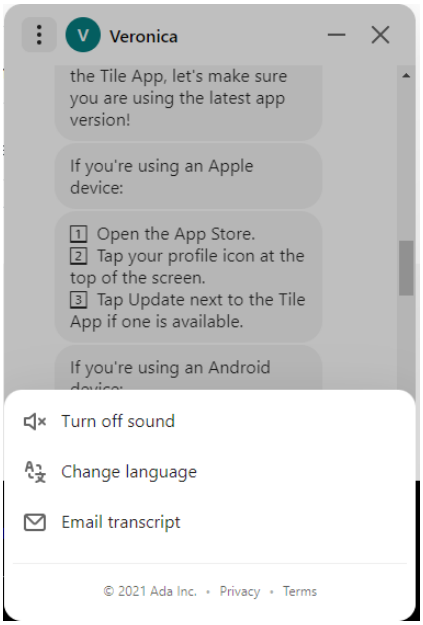


Figure 2.2.2.5 Tile can send transcript via email

Strengths

One of the strengths is Tile **flows based on topics**, it can be tough to arrange a bot efficiently in a button-driven UI when there are a lot of FAQs and related material. Tile performs an excellent job at sorting their content's high-level subjects and developing branches that assist users to discover the information they need. Next, Tile can **turn odd sounds, change language and send transcripts via email** (refer to figure 2.2.2.5).

2.2.3. Black Rifle Coffee

Functionalities and features:

Customers chat asynchronously, with seamless escalation to live support when it's needed. Meanwhile, the Black Rifle Coffee chatbot can **serve customers in real-time by entering and exiting the chat without interrupting the flow of conversation**, working entirely in the workspace with which they are familiar and comfortable. Customers can chat asynchronously, with automatic escalation to live assistance if necessary. Meanwhile, the Black Rifle Coffee chatbot can serve clients in real-time by entering and departing the chat without disturbing the flow of the discussion, allowing them to work totally in their workspace [11].

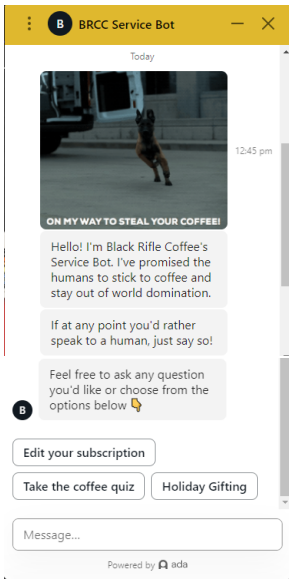


Figure 2.2.3.1 Welcome message of Black Rifle Coffee

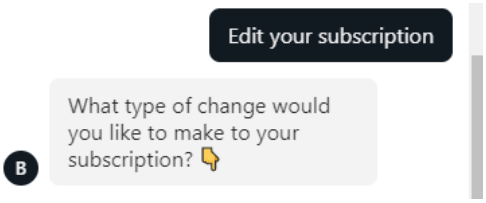


Figure 2.2.3.2 Allow clients to edit subscription

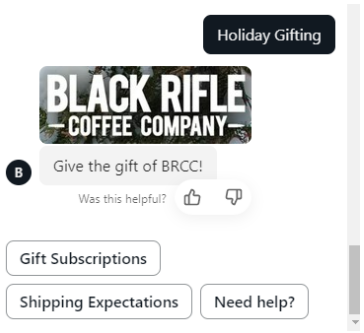


Figure 2.2.3.3 Clients can select a question from the option

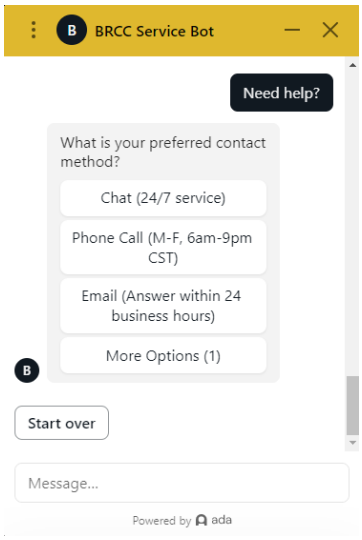


Figure 2.2.3.4 Request for customer service

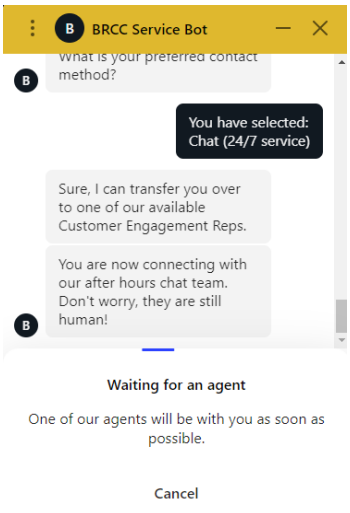


Figure 2.2.3.5 UI while waiting for a real agent

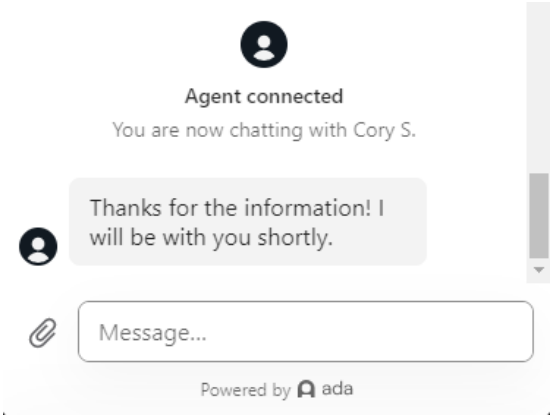


Figure 2.2.3.6 Turn to the real agent

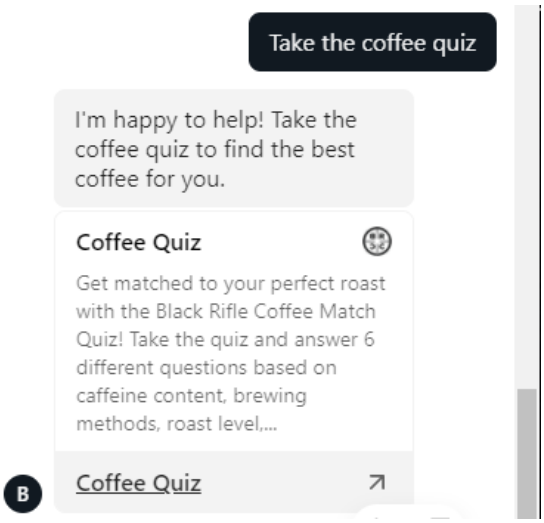


Figure 2.2.3.7 Coffee Quiz

Strengths

One of the strengths is Black Rifle Coffee is **able to edit subscription**. The Black Rifle Coffee bot allows the client to make changes to their membership right in the chatbot, including payments, delivery addresses, and goods. This makes the bot a one-stop-shop for all account requirements (refer to Figure 2.2.3.2). Next, clients can ask chatbot to alter to find for real customer agent (Figure 2.2.3.6).

2.2.4. Hello Fresh

Functionalities and features:

Brie, HelloFresh's chatbot, is utilized as a customer care bot to reduce client wait times. Brie can **automatically answer a variety of consumer questions, and many customers use the bot before contacting a human customer service agent**.

Though Brie's primary function is to assist with customer service, the bot also provides a number of other services to consumers, including a reminder service that prompts users to finish their weekly food purchase, HelloFresh's newest news, and blog articles, as well as Spotify song recommendations to keep client entertained while the client is preparing.

The bot gives HelloFresh consumers a fantastic experience and keeps them engaged with the company even after they've placed an order.

HelloFresh's Brie has proven to be a huge hit. Despite a 44 percent increase in total chats, the bot cut average customer wait time on social customer service channels by 38 percent. Brie was also employed in a Black Friday promotion that resulted in five times the number of daily users than usual [12].

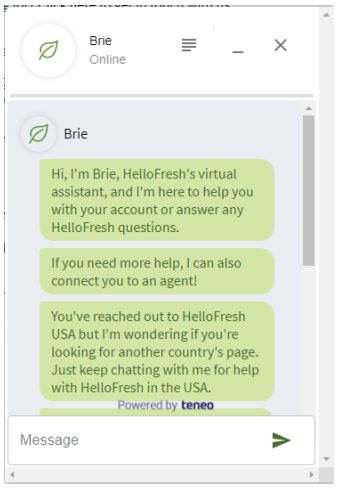


Figure 2.2.4.1 Welcome message

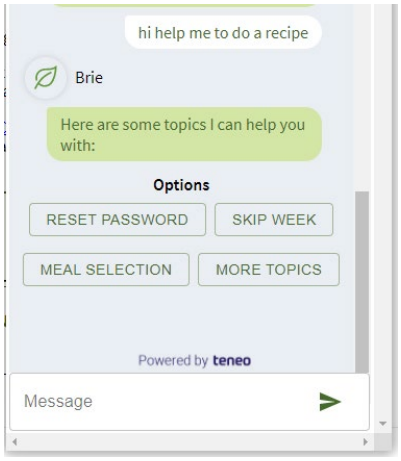


Figure 2.2.4.2 User query

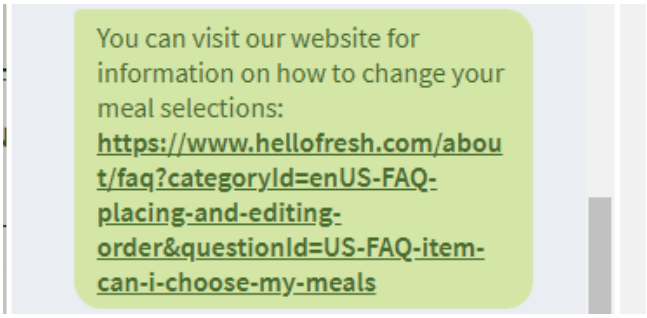


Figure 2.2.4.3 Chatbot reply

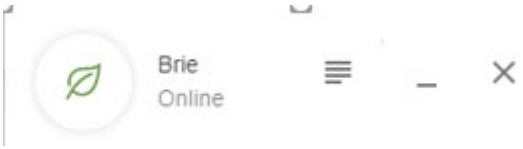


Figure 2.2.4.4 Menu button to download the chat

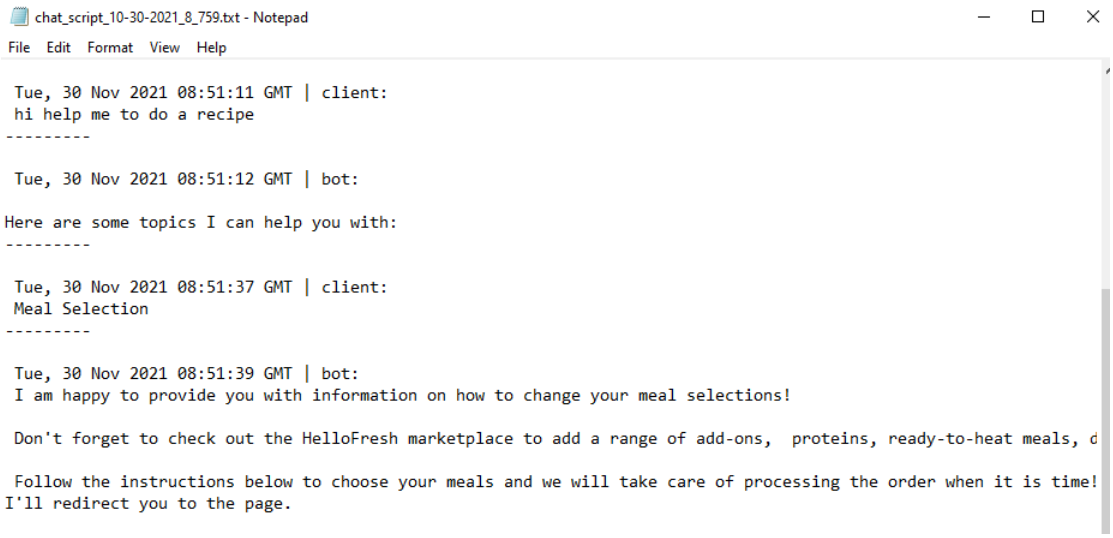


Figure 2.2.4.5 Chatbot will help clients to download the chat script among the chat

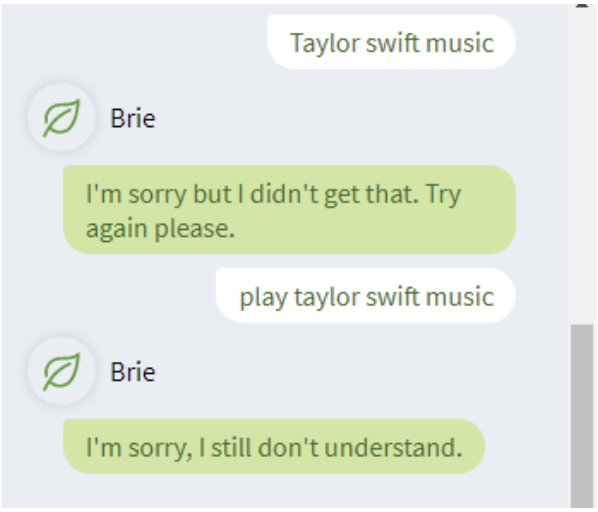


Figure 2.2.4.6 Unable to reply to the query

Strengths

One of the strengths of HelloFresh is that Brie is very friendly to the clients. Brie was also given a few entertaining features by the team to **provide customers with a memorable brand experience that would keep them interested**. Brie's Breakfast Quiz, for example, is a fun method for users to learn about HelloFresh's goods while having fun. Brie is even used by HelloFresh US to **provide music recommendations**. When the bot offers music (selected by HelloFresh's social media team) to complement the meal kit the user will be cooking that evening, they call it Get Grooving.

2.2.5. DavidsTEA

Functionalities and features:

DAVI, the DAVIDsTEA bot, is **fun and full of useful questions and FAQs to assist users with product research and assistance**. DAVI is an excellent example of how to create clever conversational UX for large libraries of useful knowledge, as it supports both English and French. DAVI can help customers to explore, discover new products, keep up to date on the latest tea-making supplies, and so much more, just like DAVIDsTEA's Tea Guides in-store and Virtual Tea Guides online [13].

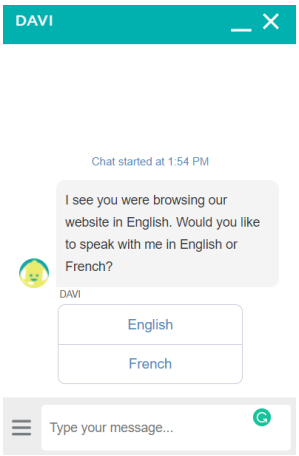


Figure 2.2.5.1 Select Language

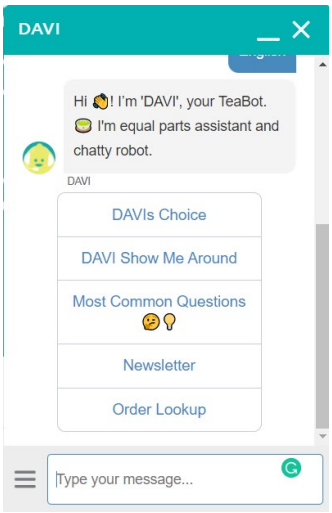


Figure 2.2.5.2 Welcome Message of Davi

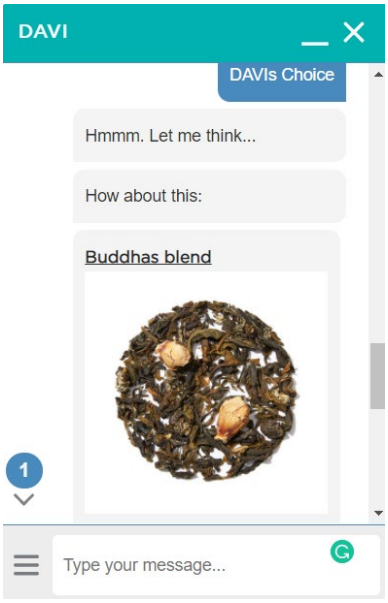


Figure 2.2.5.3 Support picture reply

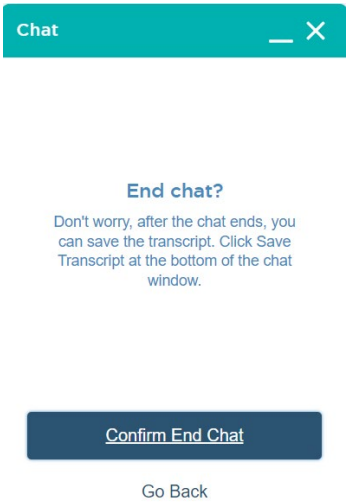


Figure 2.2.5.4 Able to get transcript after a session

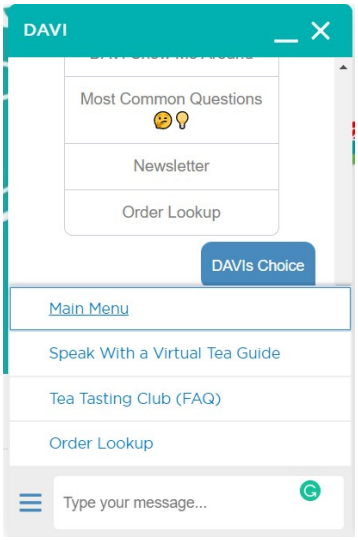


Figure 2.2.5.5 Main menu of DAVI

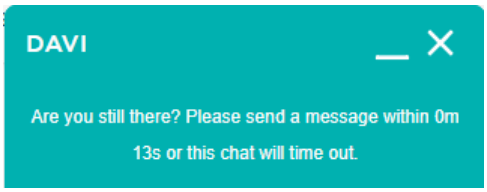


Figure 2.2.5.6 Time remaining for log out

Strengths

One of the strengths of DAVIDsTEA is it is based in Canada, has **ensured that its bot supports both English and French**. When servicing markets where a major portion of the population does not speak English as a first language, make sure the bot is sensitive to the most popular languages and more accessible to a wider audience. The client could select which language would like to interact with the chatbot (refer to figure 2.2.5.1). Moreover, Davi is able to **reply to clients by the picture** (refer to figure 2.2.5.3).

2.2.6. Limitation of Previous Studies

Several strategies and procedures have been shown to have limitations or weaknesses in previous studies. To make the suggested system or study even better, cover anything chatbot did correctly, the question is delivered on a different page on the website, have clear interaction with chatbot's base of knowledge items, chatbots do not always recognize human queries, a chatbot is text-based only are carried out.

One of the weaknesses does **not have clear interaction with Tile's base of knowledge items**. It feels like a missed opportunity since there is no obvious integration with Tile's base of knowledge articles. These articles are usually more detailed, including graphics and videos, giving clients another alternative if the bot's text-only instructions are confusing.

Next, support for a Coffee Quiz is a weakness for Black Rifle Coffee. One of the bot's key sales-driven methods is to convince users to take a coffee quiz to learn which blend best matches their preferences. However, rather than being offered within the bot, **this question is delivered on a different page on the website**. The quiz being integrated into the bot allows for more tailored material (refer to Figure 2.2.3.7).

Moreover, Brie has a limited response, which is one of HelloFresh's limitations. Chatbots provide quicker customer assistance, but they aren't without problems. Because they only respond to a restricted number of customers, not everyone will seek the answers they are looking for. As seen in Figure 2.2.4.6, **chatbots do not always recognize human queries**.

Besides, the weaknesses of Davi is text-based only for a reply message. For a product that has interesting images, animations, and videos, DAVI **chooses to ignore this helpful information**. Because it's difficult for many e-commerce firms to inspire, persuade, or help customers in a text-only UI, it's advised that your bot mix it up with other content kinds to boost engagement and understanding.

2.2.7. The Summarized Comparison of Existing Works with Proposed Project

Table 2.2.7.1 Summarized Comparison of Existing Works with Proposed Project

Existing System/Works	Functionalities and features:	Strength	Weakness
Gillette	-Information is just a few of the easy-to-find choices available through the bot	-Usage of multiple content kinds -Provides larger fonts and sound notifications	- Impossible to uncover anything Gillette did incorrectly
Tile	-More concerned with the technical setup and debugging of Tile device	-Flows based on topics -Sorting their content's high-level subjects -Turn odd sounds, change language and send transcript via email	- Do not have clear interaction with Tile's base of knowledge items
Black Rifle Coffee	-Serve customers in real-time by entering and exiting the chat without interrupting the flow of conversation	- Able to edit subscription	- Question is delivered on a different page on the website
Hello Fresh	-Automatically answer to a variety of consumer questions	-Provide customers a memorable brand experience -provide music recommendations	- Chatbots do not always recognize human queries
DavidsTEA	- Fun and full of useful questions and FAQs to assist users with product research and assistance	-Ensured that its bot supports both English and French. -Reply to clients by picture	- Understand text-based reply only

CHAPTER 3: SYSTEM METHODOLOGY/ APPROACH

3.1. System Design Diagram/Equation

3.1.1. System Architecture Diagram

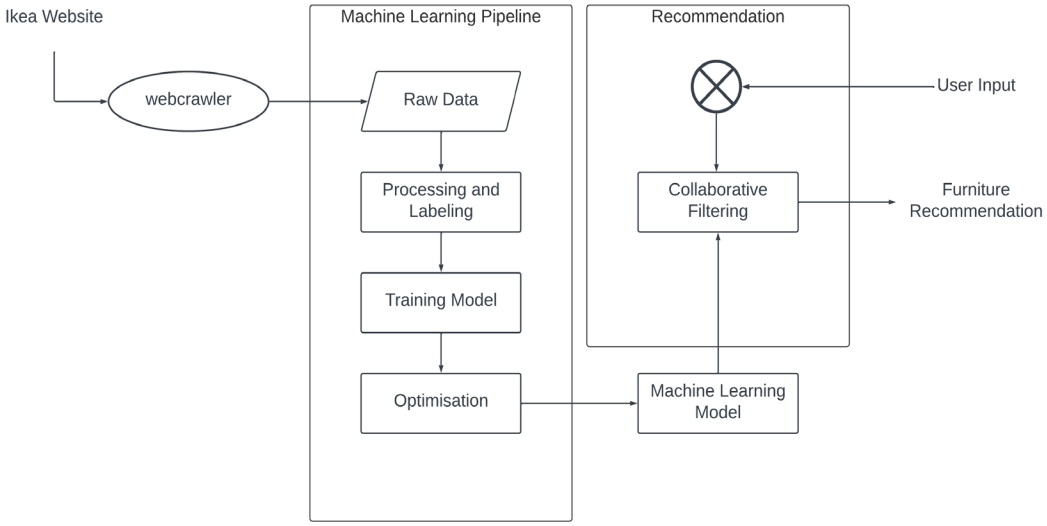


Figure 3.1.1.1 System Architecture Diagram

This project utilized a web crawler to collect a dataset of images from the Ikea website. The collected data was then sent through a machine learning pipeline, which involved several stages such as Processing and Labeling, Training Model, and Optimization. After the model was trained, it was used to provide recommendations to users based on their input. The recommendation process involved sending the user input through the previously developed machine learning model, which was then sent through a Collaborative Filtering algorithm. This resulted in the generation of furniture recommendations for the user.

3.1.2. Use Case Diagram and Description

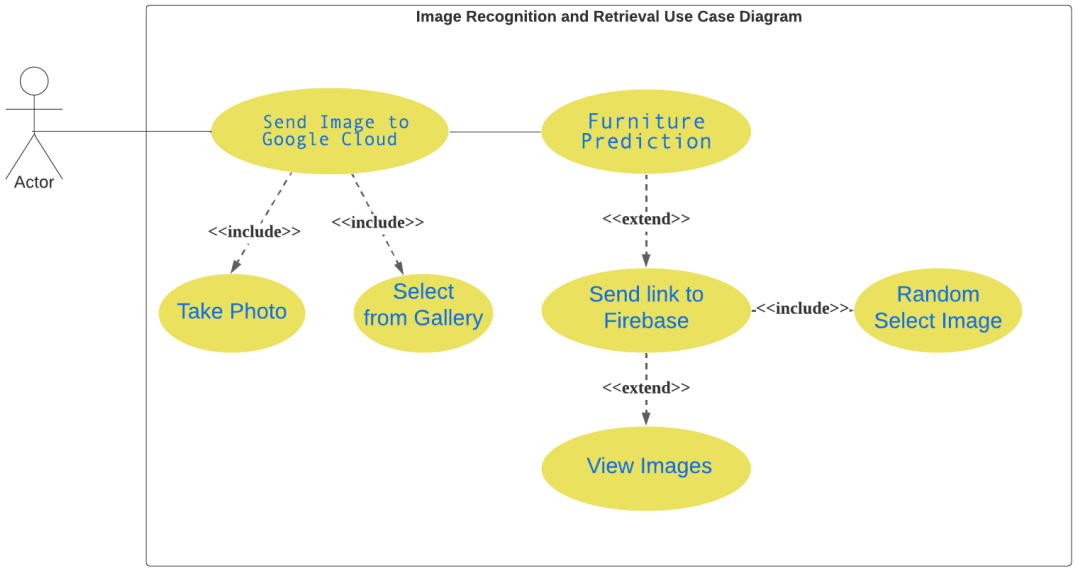


Figure 3.1.2.1 Use Case Diagram

In this project, the use case diagram describes the different ways in which the user can interact with the Ikea Furniture Finder. The main use case is for the user to submit an image through the mobile application, which triggers the prediction process through the Google Cloud Run API. Once the prediction is made, the system sends a request to Firebase to retrieve a set of images related to the predicted label, which are then displayed to the user. The use case diagram also includes additional use cases such as the option for the user to choose an image from their camera or gallery, and the ability to share the results through social media.

3.1.3. Activity Diagram

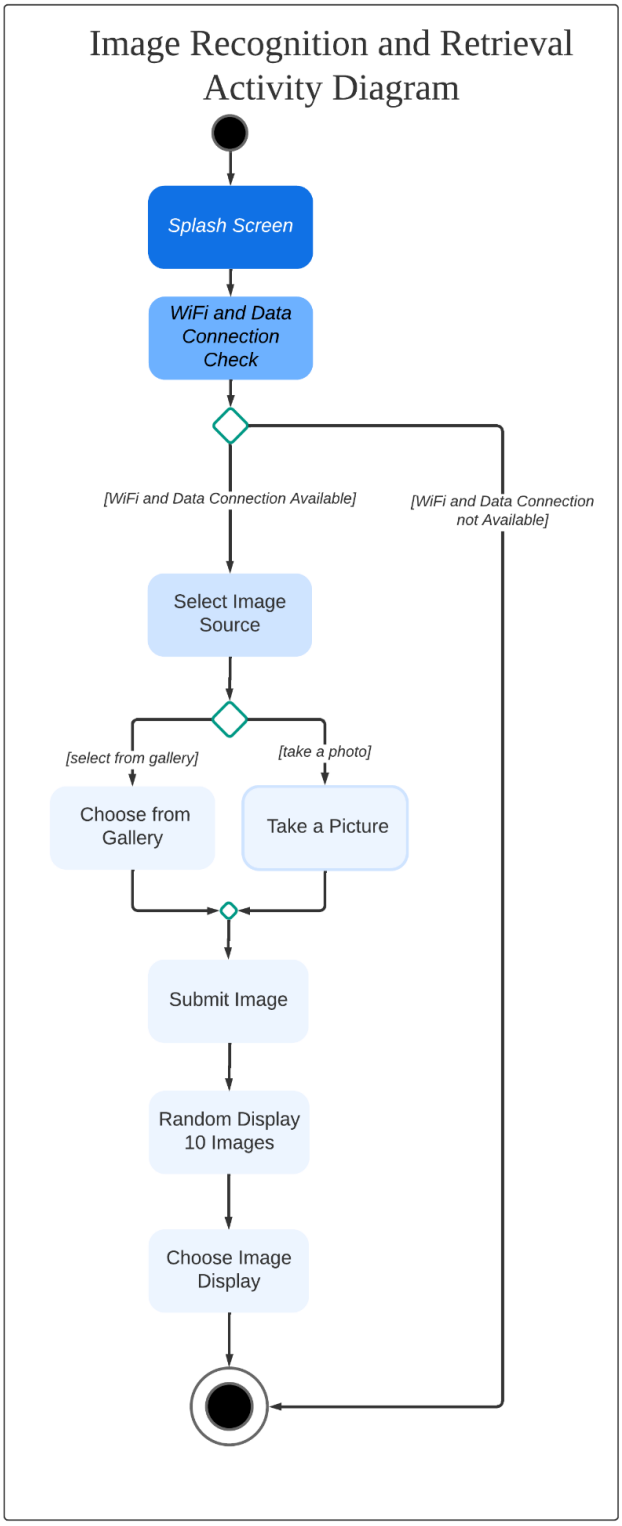


Figure 3.1.3.1 Activity Diagram

The activity diagram represents the workflow of the mobile application that allows the user to select an image from either the camera or the gallery, and then submit the image to the Flask API on Google Cloud Run for prediction. The activity diagram starts with the user launching the application and the splash screen appearing. Next, the application checks for available wifi or data connection, which is necessary for the user to access the Flask API and Firebase. Then, the user is presented with the option to select the image source from either the camera or gallery. Once an image is selected, the user clicks on the submit button, which sends the image to the Flask API on Google Cloud Run for prediction. The Flask API then returns a JSON string response with the predicted label, and the application sends a request to Firebase to retrieve a random set of images for the predicted label. Finally, the application displays the retrieved images to the user.

3.1.4. CNN Architecture Visualization

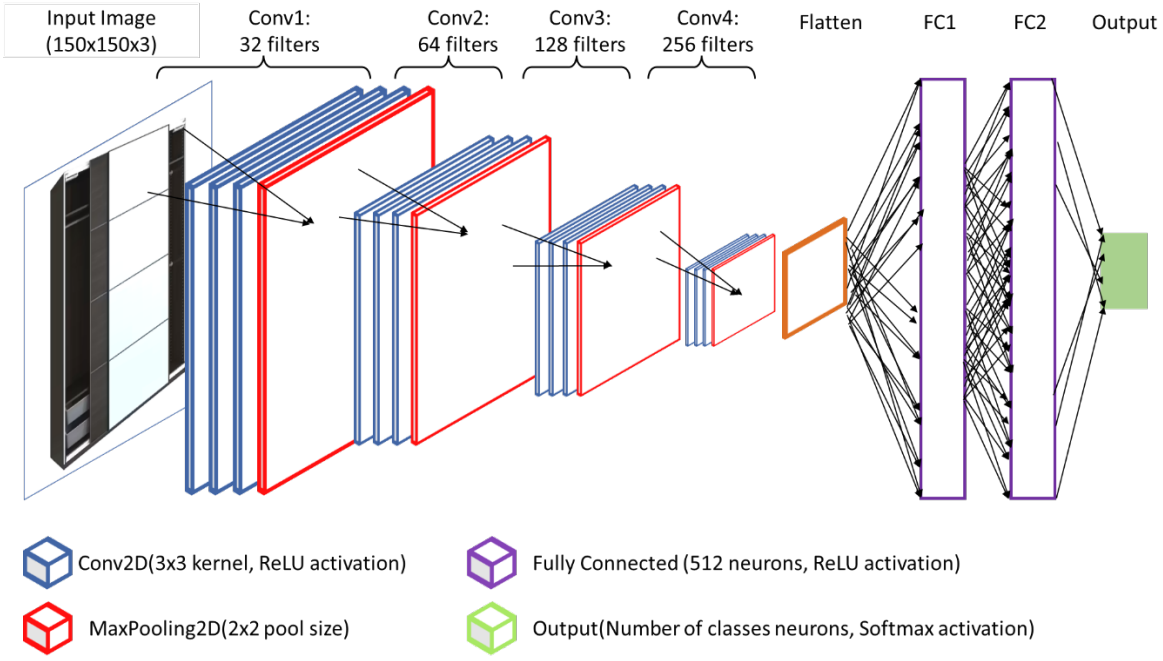
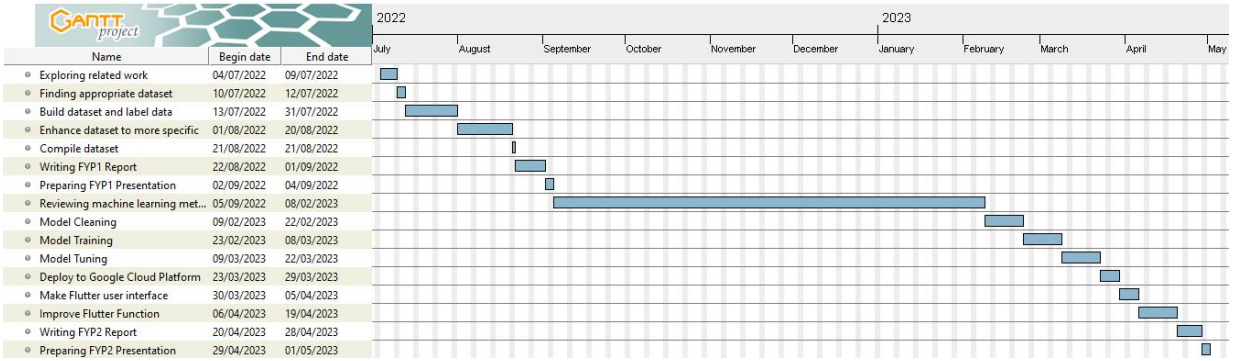


Figure 3.1.4.1 CNN Architecture Visualization for Model2_Train20

The Model2_Train20.h5 architecture consists of an input layer with a shape of (150, 150, 3), followed by 12 layers of convolution and max pooling. The convolutional layers have 32, 64, 128, and 256 filters each with a (3,3) kernel size and a ReLU activation function. Max pooling layers with a (2,2) pool size are added after every three convolutional layers. Finally, a flatten layer is added to transform the data into a 1-dimensional array, which is then fed into two fully connected layers with 512 neurons each and ReLU activation. The output layer has a number of neurons equal to the number of classes, with a Softmax activation function.

