Dissertation on

"Automated Shopping Cart"

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

UE19CS390A – Capstone Project Phase - 1

Submitted by:

Kuntal Gorai	**PES2UG19CS198**
S V S C Santosh	**PES2UG19CS346**
Skanda S	**PES2UG19CS391**
Vijay Murugan A S	**PES2UG19CS454**

Under the guidance of

Prof. Prajwala T R
Assistant Professor
PES University

January - May 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

'Automated Shopping Cart'
is a bonafide work carried out by

Kuntal Gorai	**PES2UG19CS198**
S V S C Santosh	**PES2UG19CS1346**
Skanda S	**PES2UG19CS391**
Vijay Murugan A S	**PES2UG19CS454**

In partial fulfilment for the completion of sixth semester Capstone Project Phase - 1 (UE19CS390A) in the Program of Study -Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period Jan. 2022 – May. 2022. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 6th semester academic requirements in respect of project work.

Signature	Signature	Signature
Prof Prajwala T R	Dr. Sandesh B J	Dr. B K Keshavan
Designation	Chairperson	Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. ______________________	______________________
2. ______________________	______________________

DECLARATION

We hereby declare that the Capstone Project Phase - 1 entitled **Automated Shopping Cart** has been carried out by us under the guidance of Prof. Prajwala T R, Assistant Professor and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester January – May 2022. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES2UG19CS198	**Kuntal Gorai**
PES2UG19CS346	**S V S C Santosh**
PES2UG19CS391	**Skanda S**
PES2UG19CS454	**Vijay Murugan A S**

ACKNOWLEDGEMENT

I would like to express my gratitude to Prof. Prajwala T R, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE19CS390A - Capstone Project Phase – 1.

I am grateful to the Capstone Project Coordinators, Dr.Sarasvathi V, Associate Professor and Dr. Sudeepa Roy Dey, Assistant Professor, for organizing, managing, and helping with the entire process.

I take this opportunity to thank Dr. Sandesh B J, Chairperson, Professor, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department. I would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro-Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this Capstone Project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

Shopping malls have become an integral part of city life and a lot of people commute to these marts to get their daily essentials and groceries. However, we are all forced to go through the tedious task of getting our items billed. The billing phase is the one choke point every customer must pass through. Moreover, every customer takes a variable amount of time to get their items billed, causing huge queues. Most of the current techniques available to make shopping easier are the use of RFID tags or barcodes which cannot be used for every item in the supermarkets like fruits and vegetables as they will have to be weighed and billed separately which is again time-consuming, so We aim to make this whole experience of shopping easier. For doing so we make use of ML techniques to classify the items directly, IoT devices like an Arduino/Raspberry PI along with additional components like load cell which can be used to detect weights, and an Android Studio based app which the users can use to interact and generate a bill by doing so the need of multiple billing centers and workload in supermarkets can be reduced which would prove beneficial to common people by ensuring social distancing and also reducing the average time spent in supermarkets on buying particular items.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

Our project focuses on building an application - ‘Automated shopping Cart’ which is a shopping cart with the integration of an android app and weight sensor for hassle-free shopping. The motivation for this project was to avoid the long queues that a customer has to wait for payment of the items that he/she has added to the shopping cart.

The primary features of our application include
a) An efficient deep learning model which can classify the items with very minimal error and faster classification.
We aim to make use of transfer learning with MobileNetV2 which makes it lightweight enabling it to run on resource-deficient devices, also enabling the model to be trained again in absence of the pre-trained saved model. MobileNetV2 has a very low number of parameters that can be trained, which is also an advantage in our case.
b) An android app that can perform functionalities like classifying the items placed in front of the camera, adding items to a cart, removing items from the cart, and paying the bill for items added to the cart.
c) A load cell acts as a weight sensor and helps customers estimate the weight of the item they are trying to add to their cart.
d) An Arduino board to send the weights detected by the load cell with the android app.
e) Finally a cloud backend - for which we plan to employ AWS, which will serve as a database to store the details about the items like cost, quantity, and description available in the supermarket, the order history consisting of orders that the customer had bought at the store.

While the current market offers shopping carts that make use of items with RFID tags we plan to avoid such tedious tasks by making use of image classification with help of deep learning models like MobileNetv2 to classify the item and we plan to integrate the payment section in our android app which helped customers from waiting in a long queue enabling a hassle-free shopping.

CHAPTER 2

PROBLEM DEFINITION

- With the increasing complexity and time spent on average by a person on purchasing groceries from supermarkets, the number of people coming to the supermarkets has reduced, preferring to purchase the same required items online, compromising on quality and money for time.
- With the increase in number of online customers, these online service providers tend to pay less attention to the quality of items each user is receiving at the end of the day and concentrate more on expanding their user base which is not what happens when a user has the freedom of choosing items on their own thereby not compromising on quality.
- There are 3 consistent questions that then arise throughout this process:
  1) How to make the process of shopping more efficient?
  2) How do we keep track of all the changes in the price of items and also the past orders of each user?
  3) How do we attract more customers back to supermarkets?
- Our project attempts to bridge these gaps by developing an ML-based application. It is a system that helps in the easier detection of items and classifies the commodities systematically placed into the shopping cart using Deep Learning techniques.
- This system will add the item when placed into the cart and update the details of the items on the app, retrieve necessary information from the cloud-based database available on AWS, and finally generate the bill.

CHAPTER 3

LITERATURE SURVEY

3.1 Fruit Classification for Retail Stores Using Deep Learning

Authors: Jose Luis Rojas-Aranda, Jose Ignacio Nunez-Varela, J. C. Cuevas-Tello, Gabriela Rangel-Ramirez

This paper talks about developing a simple lightweight CNN-based model for classifying the 3 types of fruits taken into consideration in this paper, which are apple, banana, and orange. Their main aim is to classify these fruits based on their color and texture, which was implemented by making use of transfer learning with a pre-trained model, i.e. MobileNet V2. The color was obtained from 3 techniques which were Single RGB Color where the color was sent as a vector of RGB values; RGB histogram where a histogram of all colors is sent and the peak from it is chosen as the color, and RGB centroid using K means where the centroid of all colors was chosen. Taking both into consideration, the accuracy was also improved and the model worked very well for predicting the desired class both when the object was placed in plastic bags and when not in plastic bags, thereby proving sufficient for achieving the same.

Advantage:

- Use of MobileNetV2 made it lightweight and computationally inexpensive. The accuracy was improved by taking into consideration other features as input, namely color.
- Also took into consideration the placement of items in plastic bags, which is what happens in daily life.

Limitation:

- The project had only 3 classes of fruits, although the number of classes in real life exceeds 100.
- Making use of RGB histogram proved background color to play a major role in determining the class of item so the background color had to be constant.