3.2. Timeline



Name	Begin date	End date
Exploring related work	04/07/2022	09/07/2022
Finding appropriate dataset	10/07/2022	12/07/2022
Build dataset and label data	13/07/2022	31/07/2022
Enhance dataset to more specific	01/08/2022	20/08/2022
Compile dataset	21/08/2022	21/08/2022
Writing FYP1 Report	22/08/2022	01/09/2022
Preparing FYP1 Presentation	02/09/2022	04/09/2022
Reviewing machine learning met...	05/09/2022	08/02/2023
Model Cleaning	09/02/2023	22/02/2023
Model Training	23/02/2023	08/03/2023
Model Tuning	09/03/2023	22/03/2023
Deploy to Google Cloud Platform	23/03/2023	29/03/2023
Make Flutter user interface	30/03/2023	05/04/2023
Improve Flutter Function	06/04/2023	19/04/2023
Writing FYP2 Report	20/04/2023	28/04/2023
Preparing FYP2 Presentation	29/04/2023	01/05/2023

Figure 3.1.4.1 Timeline

This Ikea Furniture Finder project had a projected completion date of the Friday of Year 3 Trimester 3 Week 13, which was accomplished within two trimesters. The initial project phase focused primarily on gathering data and improving the dataset for image recognition and classification. The project report was submitted on September 2nd, 2022.

During the FYP2 discussion, new project objectives were established, including the development of a TensorFlow model for accurate image classification, utilization of GCP for cloud-based deployment and scalability, integration of Firebase for efficient image storage and retrieval, and creation of a Flutter-based mobile application for user-friendly furniture recommendations.The subsequent project work included extensive training of the TensorFlow model. The project also involved the implementation of GCP and Firebase for cloud-based deployment, image storage, and retrieval. The development of the mobile application using Flutter was also undertaken to provide users with a user-friendly furniture recommendation system. The successful implementation and testing of these technologies culminated in the completion of the Ikea Furniture Finder project.

CHAPTER 4: SYSTEM DESIGN

4.1. System Block Diagram

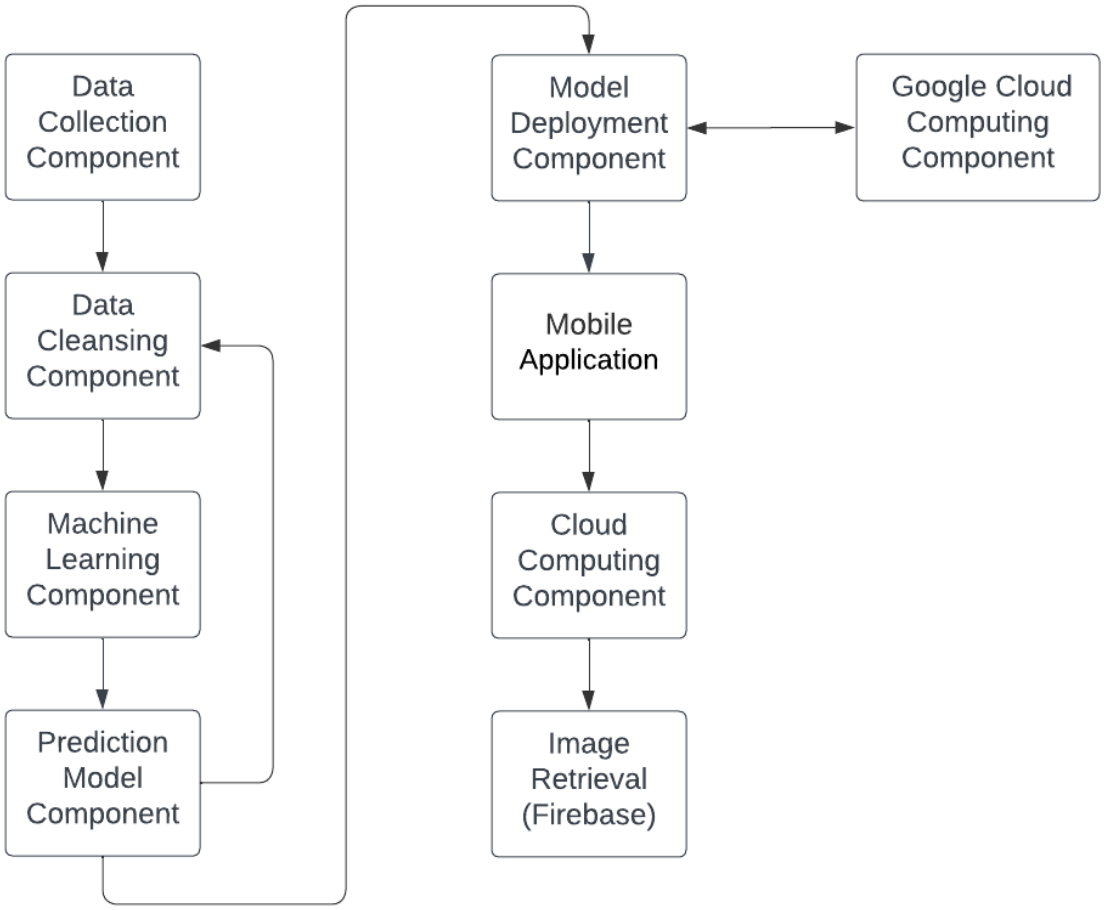


Figure 3.1.4.1 System Block Diagram

The system block diagram for this project consisted of nine main components: Data Collection, Data Cleansing, Machine Learning, Model Deployment, Cloud Computing, Mobile Application, Cloud Function Component, Cloud Computing, and Image Retrieval (Firebase).

The Data Collection component was responsible for gathering furniture image data from the IKEA website and preparing it for use in the machine learning model. The Data Cleansing component renamed the filetype, changed the pixels, and cleared the noisy data. The Machine Learning component trained a convolutional neural network (CNN) model using the collected dataset to recognize furniture items in images. The Prediction Model component used the trained machine learning model to make predictions. Due to the iterative nature of the project, the data cleansing, machine learning, and prediction model components were repeated multiple times to refine the accuracy of the system.

The Model Deployment component was responsible for repeatedly deploying the model to Docker or Google Cloud computing platform for use in image recognition tasks throughout the iterative process. Finally, the Mobile App component was responsible for allowing the user to input an image for analysis using the machine learning model deployed on the cloud. The Image Retrieval (Firebase) component was responsible for retrieving the similar category of images from the images stored in Firebase based on the analysis result.

4.2. System Components Specifications

- **Data Collection:**
  - Gather furniture image data from the IKEA website (https://www.ikea.com/my/en/cat/furniture-fu001/).
  - Save the images in a designated directory.
  - Check for and eliminate duplicate images.
- **Data Cleansing:**
  - Rename image files to change the file type or format.
  - Resize images to a uniform size(1200*1200).
  - Remove noisy data or irrelevant images.
  - Store the cleaned data in a separate folder or directory.
- **Machine Learning:**
  - Train a CNN model to recognize furniture items in images using Python programming language, TensorFlow and Keras libraries.
  - Use data augmentation techniques to increase the size of the dataset.
  - Tune hyperparameters to improve model accuracy.
  - Validate the model using a test dataset.
  - Input image size of 224x224 pixels.
  - Output classes of furniture categories from the IKEA dataset.
  - Training performed on a laptop with a minimum of 8GB RAM and NVIDIA graphics card.

- **Prediction Model:**
    - Use the trained h5 machine learning model to make predictions.
    - Output a prediction of the furniture item in the input image.
    - Output a confidence score or probability of the prediction.
- **Model Deployment:**
    - Deploy the trained machine learning model to a cloud computing platform (Google Cloud Run).
    - Set up an API to handle image recognition requests.
    - Monitor the performance and usage of the deployed model.
- **Cloud Computing:**
    - Provide the necessary infrastructure for hosting and running the machine learning model, for example: Google Cloud Platform, Google Cloud Run, Firebase.
    - Ensure high availability and scalability of the deployed model.
- **Mobile Application:**
    - Developed using Flutter programming language and Android SDK.
    - Compatible with Android operating system.
    - Must have a camera and image gallery access permissions.
    - Must be able to connect to Firebase and Google Cloud Platform services.
    - Allow users to input an image for analysis.
    - Display the predicted furniture item and confidence score.
    - Provide an option to search for similar items or view related products.

- **Cloud Function Component:**
    - Implement the image recognition functionality on the cloud.
    - Handle requests from the mobile application and trigger the machine learning model.
- **Image Retrieval (Firebase):**
    - Store a database of furniture images with associated category labels.
    - Retrieve similar category images based on the analysis result.
    - Allow for filtering or sorting of the retrieved images.
    - Firebase Realtime Database for retrieving image data and predicted results.
    - Firebase Cloud Storage for retrieving images uploaded by users.

4.3. System Components Interaction Operations

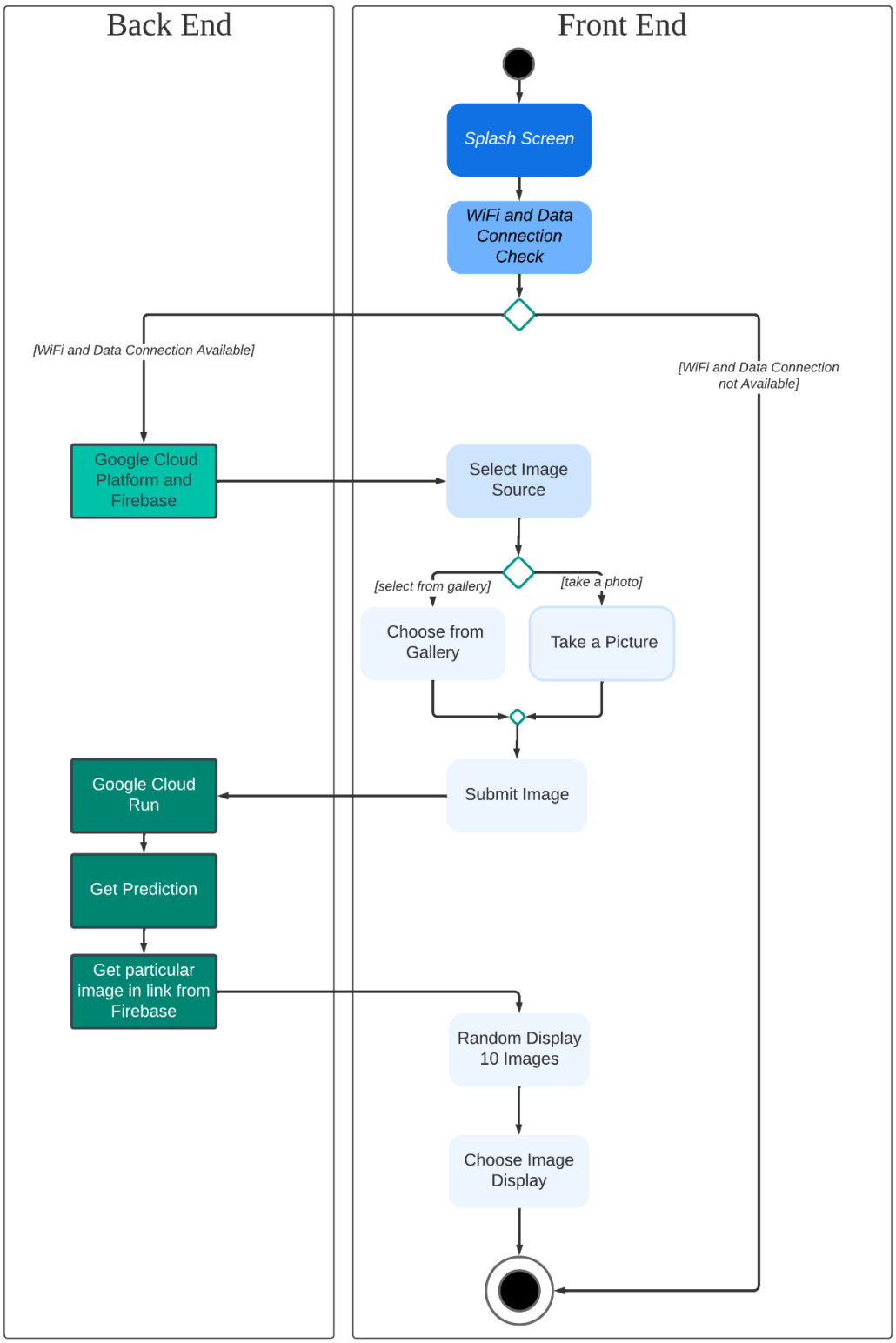


Figure 3.1.4.1 System Components Interaction Operations

- User opens the mobile application on their device.
- User see the splash screen and mobile application start to check WiFi and Data Connection Check.
- Mobile application send requests to Firebase Authentication service.
- Firebase Realtime Database service retrieves image data and sends it back to the mobile application.
- User selects an image by camera or gallery to upload for furniture recognition.
- Mobile application sends a request to Google Cloud Run to execute the trained machine learning model API.
- Google Cloud Run receives the request and starts to execute the getPrediction3 API.
- Google Cloud Run sends the request to the machine learning model API.
- Machine learning model API receives the request and performs furniture recognition on the image.
- Machine learning model API sends the predicted furniture category to Google Cloud Run.
- Google Cloud Run sends the predicted furniture category back to the mobile application.
- Mobile application holds the predicted furniture in JSON string format.
- Mobile application gets the particular needed string.
- Mobile application sends a request to Firebase Realtime Database service to retrieve all the image link based on the predicted furniture category.
- Firebase Realtime Database service retrieves all the image link and sends them back to the mobile application.
- Mobile application shuffle and displays top 10 similar category images to the user.
- User can choose one of the picture and display larger.

CHAPTER 5: SYSTEM IMPLEMENTATION

5.1. Hardware Setup

The Acer Nitro AN515-56 laptop used for the project runs on Windows 10 Home, with an 11th Gen Intel Core i5-11300H quad-core processor operating at 3.10 GHz, and an NVIDIA GeForce GTX 1650 graphics controller with 4 GB of dedicated GDDR6 graphics memory. It has 8 GB of 3200Mhz DDR4 RAM and 512 GB of SATA SSD storage [4]. The machine learning model was trained using an NVIDIA GeForce GTX 1650 GPU based on the Turing architecture, with 896 CUDA cores, 56 TMUs, and 32 ROPs. The GPU has 4 GB of GDDR5 memory with a bus width of 128 bits and a memory speed of 8 Gbps. The NVIDIA GeForce GTX 1650 is a mid-range GPU suitable for machine learning tasks, providing good performance for the price and widely used in the field[5].

1. To check GPU for this project, here are the steps on Windows computer:
    1. Right-click on the Start menu and select Device Manager.
    2. In the Device Manager window, expand the Display adapters section.
    3. The name and model of your GPU will be listed here.

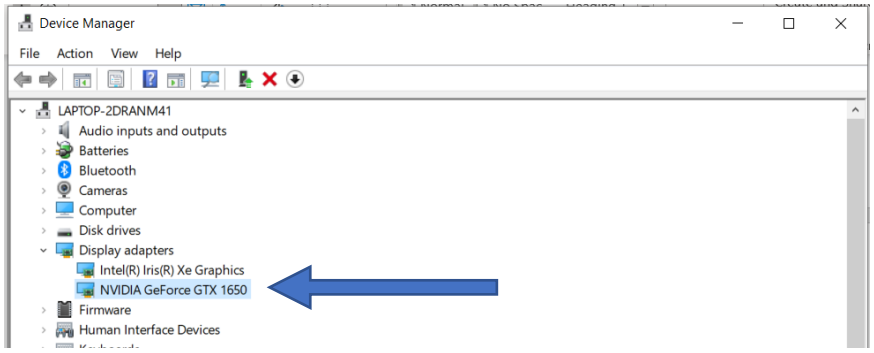


Figure 3.1.4.1 Device Manager

2. Go to the NVIDIA driver download page(https://www.nvidia.com/Download/index.aspx).

3. Choose GeForce GTX 1650 from the product list and select your operating system. Then click Search.

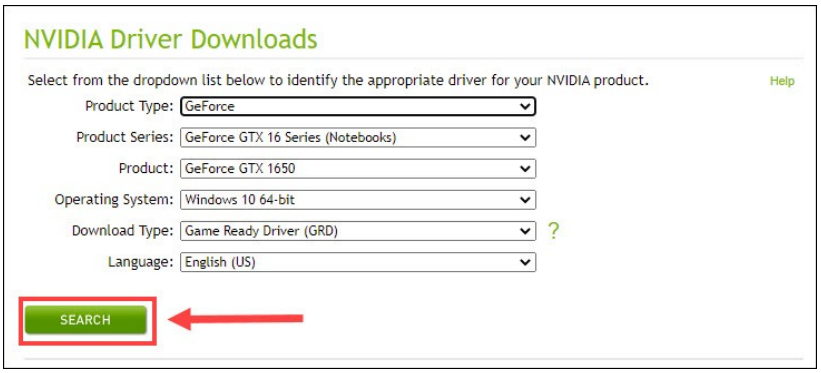


Figure 3.1.4.2 GeForce GTX 1650

4. Make sure your operating system is correctly detected. Then click the Download button.

Figure 3.1.4.3 Download

5. After the download finishes, double-click the downloaded file and follow the on-screen instruction to install the driver for your Windows.

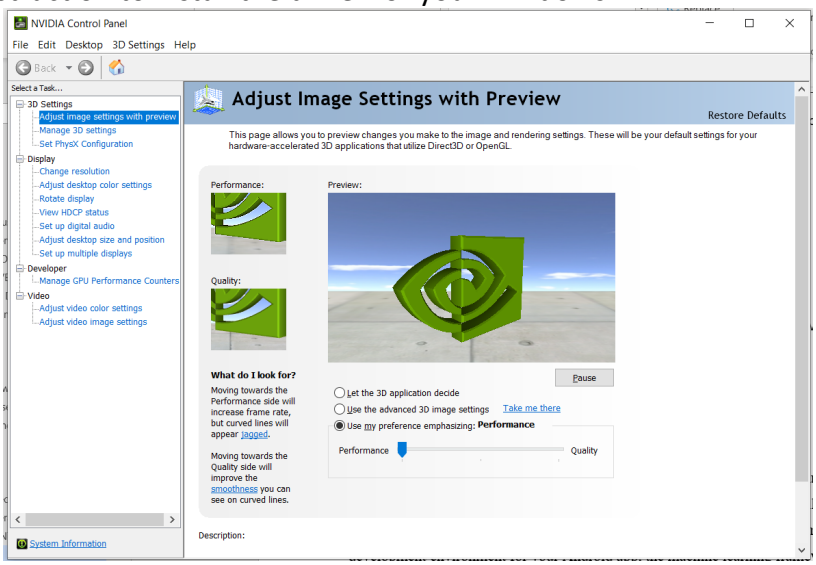


Figure 3.1.4.4 Finished Install

5.2. Software Setup

To set up the necessary environment for this project, several tools need to be downloaded, including Anaconda 2021.11 for managing the Python environment, Android Studio Bumblebee 2021.1.1 for developing the mobile application, Docker 4.18.0 for containerization, Postman 10.12.0 for API testing, PyCharm Community Edition 2023.1 for Python programming, and Google Cloud SDK, this is for Google Cloud Run and Google Cloud Storage which are cloud computing and storage, respectively. Each tool requires specific configurations and installation steps, and it is essential to follow the documentation and tutorials to ensure a smooth installation process. Additionally, it is crucial to ensure compatibility between the tools and the project requirements to avoid potential errors or compatibility issues.

Table 3.1.3.1 Software setup table

Software	Tutorial Link	Download Link
Anaconda	https://youtu.be/UTqOXwAi1pE	https://www.anaconda.com/download/
Android Studio	https://youtu.be/uCnLDDwLxVA	https://developer.android.com/studio
Docker 4.18.0	https://youtu.be/4xK-zaCRiPQ	https://www.docker.com/products/docker-desktop/
Postman 10.12.0	https://youtu.be/3eHJkcA8mTs	https://www.postman.com/downloads/
PyCharm Community Edition 2023.1	https://youtu.be/MJJpL9EmJBs	https://www.jetbrains.com/pycharm/download/#section=windows
Cuda	https://youtu.be/2TcnIzJ1RQs	https://developer.nvidia.com/cuda-11-6-0-download-archive
cuDNN		https://developer.nvidia.com/rdp/cudnn-archive
Google Cloud SDK	https://youtu.be/_tqgMBeolnw	https://cloud.google.com/sdk/docs/install

5.3. Image Dataset Preparation

Ikea Furniture consists of 11923 of furniture with 20 categories for example: 'Tables and desks', 'Chairs, stools and benches', 'Sofas', and etc (refer to Figure 5.3.1). Each of the category have their own series, for example: 'ADILS', 'ALEX', 'ANFALLARE', and etc. Each of the category have different colour, for example: 'white', 'black', 'grey' and so on. When the picture is downloaded, the file format are 'avif', 'webp', and 'jpg'. By using rename.bat, the content inside are: ren *.webp *.jpeg, ren *.avif *.jpeg,and ren *.jpg *.jpeg, this is the script to let the picture rename from 3 file format (refer to Figure 5.3.2).

F1 | ="md "&A1

	A	F
1	1_Tables_and_desks	md 1_Tables_and_desks
2	2_Chairs_stools_and_benches	md 2_Chairs_stools_and_benches
3	3_Sofas	md 3_Sofas
4	4_Armchairs_and_chaise_longues	md 4_Armchairs_and_chaise_longues
5	5_Bed_frames	md 5_Bed_frames
6	6_Wardrobes	md 6_Wardrobes
7	7_Bookcases_and_shelving_units	md 7_Bookcases_and_shelving_units
8	8_Display_cabinets_and_storage_systems	md 8_Display_cabinets_and_storage_systems
9	9_Chests_of_drawers_and_drawer_units	md 9_Chests_of_drawers_and_drawer_units
10	10_TV_and_media_furniture	md 10_TV_and_media_furniture
11	11_Trolleys_and_kitchen_islands	md 11_Trolleys_and_kitchen_islands
12	12_Outdoor_furniture	md 12_Outdoor_furniture
13	13_Bar_Tables_and_Chairs	md 13_Bar_Tables_and_Chairs
14	14_Café_furniture	md 14_Café_furniture
15	15_Sideboards_buffets_and_console_tables	md 15_Sideboards_buffets_and_console_tables
16	16_Room_dividers	md 16_Room_dividers
17	17_Children's_furniture	md 17_Children's_furniture
18	18_Nursery_furniture	md 18_Nursery_furniture
19	19_Gaming_furniture	md 19_Gaming_furniture
20	20_Furniture_sets	md 20_Furniture_sets

Figure 3.1.4.1 make directory command for 20 categories

rename.bat - Notepad

File Edit Format View Help

```
ren *.webp *.jpeg
ren *.avif *.jpeg
ren *.jpg *.jpeg
```

Figure 3.1.4.2 rename bat

Some of the symbol in category and series name will be different for example: Å and Ä will be replaced by A, Ö will e replaced by O, '/', '__' and space will be replaced by '_'. This is to reduce the error when PowerShell ran the script (refer to Figure 3.2.3).

	A	B	C	D	E
1	FIND	REPLACE WITH			
2	/	_	change symbol to open the file		
3		_	let a space become underscore		
4	Å	A	change symbol to open the file		
5	Ä	A	change symbol to open the file		
6	Ö	O	change symbol to open the file		
7	__	_	reduce duplicate underscore		

Figure 3.1.4.3 Character that need to be replaced

Compare to open the file by copy the file name and paste on the file name, this project using excel formula to generate the **id** of category, and a formula to add "**series**" at the back of each of the series (refer to Figure 5.3.4), using "md" where is make directory to let the PowerShell to run the script at once (refer to Figure 5.3.5).

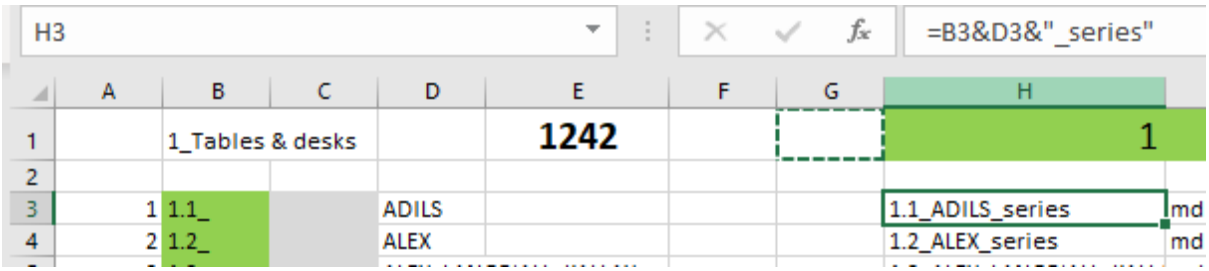


Figure 3.1.4.4 Each ID with the series name

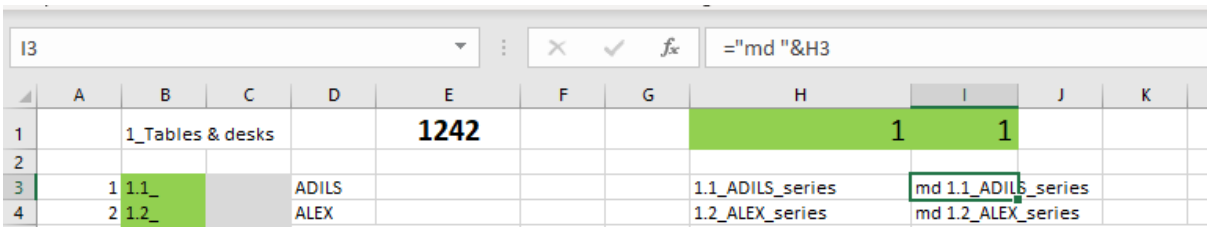


Figure 3.1.4.5 make directory command

When some of the series names and data are same, but they are different category (‘Tables and desks’, ‘Chairs, stools and benches’), this will use the “ren” which is rename function in the PowerShell. This could be shorter the time consuming for data collection (refer to Figure 5.3.6).

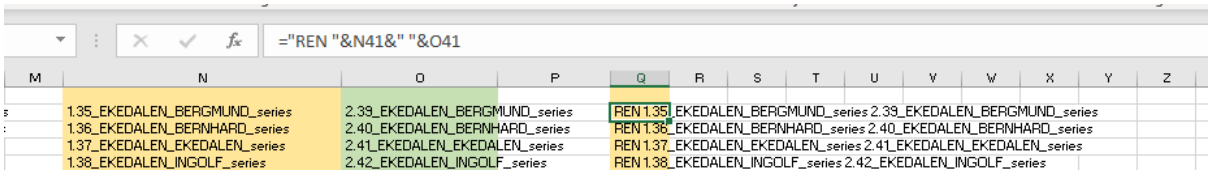


Figure 3.1.4.6 Rename command

After all of the data are collected into the file, next step is using the PowerShell command to let the series become more specific into types (‘chest-of-3-drawers-white’, ‘drawer-unit-white’) and colour (‘white’, ‘black’). By using command **dir /b /s |clip**, dir is display the list of subfolders in the dataset, /b is select the bare format, which is no any summary, heading and the file size of the dataset, /s is including all of the subfolders, while |clip is let the copy of the directory listing into the clipboard (refer to Figure 5.3.7) [14]. The list of directories in the clipboard will be pasted into excel sheet with total number of 13695 of rows (refer to Figure 5.3.8) [15].

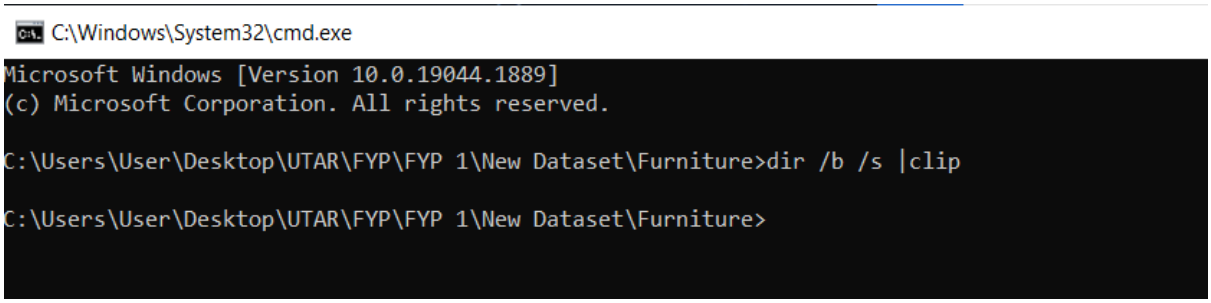


Figure 3.1.4.7 PowerShell Command to retrieve directory

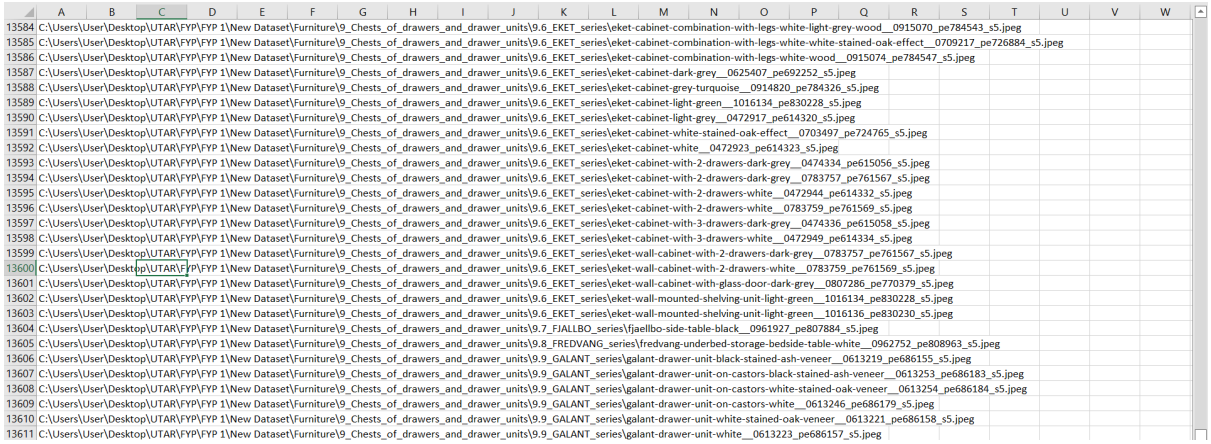


Figure 3.1.4.8 All of the directory list is pasted in Excel Sheet

Copy all of the jpeg file and paste in another sheet, with a quantity of 12029 of data(refer to Figure 5.3.9).