3.2 A Hierarchical Grocery Store Image Dataset with Visual and Semantic Labels

Authors: Marcus Klasson, Cheng Zhang, Hedvig Kjellstrom

This paper aims at helping people with visual impairments by providing support when shopping for grocery items. This project made use of both generative and deterministic deep neural networks along with a simple linear classification model like SVM. The main task in this paper was to make use of pre-trained CNNs like Densenet, AlexNet, and VGG16 to extract features and then convert the extracted features into a vector which enables them to be placed onto some classifier which enables classification based on the extracted features. This paper also made use of VAE (Variational AutoEncoder) to train based on natural images. The best accuracy for achieving the task was achieved by making use of DenseNet with an SVM classifier which can be understood due to the increase in the number of trainable layers, thereby leading to an increase in the number of parameters leading to better accuracy.

Advantages:

- Worked well even on a smaller dataset for training.
- The images as seen in real life were taken into consideration.

Limitations:

- The need for high GPU and CPU capacity and good hardware to perform fast and accurate classification because of the large number of parameters involved.
- The time needed for developing the model by training on images is quite a lot.

3.3 Deployment of Deep Learning Models on Resource-Deficient Devices for Object Detection

Authors: Ammeema Zaina, Dabeeruddin Syed

The aim of this paper was to deploy deep learning models on devices which has fewer resources like GPU which is required for training i.e mobile phones. The model that was used for training was YOLO(You Only Look Once). The model was built on a VOC data set. This data consisted of 20 labeled classes. The network had 19 convolution layers,2 fully connected layers, and a max pool layer for the reduction of dimensions. Once the model is pre-trained to make the detection to happen in real-time, they made use of darknet with CUDA, dark flow, and OpenCV. The first weights of the pre-trained model are stored in a .weights file. Next, they made use of Darkflow to convert this weight file to a protobuf file which is mobile device compatible. Once converted it is deployed into the mobile application. Finally, to be able to detect and classify images successfully, they made use of the TensorFlow module present in Android. Tensorflow module made use of three functionalities Classify, Stylize and Detect. TensorFlow Made use of two more files i.e .so (shared object) file and .jar file. These two files are built using Bazel. Once all the files are available in the Android Studio package, they can be deployed in real-time.

Advantages:

- The device detects objects in real-time, and it does not require any access to the internet.
- The object detection model was able to detect and classify objects in the mobile device within a fraction of seconds.

Limitations:

- Generation of protobuf files is quite challenging.
- Incompatibilities with versions of python and OpenCV
- Moving a huge neural network to a mobile device and making the device understand the deep layers of the network.
- Object detection for mobile devices moving at a higher speed.
- Object detection under low illumination.
- Object detection for images displayed on-screen or monitor.

3.4 Android Application for Grocery Ordering System

Authors: Shubham B Dubey, Gaurav M Kadam, Omkar Angane

This paper deals with how an Android application can be built from scratch with the help of Android Studio, which is Android's official IDE. It helps one build the highest quality applications for every android device providing extensive tools to help test the app, making it bug-free. The app developed here was an app for people to order groceries online. With a user-friendly GUI, the aim was to make sure anybody could easily order grocery items. The rest of the paper dealt with the step-by-step development of the described app.

The application consisted of activities helping with navigation and product scanning, enabling users to browse through products and choose the ones they need. The categorization of grocery products is aimed at making the search for items easier than manually searching for them. The UUID was used to connect the smart card with the app and the UID of the items that were scanned. Upon testing the workings of the app, a success rate of 100 percent was obtained.

Advantage:

- When tested on an emulator and on a smartphone, the program ran well and without issues.
- At no point during the process did the application crash.
- When testing the app's functionality, it was discovered that it had a 100 percent success rate in each direction of motion control.

Limitation:

- This application is Android-specific and does not support any iOS devices.
- The UID of all items had to be scanned.

3.5 Cross-Validation Voting for Improving CNN Classification in Grocery Products

Authors: Jamie Duque Domingo , Roberto Medina Aparicio, and Luis Miguel Gonzalez Rodrigo
In this paper, a Cross-Validation-Voting (CVV) technique for grocery product classification is presented. This technique improves several single state-of-the-art classifiers without combining different ones and avoids the problems of overfitting with respect to the training set. The training dataset is distributed in different training and validation slots so that we use all the data for training the ensemble model. Each individual model uses a part of the general training dataset as training and another part as validation. This technique was used to evaluate the improvements in three models, namely ResNeXt-101, EfficientNet B7, and Wide ResNet-101. The models showed the most improvement when the ensemble model was used with soft voting. In soft voting, the probabilities of each class are accumulated among the different models and the one with the highest probability is the winner.

Advantages: The accuracy of classification of this method was seen to go up to 93.68%.

Limitations: The training time is very high as each model in the ensemble took an hour to train when using an i9-10900K server with 128GB RAM and 2 GPU RTX-3090 with 24GB GDDR6X.

CHAPTER 4

DATA

4.1 Overview

A complete collection of images of fruits and vegetables was not readily available and had to be constructed. Multiple sources were referred and only the best were chosen to be included. Scripts were also used to scrape relevant images from websites.

4.2 Dataset

The dataset currently contains 30 unique classes segregated into train, validation and test sets in the ratio 6: 3: 1. Each training class has a minimum of 600 images and up to a maximum of 1.5k images.

CHAPTER 5

SYSTEM REQUIREMENTS SPECIFICATIONS

5.1 INTRODUCTION

The purpose of this document is to provide a detailed description of the application to be developed - “Automated Shopping Cart”.
It includes a thorough study of various requirements fulfilled by the application, use cases, various constraints on the implementation, and other technical as well as non-technical aspects involved in the project.
This document serves as a brief to outline our project and as a foundation for further developments in the future.

5.1.1 Project Scope

5.1.1.1 Objective

Develop a fully working application that generates a final bill of all the items purchased by the customer without the hassle of standing in huge lines to get them billed the conventional way, thereby saving a lot of time and making it feasible for anyone with a smartphone to use the app.

5.1.1.2 Problem

- With the increasing complexity and time spent on average by a person on purchasing groceries from supermarkets, the number of people coming to the supermarkets has reduced, preferring to purchase the same required items online, compromising on quality and money for time.
- With the increase in the number of online customers, these online service providers tend to pay less attention to the quality of items each user is receiving at the end of the day, and

concentrate more on expanding their user base which is not what happens when a user has the freedom of choosing items on their own thereby not compromising on quality.

- There are 3 consistent questions that then arise throughout this process:
  1) How to make the process of shopping more efficient?
  2) How do we keep track of all the changes in the price of items and also the past orders of each user?
  3) How do we attract more customers back to supermarkets?

5.1.1.3 Our project

- Our project attempts to bridge these gaps by developing an ML-based application. It is a system that helps in the easier detection of items and classifies the commodities systematically placed into the shopping cart using Deep Learning techniques.
- This system will add the item when placed into the cart and update the details of the items on the app, retrieve necessary information from the cloud-based database available on AWS, and finally generate the bill.