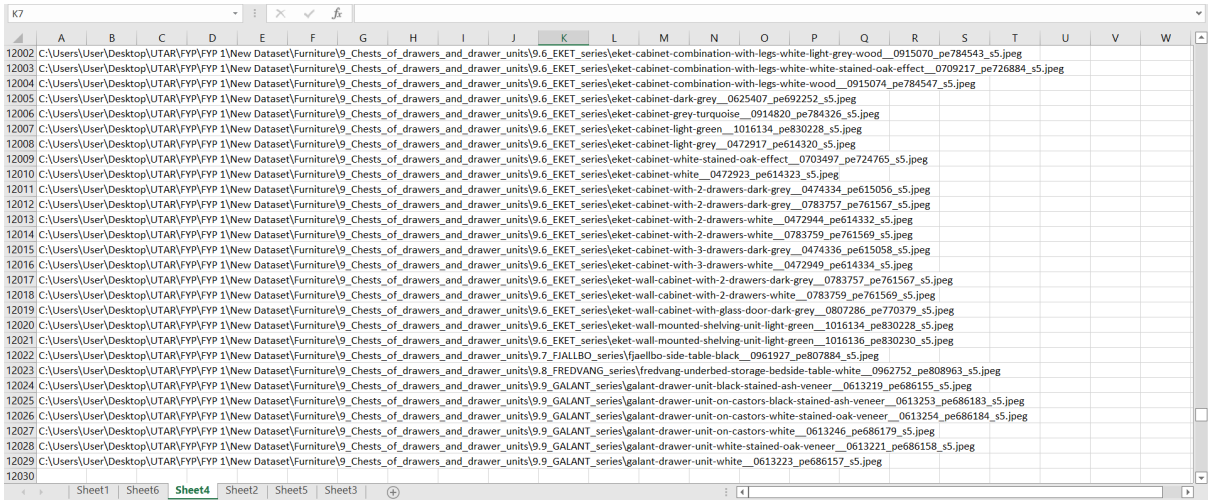


Figure 3.1.4.9 Whole dataset quantity

After all of the lists are pasted into excel files, then need to remove main folder address by pressing “Ctrl+H” to call out the find and replace function (refer to Figure 5.3.10). Then copy the same column from A to H (refer to Figure 5.3.11). Because need to make directory for the more specific dataset, so need to clear the id behind which is useless by using replace function (refer to Figure 5.3.12). After that make directory by using “md” for PowerShell scripting (refer to Figure 5.3.13).

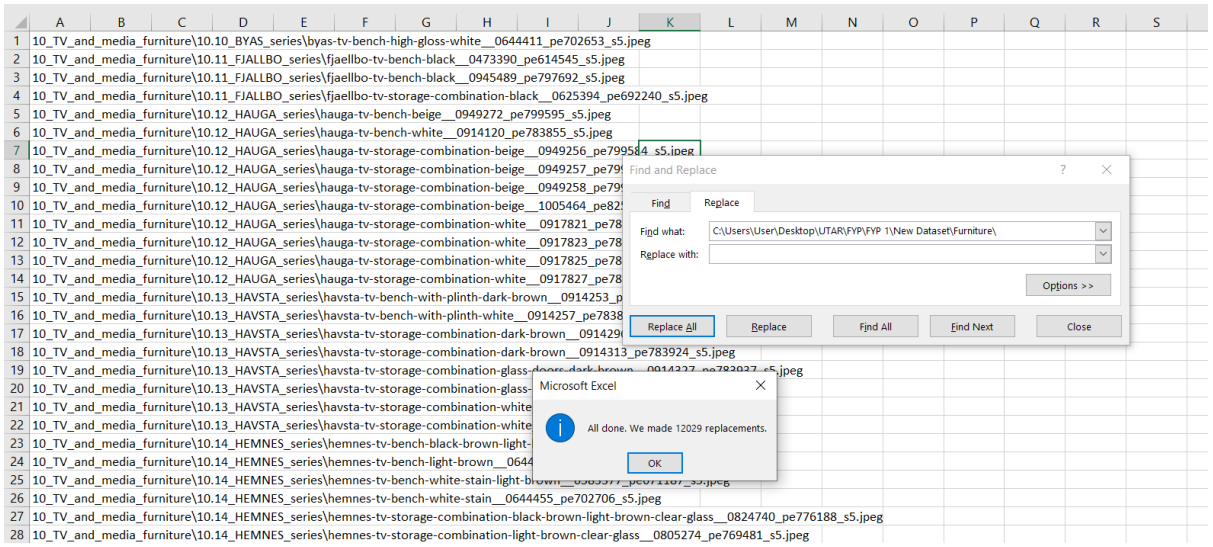


Figure 3.1.4.10 Remove the main folder address

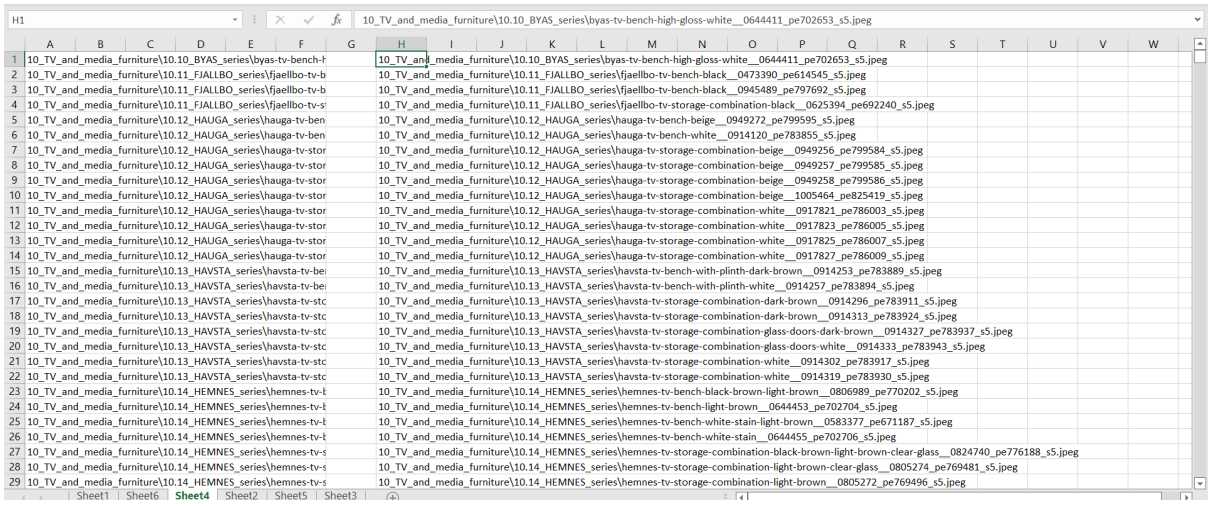


Figure 3.1.4.11 Copy the same column from A to H column

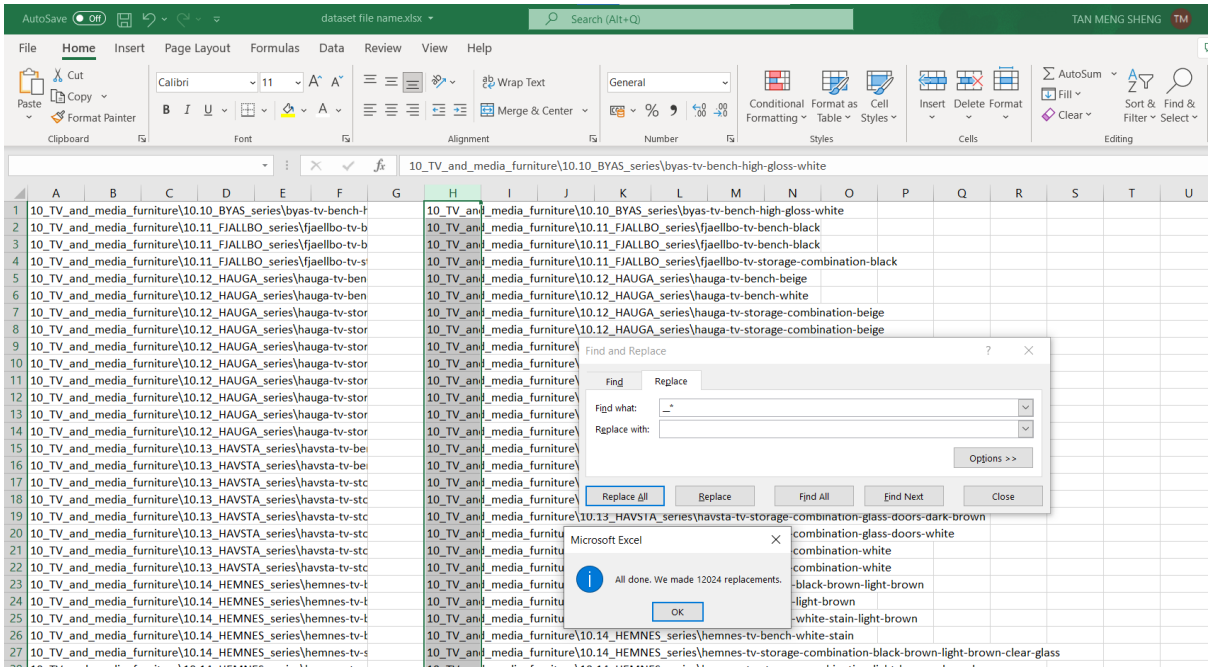


Figure 3.1.4.12 Remove the id after two underscore for opening the file

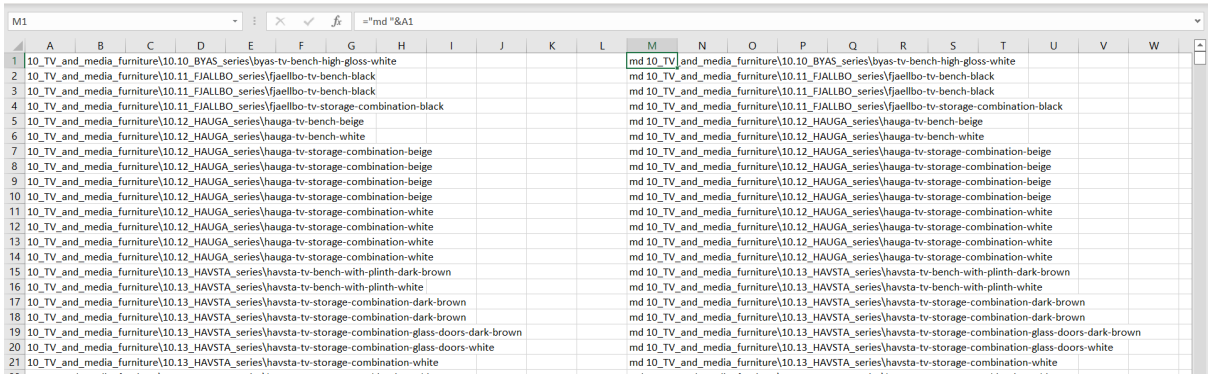


Figure 3.1.4.13 Script for making directory

After that successfully make directory by using “md” in PowerShell (refer to Figure 5.3.14), the dataset folders have been added (refer to Figure 5.3.15). To reduce the time for drag and drop from file to file, using “move” function where to move each of the jpeg dataset into regarding folder (refer to Figure 5.3.16). After that successfully move directory by using “move” in PowerShell (refer to Figure 5.3.17).

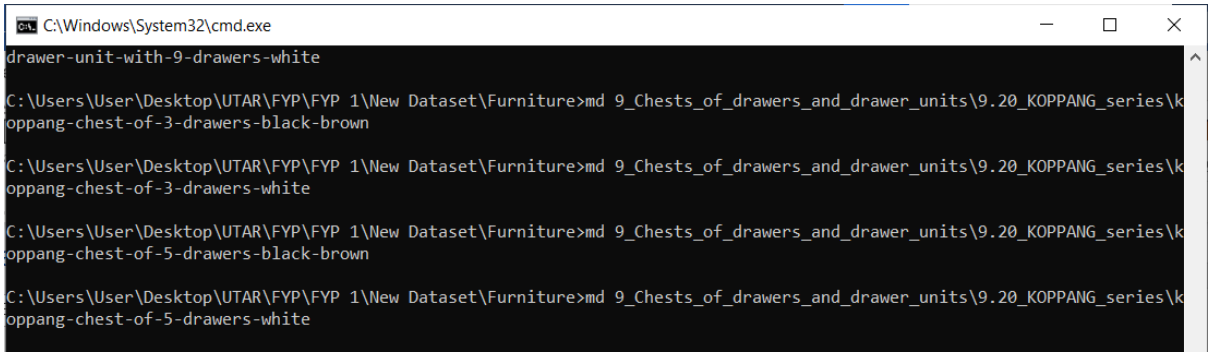


Figure 3.1.4.14 Scripting in PowerShell

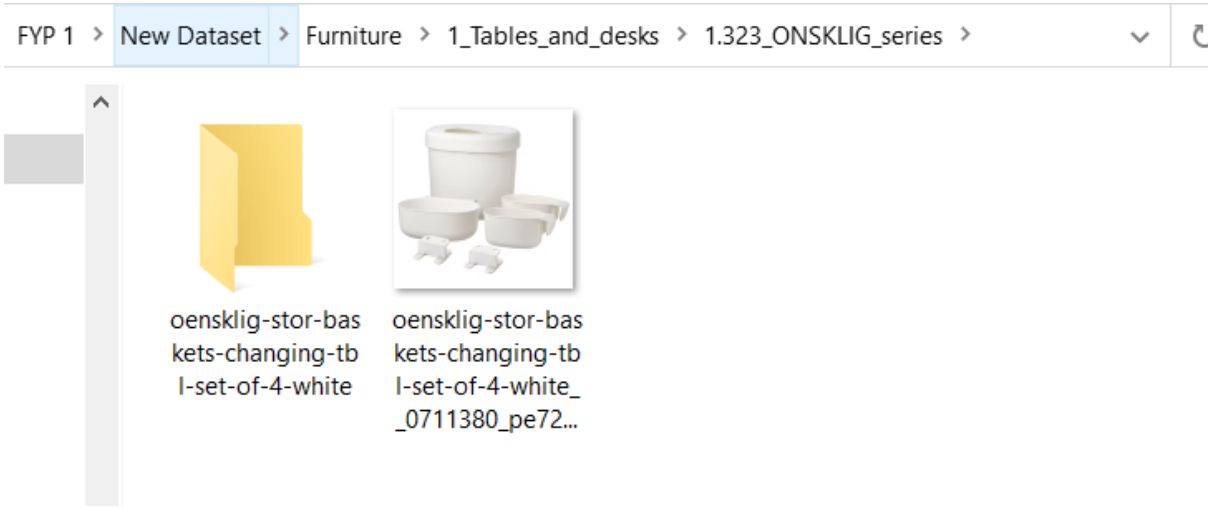


Figure 3.1.4.15 Folder that made by scripting

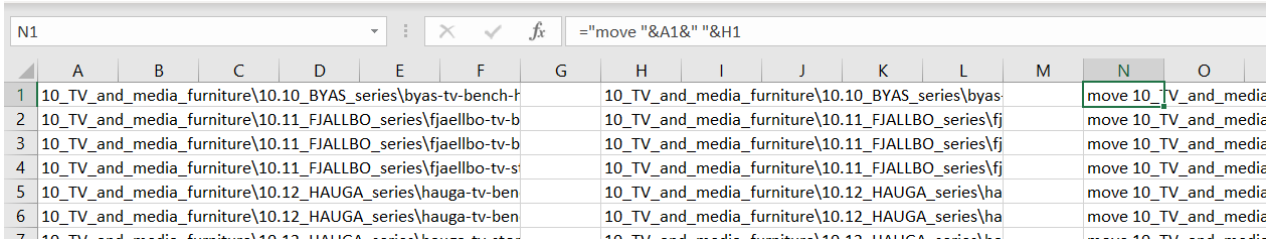


Figure 3.1.4.16 Script of move the dataset base on each of the file

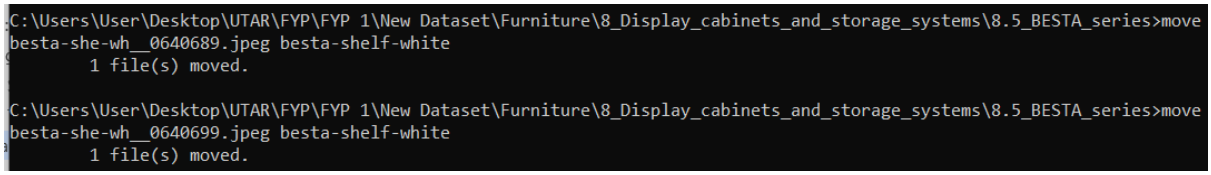


Figure 3.1.4.17 Scripting in PowerShell

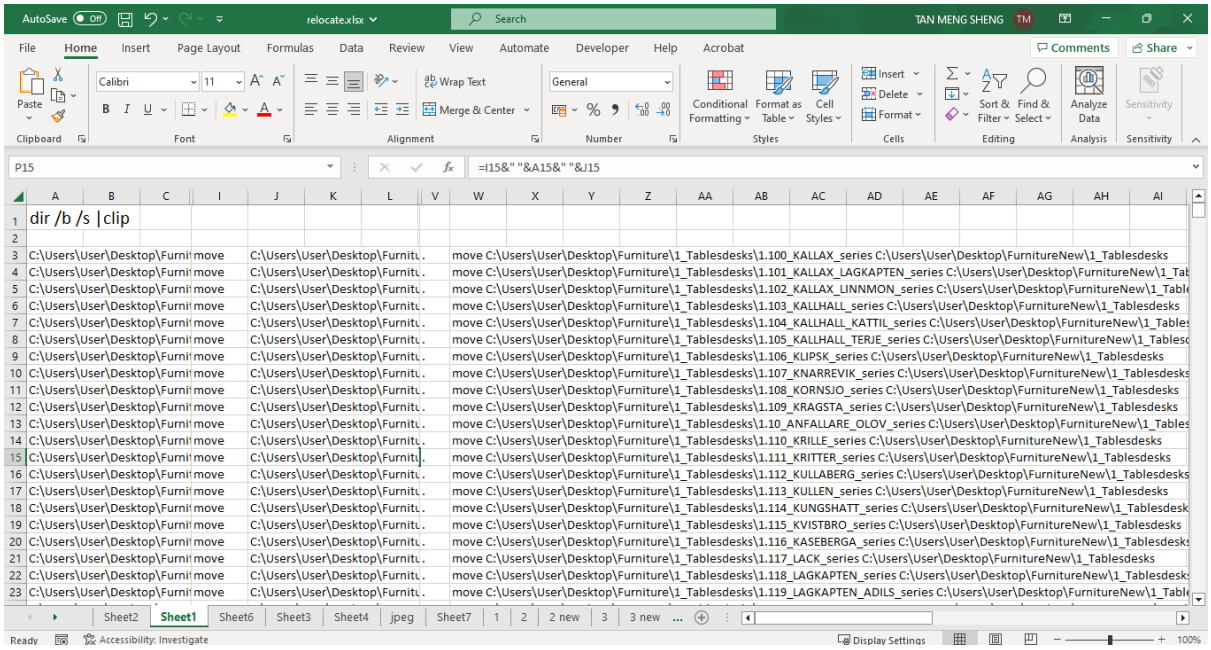


Figure 3.1.4.18 Move the directory to less layer

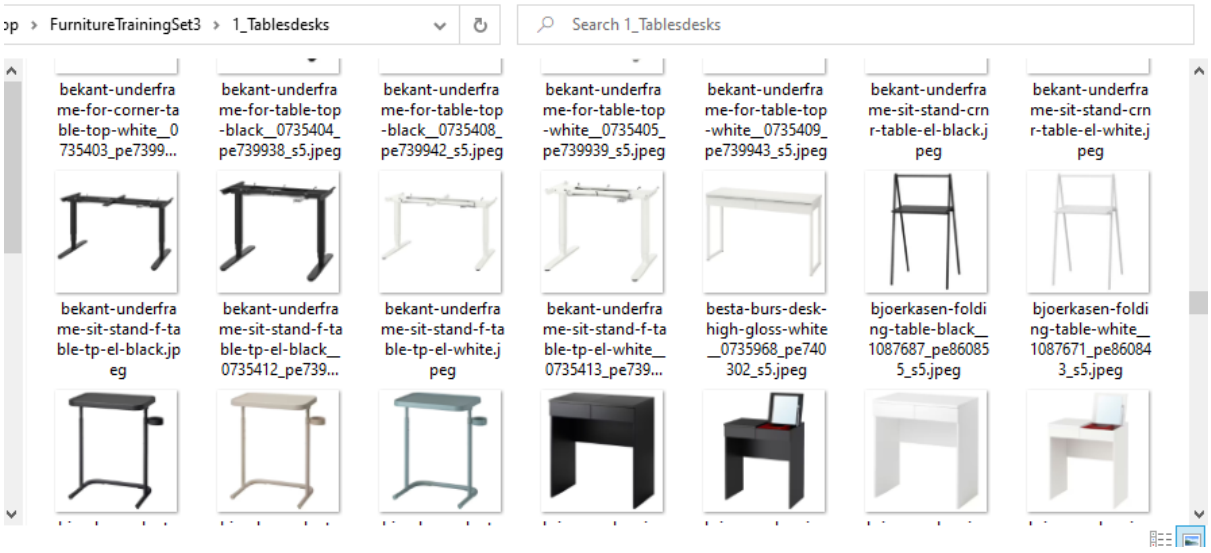


Figure 3.1.4.19 Final image dataset

Due to this dataset only need 13 class so relocate the file let all of the picture from more than 3 layer to become: Furniture/<furniture type>/<image>

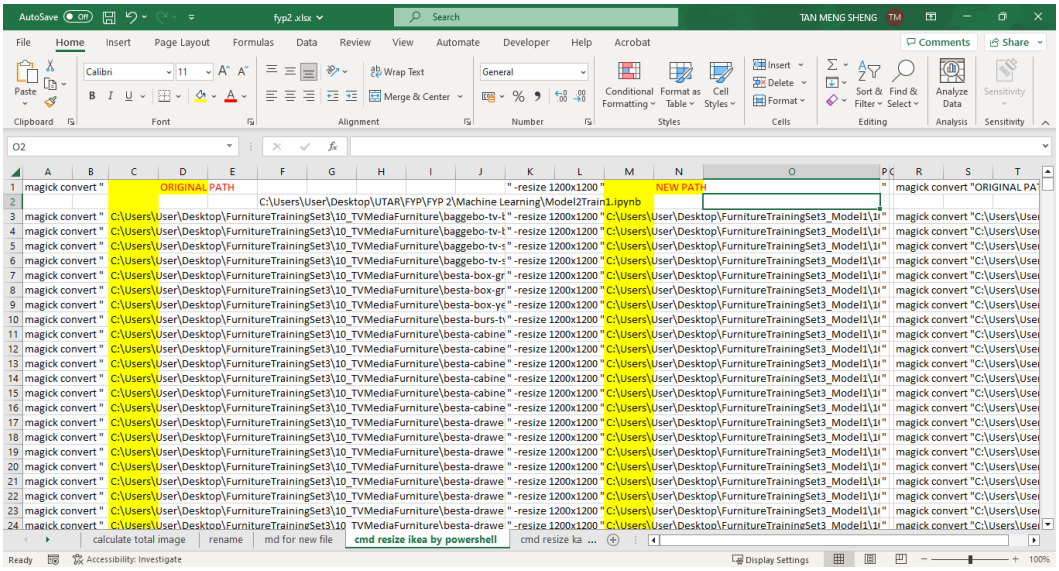


Figure 3.1.4.20 resize command

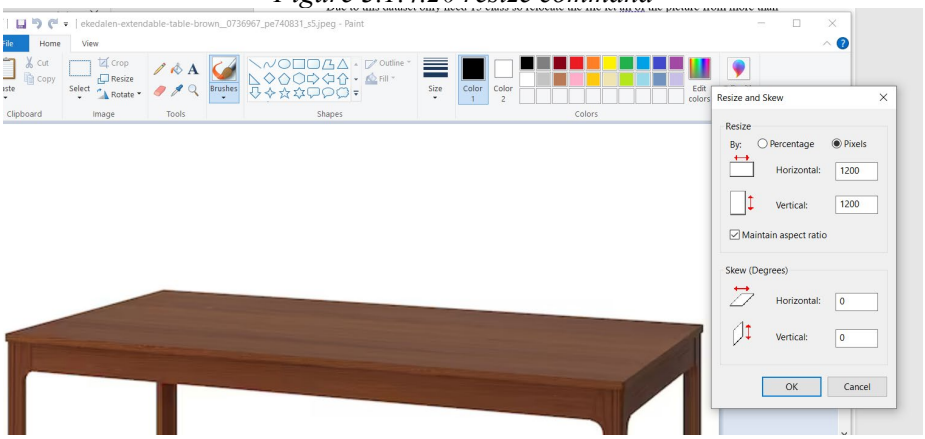


Figure 3.1.4.21 Resized image

Resizing images to a fixed resolution of 1200x1200 pixels is an important step in preparing the image dataset for computer vision analysis. This is because machine learning models require a fixed input size for the images to be processed effectively. By resizing the images to a consistent size, the machine learning model can process the images in a uniform way and make accurate predictions. This is especially important when dealing with datasets that contain images of varying sizes and resolutions.

Resizing images to a fixed resolution also has the benefit of reducing the size of the dataset, making it easier to work with and analyze. Large image sizes can take up a lot of storage space, which can be a problem for models with limited storage capacity. Resizing the images can help reduce storage space and improve the overall performance of the machine learning model.

5.4. Setting and Configuration

5.4.1. Anaconda Jupyter Notebook

Below is a list of the PIP packages used in separate image dataset to training and testing, machine learning to make h5 model, and make prediction by the h5 model produced after trained. Here are the code and the modules or libraries used for Anaconda Jupyter Notebook:

Table 5.4.1.1 Code and the modules or libraries used for Anaconda Jupyter Notebook

!pip install tensorflow from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array	1. tensorflow: An open-source machine learning framework developed by Google, used for building and training deep learning models. a. tensorflow.keras.layers: Provides classes for building different layers of a neural network, including: i. Dense: Fully connected layer ii. Conv2D: 2D convolutional layer iii. MaxPooling2D: 2D pooling layer iv. Flatten: Flattens the input b. tensorflow.keras.preprocessing.image: Provides tools for image preprocessing and data augmentation, including: i. ImageDataGenerator: Generates batches of tensor image data with real-time data augmentation ii. load_img: Loads an image from a file and returns a PIL Image object iii. img_to_array: Converts a PIL Image instance to a Numpy array

!pip install opencv-python import cv2	2. opencv-python: A library used for computer vision and image processing. a. cv2: Provides tools for loading, manipulating, and saving images.
!pip install matplotlib import matplotlib.pyplot as plt	3. matplotlib: A library used for creating visualizations, including: a. pyplot: Provides a way of plotting graphs and charts.
!pip install scikit-learn import sklearn	4. scikit-learn: A library providing tools for machine learning, including algorithms for classification, regression, clustering, and dimensionality reduction.
!pip install pillow from PIL import Image	5. pillow: A library used for image processing, including: a. Image: Provides a way of opening, manipulating, and saving image files.

After installed all of the module and library that are needed, Three steps are required to make:

1) Separate file to training(80%) and testing(20%). (Model2Separate09.ipynb)

```
# Import necessary libraries
import shutil
import random
import numpy as np
from sklearn.model_selection import train_test_split

# Set up data paths
data_dir = 'C:/Users/User/Desktop/FurnitureTrainingSet3'
train_val_dir = 'C:/Users/User/Desktop/FurnitureTrainingSet3_Model1'

# Create train and val directories
train_dir = os.path.join(train_val_dir, 'train')
val_dir = os.path.join(train_val_dir, 'val')
for dir in [train_dir, val_dir]:
    if not os.path.exists(dir):
        os.makedirs(dir)

# Split the dataset into train and val sets
class_names = os.listdir(data_dir)
for class_name in class_names:
    class_dir = os.path.join(data_dir, class_name)
    images = os.listdir(class_dir)
    random.shuffle(images)
    train_images = images[:int(0.8 * len(images))]
    val_images = images[int(0.8 * len(images)):]
    for image in train_images:
        src_path = os.path.join(class_dir, image)
        dst_path = os.path.join(train_dir, class_name, image)
        os.makedirs(os.path.dirname(dst_path), exist_ok=True)
        shutil.copy(src_path, dst_path)
    for image in val_images:
        src_path = os.path.join(class_dir, image)
        dst_path = os.path.join(val_dir, class_name, image)
        os.makedirs(os.path.dirname(dst_path), exist_ok=True)
        shutil.copy(src_path, dst_path)
```

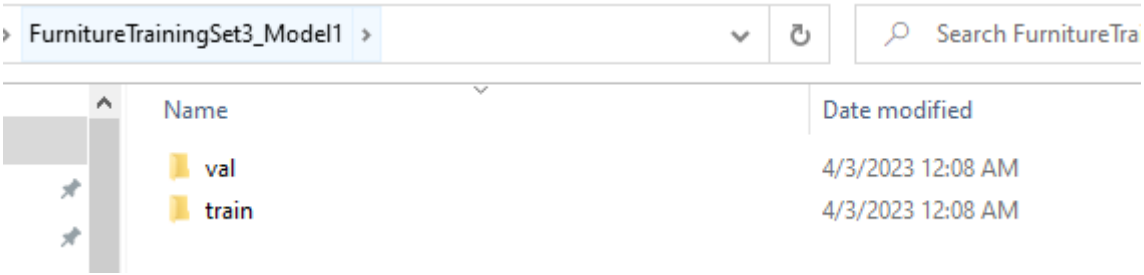


Figure 5.4.1.1 Separated file

2) Train the h5 model (Model2Train20.ipynb)

```
# Import necessary libraries
import os
import numpy as np
import tensorflow as tf
import h5py
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten

#Code to increase the memory of GPU
physical_devices = tf.config.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(physical_devices[0], True)

# Set up data paths
data_dir =  'C:/Users/User/Desktop/FurnitureTrainingSet3_Model1'
# Set up data paths
train_dir = 'C:/Users/User/Desktop/FurnitureTrainingSet3_Model1/train'
val_dir = 'C:/Users/User/Desktop/FurnitureTrainingSet3_Model1/val'

# Set up data generators for data augmentation
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
val_datagen = ImageDataGenerator(rescale=1./255)

# Set up data generator to flow from directory
train_generator = train_datagen.flow_from_directory(train_dir,
                                                    target_size=(150, 150),
                                                    batch_size=32,
                                                    class_mode='categorical')

val_generator = val_datagen.flow_from_directory(val_dir,
                                                target_size=(150, 150),
                                                batch_size=32,
                                                class_mode='categorical')

# Build the CNN model
model = Sequential()

# Convolutional layer 1
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)))
model.add(MaxPooling2D((2, 2)))

# Convolutional layer 2
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
```

```
# Convolutional layer 3
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Convolutional layer 4
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Flatten and fully connected layer
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(train_generator.num_classes, activation='softmax'))

# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
history = model.fit(train_generator,
                    steps_per_epoch=len(train_generator),
                    epochs=50,
                    validation_data=val_generator,
                    validation_steps=len(val_generator))

path = ('Model2_Train20.h5')

# Save the model
model.save(path)

print('Finished Training')
print(path +' saved')
```

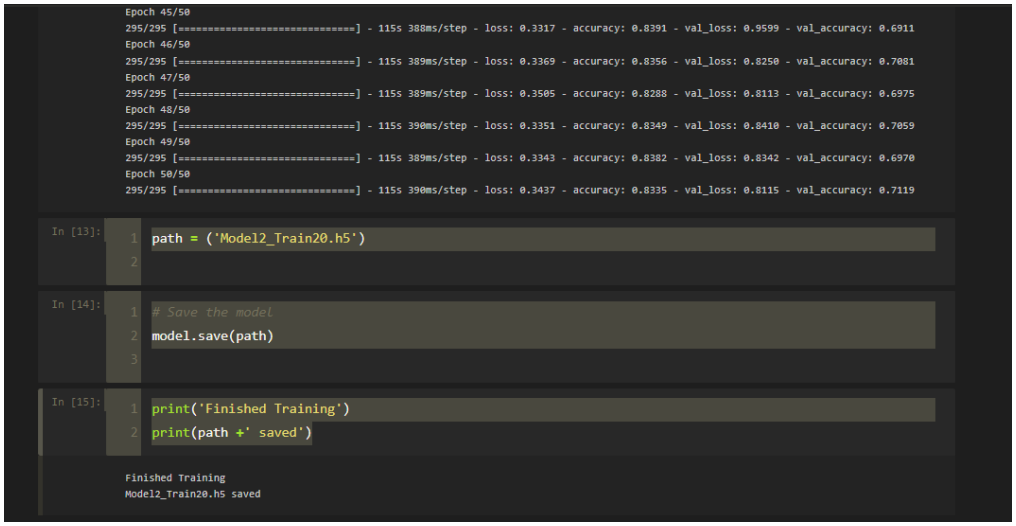


Figure 5.4.1.2 Trained model had been saved

3) Try to use the model to make prediction in Jupyter notebook (prediction_h5.ipynb)

```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img,
img_to_array
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten
import cv2
import matplotlib.pyplot as plt

# Code to increase the memory of GPU
physical_devices = tf.config.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(physical_devices[0], True)

# Set up data paths
data_dir = 'C:/Users/User/Desktop/FurnitureTrainingSet2'

# Set up data generators for data augmentation
test_datagen = ImageDataGenerator(rescale=1./255)

# Set up data generator to flow from directory
test_generator = test_datagen.flow_from_directory(
        data_dir,
        target_size=(150, 150),
        batch_size=1,
        class_mode='categorical',
        shuffle=False)

# Load the trained model
model = tf.keras.models.load_model('C:/Users/User/Desktop/UTAR/FYP/FYP 2/Machine
Learning/Week 8/Model2_Train20.h5')

# Load the test image with OpenCV
img_path = 'C:/Users/User/Desktop/UTAR/FYP/FYP 2/Machine Learning/Week
8//table2.jpeg'
test_image = cv2.imread(img_path)

# Resize the image to the correct size
test_image = cv2.resize(test_image, (150, 150))

# Convert the image to a numpy array and normalize the pixel values
test_image = np.array(test_image, dtype="float32")/255.0

# Add a batch dimension to the image (the model expects a batch of images)
test_image = np.expand_dims(test_image, axis=0)
```

```
# Predict the class of the test image
prediction = model.predict(test_image)
predicted_class = np.argmax(prediction[0])
class_prob = prediction[0][predicted_class]
class_label =
list(test_generator.class_indices.keys())[list(test_generator.class_indices.values()).index(pr
edicted_class)]

print(f"Predicted class: {class_label}")
print(f"Probability: {class_prob}")

# Get the list of file paths for images in the predicted class
image_paths = [os.path.join(data_dir, class_label, file_name) for file_name in
os.listdir(os.path.join(data_dir, class_label))]

# Shuffle the list of image paths
import random
random.shuffle(image_paths)

# Display similar images
num_similar_images = min(5, len(image_paths))  # adjust num_similar_images
fig, axs = plt.subplots(1, num_similar_images, figsize=(30, 30))  # adjust figsize
for i in range(num_similar_images):
    img = cv2.imread(image_paths[i])
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    axs[i].imshow(img)
    axs[i].axis('off')

plt.show()
```

Figure 5.4.1.3 Validation for h5 model

5.4.2. PyCharm

These are the files need to deploy to docker, are required to make by PyCharm

1. test directory: This directory contains a Python script named "test.py" along with several image files in JPEG format. The purpose of this directory is to provide a testing environment for the main program.

a) test.py

```
import requests

url = "http://127.0.0.1:5000/"
#url = "https://getprediction3-biptoq2fwq-uw.a.run.app"

picture = 'table.jpeg'
files = {'file': open(picture, 'rb')}

resp = requests.post(url, files=files)
```

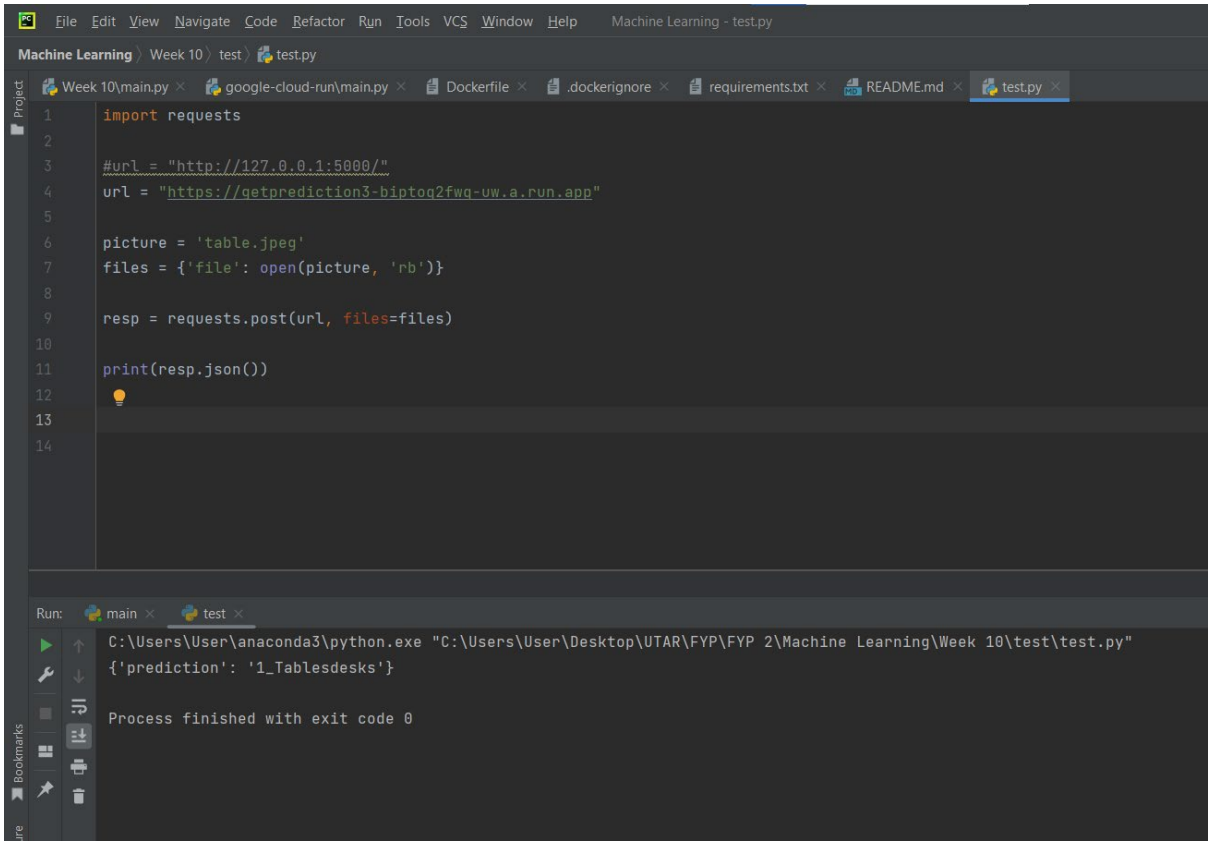


Figure 5.4.2.1 Test the accuracy by local URL

2. .dockerignore: This file specifies files and directories that should be excluded from the Docker build context. This file is used by Docker to determine which files to exclude from the image build process.

```
Dockerfile
README.md
*.pyc
*.pyo
*.pyd
__pycache__
.pytest_cache
.idea
test
```

3. README.md: This file contains information about the project, such as instructions for installation and usage.

```
### 1. Write App (Flask, TensorFlow)
- The code to build, train, and save the model is in the
`test` folder.
- Implement the app in `main.py`
### 2. Setup Google Cloud
- Create new project
- Activate Cloud Run API and Cloud Build API

### 3. Install and init Google Cloud SDK
- https://cloud.google.com/sdk/docs/install

### 4. Dockerfile, requirements.txt, .dockerignore
- https://cloud.google.com/run/docs/quickstarts/build-and-
deploy#containerizing

### 5. Cloud build & deploy
```
gcloud builds submit --tag
gcr.io/<project_id>/<function_name>
gcloud run deploy --image gcr.io/<project_id>/<function_name>
--platform managed
```

### Test
- Test the code with `test/test.py`

### Watch the video tutorial
- How To Deploy ML Models With Google Cloud Run

[![Alt
text](https://img.youtube.com/vi/vieoHqt7pxo/hqdefault.jpg)](
https://youtu.be/vieoHqt7pxo)
```

4. Model2_Train20.h5: This is a machine learning model file that has been trained to recognize different types of furniture.

5. requirements.txt: This file contains a list of Python dependencies that need to be installed in order for the program to run.

```
Flask
gunicorn
tensorflow
numpy
pillow
```

6. Dockerfile: This file is used to build the Docker image. It specifies the base image to use, sets up the environment, installs dependencies, and copies the necessary files to the image.

```
# Use the official lightweight Python image.
# https://hub.docker.com/_/python
FROM python:3.10-slim

# Allow statements and log messages to immediately appear in the
Knative logs
ENV PYTHONUNBUFFERED True

# Copy local code to the container image.
ENV APP_HOME /app
WORKDIR $APP_HOME
COPY . ./

# Install production dependencies.
RUN pip install -r requirements.txt

# Run the web service on container startup. Here we use the gunicorn
# webserver, with one worker process and 8 threads.
# For environments with multiple CPU cores, increase the number of
workers
# to be equal to the cores available.
# Timeout is set to 0 to disable the timeouts of the workers to allow
Cloud Run to handle instance scaling.
CMD exec gunicorn --bind :$PORT --workers 1 --threads 8 --timeout 0
main:app
```

7. main.py: This is the main Python script for the program. It uses the machine learning model to classify images of furniture and returns the results. The program can be run using a web API or from the command line.

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

import io
import tensorflow as tf
from tensorflow import keras
import numpy as np
from PIL import Image

from flask import Flask, request, jsonify

model_path = "Model2_Train20.h5"
model = keras.models.load_model(model_path)


class_labels = {
    0: "10_TVMediaFurniture",
    1: "11_OutdoorFurniture",
    2: "12_ChildrensFurniture",
    3: "13_NurseryFurniture",
    4: "1_Tablesdesks",
    5: "2_ChairsStoolsbenches",
    6: "3_Sofas",
    7: "4_ArmchairsChaiseLongues",
    8: "5_BedFrames",
    9: "6_Wardrobes",
    10: "7_BookcasesShelvingUnits",
    11: "8_DisplayCabinetsStorageSystems",
    12: "9_ChestsOfDrawersDrawerUnits"
}

def transform_image(pillow_image):
    # Resize image to (150, 150) and convert to RGB
    pillow_image = pillow_image.resize((150, 150)).convert('RGB')
    data = np.asarray(pillow_image)
    data = data / 255.0
    data = data[np.newaxis, ...]
    # --> [1, 150, 150, 3]
    return data

def predict(x):
    pred = model(x)
    pred = tf.nn.softmax(pred)
    label = np.argmax(pred)
    return label

app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":
        file = request.files.get('file')
        if file is None or file.filename == "":
            return jsonify({"error": "no file"})
```

```
        try:
            image_bytes = file.read()
            pillow_img =
Image.open(io.BytesIO(image_bytes)).convert('L')
            tensor = transform_image(pillow_img)
            prediction = predict(tensor)
            label = class_labels[prediction]
            data = {"prediction": label}
            return jsonify(data)
        except Exception as e:
            return jsonify({"error": str(e)})

    return "OK"


if __name__ == "__main__":
    app.run(debug=True)
```

5.4.3. Docker

1) After make all the files in Pycharm, go to the directory where save all of the files and type cmd.exe to open command prompt.

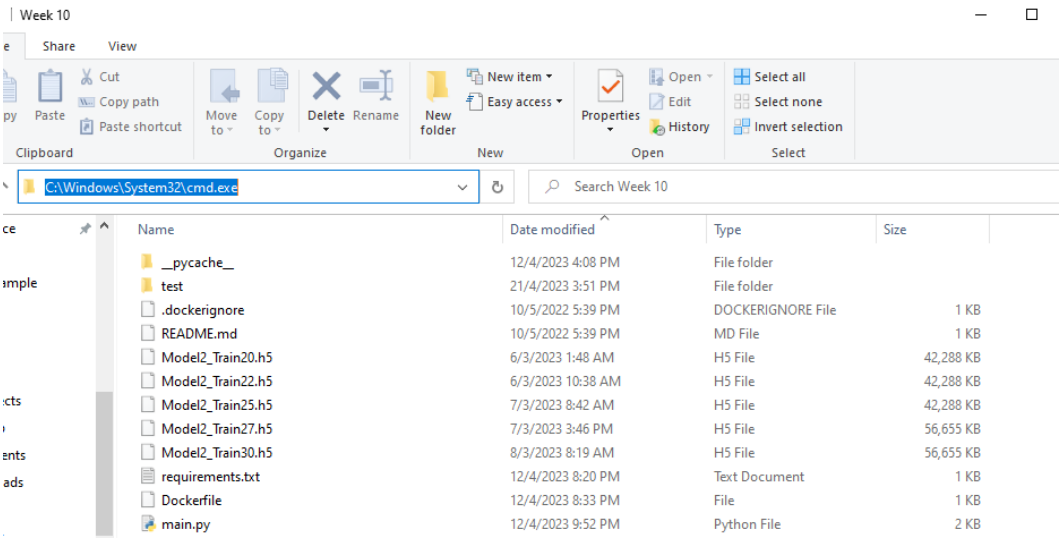


Figure 5.4.3.1 Open command prompt in particular directory

2) Then execute this code in the command prompt, need some time to build the docker image in Docker.

```
docker build -f Dockerfile -t furniture_docker_file .
```

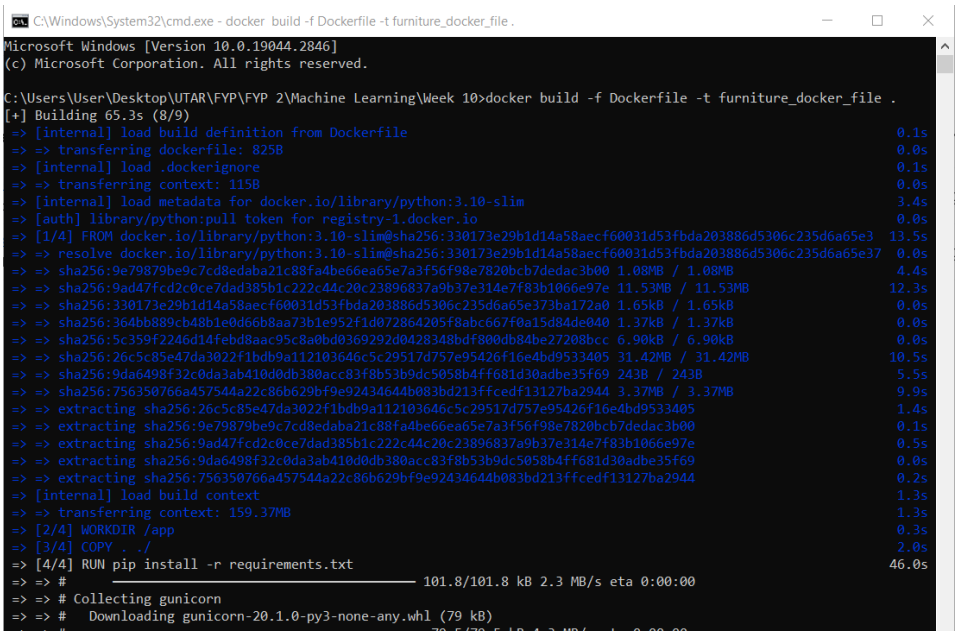


Figure 5.4.3.2 Build the docker image in Docker

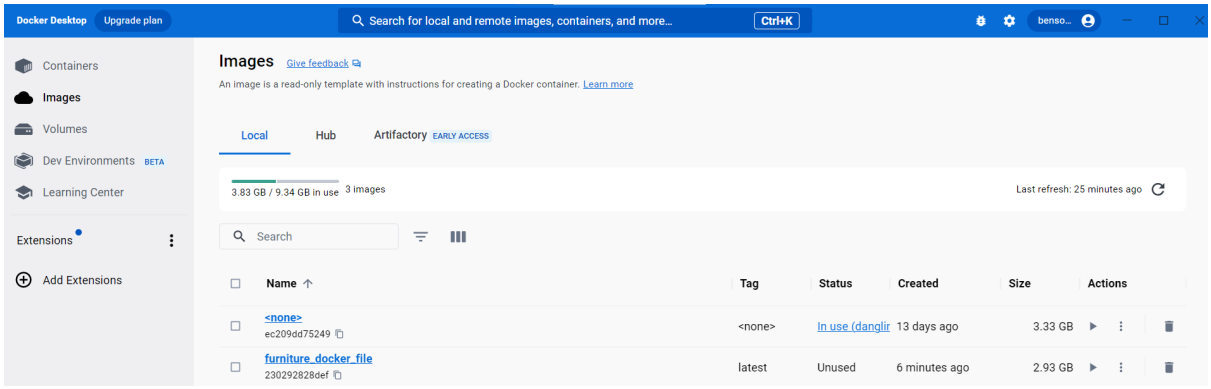


Figure 5.4.3.3 Uploaded Docker Image

3) Then create a container by this code

```
docker run -ti furniture_docker_file /bin/bash
```

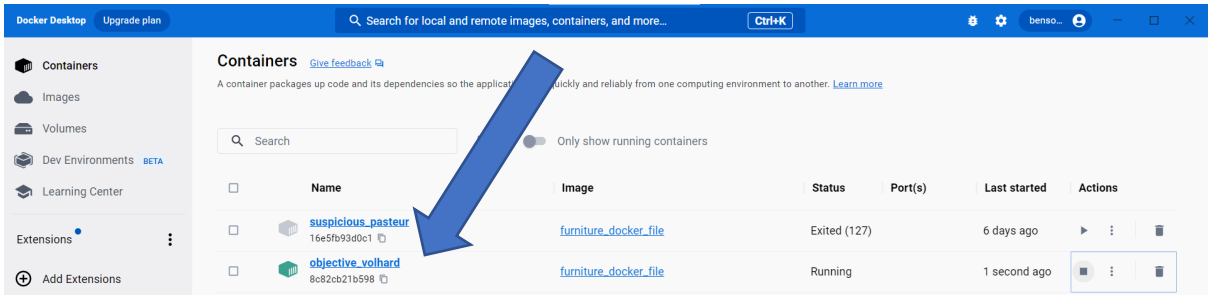


Figure 5.4.3.4 Create a container

or use this code to go the same container

```
docker exec -it objective_volhard /bin/bash
```

4) then list all the name, change the file from app to src

```
mv app src
ls
cd src
ls
```

```
root@4b2c60a2f865:/app# cd ../
root@4b2c60a2f865:/# mv app src
root@4b2c60a2f865:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  src  srv  sys  tmp  usr  var
root@4b2c60a2f865:/# cd src
root@4b2c60a2f865:/src# ls
Model2_Train20.h5  Model2_Train25.h5  Model2_Train30.h5  docker_data  requirements.txt
Model2_Train22.h5  Model2_Train27.h5  docker             main.py
root@4b2c60a2f865:/src#
```

Figure 5.4.3.5 List the file in src file

5.4.4. Google Cloud Platform

Google Cloud Platform (GCP) is a cloud computing platform offered by Google. It provides various services such as computing, storage, machine learning, and data analytics that are accessible over the internet.

GCP offers a 90-day free trial with a credit of $300 to explore its services. This trial period is intended to help users understand and test the platform's capabilities. During the trial period, users can try out various services and features of GCP without incurring any cost up to the limit of the credit. Once the credit is exhausted or the trial period ends, users will be charged for the services they continue to use.

1) Create new project

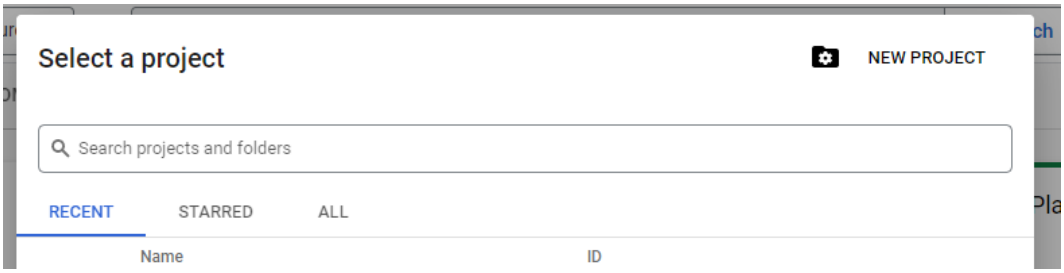


Figure 5.4.4.1 Create new project

2) Set the name to furniture, then click 'create'

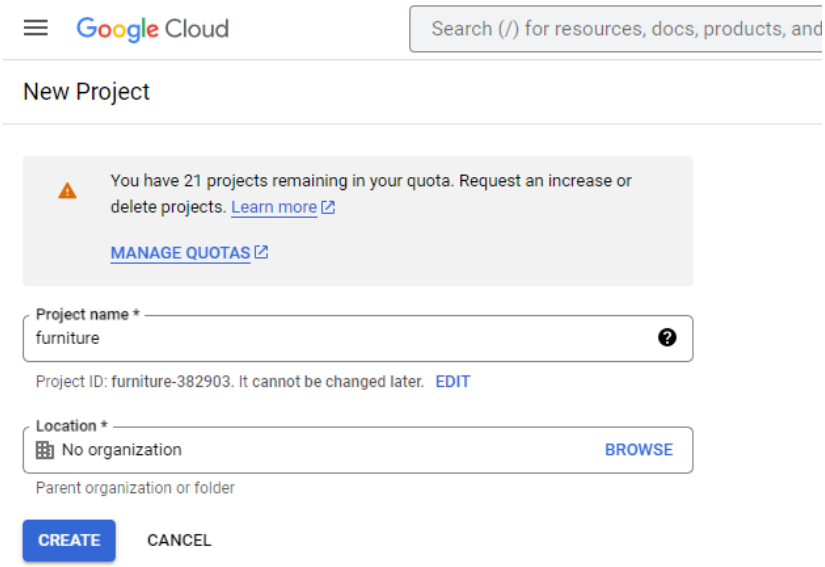


Figure 5.4.4.2 Define project name

3) New project created.

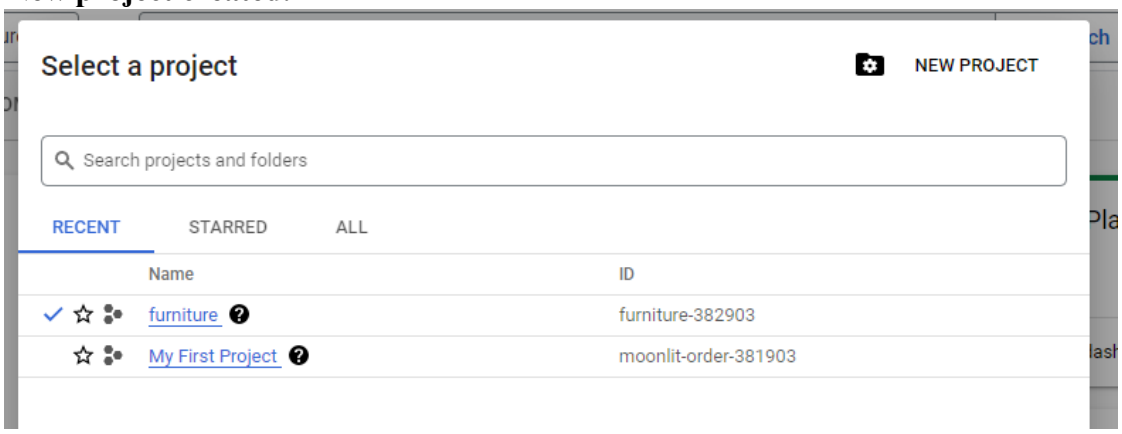


Figure 5.4.4.3 Created project

4) Create new service for Google Cloud Run and Cloud Storage

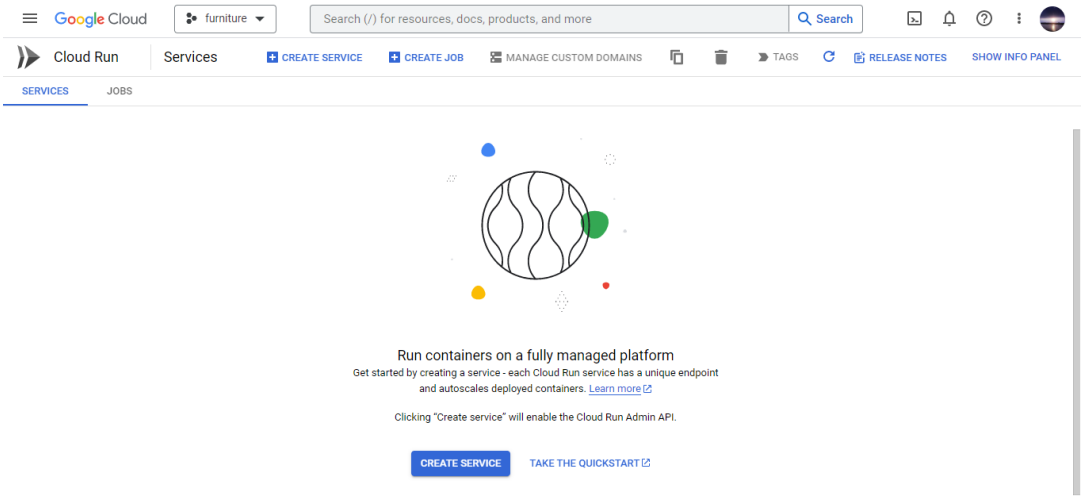


Figure 5.4.4.4 Create new service for Google Cloud Run

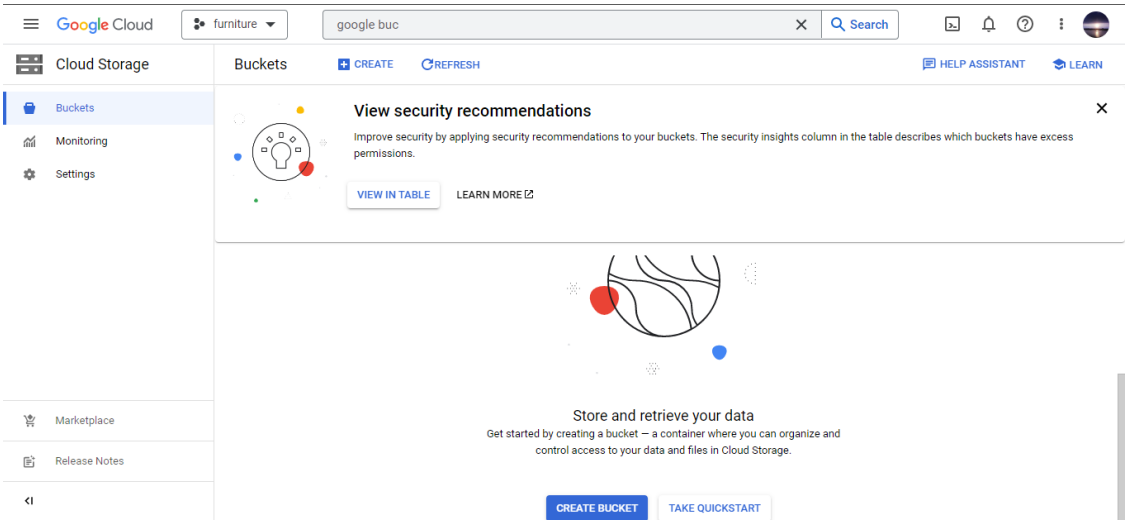


Figure 5.4.4.5 Create new service for Cloud Storage

5.4.5. Firebase

Firebase is a cloud-based platform that provides a range of services, including cloud storage for storing and retrieving files such as images. You can use Firebase Storage to store your image data and retrieve them later.

To store an image in Firebase Storage, need to initialize a Firebase App and configure Firebase Storage in your code. Then, use the Firebase Storage SDK to upload the image file to Firebase Storage. Once uploaded, you can access the image by its download URL or other metadata.

This is a command to copy files or directories from this project local machine (C:/Users/User/Desktop/FurnitureTrainingSet2) to a Google Cloud Storage bucket (gs://furniture-382903.appspot.com).

gsutil is a command-line tool provided by Google Cloud SDK to interact with Google Cloud Storage. -m is a flag to enable multi-threaded transfers for better performance. cp is the command to copy, and -r is a flag to copy directories recursively.

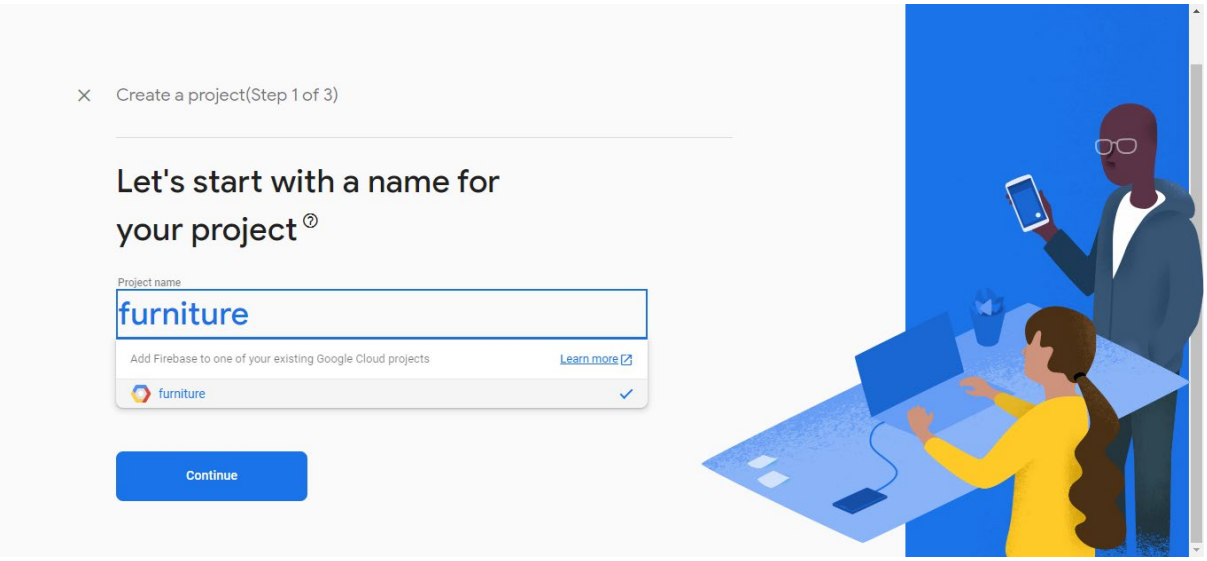


Figure 5.4.5.1 Step 1 to create Firebase project

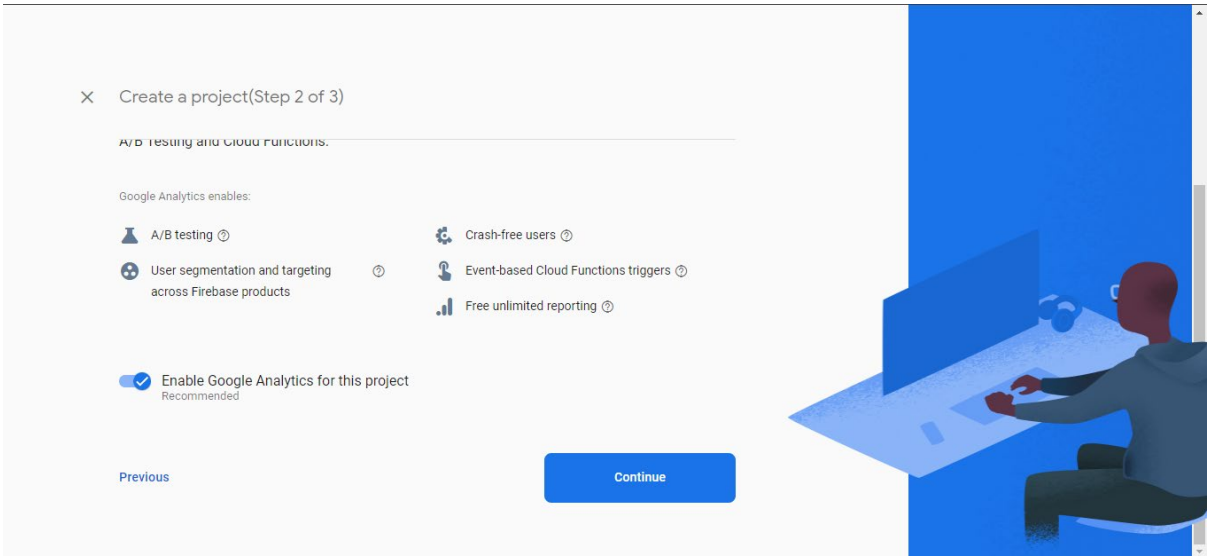


Figure 5.4.5.2 Step 2 to create Firebase project

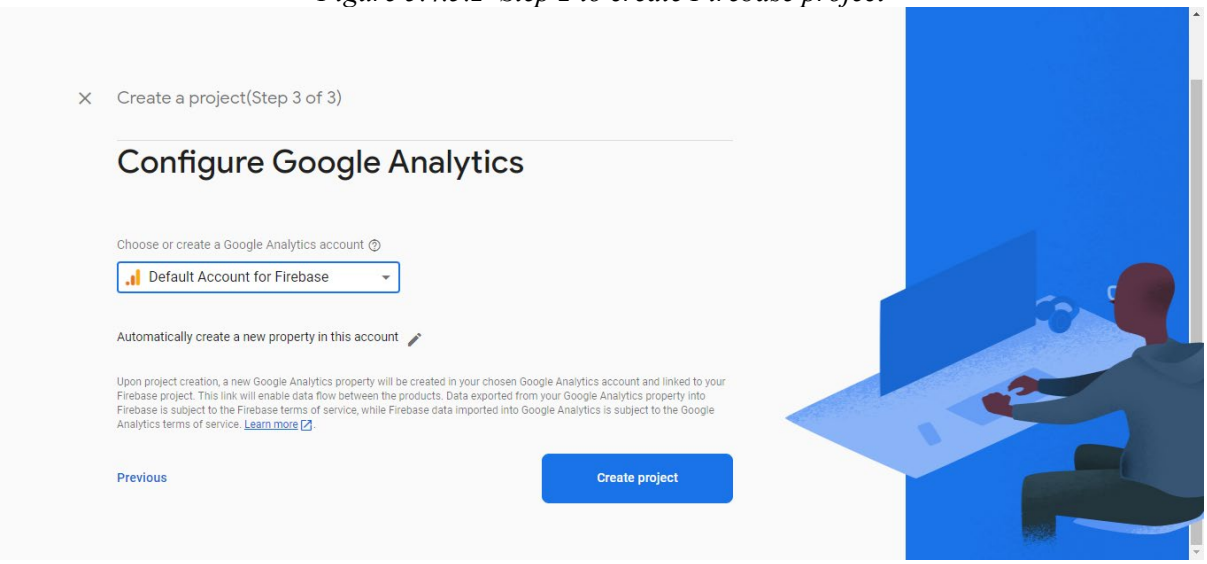


Figure 5.4.5.3 Step 3 to create Firebase project

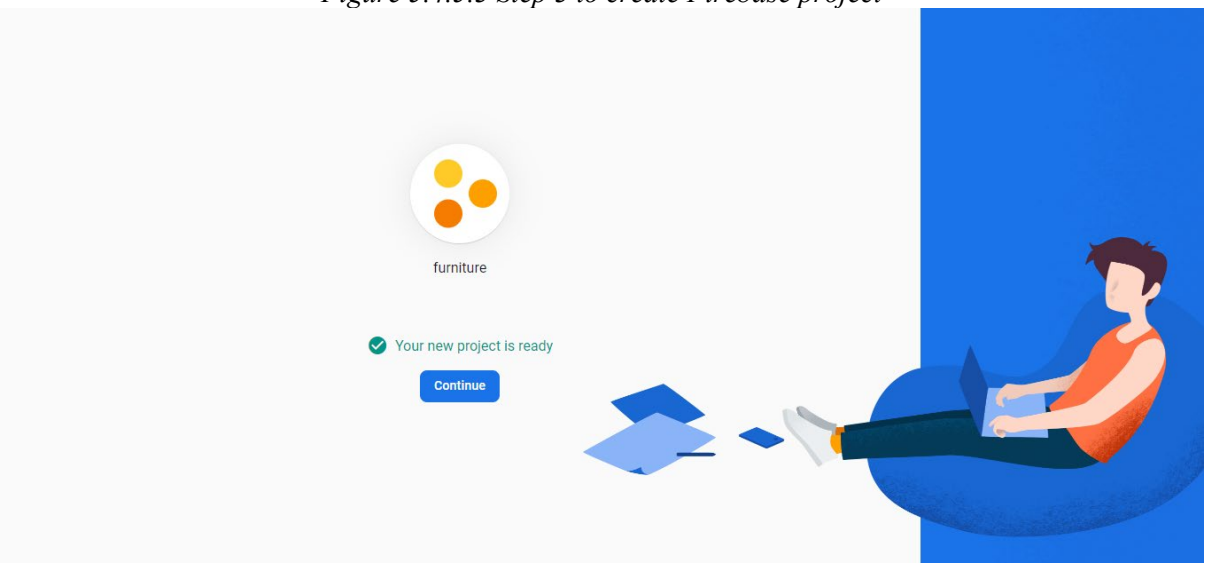


Figure 5.4.5.4 Success to create Firebase project

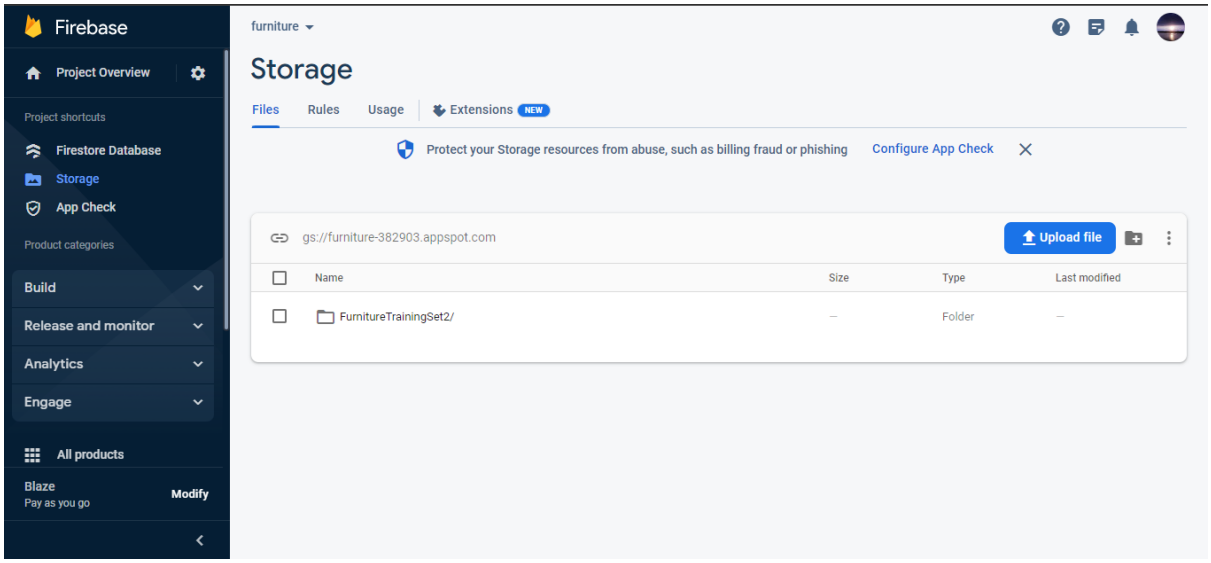


Figure 5.4.5.5 Firebase storage is created

So in summary, this command will copy the FurnitureTrainingSet2 directory and all of its contents from this project local machine to a Google Cloud Storage bucket named furniture-382903.appspot.com.