5.1.1.4 Limitations

1) Our app is designed to work only on an android-based mobile device and cannot be used on the iOS platform.
2) Users must be familiar with the working of this android application.
3) We make use of third-party services like RazorPay to proceed with the payment at the end of shopping.
4) We have to work on improving the accuracy of our ML-based algorithm so that the number of false negatives and false positives are minimized. Example:- Our model must not predict a cucumber as an Ivy gourd which looks very similar in characteristics.
5) We plan to work on making our application more user-friendly and less prone to crashes.

5.2 PRODUCT PERSPECTIVE

In the day-to-day lives of people who visit shopping marts to purchase their needs, they spend an enormous amount of time waiting in long queues to get the billing done the conventional way. Hence, with the help of this app, we aim to cut down the time spent by customers in queues and make the whole shopping experience better.

5.2.1 Product Features

1) **Login/Sign up:** A basic login or sign-up feature (High priority).
2) **Classify Items:** This feature will classify the items based on the pre-trained model, and uses TensorFlow in Android Studio for real-time detection (High priority).
3) **Detect weights:** This feature makes use of a load cell to get the weight of the item placed on the cart (High priority).
4) **Generate Bill:** Finally Generates a bill based on items calssified and the weights detedcted which can be then paid off by the user at time of exit.
5) **Payment**: The payment part of this project is being handled by Third party providers like razor pay.

5.2.2 User Classes and Characteristics

1. User

- Show items to our app to detect them.
- Add items to our cart containing load cell
- Verify the quantity and press add button on Arduino
- Verify the items with items on the app
- Pay the final item through various payment gateways

2. Admin

- Make updates to the item based on stock available in mart
- Add users to database based on user details sent to app

5.2.3 Operating Environment

- **Environment:** The software will work as an application restricted to Android-based smartphones.
- **Operating System:** Android 8.1 and above.
- **Database:** AWS, Firebase.
- **Software Component:** Inputs include real-time image capturing and weight from the load cell. Outputs include updating of cart items and bill generation.

5.2.4 General Constraints, Assumptions and Dependencies

5.2.4.1 Assumptions and Dependencies

- Customers should have a basic understanding of the workings of an application.
- Objects are placed in a transparent bag
- Final Payment is handled by some 3rd party interface.
- The customer has an Android phone with minimal computational resources to use the app.
- Dependencies: Keras, TensorFlow, OpenCV, DarkFlow

5.2.4.2 Constraints

- User Interface Constraints: The user must be familiar with the workings of this Android application.
- Hardware Constraints:
    - The application would work on Android devices which use Android version 8.1 and above and would also require a functioning camera.
    - Inaccuracy of load cell
    - Training requires powerful GPUs

- Software Constraints:
  - The system will be intended to run on Android 8.1 and above.
  - Time optimization of algorithms used to classify images
  - Time to update the bill in the app.
  - A customer shows a cheaper product and places the expensive product into the cart i.e., fooling the system.

- Design Standards Compliance: The application will be implemented using Android Studio.

- Operational Constraints: The application supports multiple users operating it at the same time.

5.2.5 Risks

- Security protocols may not be followed by the admin, which could be a breach of privacy as the application requires access to the user's device gallery and camera.
- Instances on the cloud that are not being used may not be terminated and this may cost a lot of money.
- Model using up too many computational resources causing the app to crash.
- Failure of Hardware

5.3 FUNCTIONAL REQUIREMENTS

- User login and authentication with Google Firebase.
- The ability of the system to classify items shown by users accurately.
- Fetch item details and add items to the bill.
- The app is robust and is always connected to the Arduino when in use to receive information about the weights of items being placed in the cart.

5.4 EXTERNAL INTERFACE REQUIREMENTS

5.4.1 User Interfaces

- The screen format should be limited to the respective mobile screen size.
- Supported platforms are Android only.
- Screen orientation should be limited to portrait layout only.
- GUI Standards that will be followed are:
  - The usage of a bottom navigation bar allows for easy navigation.
  - The software has a simple UI to make it more user-friendly.
  - A Google Pixel 2 XL will be used to create the standard screen. The title of the screen will be displayed in the app bar on all screens.
  - A floating action button or a conventional button, whichever is appropriate, should be utilized for functions requiring user interaction.
  - Snackbar messages will be used to display error messages.

5.4.2 Hardware Requirements

- Android devices must have internet hardware and support HTTP protocols.
- The camera of the phone must be functional.
- A load cell to measure the weight of the item being placed on the cart
- Arduino board for communication between the android app and the load cell

5.4.3 Software Requirements

- Classification Of Grocery Items:
  - Name: Transfer Learning with MobileNetV2
  - Tools and libraries: TensorFlow, Keras, Jupyter Notebook
  - Source: https://www.tensorflow.org/tutorials/images/transfer_learning
  -

- Generation of protobuf file :
    - Name: DarkFlow with OpenCV
    - Tools: OpenCV and Cython
    - Source: http://www.darkflow.org/docs/home.en/

- App UI:
    - Operating System: Android
    - Version: Android 8 and above
    - Tools: Figma (For High-Level Design and Prototyping)

5.4.4 Communication Interfaces

- The HTTPS network protocol is used to host the AWS cloud platform. Google Firebase, which also runs on HTTPS, port number 443, could be used for user authentication.
- The app directly communicates with the Arduino board continuously sending information about a load of items placed into the shopping cart being detected by the load cell. This communication could either be wired or be enabled for a very short distance communication by making use of Bluetooth for sending and receiving data.

5.5 NON-FUNCTIONAL REQUIREMENTS

5.5.1 Performance Requirement

Availability

The shopping cart that we are planning to implement with the integration of load cell and Arduino should be available to users during shopping. The application's key features like adding items to the cart, and removing items from the cart should be made available once the user has connected to the Arduino board present in the shopping cart. User should also be able to access their previous order on the app that he/she has made in the shopping cart.

Integrity & Safety

An encrypted login service is provided to ensure that access to the account is restricted to only the user and persons authorized by him/her to prevent misuse of the user's resources stored in his/her cloud.

In terms of data loss and protection, because we use AWS, a backup of the data (multiple copies of the files) stored on the cloud is maintained so that the user can always retrieve the data in the event of device failure or other unforeseen circumstances.

Performance

The performance we try to achieve is to make use of a model which is very fast i.e able to predict the class of vegetable very quickly within 3seconds to ensure that the user doesn't have to wait for a very long time also we aim at developing the model with an accuracy of close to 93% plus to ensure that the class is correctly classified preventing the user from getting tired of the misclassifications.

Correctness

All algorithms implemented in the system must be correct, which means they must perform as expected. The testing phase ensures the software's correctness by running through all possible scenarios, ensuring that the algorithms are ready to handle any exceptions that may arise.

Reliability & Robustness

We ensure all-around reliability and robustness by ensuring data protection, user privacy, and quickly and accurately classifying items placed in front of the camera.