```
gsutil -m cp -r C:/Users/User/Desktop/FurnitureTrainingSet2 gs://furniture-382903.appspot.com
```

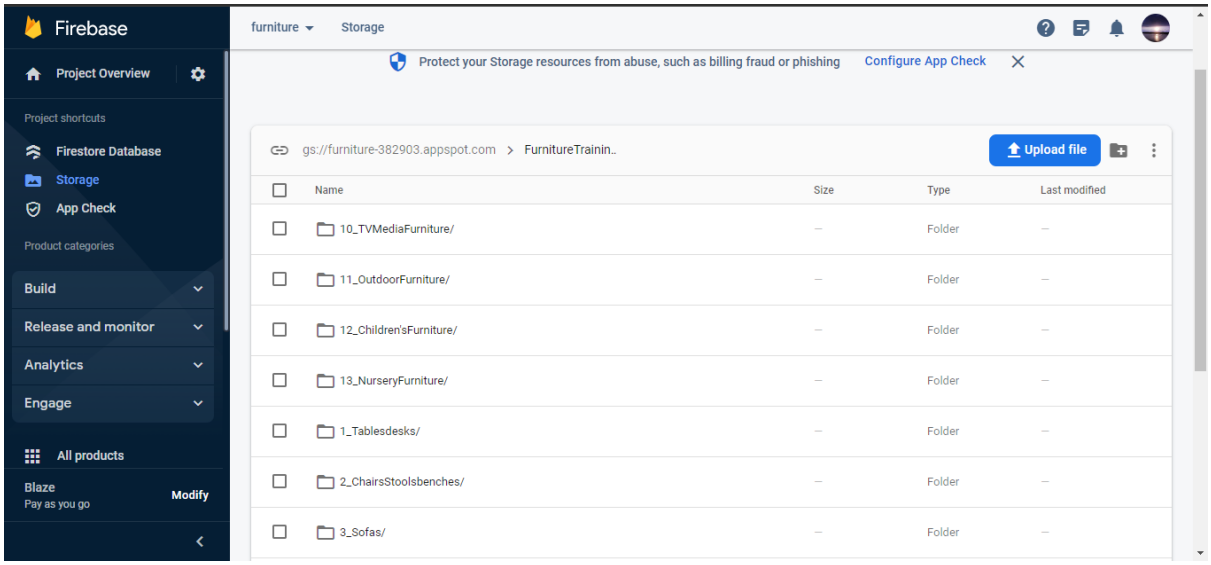


Figure 5.4.5.6 Furniture set 2 in Firebase

5.4.6. Android Studio

Android Studio is a powerful integrated development environment (IDE) used for developing Android applications. In this project, Android Studio is used to create the user interface, write code for the application logic, and test and debug the app. In addition, Android Studio integrates with other tools and services that can be useful for this project, such as Firebase for backend services and Google Cloud Platform.

Packages and settings listed in the pubspec.yaml

Table 5.4.6.1 Packages and Settings Listed in the dependencies of pubspec.yaml

Dependencies	Definitions
fluttertoast: ^8.1.1	This package is used to display toast messages in the app.
motion_toast: ^2.4.3	This package is used to display motion-based toast messages in the app.
flutter_app_name: ^0.1.1	This package is used to get the name of the app as defined in the app's pubspec.yaml file.
image_picker: ^0.8.7	This package is used to select images from the device's gallery or camera.
camera: ^0.10.3+2	This package is used to capture images using the device's camera.
googleapis: ^10.1.0	This package is used to access Google APIs from the app.
http: ^0.13.4	This package is used to make HTTP requests and receive responses in the app.
firebase_core: ^2.9.0	This package is required to use Firebase services in the app.
firebase_storage: ^11.1.0	This package is used to store files in the Firebase Cloud Storage.
cloud_firestore: ^4.5.0	This package is used to store and retrieve data from the Firebase Cloud Firestore.
transparent_image: ^2.0.1	This package is used to display transparent images in the app.
file_picker: ^5.2.9	This package is used to select files from the device's file system.

flutter_secure_storage: ^8.0.0	This package is used to store sensitive data securely in the app.
firebase_app_check: ^0.1.2	This package is used to verify the authenticity of requests made to Firebase services.
permission_handler: ^10.2.0	This package is used to handle permissions for accessing device features such as camera, gallery, location, etc.
gallery_saver: ^2.3.2	This package is used to save images or videos to the device's gallery.
firebase_analytics: ^10.2.0	This package is used to track user behavior and app usage statistics.
flutter_native_splash: ^2.0.1+1	This package is used to add a custom splash screen to the app.
connectivity: ^3.0.6	This package is used to monitor the device's network connectivity.
app_settings: ^4.2.0	This package is used to open the device's app settings from within the app.

The code need to import in storage_service.dart:

```
import 'package:firebase_storage/firebase_storage.dart' as
firebase_storage;
import 'package:firebase_core/firebase_core.dart' as firebase_core;
import 'dart:io';
```

The code need to import in main.dart:

```
import 'dart:async';
import 'dart:io';
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:ikea_furniture_finder/storage_service.dart';
import 'package:image_picker/image_picker.dart';
import 'package:camera/camera.dart';
import 'package:http/http.dart' as http;
import 'package:firebase_storage/firebase_storage.dart' as
firebase_storage;
import 'package:firebase_core/firebase_core.dart' as firebase_core;
import 'dart:convert';
import 'package:firebase_app_check/firebase_app_check.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:flutter_native_splash/flutter_native_splash.dart';
import 'package:connectivity/connectivity.dart';
```

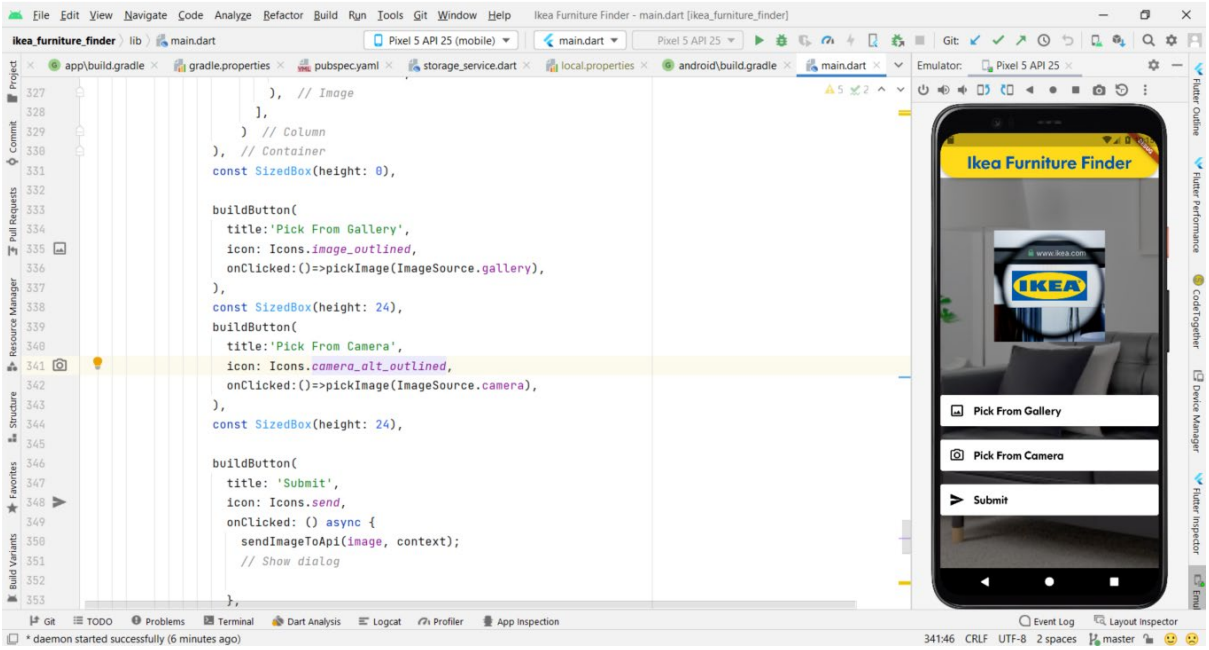


Figure 5.4.6.1 Android Studio interface

The configuration settings related to the app's environment SDK, splash screen, icons, assets (including images), and fonts in pubspec.yaml.

Table 5.4.6.2 Set up for pubspec.yaml

The use of the elements	Code
Environment of sdk	`environment:` `  sdk: '>=2.18.0 <3.0.0'`

flutter_native_splash
Specifies the app's splash screen settings such as background image and platform support (Android and iOS).
	`flutter_native_splash:`
	`  background_image:`
	`'assets/ikealaunch.png'`
	`  android: true`
	`  ios: true`

flutter_icons
Configures the app's icon for different platforms (Android, iOS, web, and Windows).
	`flutter_icons:`
	`  android: true`
	`  ios: true`
	`  remove_alpha_ios: true`
	`  image_path: "assets/ikeaapp.png"`
	`  min_sdk_android: 21 # android min sdk min:16, default 21`
	`  web:`
	`  generate: true`
	`  image_path:`
	`"assets/ikeaapp.png"`
	`  background_color: #ffffff`
	`  theme_color: #ffffff`
	`  windows:`
	`  generate: true`
	`  image_path:`
	`"assets/ikeaapp.png"`
	`  icon_size: 48 # min:48, max:256, default: 48`

Assets
Lists all the app's assets (images in this case) and their respective paths.
Six background image will repeat to show at background of the app
Ikealaunch.png is a splash screen when user use the app
	`assets:`
	`  - assets/IkeaLogo.png`
	`  - assets/ikeaapp.png`
	`  - assets/background_image_1.png`
	`  - assets/background_image_2.png`
	`  - assets/background_image_3.png`
	`  - assets/background_image_4.png`
	`  - assets/background_image_5.png`
	`  - assets/background_image_6.png`
	`  - assets/ikealaunch.png`

Fonts
Specifies the app's font family and its corresponding font files.
	`fonts:`
	`  - family: IKEA-Sans`
	`  fonts:`
	`  - asset: fonts/IKEA-Sans-Regular.ttf`

Figure 5.4.6.2 IkeaLogo.png

Figure 5.4.6.3 ikeaapp.png

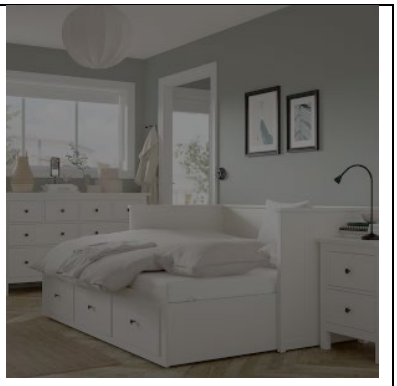


Figure 5.4.6.4 background_image_1.png

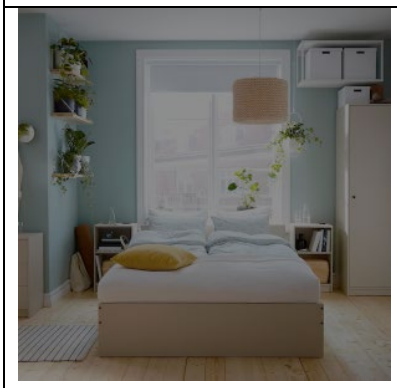


Figure 5.4.6.5 background_image_2.png

Figure 5.4.6.6 background_image_3.png

Figure 5.4.6.7 background_image_4.png

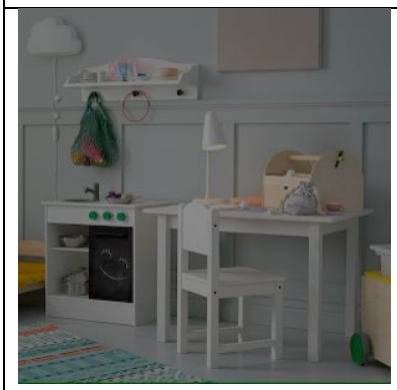


Figure 5.4.6.8 background_image_5.png

Figure 5.4.6.9 background_image_6.png

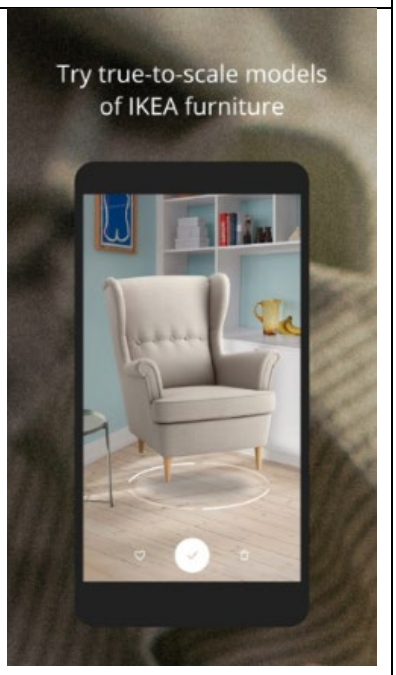


Figure 5.4.6.10 ikealaunch.png

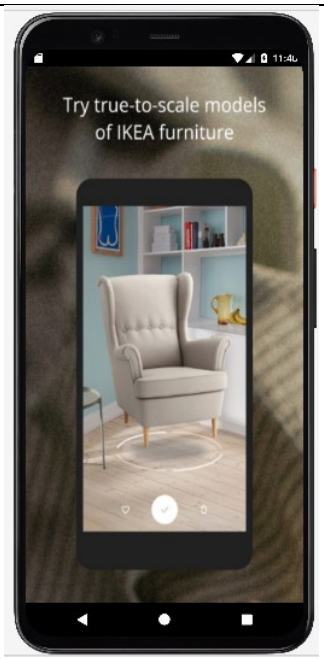


Figure 5.4.6.11 Splash Screen

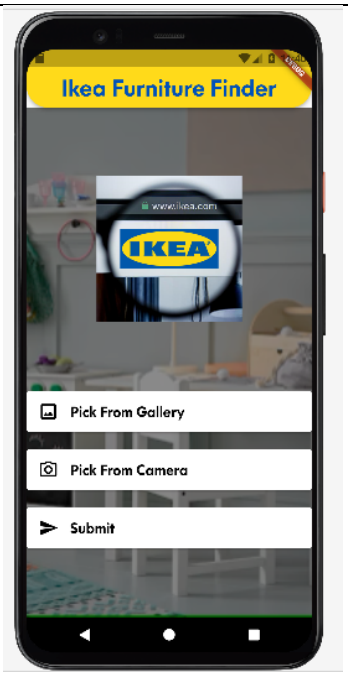


Figure 5.4.6.12 Background Picture demo 1

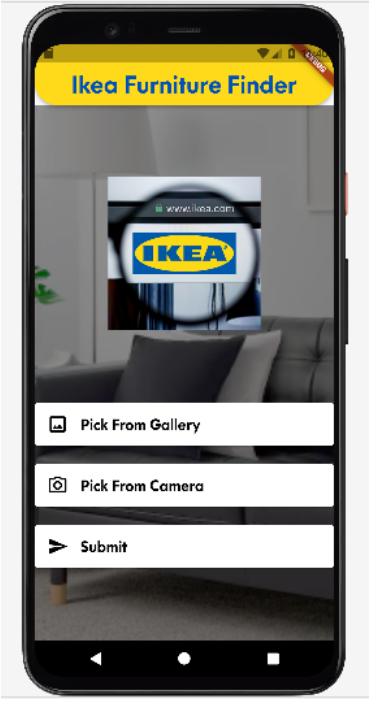


Figure 5.4.6.13 Background Picture demo 2

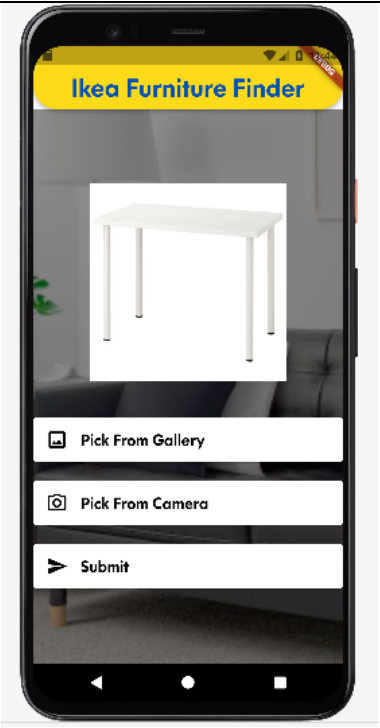


Figure 5.4.6.14 Picture will change after inserting image

Figure 5.4.6.11 showed that in the splash screen, Figure 5.4.6.12 and Figure 5.4.6.13 showed that in the main screen, it had 3 buttons, Pick from gallery, Pick from camera, and submit, the background image will looping. Figure 5.4.6.14 show the image picked by user is showed and replace the 'Ikea' image.

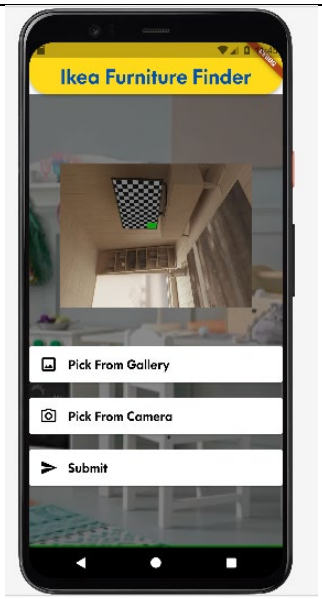


Figure 5.4.6.15 The image will fit the original orientation

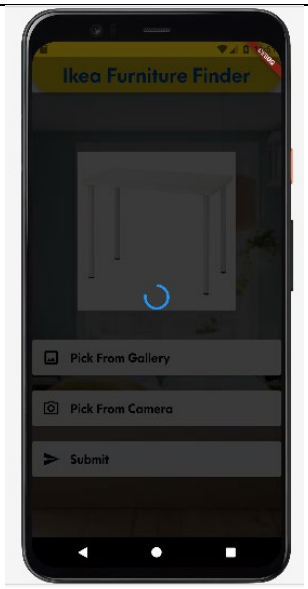


Figure 5.4.6.16 After clicking submit will show transparent grey overlay

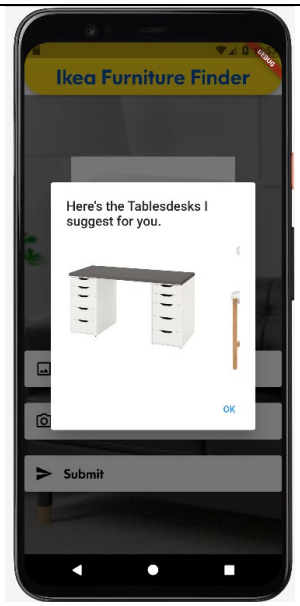


Figure 5.4.6.17 Will show the type of Furniture and show image from Firebase

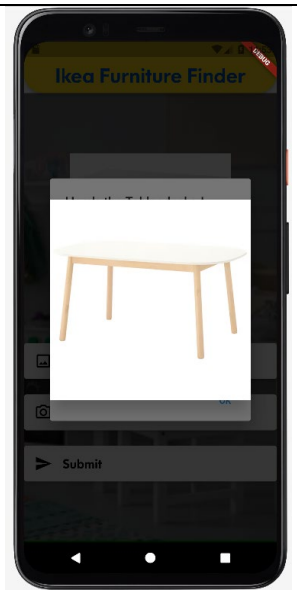


Figure 5.4.6.18 Can select either image and bigger the image

Also, this project Wi-Fi and data check after splash screen. Before use the Camera, it will ask for the permission to use camera. While click submit, it will show a transparent grey overlay cover the screen and after loading (Figure 5.4.6.16), it will show the type of the furniture: Tabledesk (Figure 5.4.6.16) predicted by GCP Cloud Run and 10 image. In Figure 5.4.6.18, user can select either image and bigger the image.

main.dart

```
import 'dart:async';
import 'dart:io';
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:ikea_furniture_finder/storage_service.dart';
import 'package:image_picker/image_picker.dart';
import 'package:camera/camera.dart';
import 'package:http/http.dart' as http;
import 'package:firebase_storage/firebase_storage.dart' as
firebase_storage;
import 'package:firebase_core/firebase_core.dart' as firebase_core;
import 'dart:convert';
import 'package:firebase_app_check/firebase_app_check.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:flutter_native_splash/flutter_native_splash.dart';
import 'package:connectivity/connectivity.dart';


final Storage storage = Storage();

Future main() async{
  WidgetsFlutterBinding.ensureInitialized();
  await Future.delayed(const Duration(seconds: 3));
  FlutterNativeSplash.remove();
  await Firebase.initializeApp();
  var connectivityResult = await Connectivity().checkConnectivity();
  if (connectivityResult != ConnectivityResult.wifi && connectivityResult
!= ConnectivityResult.mobile) {
    // If not connected to WiFi or mobile data, show an error message and
exit the app
    runApp(MaterialApp(
      home: Builder(
        builder: (context) => AlertDialog(
          title: Text('No Internet Connection'),
          content: Text('Please connect to a WiFi or mobile data network
to use Ikea Furniture Finder.'),
          actions: [
            TextButton(
              child: Text('OK'),
              onPressed: () => exit(0),
            ),
          ],
        ),
      ),
    ));
    return;
  }

  runApp(MaterialApp(
    home: MyApp(),
  ));
}

Future<void> sendImageToApi(File? image, BuildContext context) async {
  if (image == null) return;
```

```
  // Get an App Check token
  final appCheckToken = await FirebaseAppCheck.instance.getToken();

  // show a transparent grey overlay over the screen
  showDialog(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) {
      return Container(
        color: Colors.black.withOpacity(0.5),
        child: Center(
          child: CircularProgressIndicator(),
        ),
      );
    },
  );

  // Send the image to the API endpoint
  final url = Uri.parse("https://getprediction3-biptoq2fwq-
uw.a.run.app");
  final request = http.MultipartRequest("POST", url);
  request.headers.addAll({'Authorization': 'Bearer $appCheckToken'});
  request.files.add(await http.MultipartFile.fromPath("file",
image.path));
  final response = await request.send();
  final responseBody = await response.stream.bytesToString();
  final jsonMap = json.decode(responseBody);
  final prediction = jsonMap['prediction'];

  // Modify the prediction value to include the path
  final String furnitureType = prediction;
  final furnitureTypeWithoutPrefix = furnitureType.split("_")[1]; //
"Sofas"
  final String furnitureTypeWithoutPrefixAndSplit =
furnitureTypeWithoutPrefix.split(RegExp(r'(?=[A-Z])')).join(' ');
  print(furnitureTypeWithoutPrefixAndSplit);//Prints "Sofas"
  print(furnitureType); // Prints "3_Sofas"

  // Show a message to the user
  final message = furnitureType.isNotEmpty ? "Image sent successfully" :
"Failed to send image";
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text(message),
      duration: Duration(seconds: 2),
    ),
  );


  final storage = FirebaseStorage.instance;
  final ref =
storage.ref().child('FurnitureTrainingSet2/$furnitureType');

  final result = await ref.listAll();

  // Get a list of all the JPEG file URLs under the specified directory
  final List<String> jpegUrls = await Future.wait(result.items
      .where((item) => item.name.toLowerCase().endsWith('.jpeg'))
      .map((item) => item.getDownloadURL())
      .toList());
```

```
  // Shuffle the list of JPEG URLs
  jpegUrls.shuffle();

  // Get the top 10 URLs and show them in a horizontally scrollable view
  final top10Urls = jpegUrls.take(10).toList();
  final List<Widget> images = await Future.wait(top10Urls.map((url) async
{
    final response = await http.get(Uri.parse(url));
    final bytes = response.bodyBytes;
    return GestureDetector(
      onTap: () {
        showDialog(
          context: context,
          builder: (_) => Dialog(
            child: Image.memory(
              bytes,
              fit: BoxFit.contain,
            ),
          ),
        );
      },
      child: Image.memory(
        bytes,
        width: 200,
        height: 200,
        fit: BoxFit.cover,
        errorBuilder: (context, error, stackTrace) {
          return Image.asset(
            'assets/images/error.png',
            width: 200,
            height: 200,
            fit: BoxFit.cover,
          );
        },
      ),
    );
  }));

  // remove the grey overlay and show the dialog
  Navigator.of(context).pop(); // remove the grey overlay
  showDialog(
    context: context,
    builder: (_) => AlertDialog(
      title: Text('Here\'s the $furnitureTypeWithoutPrefixAndSplit I
suggest for you.'),
      content: SingleChildScrollView(
        scrollDirection: Axis.horizontal,
        child: Row(
          children: images,
        ),
      ),
      actions: [
        TextButton(
          child: Text('OK'),
          onPressed: () => Navigator.of(context).pop(),
        ),
      ],
    ),
  );
```

```
}

class MyApp extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    return MyAppState();
  }
}

class MyAppState extends State<MyApp>{

  final colorYellow = Color.fromRGBO(255, 218, 26, 1);
  final colorBlue = Color.fromRGBO(0, 81, 186, 1);

  List<AssetImage> _backgroundImages = [
    AssetImage('assets/background_image_1.png'),
    AssetImage('assets/background_image_2.png'),
    AssetImage('assets/background_image_3.png'),
    AssetImage('assets/background_image_4.png'),
    AssetImage('assets/background_image_5.png'),
    AssetImage('assets/background_image_6.png'),];

  int _currentIndex = 0;
  int _nextIndex = 1;

  @override
  void initState() {
    super.initState();
    // Start the timer to switch the background image every 3 seconds
    Timer.periodic(Duration(seconds: 3), (timer) {
      setState(() {
        _currentIndex = _nextIndex;
        _nextIndex = (_nextIndex + 1) % _backgroundImages.length;
      });
    });
  }


  File?image;
  Future pickImage(ImageSource source) async {
    try {
      final image = await ImagePicker().pickImage(source: source);
      if (image == null) return;

      final imageTemporary = File(image.path);
      setState(() => this.image = imageTemporary);
    } on PlatformException catch (e) {
      print("Fail to pick image: $e");
      // TODO
    }
  }

  Future<Size> _getImageSize(ImageProvider imageProvider) async {
    Completer<Size> completer = Completer();
    Image image = Image(image: imageProvider);
    image.image.resolve(ImageConfiguration()).addListener(
      ImageStreamListener(
            (ImageInfo imageInfo, bool _) {
          completer.complete(Size(
            imageInfo.image.width.toDouble(),
            imageInfo.image.height.toDouble(),
```

```
          ));
        },
      ),
    );
    return completer.future;
  }


  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Ikea Furniture Finder',
          style: TextStyle(color: Color.fromRGBO(0, 81, 186, 1),
              fontSize: 32,
              //fontWeight: FontWeight.bold,
              fontFamily: 'IKEA-Sans'
          ),