5.5.2 Safety Requirements

1) User passwords must be encrypted.
2) Ensure one-way communication from the load cell to the Arduino and then to the Android app.
3) Camera access must not be misused by the admin and should be used only to serve their respective purposes.
4) Only the admin of the Super Market would be able to modify the weights, price, and other necessary details of that supermarket.

5.5.3 Security Requirements

1) As the application's cloud storage should be accessible from a variety of devices, the user's account or profile must be secured to prevent unauthorized personnel from accessing his or her account and its contents.
2) For classifying items, since access to the camera is required, we need to ensure that this access is approved by the user and that the necessary permissions are obtained.
3) Ensuring all security feature required for payments is enabled by the third-party payment service.

CHAPTER 6

SYSTEM DESIGN

6.1 INTRODUCTION

Shopping malls have become an integral part of city life and a lot of people commute to these marts to get their daily essentials and groceries. However, we are all forced to go through the tedious task of getting our items billed. We aim to make this whole experience of shopping easier. The proposed device will classify items as and when they are put into the cart and append their net price to the bill accessible via the mobile application for easy payment.

6.2 CURRENT SYSTEM

Existing systems use RFID tags and barcodes to bill items which is not feasible, especially for grocery items such as fruits and vegetables. The problem with this system is that it is infeasible to tag individual pieces of fruits and vegetables being a waste of time and effort.

6.3 DESIGN CONSIDERATIONS

6.3.1 Design Goals

- We aim to build a smooth and friendly interface for our application, making it intuitive and easy to navigate through the app.
- The device is not clunky and does not hinder the shopping experience of the user.
- The developed Android application is compatible with any regular Android smartphone even with little processing capacity.
- The sensitive user data is encrypted protecting the privacy of the users.

Workflow:

- The working of the system stars off when the user logins in to the app
- They then connect their mobile device to the aurdrino board connected to the shopping cart.

- Now the user can start shopping by choosing the desire items and show that to the camera of his mobile phone.
- The class of item being shown is identified within 3 seconds after which the user can place the desired weight of that item into the cart.
- The load of item added is sent from loadcell to aurdrino continuously.
- Aurdrino transmits the same to our app
- Based on the class identified by the model and prices from the database we finally generate a bill.
- User is then redircected to the payments section after which user will be able to end his shopping.

6.3.2 Architecture Choices

6.3.2.1 Classification Task for Identifying the class of vegetable

- Makes use of transfer learning-based MobileNetV2 to identify the correct class of grocery item.
- The main advantage of choosing this would include the fact that MobileNetV2 is lightweight and is robust in nature to enable it to run on any resource-deficient device.
- Another advantage of this would include the fact that the amount parameters which are trainable, taking very less time to train the model completely.

6.3.2.2 Arduino Based Connection:

- Establish a connection between the load cell and Arduino by making use of wires enabling the continuous transfer of load cell signal to Arduino.
- Establish a connection between Arduino and android app to send data of weight being placed into cart.

6.3.2.3 Mobile application

- Mobile apps are developed specifically for Android devices.

- Application development is done in Android Studio.
- This application is compatible with Android version 8 and above.
- Another option: You can use Flutter to create a common Android and iOS application, but native Android applications are lightweight and easy to use.

- Pros for using Android
  - Applications can be created easily and quickly.
  - Native Android applications are lightweight and memory efficient.
  - Easy access to camera and microphone permissions.
- Cons
  - iOS mobile devices are not supported.

6.3.2.4 Cloud Platform

- Options considered: AWS, Google, Azure
- Platform: AWS
- Pros:
  - Open source to a large extent
  - Model of flexible pricing
  - A large number of software options are available on the market.
  - A more established cloud ecosystem
  - Individual machine access and configuration is possible.
  - User permission controls that are second to none
  - S3 storage backup are quite dependable.
  - Provides a mobile development suite for quick app integration.

- Cons:
  - Hybrid cloud computing isn't as good.
  - Limitations on service (Ex: only 20 EC2 instances per region)
  - Outside of the United States, there are some features that aren't available.
  - Concerns about data privacy
- The configuration choices for Google Cloud's Firebase for Android are restricted, despite the fact that it is a fairly straightforward integration.

- Azure has the highest availability of any vendor, as well as stringent security policies. However, there are few Linux possibilities, and Microsoft doesn't do much to include the open-source community in Azure.

6.3.3 Constraints, Assumptions, and Dependencies

➔ Assumptions and Dependencies

- Customers should have a basic understanding of the workings of an application.
- Objects are placed in a transparent bag
- Final Payment is handled by some 3rd party interface.
- The customer has an Android phone with minimal computational resources to use the app.
- Dependencies: Keras, TensorFlow, OpenCV, DarkFlow

➔ Constraints

- User Interface Constraints: The user must be familiar with the workings of this Android application.
- Hardware Constraints:
  - The application would work on Android devices which use Android version 8.1 and above and would also require a functioning camera.
  - Inaccuracy of load cell
  - Training requires powerful GPUs
- Software Constraints:
  - The system will be intended to run on Android 8.1 and above.
  - Time optimization of algorithms used to classify images
  - Time to update the bill in the app.
  - A customer shows a cheaper product and places the expensive product into the cart i.e., fooling the system.
- Design Standards Compliance: The application will be implemented using Android Studio.

- Operational Constraints: The application supports multiple users operating it at the same time.

6.4 HIGH-LEVEL SYSTEM DESIGN

6.4.1 Data Flow Diagram

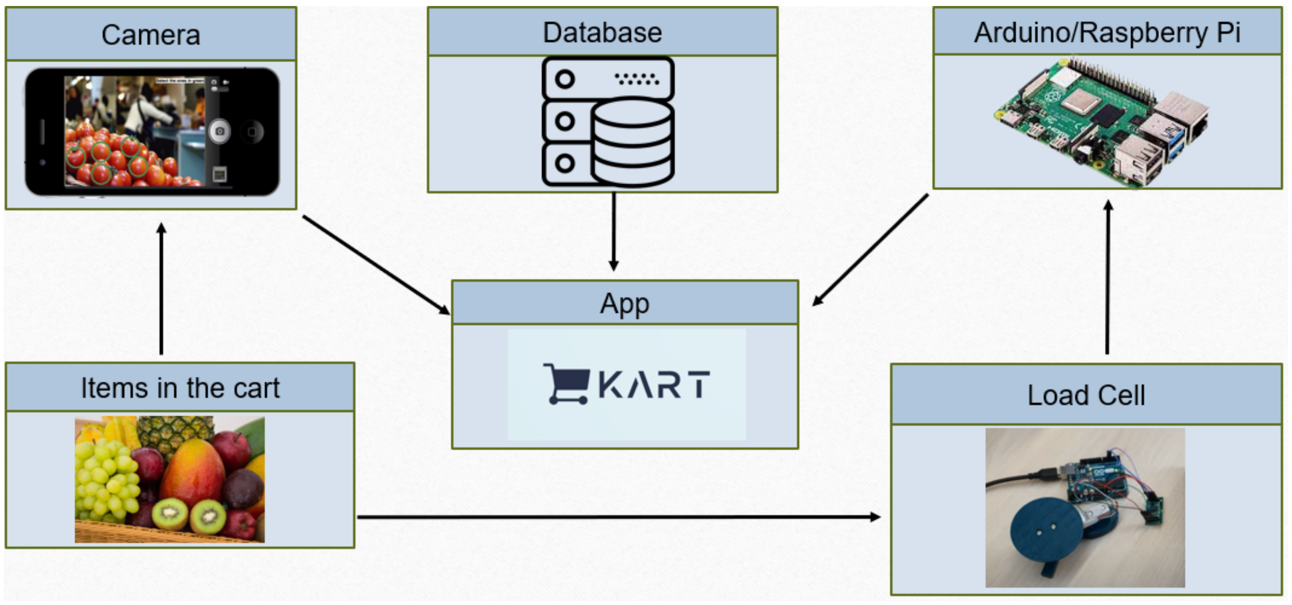


Fig 6.4.1 Data Flow Diagram

Details: This is a level 1 data flow diagram illustrating the logical data flow overview.

6.4.2 System Architecture Diagram

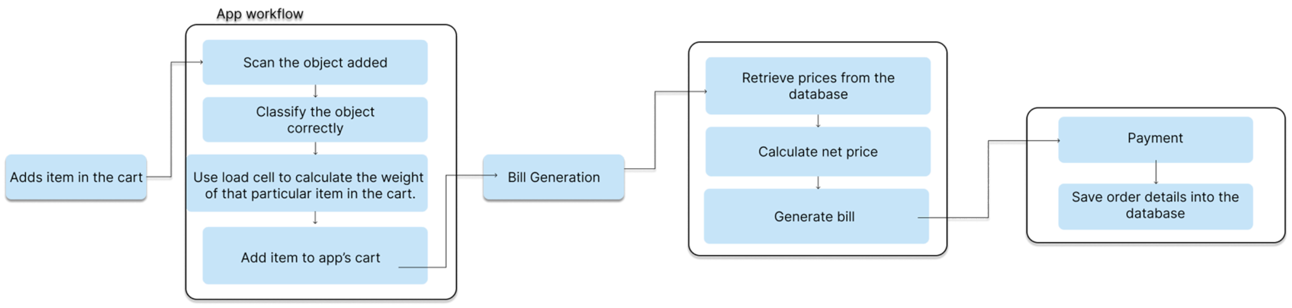

App workflow
Adds item in the cart
Scan the object added
Classify the object correctly
Use load cell to calculate the weight of that particular item in the cart.
Add item to app's cart
Bill Generation
Retrieve prices from the database
Calculate net price
Generate bill
Payment
Save order details into the database

Fig 6.4.2 System Architecture Design

The above diagram describes the different components and services that interact with each other to form the entire application and helps us visualize the flow of control.

6.4.3 Identifying system elements from different perspectives

i)Sequence Diagram - illustrates the Process and workflow of the system and the functioning of the application's elements as a whole in unison with each other is depicted.

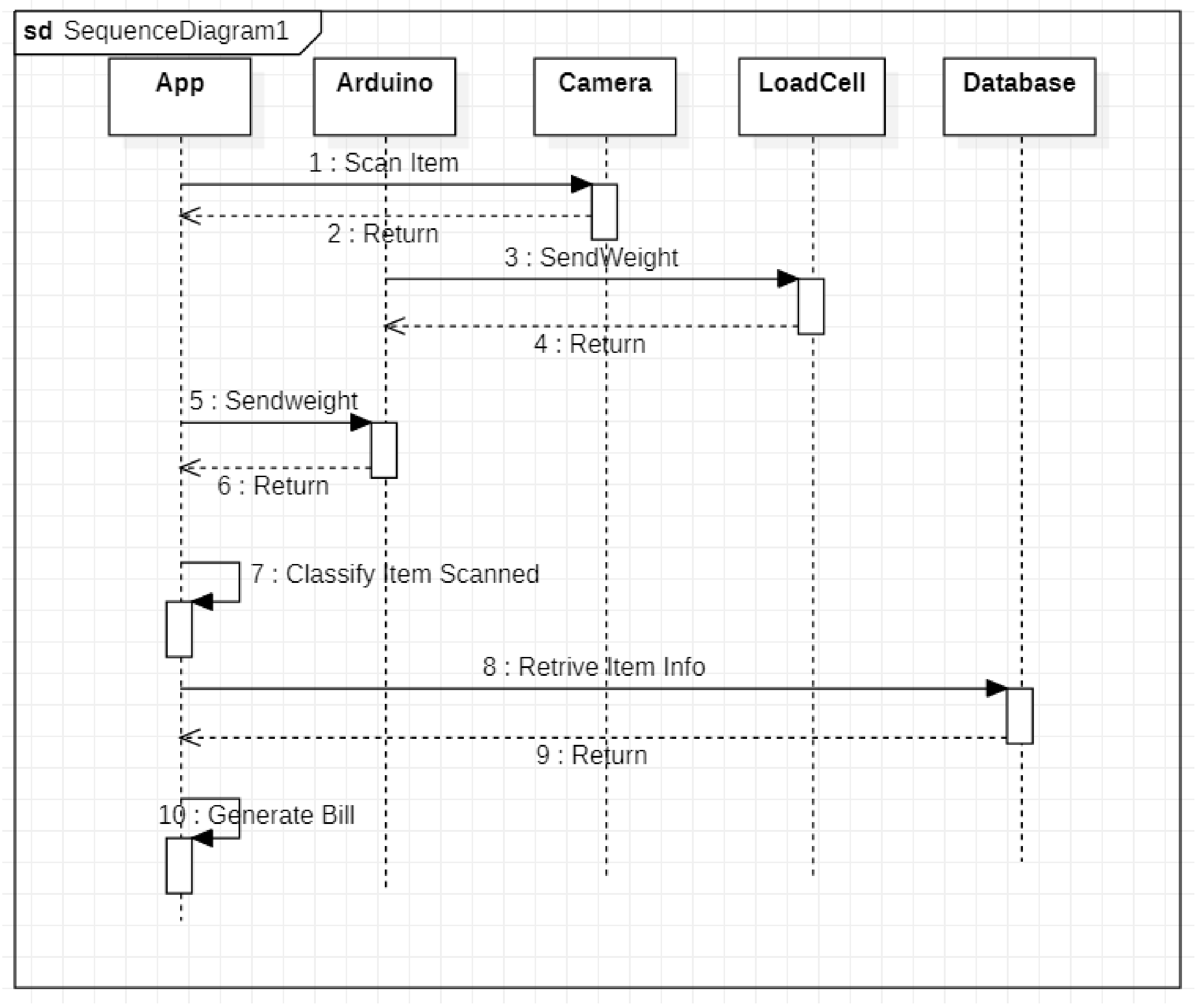


Fig 6.4.3.i Sequence Diagram

ii) Deployment Diagram - shown serves to provide an overview of the different software and hardware components and how they communicate with each other and integrate to form the system as a whole.