        ),
        centerTitle: true,
        backgroundColor: colorYellow,
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.only(
              bottomLeft: Radius.circular(30),
              bottomRight:  Radius.circular(30)
          ),
        ),
      ),

      body: Stack(
          children: <Widget>[
            Container(
              decoration: BoxDecoration(
                image: DecorationImage(
                  image: _backgroundImages[_currentIndex],
                  fit: BoxFit.cover,
                ),
              ),
            ),
            AnimatedContainer(
              duration: Duration(seconds: 3),
              decoration: BoxDecoration(
                image: DecorationImage(
                  image: _backgroundImages[_nextIndex],
                  fit: BoxFit.cover,
                ),
              ),
              onEnd: () {
                setState(() {
                  _currentIndex = _nextIndex;
                  _nextIndex = (_nextIndex + 1) %
_backgroundImages.length;
                });
              },
            ),
            Positioned.fill(
              child: Column(
                children: <Widget>[
                  Container(
```

```
                    height: 360,
                    width: 360,
                    padding: EdgeInsets.all(32.0),
                    margin: EdgeInsets.fromLTRB(0, 30, 0, 0),
                    child: Column(
                      children: [
                        const SizedBox(height: 32),
                        image != null ? FutureBuilder<Size>(
                          future: _getImageSize(FileImage(image!)),
                          builder: (context, snapshot) {
                            if (snapshot.hasData) {
                              Size imageSize = snapshot.data!;
                              if (imageSize.width > imageSize.height) {
                                // Landscape image
                                return Image.file(
                                  image!,
                                  width: double.infinity,
                                  height: 250,
                                  fit: BoxFit.fitWidth,
                                );
                              } else {
                                // Portrait image
                                return Image.file(
                                  image!,
                                  height: 250,
                                  width: null,
                                  fit: BoxFit.fitHeight,
                                );
                              }
                            } else {
                              return CircularProgressIndicator();
                            }
                          },
                        ) :Image(
                          image:AssetImage('assets/ikeaapp.png'),
                          width: 200,
                          height: 200,
                          fit: BoxFit.cover,
                        ),
                      ],
                    )
                ),
                const SizedBox(height: 0),

                buildButton(
                  title:'Pick From Gallery',
                  icon: Icons.image_outlined,
                  onClicked:()=>pickImage(ImageSource.gallery),
                ),
                const SizedBox(height: 24),
                buildButton(
                  title:'Pick From Camera',
                  icon: Icons.camera_alt_outlined,
                  onClicked:()=>pickImage(ImageSource.camera),
                ),
                const SizedBox(height: 24),

                buildButton(
                  title: 'Submit',
                  icon: Icons.send,
                  onClicked: () async {
```

```
                        sendImageToApi(image, context);
                        // Show dialog

                      },
                    ),

                    Spacer(),

                  ],
                ),
              ),

            ]
        ),
    );
  }

  Widget buildButton({
    required String title,
    required IconData icon,
    required VoidCallback onClicked,
  })=>
      ElevatedButton(
        style: ElevatedButton.styleFrom(
          minimumSize: Size.fromHeight(56),
          primary: Colors.white,
          onPrimary: Colors.black,
          textStyle: TextStyle(fontSize: 20, fontFamily: 'IKEA-Sans' ),
        ),
        child: Row(
          children: [
            Icon(icon, size: 28),
            const SizedBox(width: 16),
            Text(title),
          ],
        ),
        onPressed: onClicked,
      );
}
```

storage_service.dart

```
import 'package:firebase_storage/firebase_storage.dart' as
firebase_storage;
import 'package:firebase_core/firebase_core.dart' as firebase_core;
import 'dart:io';
import 'dart:math';

class Storage {

  final firebase_storage.FirebaseStorage storage =
      firebase_storage.FirebaseStorage.instance;
  Future<void> uploadFile(
      String filePath,
      String fileName,
      ) async {
    File file = File(filePath);

    try {
      await
storage.ref('FurnitureTrainingSet2/$fileName/').putFile(file);
      print("18"+'FurnitureTrainingSet2/$fileName/');
    } on firebase_core.FirebaseException catch (e) {
      print(e);
    }
  }

  Future<String>downloadURL(String imageName) async{
    String downloadURL = await
storage.ref('FurnitureTrainingSet2/$imageName+/').getDownloadURL();
    print("26"+downloadURL);
    return downloadURL;
  }

  Future<List<String>> getJpegFileUrls(String prediction) async {
    List<String> imageUrls = [];
    try {
      firebase_storage.ListResult result =
      await firebase_storage.FirebaseStorage.instance
          .ref('FurnitureTrainingSet2/$prediction/')
          .listAll();

      for (firebase_storage.Reference ref in result.items) {
        if (ref.name.endsWith('.jpeg')) {
          String downloadUrl = await ref.getDownloadURL();
          imageUrls.add(downloadUrl);
        }
      }
    } catch (e) {
      print(e);
    }
    return imageUrls;
  }

  Future<List<String>> getRandomImages(String prediction) async {
    firebase_storage.ListResult listResult =
    await firebase_storage.FirebaseStorage.instance
        .ref('FurnitureTrainingSet2/$prediction'+'/')
        .listAll();
    List<firebase_storage.Reference> images = listResult.items;
```

```
    List<String> imageUrls = [];

    // Shuffle the images and pick the first ten
    images.shuffle();
    for (int i = 0; i < 10 && i < images.length; i++) {
      if (images[i].name.endsWith('.jpeg')) { // only add JPEG files
        String downloadUrl = await images[i].getDownloadURL();
        imageUrls.add(downloadUrl);
      }
    }

    return imageUrls;
  }

}
```

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.ikea_furniture_finder">
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"/>

   <application
        android:requestLegacyExternalStorage="true"
        android:label="Ikea Furniture Finder"
        android:icon="@mipmap/ic_launcher">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:launchMode="singleTop"
            android:theme="@style/LaunchTheme"

android:configChanges="orientation|keyboardHidden|keyboard|screenSize|sma
llestScreenSize|locale|layoutDirection|fontScale|screenLayout|density|uiM
ode"
            android:hardwareAccelerated="true"
            android:windowSoftInputMode="adjustResize">


            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category
android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <!-- Don't delete the meta-data below.
             This is used by the Flutter tool to generate
GeneratedPluginRegistrant.java -->
       <meta-data
           android:name="flutterEmbedding"
           android:value="2" />
       <meta-data
android:name="com.google.firebase.appcheck.provider.FirebaseAppCheckProvi
der"

android:value="com.google.firebase.appcheck.debug.DebugAppCheckProvider"
/>

    </application>
    <meta-data
        android:name="flutterEmbedding"
        android:value="2" />
</manifest>
```

5.5. System Operation (with Screenshot)

In this System Operation section, this report will explain how the system operates in detail. This includes providing a step-by-step guide on how to use the system and its different functionalities. Screenshots and images can be included to help illustrate the steps and provide visual aids. It is important to explain how the various components of the system work together and how users can interact with them. A clear and concise description of the system's operation will make it easier for users to navigate and utilize its features.

Uploading Docker Image to Google Cloud Run involves deploying a Docker container to the Cloud Run service, which automatically scales to handle incoming requests. Linking Firebase with Google Cloud Storage Bucket enables users to easily store and retrieve data, including images and other files, in the Google Cloud Storage Bucket, which can be accessed from Firebase. Integrating Google Cloud Platform and Firebase with Android Studio enables developers to leverage the powerful capabilities of both platforms to create high-quality mobile apps. By linking Firebase and Google Cloud Platform, developers can use Firebase Authentication, Realtime Database, and Cloud Storage in their Android apps, while taking advantage of the scalability and flexibility of Google Cloud Platform.

5.5.1. Uploading Docker Container to Google Cloud Run

1) Go to the file which made Docker file (C:\Users\User\Desktop\UTAR\FYP\FYP 2\Machine Learning\Week 10>)
2) Type the following code to upload the file from Docker Container to Google Cloud Run

```
gcloud builds submit --tag gcr.io/furniture-382903/index

gcloud run deploy --image gcr.io/furniture-382903/index --platform managed

Service name (index):  getprediction3
Please enter numeric choice or text value (must exactly match list item):  34
Allow unauthenticated invocations to [getprediction3] (y/N)?  y

Service [getprediction4] revision [getprediction3-00001-vuv] has been deployed and is serving 100 percent of traffic.
Service URL: https://getprediction4-biptoq2fwq-uw.a.run.app
```

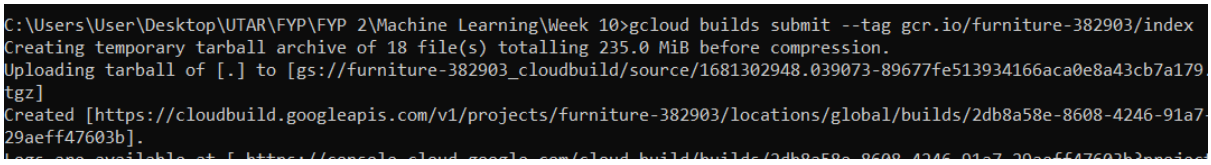


Figure 5.5.1.1 upload the file from Docker Container to Google Cloud

Figure 5.5.1.2 Deployed successfully on google cloud

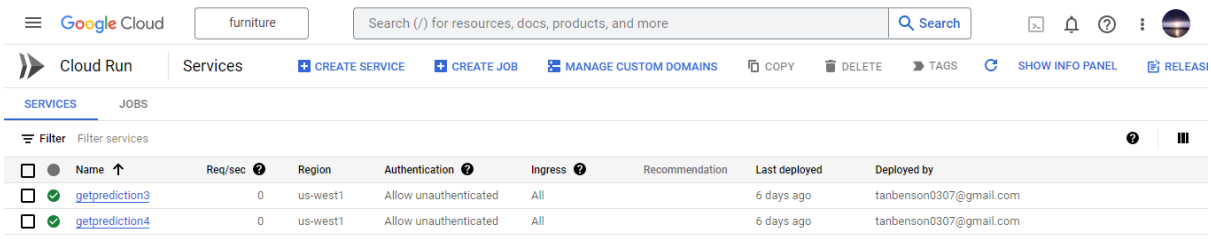


Figure 5.5.1.3 Google cloud appeared deployed model

5.5.2. Linking Firebase with Google Cloud Storage Bucket

1) Go to the Firebase Console and select your project.
2) Click on the "Storage" tab in the left menu.
3) Click on the "Get started" button to set up Firebase Storage for your project.
4) Follow the instructions to create a new Cloud Storage Bucket.
5) Once the bucket is created, click on the "Rules" tab in the Firebase Storage menu.
6) Update the rules to allow read and write access to authenticated users by replacing the default rule with the following code:
7) Click on the "Get started" button in the Cloud Storage menu.
8) Select "Browser" from the menu on the left to open the Cloud Storage Browser.
9) Find your newly created bucket and click on it.