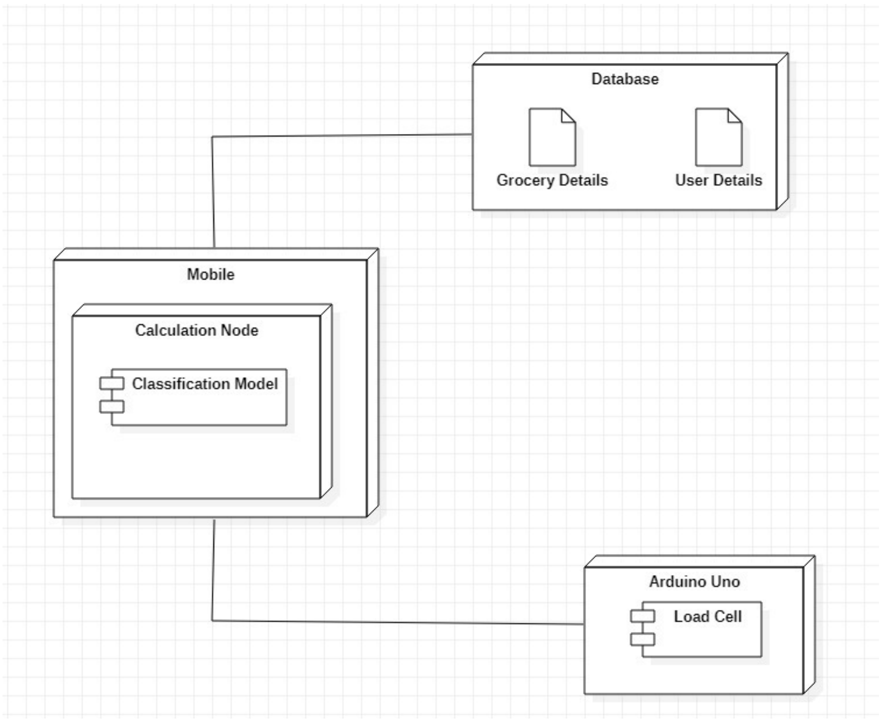


Fig 6.4.3.ii Deployment Diagram

iii) Component Diagram - which represents all the various subunits and the functional operations that tie them together.

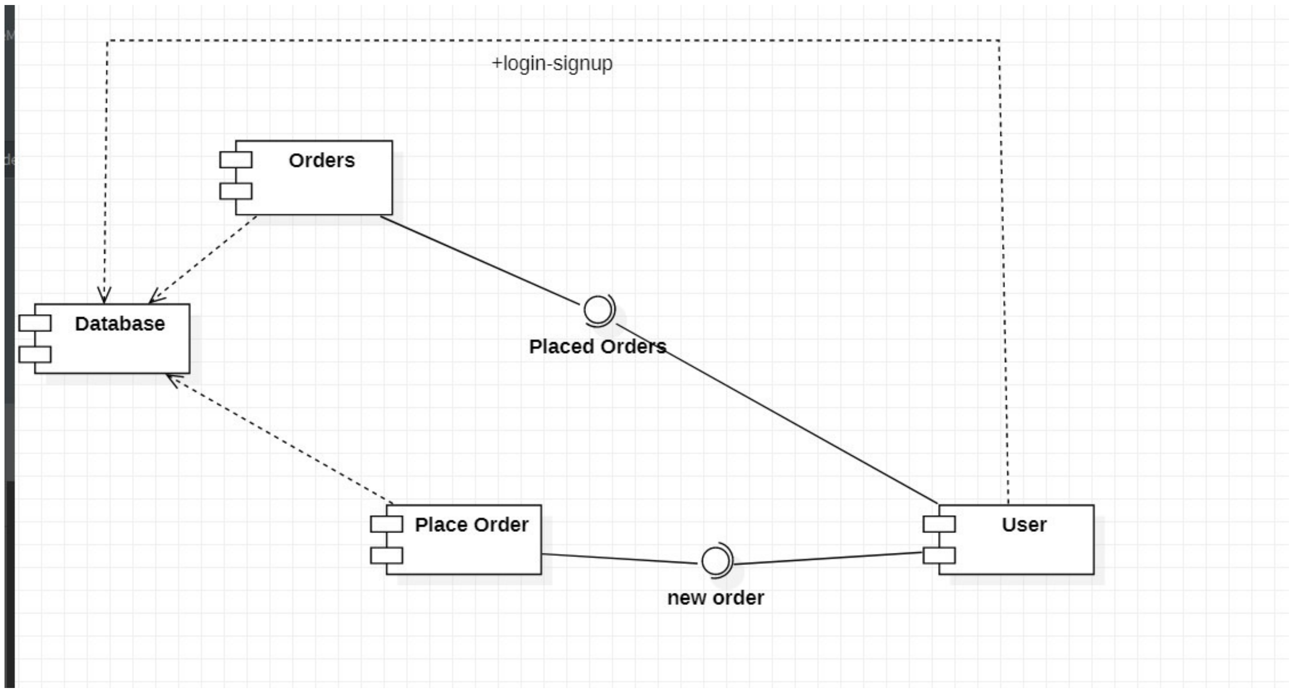


Fig 6.4.3.iii Component Diagram

6.5 DESIGN DESCRIPTION

6.5.1 Master Class Diagram

Our app is consisting of a virtual shopping cart where items are first classified using a camera and then added to the cart. The shopping cart has a list of items placed on it as the user is adding it to the app. It is composed of a load cell to get the weight of the item that the user is placing in the cart. For classification, the app makes use of a deep learning model which was pre-trained to classify the item. Once it is able to classify the image it links with the item class where we can get all details about the item. Once all the items are added and the user is done with adding the app uses the billing class to make the final payment and store the details of payment in our database.