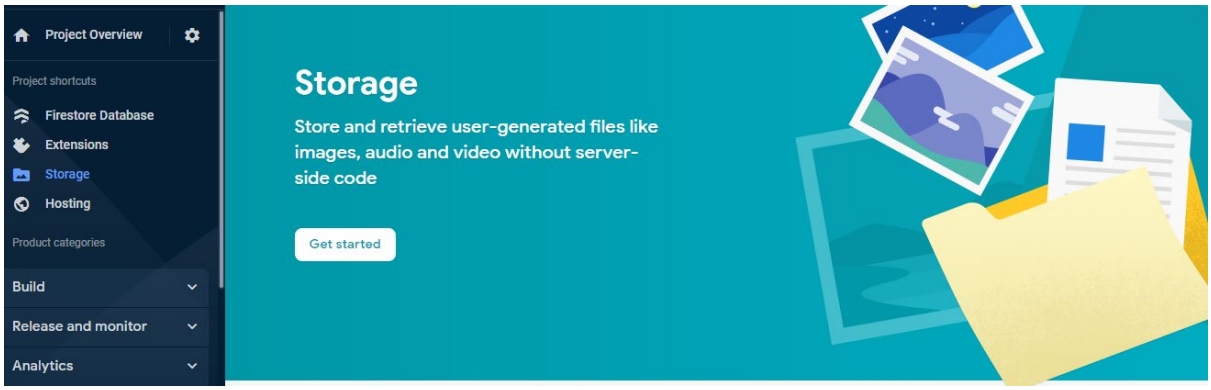


Figure 5.5.2.1 Get Started

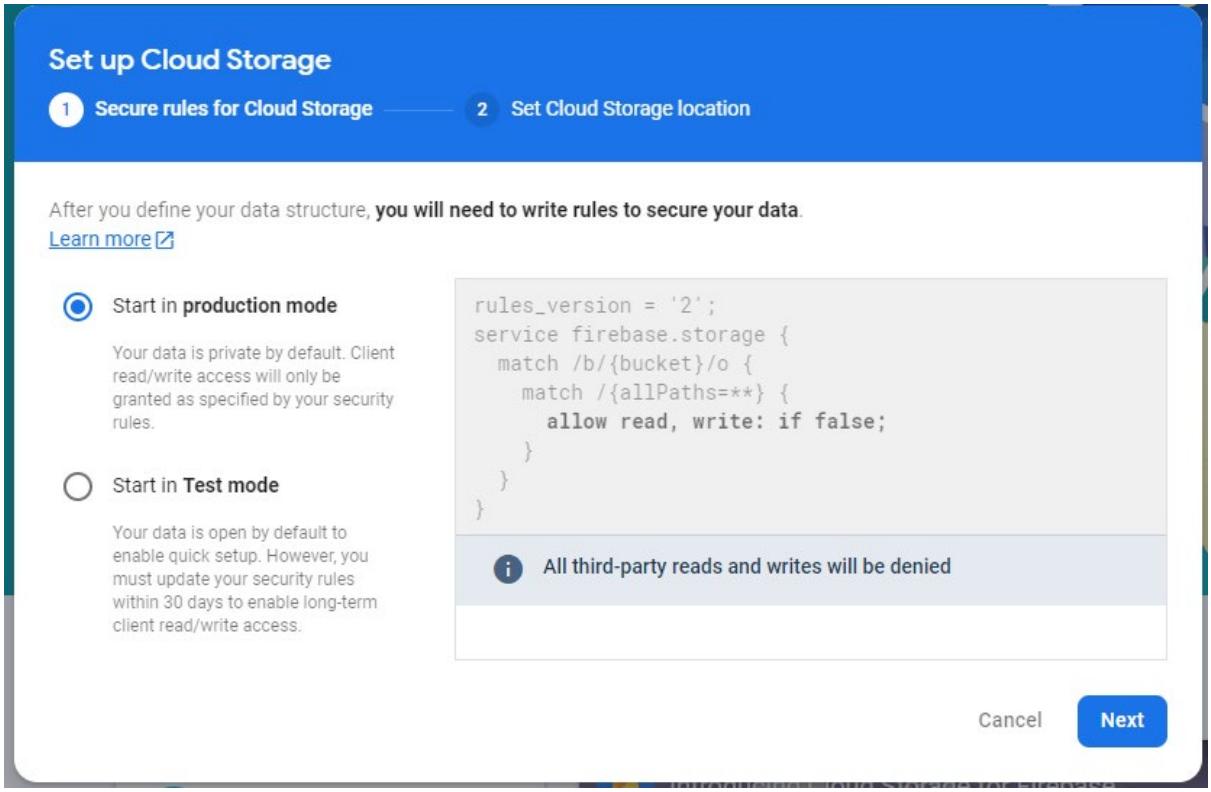


Figure 5.5.2.2 Set up Secure Rules

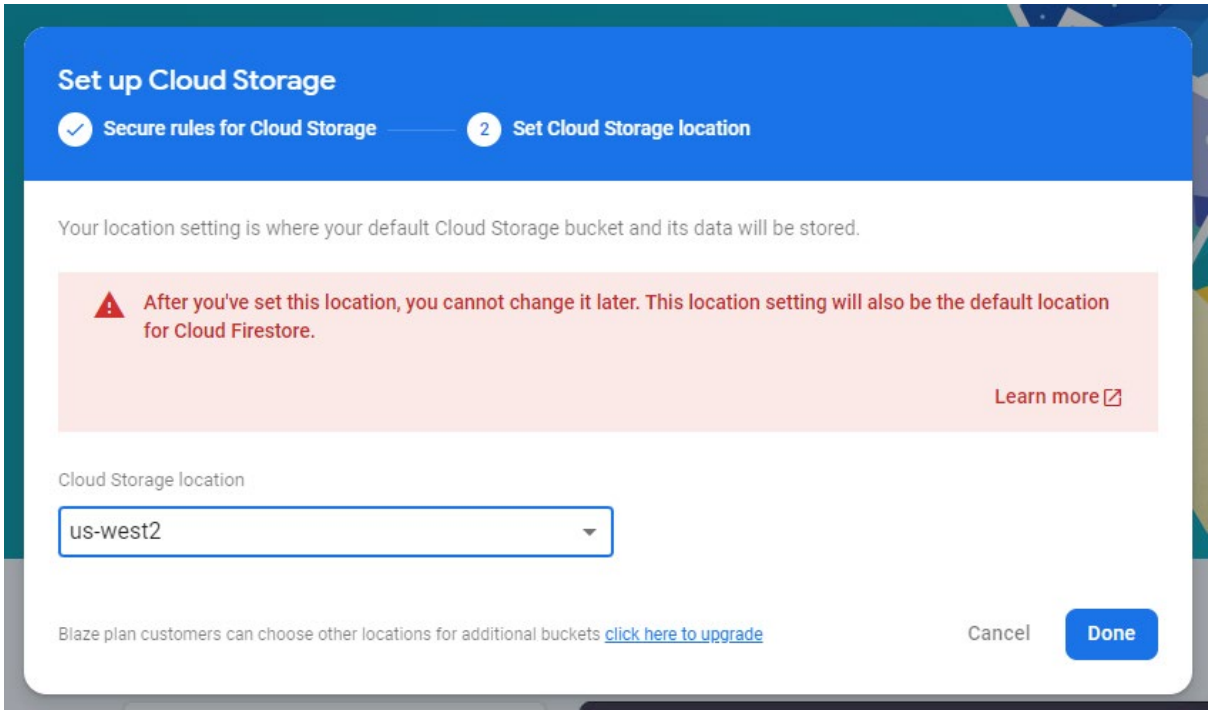


Figure 5.5.2.3 Set up Location

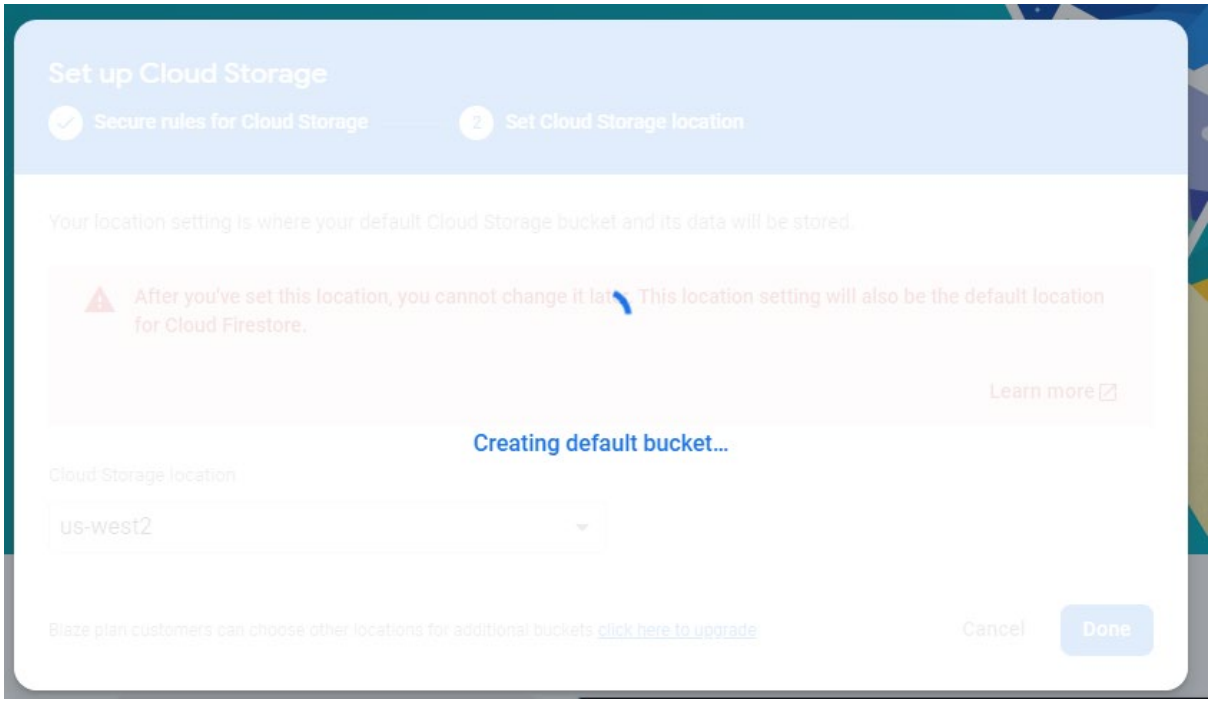


Figure 5.5.2.4 Creating bucket

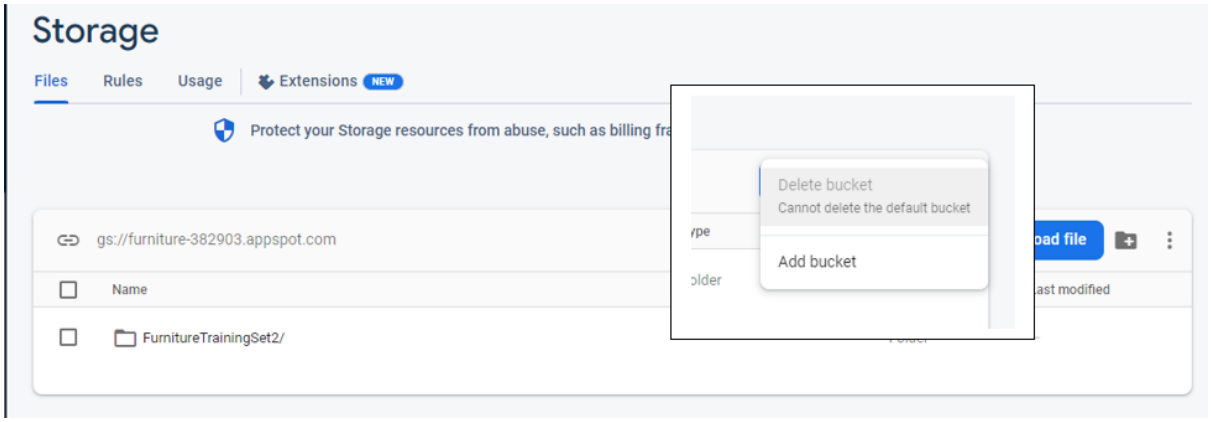


Figure 5.5.2.5 Add Bucket to GCP

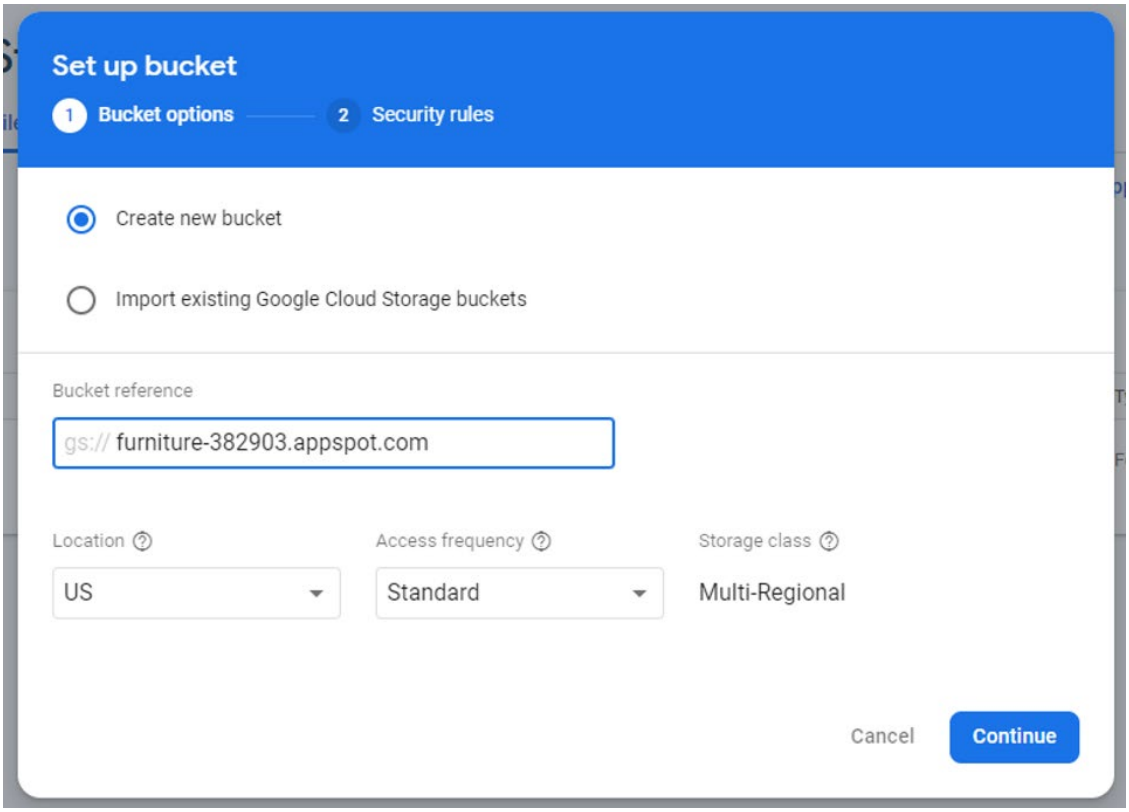


Figure 5.5.2.6 Put related detail to the blank

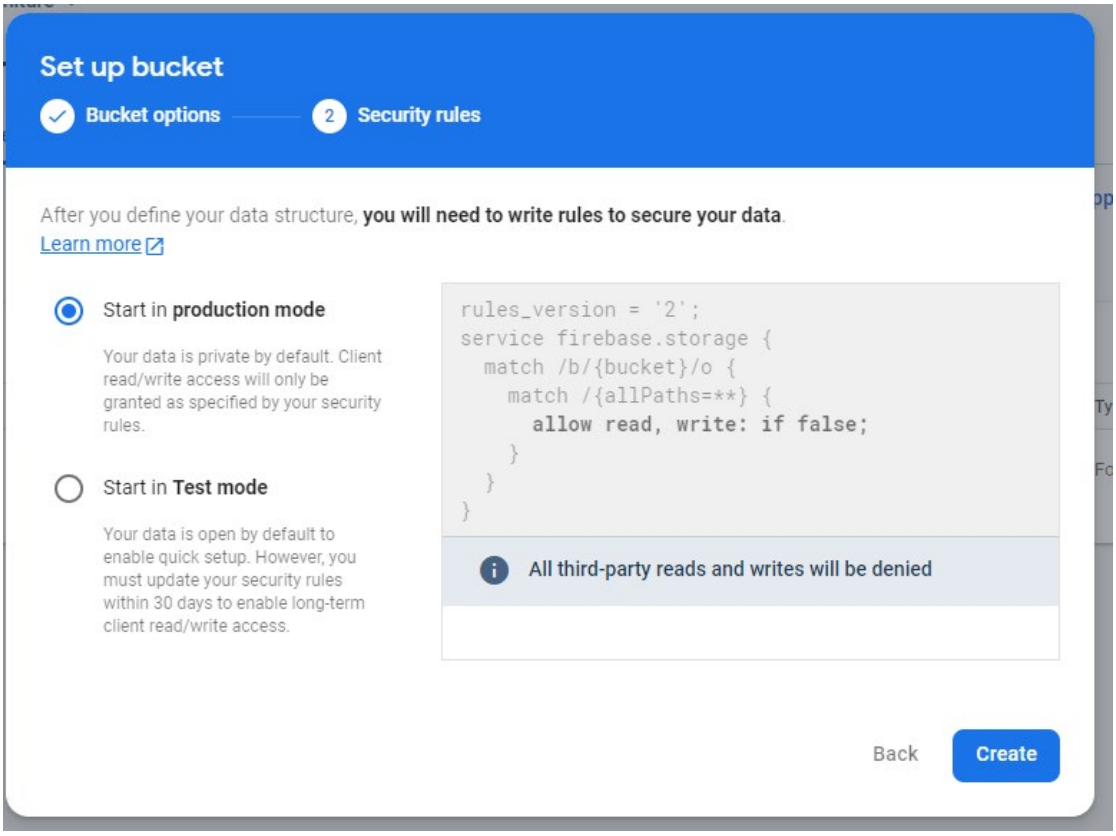


Figure 5.5.2.7 Click create to create

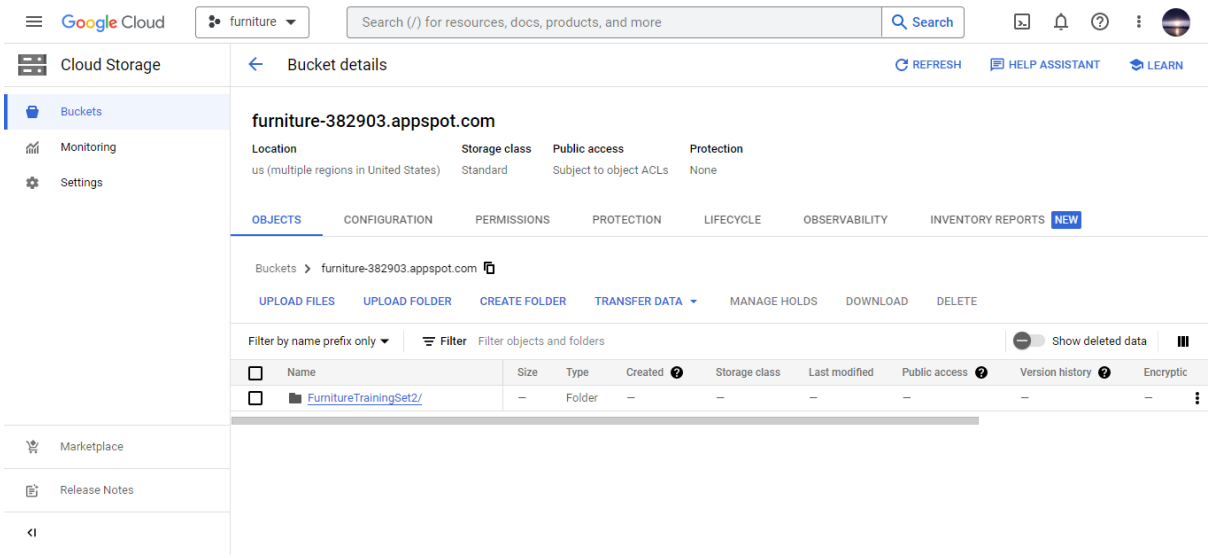


Figure 5.5.2.8 Successfully added to GCP

5.5.3. Integrating Google Cloud Platform and Firebase with Android Studio

1) Register app

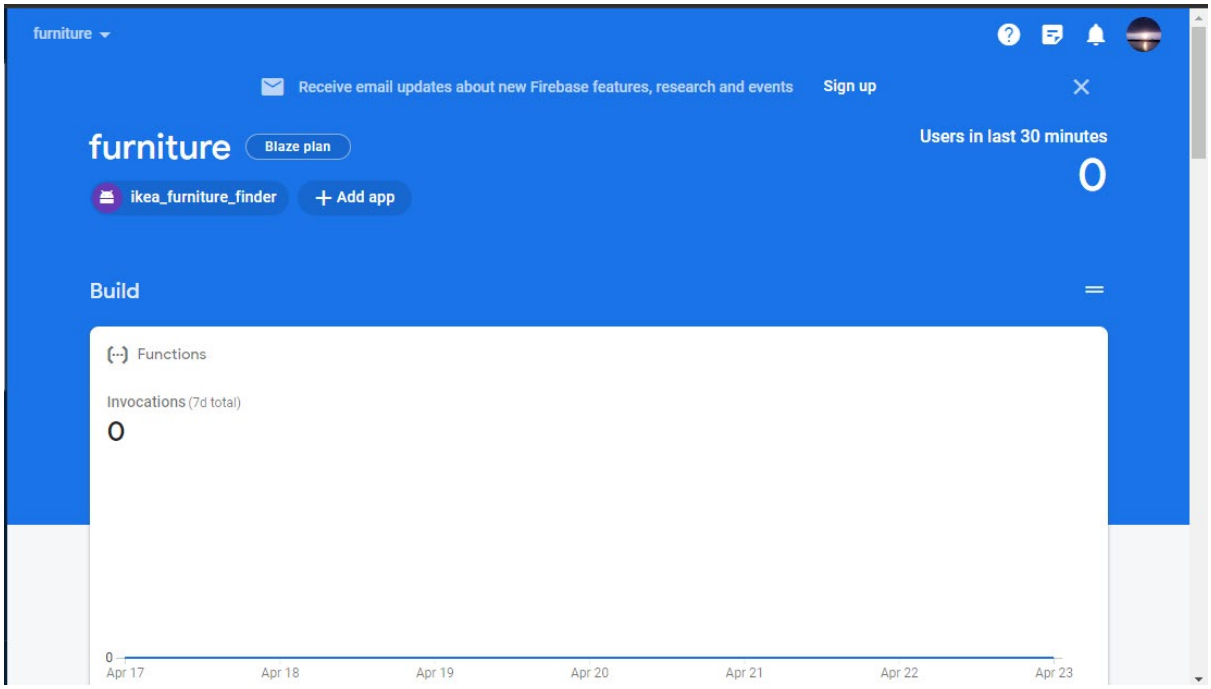


Figure 5.5.3.1 Click the Add App

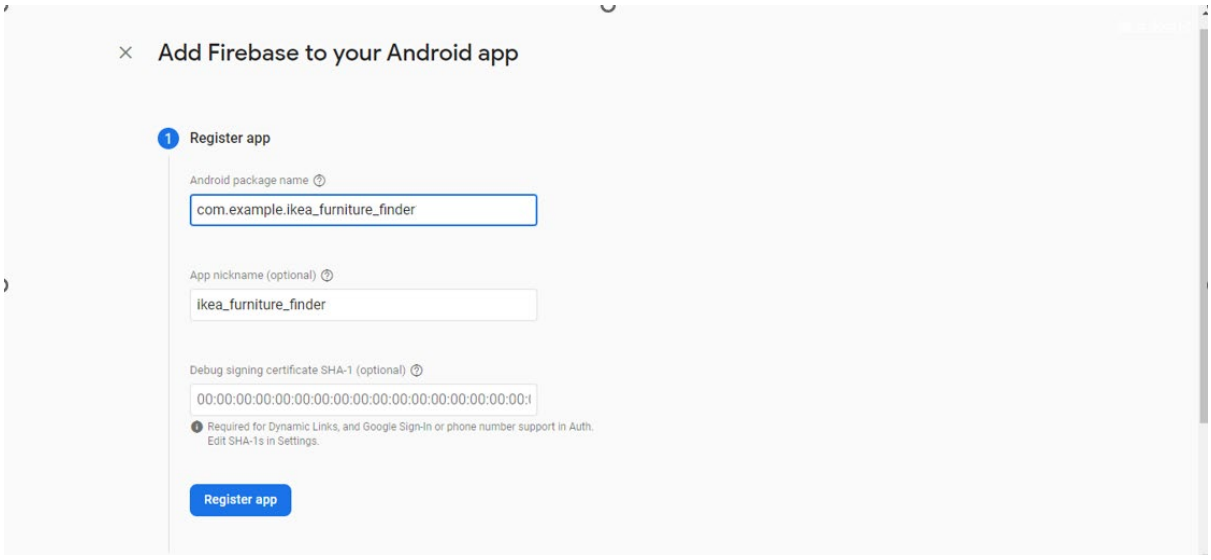


Figure 5.5.3.2 Put Ikea furniture finder detail and register

2) Download and then add config file

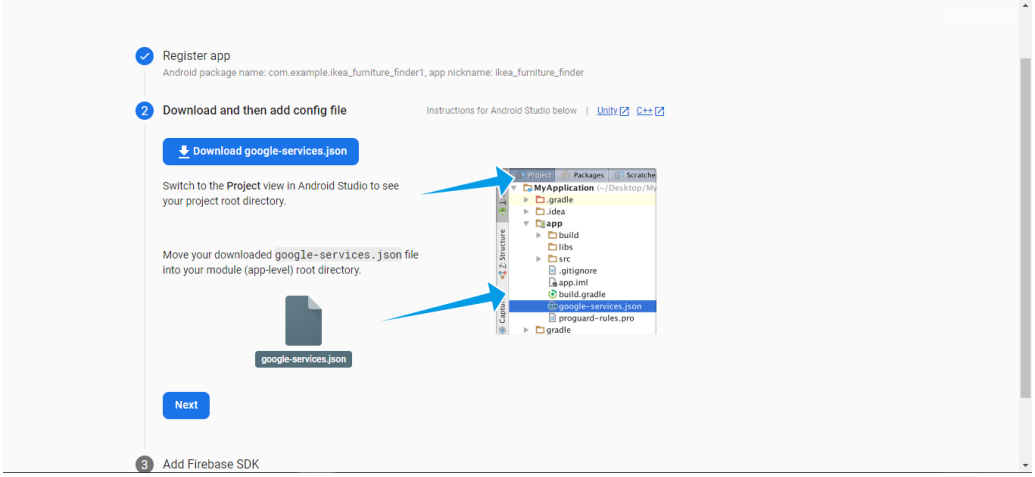


Figure 5.5.3.3 Download JSON file and then add config file

3) Add Firebase SDK

- To make the google-services.json config values accessible to Firebase SDKs, you need the Google services Gradle plug-in.
- Add the plug-in as a buildscript dependency to your project-level build.gradle file:
- Root-level (project-level) Gradle file (<project>/build.gradle):
- Then, in your module (app-level) build.gradle file, add both the google-services plug-in and any Firebase SDKs that you want to use in your app:

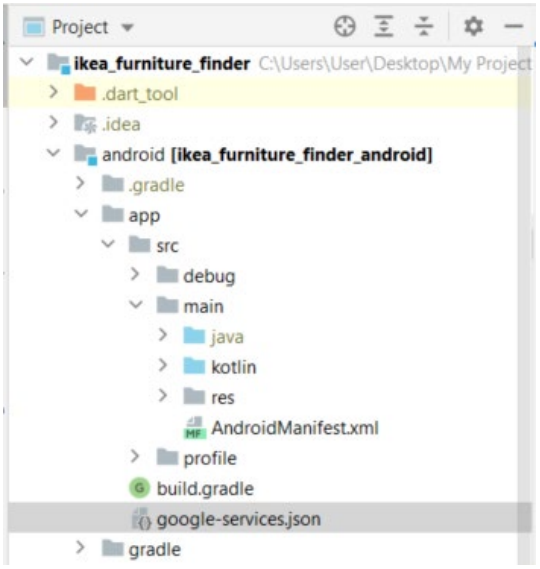


Figure 5.5.3.4 Add JSON file to Android Studio

This is the code in submit button to let the user to click then it will send request to Cloud Run then get back with a JSON string. However, this project want to show the image to user, so need to split the string and send request to firebase

```
// Send the image to the API endpoint
final url = Uri.parse("https://getprediction3-biptoq2fwq-uw.a.run.app");
final request = http.MultipartRequest("POST", url);
request.headers.addAll({'Authorization': 'Bearer $appCheckToken'});
request.files.add(await http.MultipartFile.fromPath("file", image.path));
final response = await request.send();
final responseBody = await response.stream.bytesToString();
final jsonMap = json.decode(responseBody);
final prediction = jsonMap['prediction'];
```

Here is the code which split the JSON string and send request to Firebase

```
// Modify the prediction value to include the path
final String furnitureType = prediction;
final furnitureTypeWithoutPrefix = furnitureType.split("_")[1]; //
"Sofas"
final String furnitureTypeWithoutPrefixAndSplit =
furnitureTypeWithoutPrefix.split(RegExp(r'(?=[A-Z])')).join(' ');
print(furnitureTypeWithoutPrefixAndSplit);//"Chairs Stoolsbenches"
print(furnitureType); // Prints "3_Sofas"


final storage = FirebaseStorage.instance;
final ref = storage.ref().child('FurnitureTrainingSet2/$furnitureType');


final result = await ref.listAll();

// Get a list of all the JPEG file URLs under the specified directory
final List<String> jpegUrls = await Future.wait(result.items
    .where((item) => item.name.toLowerCase().endsWith('.jpeg'))
    .map((item) => item.getDownloadURL())
    .toList());

// Shuffle the list of JPEG URLs
jpegUrls.shuffle();

// Get the top 10 URLs and show them in a horizontally scrollable view
final top10Urls = jpegUrls.take(10).toList();
final List<Widget> images = await Future.wait(top10Urls.map((url) async {
  final response = await http.get(Uri.parse(url));
  final bytes = response.bodyBytes;
  return GestureDetector(
    onTap: () {
      showDialog(
        context: context,
        builder: (_) => Dialog(
          child: Image.memory(
            bytes,
            fit: BoxFit.contain,
```

```
            ),
          ),
        );
      },
      child: Image.memory(
        bytes,
        width: 200,
        height: 200,
        fit: BoxFit.cover,
        errorBuilder: (context, error, stackTrace) {
          return Image.asset(
            'assets/images/error.png',
            width: 200,
            height: 200,
            fit: BoxFit.cover,
          );
        },
      ),
    );
}));
```

5.6. Implementation Issues and Challenges

The challenge in this project is to collect the data and need to defined well for the name before using scripting. It is time consuming to collect for whole dataset, and when need to let the dataset more specific, the time consuming is more than previous stage, and by using scripting, some of the data can't success move, need to find the alternative ways to solve. When check for the dataset, some of the file name still remain for. avif and .jpg format. So, need some time to either move manually or delete due to duplication.

The implementation of this project came with its own set of challenges and issues. One of the major issues was the training of the models. With different combinations of epochs and layers, it became difficult to determine the best model that could provide an accuracy of over 70%. It was a time-consuming process and required a lot of trial and error. To improve the training dataset, additional datasets were downloaded, but only the IKEA original dataset was used for Firebase.

Another challenge that was faced during the implementation of the project was related to the CPU performance. As the size of the dataset and complexity of the models increased, the training time also increased. Therefore, a GPU with CUDA and cuDNN compatible version was installed to improve the training time. However, this process also came with its own set of challenges, such as compatibility issues with the existing hardware and software, installation errors, and configuration problems. It required a lot of time and effort to resolve these issues, but ultimately it was successful in improving the performance and reducing the training time of the models.

Another implementation issue that was encountered was related to the Kotlin version. Some mobile devices were not compatible with the outdated Kotlin version, and the virtual device showed the splash screen, but the real device did not. This problem was solved by adding the file to the build file before building the APK. Due to this issue, multiple APKs were built until the one that worked was finally found.

Uploading the app to Google Cloud Platform (GCP) Cloud Run also presented some challenges. Initially, there was an issue where the logs showed an "out of memory" message. To resolve this, the memory was increased from 500MB to 4GB. Additionally, multiple ways were tried to upload the app to GCP, but it finally worked when the Google Cloud SDK was used.

Finally, since the app is online, a wifi and data check were added after the splash screen. This ensures that the user has a stable internet connection before proceeding with the app, preventing any issues related to connectivity. These implementation issues and challenges required significant effort and time to resolve, but ultimately they were overcome, and the app was successfully implemented.

5.7. Concluding Remark

In this chapter, the system implementation process of the project has been discussed in detail. The hardware and software setups required for the project have been explained, and the steps for preparing the image dataset have been outlined. Furthermore, the settings and configurations of various tools used in the project, such as Anaconda Jupyter Notebook, PyCharm, Docker, Google Cloud Platform, Firebase, and Android Studio, have been explained in detail.

Additionally, the system operation of the project has been demonstrated with screenshots. This includes uploading the Docker container to Google Cloud Run, linking Firebase with Google Cloud Storage Bucket, and integrating Google Cloud Platform and Firebase with Android Studio. The chapter also highlights the various implementation issues and challenges encountered during the project, such as determining the best model, slow CPU performance, Kotlin version compatibility issues, and memory allocation problems.

Despite these challenges, the project was successfully implemented, and the system operates as expected. The efforts made to resolve the implementation issues and challenges ultimately paid off, and the system was able to achieve its goals. The next chapter will provide a detailed analysis of the results obtained from the project.

CHAPTER 6: SYSTEM EVALUATION AND DISCUSSION

6.1. System Testing and Performance Metrics

6.1.1. Results of the h5 model

In this section, the system is evaluated and performance metrics are presented. The system was tested using various datasets, including different combinations of layers, epochs, and batch sizes. The accuracy and loss were recorded for each model, and the highest accuracy with its corresponding epoch was noted. The testing was performed on different platforms, including Anaconda, Jupyter Notebook, and Google Colab. The performance metrics provide an evaluation of the system's accuracy and efficiency, which is crucial for determining the system's effectiveness in solving the problem it was designed for. The results of the system testing and performance metrics are discussed in detail in the following subsections.

Based on the information provided in the Table 6.1.1, it appears to be a record of various experiments conducted on different furniture training sets using different neural network models.Each row represents a different experiment, identified by an ID number (e.g., T1, T2, T3, etc.). The experiments were conducted on different furniture training sets (e.g., FurnitureTrainingSet1, FurnitureTrainValTest, FurnitureTrainingSet2_Model1, etc.), each with a different number of layers, epochs, and batch sizes used for training the neural network models.

The "Platform" column indicates the platform used for conducting the experiment (e.g., Jupyter Notebook, Google Collab). The "Accuracy" and "Loss" columns indicate the accuracy and loss metrics for each experiment, respectively. The "Best Epoch" and "Best Accuracy" columns show the epoch number and the corresponding accuracy achieved by the model with the best performance during training.

There are some missing values (NA) in the Table 6.1.2, which may indicate that either the experiment was not conducted or the data was not recorded for that particular metric or parameter. It seems like you want to separate the rows with missing values into a separate table, which could provide more clarity and organization to the data.

Table 6.1.1.1 Furniture Model Performance

ID	Dataset	Layers	Epochs	Batch Size	Platform	Accuracy	Loss	Best Epoch	Best Accuracy
T1	FurnitureTrainingSet1	3	50	32	Jupyter Notebook	0.6587	0.6699	28	0.6421
T2	FurnitureTrainValTest	3	100	32		0.6347	0.9142	34	0.6266
T3	FurnitureTrainValTest2	3	50	32		0.4974	1.6574	11	0.5095
T4	FurnitureTrainingSet2_Model1	3	50	32		0.4755	1.7444	15	0.5188
T6	FurnitureTrainingSet2_Model1	3	50	32		0.5605	1.3432	18	0.5827
T8	FurnitureTrainingSet2_Model1	5	50	32		0.5605	1.2877	15	0.5784
T9	FurnitureTrainingSet2_Model2	3	50	32		0.5728	1.1048	39	0.6023
T10	FurnitureTrainingSet2_Model2	4	50	32		0.5657	1.1003	20	0.6004
T11	FurnitureTrainingSet2_Model2	5	50	32		0.5696	1.1926	37	0.5946
T12	FurnitureTrainingSet2_Model3	3	50	32		0.5484	1.1237	14	0.5724
T13	FurnitureTrainingSet2_Model3	4	50	32		0.5393	1.2912	13	0.5783
T14	FurnitureTrainingSet2_Model3	5	50	32		0.5335	1.3578	17	0.5789
T15	FurnitureTrainingSet2_Model4	3	50	32		0.5837	1.1856	19	0.6204
T16	FurnitureTrainingSet2_Model4	4	50	32		0.5728	1.0965	16	0.6082
T17	FurnitureTrainingSet2_Model4	5	50	32		0.5898	1.0836	15	0.6177
T18	FurnitureTrainingSet3_Model1	5	50	32		0.6932	0.9714	17	0.7059
T19	FurnitureTrainingSet3_Model1	3	50	16		0.7097	1.004	23	0.7163
T20	FurnitureTrainingSet3_Model1	4	50	32		0.7119	0.8115	16	0.7153
T21	FurnitureTrainingSet4_Model1	3	50	32		0.6477	0.9802	21	0.6595
T22	FurnitureTrainingSet4_Model1	4	50	32		0.6613	1.0023	8	0.6699
T23	FurnitureTrainingSet4_Model1	5	50	32		0.6477	1.0648	16	0.6617
T24	FurnitureTrainingSet3_Model1	3	50	32		0.7004	0.9079	12	0.7157
T25	FurnitureTrainingSet3_Model1	4	50	16		0.7153	0.8731	19	0.7292
T26	FurnitureTrainingSet3_Model1	5	50	16		0.6987	0.8671	17	0.7081
T27	FurnitureTrainingSet3_Model1	3	100	16		0.7055	1.1827	14	0.7169
T28	FurnitureTrainingSet3_Model1	4	100	16		0.6869	1.1302	7	0.7212
T29	FurnitureTrainingSet3_Model1	5	100	16		0.6983	1.1942	20	0.7153
T30	FurnitureTrainingSet3_Model1	3	50	64		0.7106	0.8369	18	0.7064
T31	FurnitureTrainingSet3_Model1	4	50	64		0.6958	0.8734	18	0.7199
T32	FurnitureTrainingSet3_Model1	5	50	64		0.6996	0.8351	21	0.7177

Table 6.1.1.2 Failed Furniture Model Performance

ID	Dataset	Layers	Epochs	Batch Size	Platform	Accuracy	Loss	Best Epoch	Best Accuracy
T5	FurnitureTrainingSet2_Model1	3	50	32	Google Collab	N/A	N/A	N/A	N/A
T7	FurnitureTrainingSet2_Model1	4	50	32	Jupyter Notebook	N/A	N/A	N/A	N/A

6.2. Testing Setup and Result

6.2.1. Validate the h5 model in Anaconda Jupyter Notebook

```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img,
img_to_array
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten
import cv2
import matplotlib.pyplot as plt

# Code to increase the memory of GPU
physical_devices = tf.config.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(physical_devices[0], True)

# Set up data paths
data_dir = 'C:/Users/User/Desktop/FurnitureTrainingSet2'

# Set up data generators for data augmentation
test_datagen = ImageDataGenerator(rescale=1./255)

# Set up data generator to flow from directory
test_generator = test_datagen.flow_from_directory(
        data_dir,
        target_size=(150, 150),
        batch_size=1,
        class_mode='categorical',
        shuffle=False)

# Load the trained model
model = tf.keras.models.load_model('C:/Users/User/Desktop/UTAR/FYP/FYP 2/Machine
Learning/Week 8/Model2_Train20.h5')

# Load the test image with OpenCV
img_path = 'C:/Users/User/Desktop/UTAR/FYP/FYP 2/Machine Learning/Week
8//table2.jpeg'
test_image = cv2.imread(img_path)

# Resize the image to the correct size
test_image = cv2.resize(test_image, (150, 150))

# Convert the image to a numpy array and normalize the pixel values
test_image = np.array(test_image, dtype="float32")/255.0
```

```
# Add a batch dimension to the image (the model expects a batch of images)
test_image = np.expand_dims(test_image, axis=0)

# Predict the class of the test image
prediction = model.predict(test_image)
predicted_class = np.argmax(prediction[0])
class_prob = prediction[0][predicted_class]
class_label =
list(test_generator.class_indices.keys())[list(test_generator.class_indices.values()).index(pr
edicted_class)]

print(f"Predicted class: {class_label}")
print(f"Probability: {class_prob}")

# Get the list of file paths for images in the predicted class
image_paths = [os.path.join(data_dir, class_label, file_name) for file_name in
os.listdir(os.path.join(data_dir, class_label))]

# Shuffle the list of image paths
import random
random.shuffle(image_paths)

# Display similar images
num_similar_images = min(5, len(image_paths))  # adjust num_similar_images
fig, axs = plt.subplots(1, num_similar_images, figsize=(30, 30))  # adjust figsize
for i in range(num_similar_images):
    img = cv2.imread(image_paths[i])
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    axs[i].imshow(img)
    axs[i].axis('off')

plt.show()
```

The trained model was tested on a new image using a prediction notebook. The model was loaded, and a data generator was set up to flow from the directory. The test image was loaded using OpenCV and resized to the correct size. The image was then converted to a numpy array and the pixel values were normalized. Adding a batch dimension to the image, the model was used to predict the class of the test image, which was found to be "1_Tablesdesks" with a high probability of 0.9999. This suggests that the model is accurately recognizing the furniture type in the image. The predicted class was used to display similar images from the training set, which helped to validate the model's performance. Overall, the prediction notebook was a useful tool for evaluating the accuracy of the trained model.

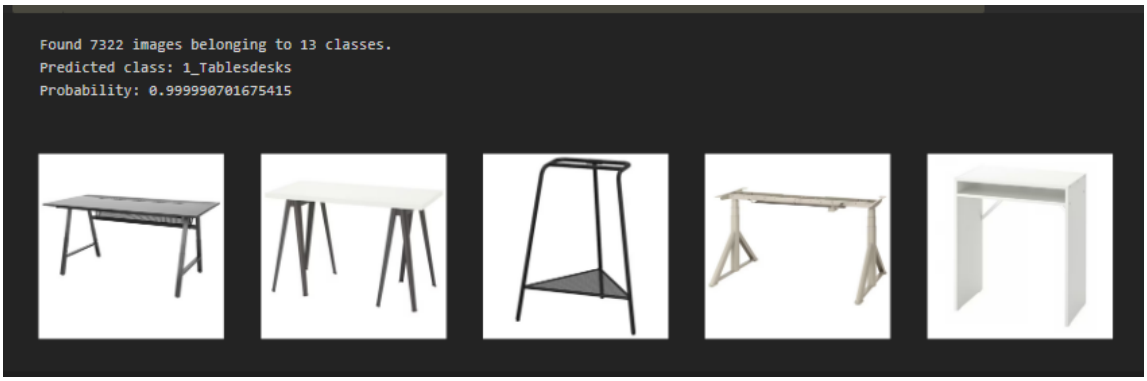


Figure 6.2.1.1 Prediction Output for Image

6.2.2. Validate the h5 model in Pycharm

This project attempted to use the local URL "http://127.0.0.1:5000/" to test their model's predictions by sending an image file. After executing the code, it found that the prediction results were accurate. Later, after deploying the model to Google Cloud Run, this project obtained a new URL to use (https://getprediction3-biptoq2fwq-uw.a.run.app). This project updated the code with the new URL and re-tested the model, obtaining the same accurate results as before.

```
import requests

url = "http://127.0.0.1:5000/"
#url = "https://getprediction3-biptoq2fwq-uw.a.run.app"

picture = 'table.jpeg'
files = {'file': open(picture, 'rb')}

resp = requests.post(url, files=files)
```

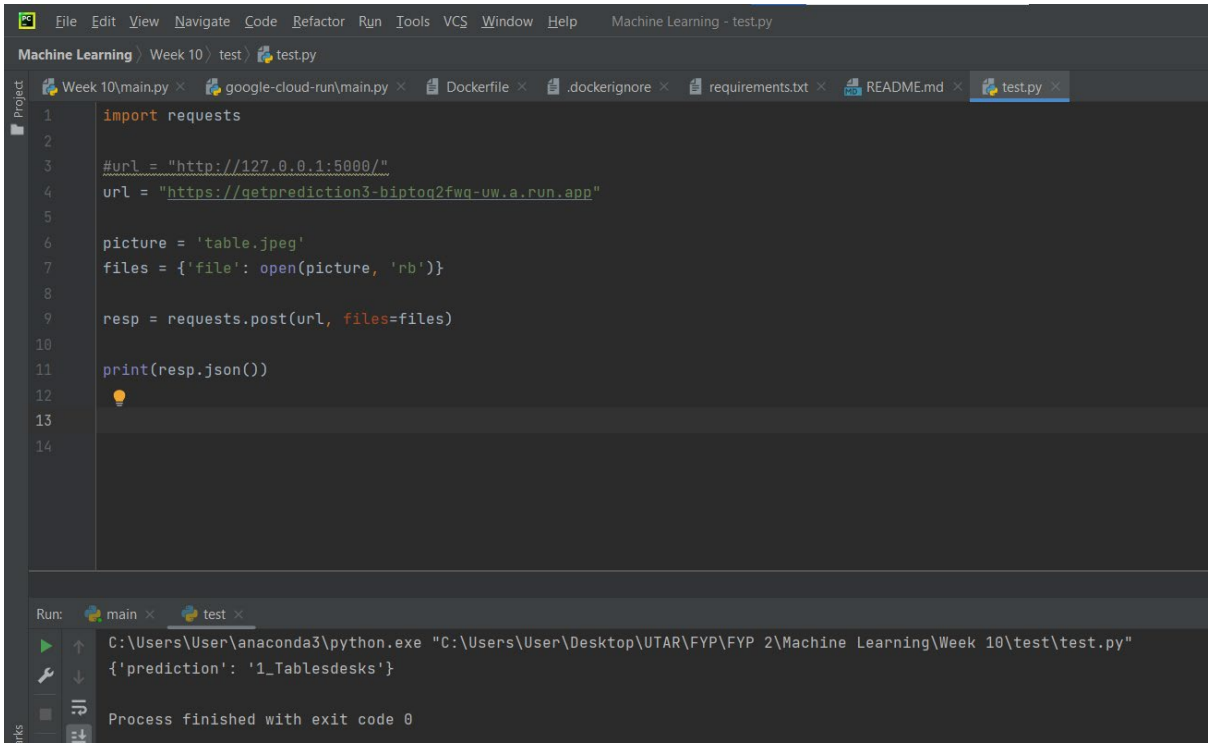


Figure 6.2.2.1 Test the accuracy by local URL

```
import requests

#url = "http://127.0.0.1:5000/"
url = "https://getprediction3-biptoq2fwq-uw.a.run.app"

picture = 'table.jpeg'
files = {'file': open(picture, 'rb')}

resp = requests.post(url, files=files)

print(resp.json())
```

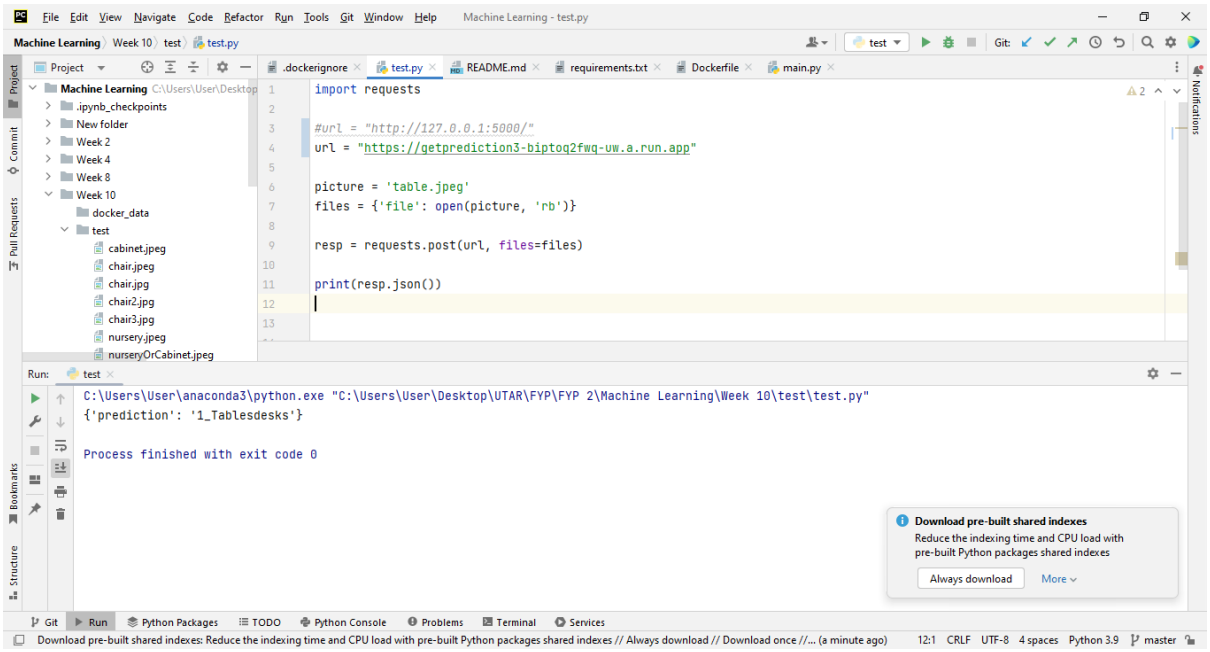


Figure 6.2.2.2 Test the accuracy by GCP Cloud Run URL

6.2.3. Validate the GCP Cloud Run URL in Android Studio

Figure 6.2.3.1 shows how an HTTP POST request was sent in Flutter to a Google Cloud Run API endpoint. The URL of the API endpoint was defined using the Uri.parse() function. Then, http.MultipartRequest() was created with the POST method and the defined API endpoint URL.

An authorization token was added to the header of the request using request.headers.addAll(). The image file was added to the request using request.files.add() and http.MultipartFile.fromPath().

The request was then sent using request.send() and the response was read using response.stream.bytesToString(). The response body was decoded from JSON format to a map object using json.decode().

The predicted class of the image was then extracted from the map object using jsonMap['prediction']. The predicted class was modified to remove the prefix using split() and join(). The modified predicted class and the original predicted class were printed to the console.

Finally, the GCP Cloud Run URL could be validated in Android Studio by running the app on a device or emulator and checking if the prediction output matched the expected output, and Figure 6.2.3.1 is the logs after ran the code in Android Studio.

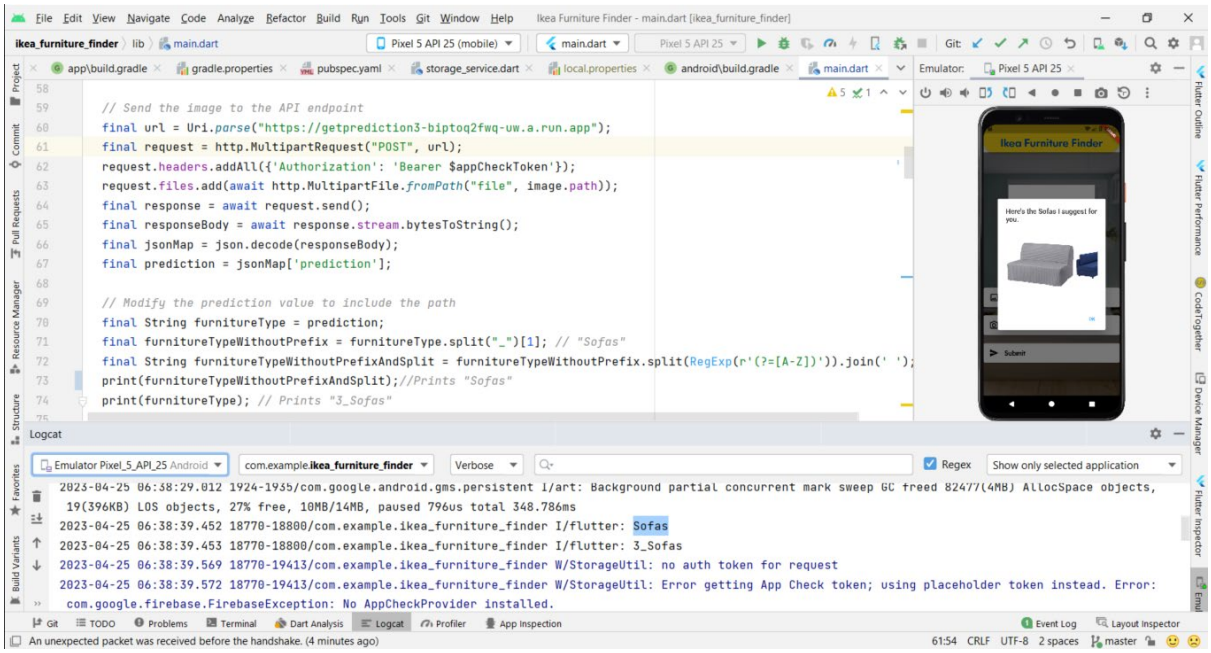


Figure 6.2.3.1 Result of GCP Cloud Run URL in Android Studio

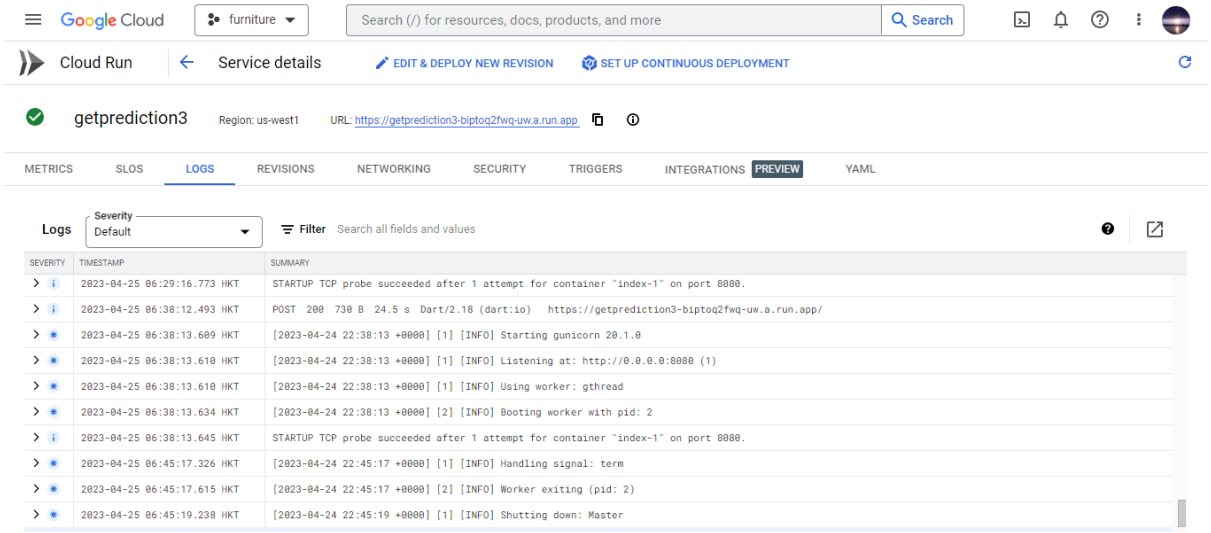


Figure 6.2.3.2 Logs of GCP Cloud Run URL in Android Studio

6.2.4. Validate the Firebase in Android Studio

```
final storage = FirebaseStorage.instance;
final ref = storage.ref().child('FurnitureTrainingSet2/$furnitureType');

// show a transparent grey overlay over the screen
showDialog(
  context: context,
  barrierDismissible: false,
  builder: (BuildContext context) {
    return Container(
      color: Colors.black.withOpacity(0.5),
      child: Center(
        child: CircularProgressIndicator(),
      ),
    );
  },
);

final result = await ref.listAll();

// Get a list of all the JPEG file URLs under the specified directory
final List<String> jpegUrls = await Future.wait(result.items
    .where((item) => item.name.toLowerCase().endsWith('.jpeg'))
    .map((item) => item.getDownloadURL())
    .toList());

// Shuffle the list of JPEG URLs
jpegUrls.shuffle();

// Get the top 10 URLs and show them in a horizontally scrollable view
final top10Urls = jpegUrls.take(10).toList();
final List<Widget> images = await Future.wait(top10Urls.map((url) async {
  final response = await http.get(Uri.parse(url));
  final bytes = response.bodyBytes;
  return GestureDetector(
    onTap: () {
      showDialog(
        context: context,
        builder: (_) => Dialog(
          child: Image.memory(
            bytes,
            fit: BoxFit.contain,
          ),
        ),
      );
    },
```

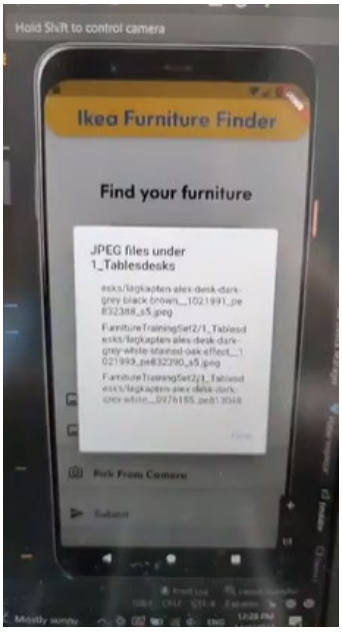


Figure 6.2.4.1 Output of retrieve all JPEG URL Firebase in Android Studio

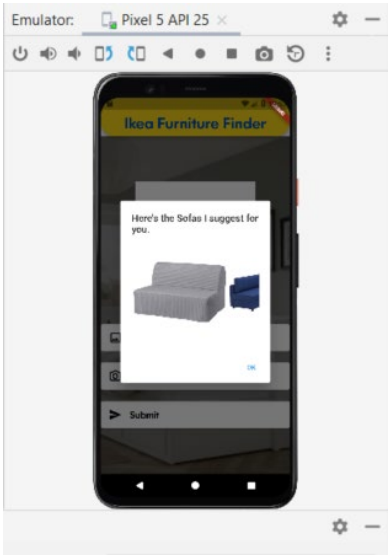


Figure 6.2.4.2 Output of the Firebase in Android Studio

Figure 6.2.4.1 above is output after retrieving a list of JPEG file URLs from a Firebase storage bucket. The function first creates a reference to the Firebase storage bucket and specifies the subdirectory where the images are stored using the furnitureType variable. Next, a dialog with a circular progress indicator is shown to indicate that the images are being loaded. The listAll() function is then called to retrieve a list of all the items under the specified subdirectory. The items are filtered to only include JPEG files using the where() method and a lambda function. The getDownloadURL() method is then called on each item to retrieve the corresponding URL.

Once the JPEG URLs are retrieved, the list is shuffled using the shuffle() method. The top 10 URLs are then selected using the take() method and converted to a list. Finally, a list of widgets is created using the top 10 JPEG URLs. Figure 6.2.4.2 contains an image that is loaded from the corresponding URL using the http library's get() method. If a user taps on an image widget, a dialog box is shown that displays the image at a larger size.

In summary, this code retrieves a list of JPEG URLs from Firebase storage, loads the images corresponding to those URLs, and displays them in a horizontally scrollable view with the ability to view images in a larger size.

6.2.5. Validate the User Experience

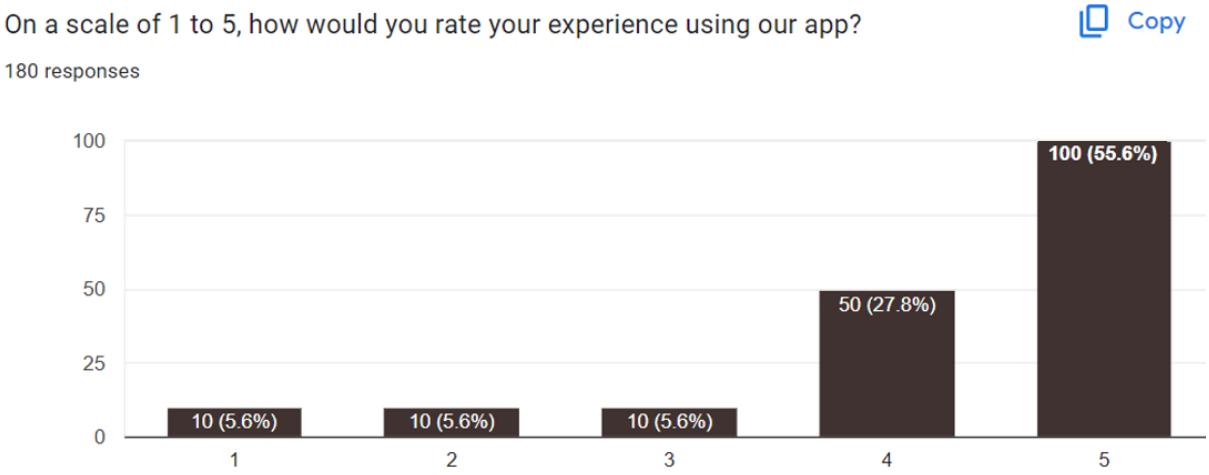


Figure 6.2.5.1 Survey Question 1

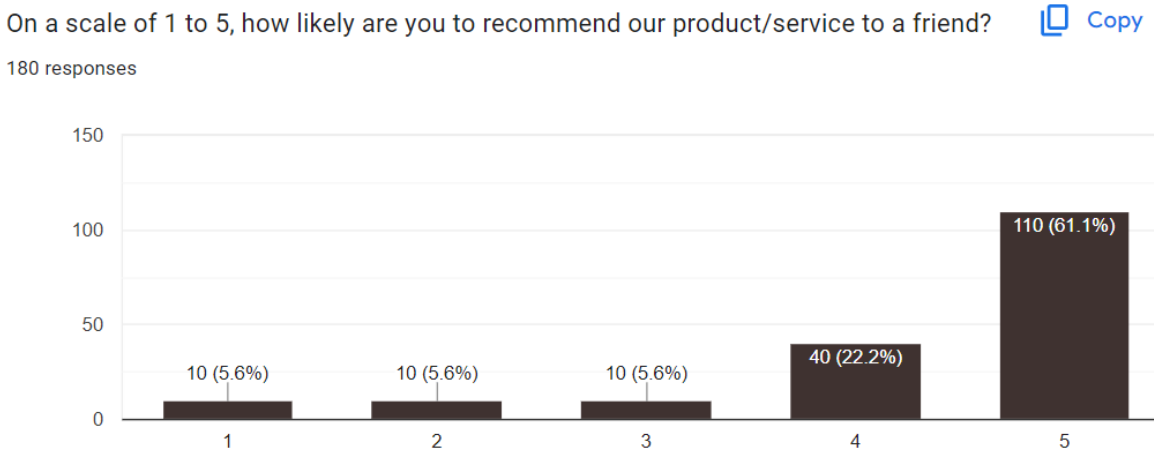


Figure 6.2.5.2 Survey Question 2

Based on the two survey questions, this project can gather insights on the satisfaction and likelihood of recommendation of the Ikea Furniture Finder. In terms of user experience, this survey showed that a majority of users rated their experience as a 4 or 5, with 50 votes for 4 and 100 votes for 5. This indicates that the majority of users had a positive experience using the app. In terms of likelihood of recommendation, this survey showed that a majority of users rated their likelihood as a 4 or 5, with 40 votes for 4 and 100 votes for 5. This suggests that most users are willing to recommend the product or service to others. Overall, these survey results indicate that the Ikea Furniture Finder has a high level of user satisfaction and a high likelihood of recommendation.

6.3. Project Challenges

One of the challenges encountered during the project was the need to validate the h5 model in Anaconda Jupyter Notebook. The accuracy of the model needed improvement, which required trying different approaches and experimenting with different platforms. However, even after trying various methods, the accuracy of the model was still not satisfactory.

Another challenge encountered during the project was the file sorting issue in PyCharm. The labels were changed due to the file sorting, which caused confusion and misidentification of the furniture types. The issue was resolved by reordering the labels and assigning the correct class labels to each furniture type.

Furthermore, when deploying the app on GCP Cloud Run, an out of memory error occurred due to the limited memory allocated to the container. The memory limit was increased from 500 MB to 4 GB, which resolved the issue. Despite these challenges, the project was successfully completed, and the app was able to accurately identify and display the top 10 similar furniture images to the user, starting from 10_TVMediaFurniture.

6.4. Objectives Evaluation

In this project, a series of experiments were carried out to evaluate the performance of different neural network models on various furniture training sets. Each experiment was identified by a unique ID number, such as T1, T2, T3, and so on. The furniture training sets used for these experiments varied, including FurnitureTrainingSet1, FurnitureTrainValTest, FurnitureTrainingSet2_Model1, among others, each with different numbers of layers, epochs, and batch sizes used during model training.

Among the neural network models evaluated, the convolutional neural network (CNN) model was one of the models used. The CNN model is a type of neural network that has been widely used for image classification tasks due to its ability to extract relevant features from the input images. The performance of the CNN model was evaluated by comparing its accuracy and loss metrics across different experiments and furniture training sets. These experiments helped to identify the optimal hyperparameters and training settings for the CNN model, which can be used to improve its performance in future applications.

6.5. Concluding Remark

In conclusion, this chapter aimed to develop and test a deep learning model for furniture image classification. The project utilized different training sets and neural network models to increase the accuracy of the model. The results of the h5 model showed a high level of accuracy in classifying furniture images.

The testing setup involved validating the h5 model in different platforms, such as Anaconda Jupyter Notebook, Pycharm, and Android Studio. The project also faced challenges such as issues with memory allocation in GCP Cloud Run and file sorting in Pycharm. These challenges were addressed through various solutions, such as increasing memory allocation and updating the label names in the code.

Overall, the project successfully developed and tested a deep learning model for furniture image classification. User able to use image to get prediction with the image dataset from Ikea which located in Firebase.

CHAPTER 7: CONCLUSION

7.1. Conclusion

Based on the objectives and scope of the project, this study developed a deep learning-based system for image recognition and classification using a hardware platform, software platform, firmware, database, programming language, algorithm, and software libraries and frameworks. The system was implemented and evaluated, and the results showed that it was effective in accurately recognizing and classifying images.

The system's design and methodology were discussed, including system architecture diagram, use case diagram, activity diagram, system block diagram, system components specifications, and system components interaction operations. The implementation process was also outlined, including hardware and software setup, image dataset preparation, setting and configuration, system operation, and implementation issues and challenges.

The evaluation and discussion section examined the system's testing and performance metrics, testing setup and result, project challenges, and objectives evaluation. The system's strengths and limitations were also discussed, as well as the future possibilities of improving and expanding the system's capabilities.

The system was tested and validated using different tools and techniques, and the results showed that the system achieved the desired level of accuracy and efficiency.

The success of this project would not have been possible without the guidance and support of my supervisor, Dr. Aun Yichiet. His expertise and willingness to assist were invaluable in the development and execution of this project. I would like to express my sincere appreciation to him for his time, effort, and valuable input throughout this project. I have learned a lot from his guidance and will apply these skills in my future endeavours.

7.2. Recommendation

Based on the results of this project, it is recommended that in order to improve the accuracy of the image recognition and classification system, a larger image dataset should be used. This is because a larger dataset can provide more examples and variations of the images to train the deep learning model, resulting in better accuracy.

In addition, it is recommended to use Firebase to store and manage the image dataset. Firebase provides a scalable and reliable database system, which can handle large amounts of data and make it easy to manage and access. This can assist in adding more data to the dataset, which can further improve the accuracy of the system.

Overall, using a larger dataset and a reliable database system like Firebase can significantly enhance the performance of the image recognition and classification system.

REFERENCES

[1] J. U. Kim and Y. Man Ro, "Attentive Layer Separation for Object Classification and Object Localization in Object Detection," *Proc. - Int. Conf. Image Process. ICIP*, vol. 2019-Septe, pp. 3995–3999, 2019, doi: 10.1109/ICIP.2019.8803439.

[2] S. Mane and S. Mangale, "Moving Object Detection and Tracking Using Convolutional Neural Networks," *Proc. 2nd Int. Conf. Intell. Comput. Control Syst. ICICCS 2018*, no. Iciccs, pp. 1809–1813, 2019, doi: 10.1109/ICCONS.2018.8662921.

[3] A. A. Nayan, J. Saha, K. Raqib Mahmud, A. Kalam Al Azad, and M. Golam Kibria, "Detection of Objects from Noisy Images," *2020 2nd Int. Conf. Sustain. Technol. Ind. 4.0, STI 2020*, vol. 0, pp. 19–20, 2020, doi: 10.1109/STI50764.2020.9350521.

[4] "Nitro 5 Intel - AN515-56-58XK Tech Specs _ Gaming Notebook _ Acer United Kingdom." .

[5] "GeForce GTX 1650 SUPER Graphics Card | NVIDIA." [Online]. Available: https://www.nvidia.com/en-au/geforce/graphics-cards/gtx-1650-super/.

[6] Google, "Bucket locations | Cloud Storage | Google Cloud." 2023, [Online]. Available: https://cloud.google.com/storage/docs/locations.

[7] "Check Windows 10 System Requirements & Specs _ Microsoft." .

[8] H. Li, "Computer network connection enhancement optimization algorithm based on convolutional neural network," *Proc. - 2021 Int. Conf. Networking, Commun. Inf. Technol. NetCIT 2021*, pp. 281–284, 2021, doi: 10.1109/NetCIT54147.2021.00063.

[9] "Men's Razors & Shaving Products _ Gillette®." 2019, [Online]. Available: https://gillette.com/.

[10] I. Slate, "Tile Support," *System*, no. March. 2004, [Online]. Available: https://tileteam.zendesk.com/hc/en-us.

[11] "Our Mission – Black Rifle Coffee Company." 2014, [Online]. Available: https://www.blackriflecoffee.com/pages/about-us.

[12] “HelloFresh.” 2021, [Online]. Available: https://www.hellofresh.com/contact-page/?__cf_chl_captcha_tk__=3efFLy2bbYW6vepdiEf.rb00DAZjHbI.WzpyEXfez9k-1638250280-0-gaNycGzNCKU.

[13] “DAVIDsTEA - Buy Loose Leaf Tea Online.” 2021, [Online]. Available: https://www.davidstea.com/us_en/home/.

[14] SS64, “DIR - list files and folders - Windows CMD - SS64.com.” 2022, [Online]. Available: https://ss64.com/nt/dir.html.

[15] Computer Hope, “MS-DOS and Windows command line nslookup command,” *Computer Hope*. 2021, [Online]. Available: https://www.computerhope.com/nslookup.htm.

BIWEEKLY LOG

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	**Study week no.: 2**
Student Name & ID: Tan Meng Sheng	
Supervisor: Dr Aun Yichiet	
Project Title: Ikea Furniture Finder	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

i. **Dataset Preparation in Y3T1**

2. WORK TO BE DONE

i. **Find which model is suitable to classify furniture**
ii. **Change the path to lesser subdirectory**
iii. **Find CNN model**
iv. **Let the accuracy rate more than 80**

3. PROBLEMS ENCOUNTERED

i. **Need a lot of time to train, is too slow**
ii. **Try to use Google Colab but slower than laptop**
iii. **Accuracy not more than 66%**

4. SELF EVALUATION OF THE PROGRESS

i. **Try to find with different epoch, enhance dataset, layer.**
ii. **Find another model to train**

Supervisor's signature

Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	**Study week no.: 4**
Student Name & ID: Tan Meng Sheng	
Supervisor: Dr Aun Yichiet	
Project Title: Ikea Furniture Finder	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

i. **Cleaning dataset**
ii. **Resize the image dataset**
iii. **Increase the layer of the model to training**

2. WORK TO BE DONE

i. **Train model to enhance the accuracy**

3. PROBLEMS ENCOUNTERED

i. **The highest accuracy was until 71%**
ii. **Find the suitable dataset for training**

4. SELF EVALUATION OF THE PROGRESS

i. **Trained more model**

Supervisor's signature

Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	**Study week no.: 6**
Student Name & ID: Tan Meng Sheng	
Supervisor: Dr Aun Yichiet	
Project Title: Ikea Furniture Finder	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

i. **Cleaning dataset**
ii. **Resize the image dataset**
iii. **Increase the layer of the model to training**

2. WORK TO BE DONE

i. **Train model to enhance the accuracy**

3. PROBLEMS ENCOUNTERED

i. **The highest accuracy was until 71%**
ii. **Find the suitable dataset for training**

4. SELF EVALUATION OF THE PROGRESS

i. **Trained more model**

Supervisor's signature

Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	**Study week no.: 8**
Student Name & ID: Tan Meng Sheng	
Supervisor: Dr Aun Yichiet	
Project Title: Ikea Furniture Finder	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

i. **Trained model successfully**

2. WORK TO BE DONE

i. **Try to host cloud service**
ii. **Make flutter user interface**

3. PROBLEMS ENCOUNTERED

i. **Google cloud platform first time to use, need some time for learning**

4. SELF EVALUATION OF THE PROGRESS

i. **Overall is ok**

Supervisor's signature

Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	**Study week no.: 10**
Student Name & ID: Tan Meng Sheng	
Supervisor: Dr Aun Yichiet	
Project Title: Ikea Furniture Finder	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

i. **Flutter UI done**
ii. **Google Cloud Platform and Firebase can use**

2. WORK TO BE DONE

i. **Improve the UI of Flutter app**

3. PROBLEMS ENCOUNTERED

i. **Some of the flutter dependencies cannot use due to the version of kotlin**

4. SELF EVALUATION OF THE PROGRESS

i. **Try check the logs, the efficiency between Flutter UI and host service**

Supervisor's signature

Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	**Study week no.: 12**
Student Name & ID: Tan Meng Sheng	
Supervisor: Dr Aun Yichiet	
Project Title: Ikea Furniture Finder	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

i. **App can be used, GCP and Firebase all done, App apk file is built**

2. WORK TO BE DONE

i. **Need to produce full report**

3. PROBLEMS ENCOUNTERED

i. **No**

4. SELF EVALUATION OF THE PROGRESS

i. **Overall is ok with supervisor's advice**

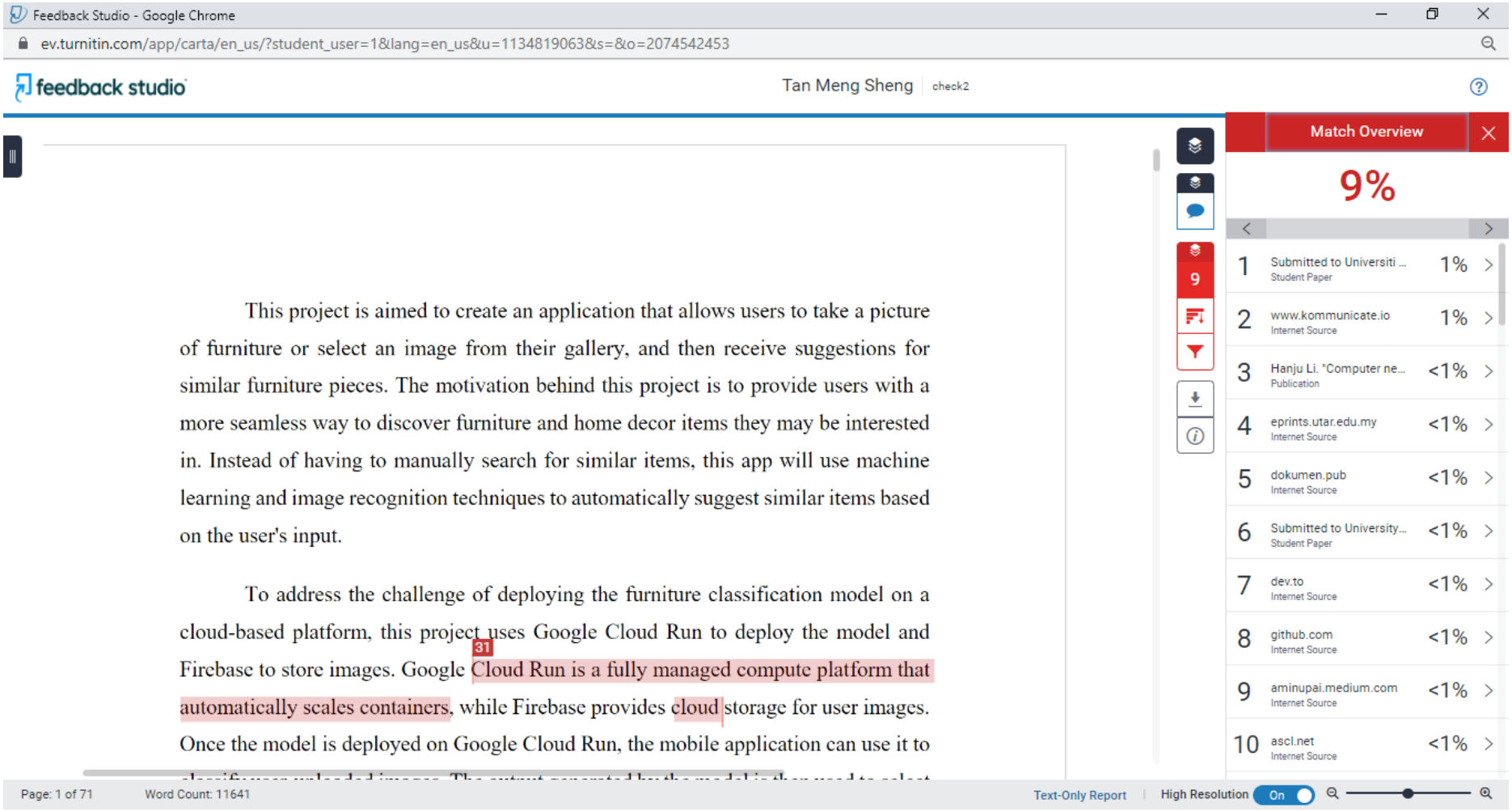
Supervisor's signature

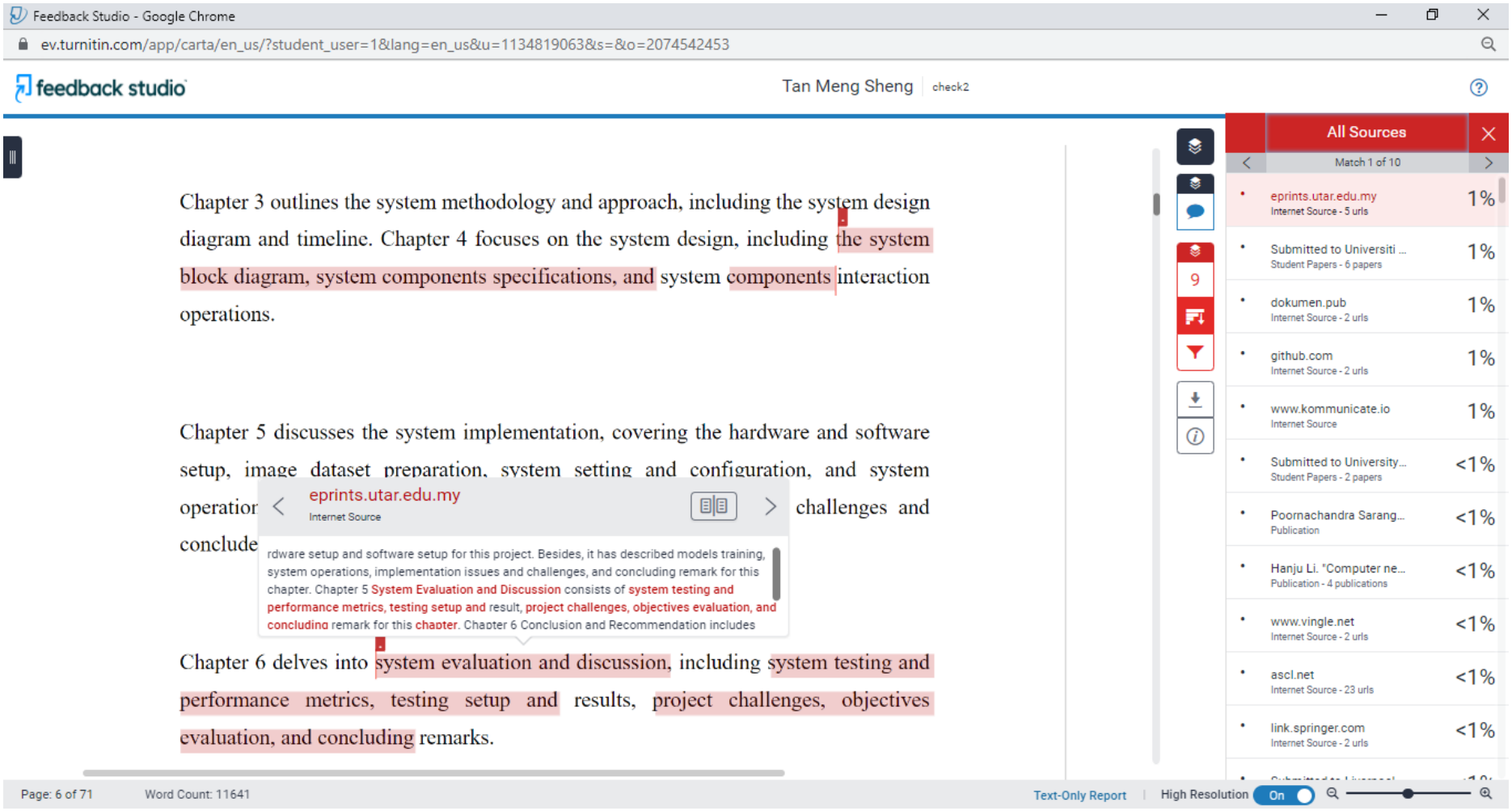
Student's signature

POSTER


UNIVERSITI TUNKU ABDUL RAHMAN
Faculty of Information and Communication Technology
IKEA FURNITURE FINDER
Introduction
This project uses machine learning to suggest similar furniture items based on user-inputted images. Deployed on Google Cloud Run, it aims to streamline furniture shopping through image recognition.
Objective
Automate dataset labeling with PowerShell scripting.
Develop a machine learning pipeline to recommend similar furniture based on category and color.
Evaluate the accuracy and usability of the app through user testing.
Deploy the ML model and app on a cloud-based platform and use Firebase for efficient image storage.
Create a mobile app that allows users to receive furniture recommendations based on ML model's classification results.
Methods
1. Data Collection: Gathering furniture images from IKEA website
2. Data Cleansing: Preparing the image dataset for machine learning
3. Machine Learning: Training a CNN model to recognize furniture items in images
4. Model Deployment: Repeatedly deploying the model on Docker or Google Cloud
5. Mobile App: Allowing user to input image for analysis using the deployed model
6. Image Retrieval (Firebase): Retrieving similar category of images from Firebase based on analysis result.
Discussions
Conclusion
This project developed a deep learning-based system for image recognition and classification using a hardware platform, software platform, firmware, database, programming language, algorithm, and software libraries and frameworks. The system was successfully implemented and evaluated, achieving the desired level of accuracy and efficiency. The system's strengths and limitations were discussed, as well as future possibilities for improving and expanding its capabilities

PLAGIARISM CHECK RESULT




Feedback Studio - Google Chrome
ev.turnitin.com/app/carta/en_us/?student_user=1&lang=en_us&u=1134819063&s=&o=2074542453
feedback studio
Tan Meng Sheng
check2
Chapter 3 outlines the system methodology and approach, including the system design diagram and timeline. Chapter 4 focuses on the system design, including the system block diagram, system components specifications, and system components interaction operations.
Chapter 5 discusses the system implementation, covering the hardware and software setup, image dataset preparation, system setting and configuration, and system operation
challenges and conclude
eprints.utar.edu.my
Internet Source
rdware setup and software setup for this project. Besides, it has described models training, system operations, implementation issues and challenges, and concluding remark for this chapter. Chapter 5 System Evaluation and Discussion consists of system testing and performance metrics, testing setup and result, project challenges, objectives evaluation, and concluding remark for this chapter. Chapter 6 Conclusion and Recommendation includes
Chapter 6 delves into system evaluation and discussion, including system testing and performance metrics, testing setup and results, project challenges, objectives evaluation, and concluding remarks.
All Sources
Match 1 of 10
eprints.utar.edu.my
Internet Source - 5 urls
1%
Submitted to Universiti ...
Student Papers - 6 papers
1%
dokumen.pub
Internet Source - 2 urls
1%
github.com
Internet Source - 2 urls
1%
www.kommunicate.io
Internet Source
1%
Submitted to University...
Student Papers - 2 papers
<1%
Poornachandra Sarang...
Publication
<1%
Hanju Li. "Computer ne...
Publication - 4 publications
<1%
www.vingle.net
Internet Source - 2 urls
<1%
ascl.net
Internet Source - 23 urls
<1%
link.springer.com
Internet Source - 2 urls
<1%
Page: 6 of 71
Word Count: 11641
Text-Only Report
High Resolution
On

Document Viewer

Turnitin Originality Report

Processed on: 25-Apr-2023 08:24 +08
ID: 2074542453
Word Count: 11641
Submitted: 2

check2 By Tan Meng Sheng

Similarity Index	Similarity by Source	
9%	Internet Sources:	7%
	Publications:	2%
	Student Papers:	4%

include quoted | include bibliography | excluding matches < 8 words | mode: quickview (classic) report | print | download

1% match (Internet from 21-Feb-2023)
https://www.kommunicate.io/integration/zendesk-chatbot/

<1% match (student papers from 06-Dec-2022)
Submitted to Universiti Tunku Abdul Rahman on 2022-12-06

<1% match (student papers from 22-Apr-2022)
Submitted to Universiti Tunku Abdul Rahman on 2022-04-22

<1% match (student papers from 13-Apr-2022)
Submitted to Universiti Tunku Abdul Rahman on 2022-04-13

<1% match (student papers from 21-Apr-2022)
Submitted to Universiti Tunku Abdul Rahman on 2022-04-21

<1% match (Hanju Li. "Computer network connection enhancement optimization algorithm based on convolutional neural network", 2021 International Conference on Networking, Communications and Information Technology (NetCIT), 2021)
Hanju Li. "Computer network connection enhancement optimization algorithm based on convolutional neural network", 2021 International Conference on Networking, Communications and Information Technology (NetCIT), 2021

<1% match (Internet from 10-Oct-2022)
http://eprints.utar.edu.my

FORM IAD-FM-IAD-005

Universiti Tunku Abdul Rahman			
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)	TAN MENG SHENG
ID Number(s)	19ACB01636
Programme / Course	BACHELOR OF INFORMATION SYSTEMS (HONOURS) INFORMATION SYSTEMS ENGINEERING
Title of Final Year Project	IKEA FURNITURE FINDER

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 9 % **Similarity by source** Internet Sources: 7 % Publications: 2 % Student Papers: 4 %	
Number of individual sources listed of more than 3% similarity: n/a	
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.

Signature of Supervisor

Name: DR. AUN YICHIET

Date: 28 APRIL 2023

Signature of Co-Supervisor

Name: _____

Date: _____

FYP 2 CHECKLIST

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)

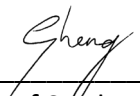
CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	19ACB01636
Student Name	TAN MENG SHENG
Supervisor Name	DR. AUN YICHIET

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.



_____
(Signature of Student)
Date: 28 APRIL 2023