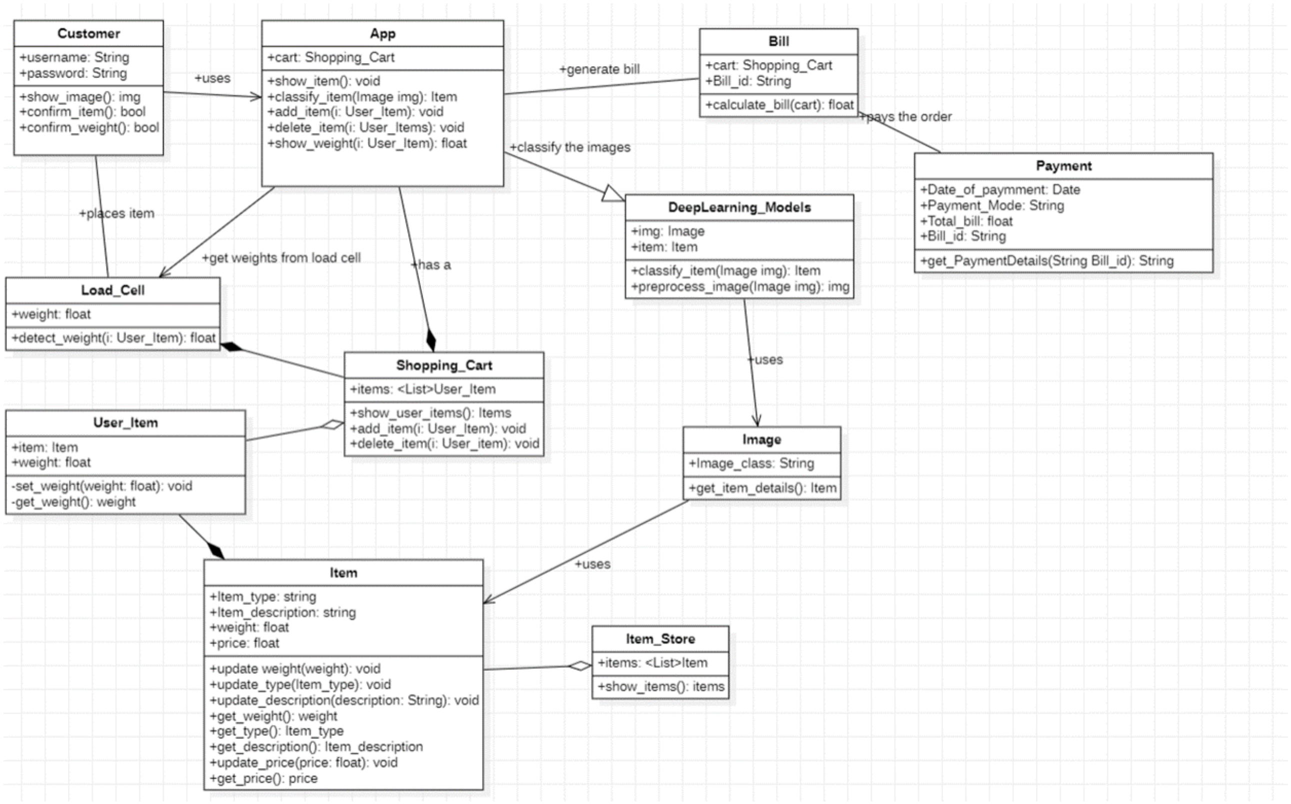


Fig 6.5.1 Master Class Diagram

6.5.2 Reusability Considerations

- For the android studio component, the same code will be used for each part of the front end, with a few minor changes; i.e. the backbone of the front end code remains the same for each functionality, with a few minor changes.

6.6 STATE DIAGRAM

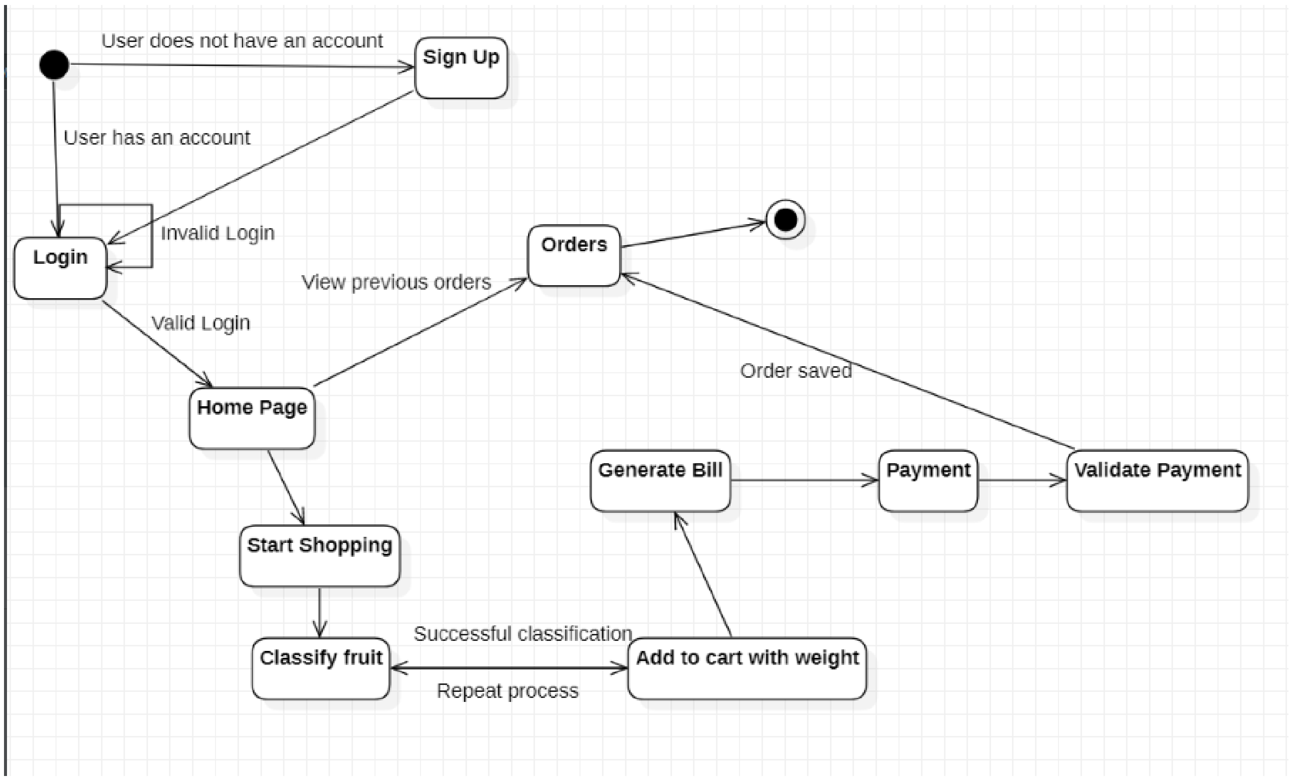


Fig 6.6 State Diagram

6.7 USE CASE DIAGRAM:

This diagram provides a detailed view of use cases and key actors involved in the application.

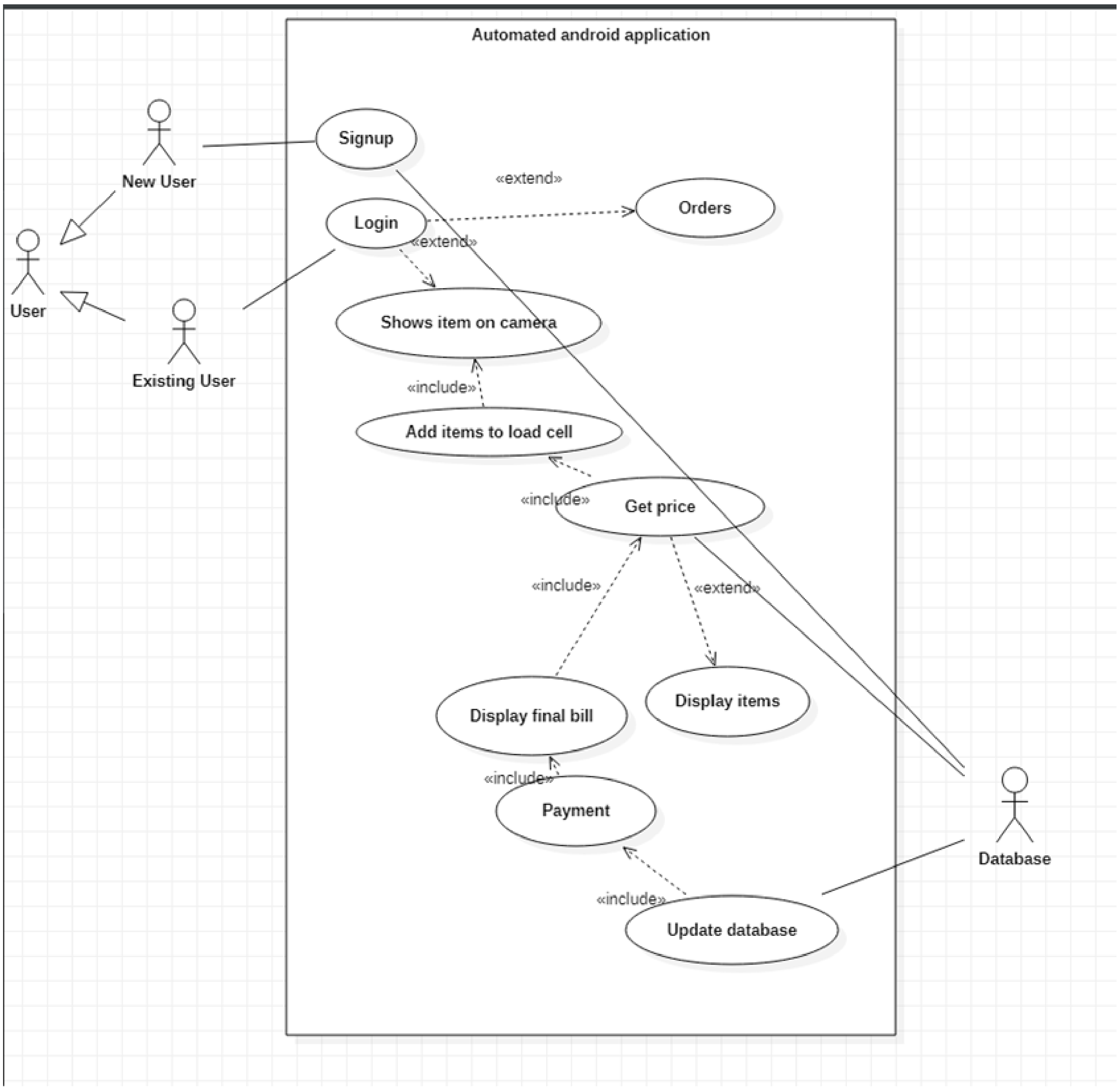


Fig 6.7 Use Case Diagram

6.8 EXTERNAL INTERFACES

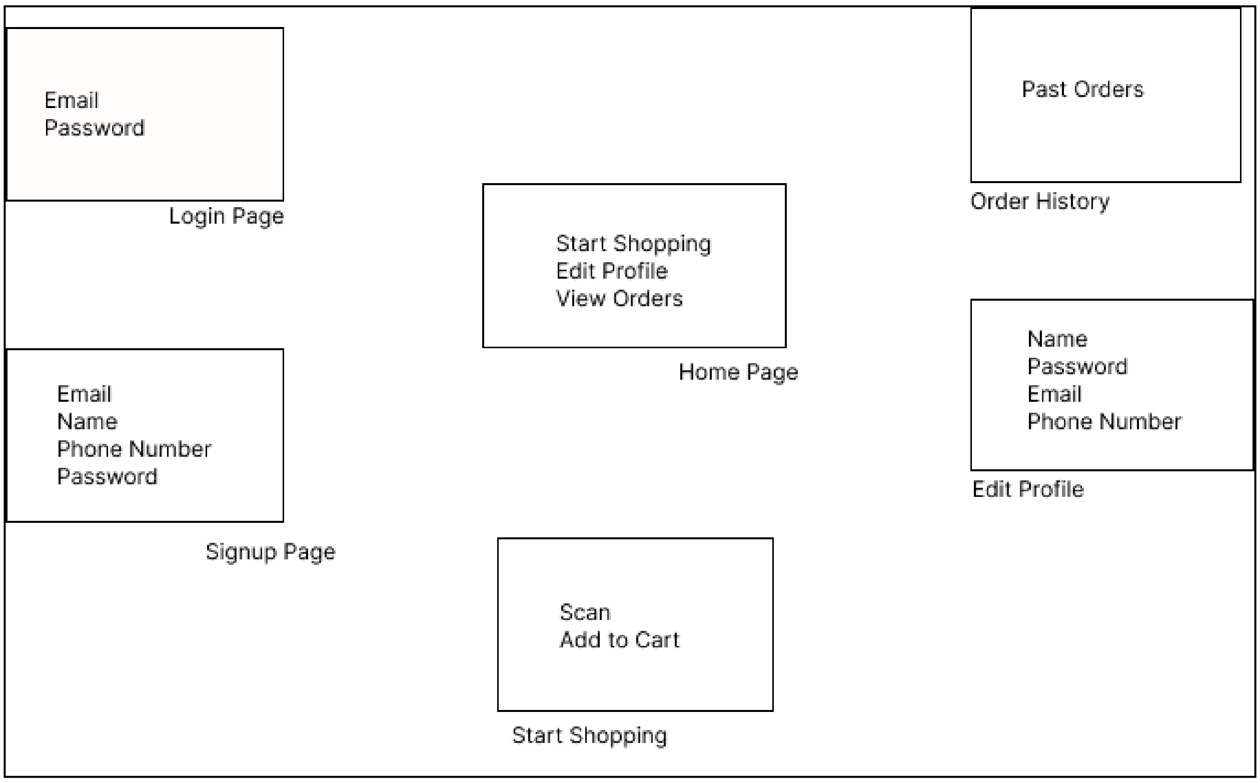


Fig 6.8 External Interfaces

6.9 ER Diagram

This gives us an idea of the architecture of database which we plan to use inorder to retrieve and store information of items and related to users.

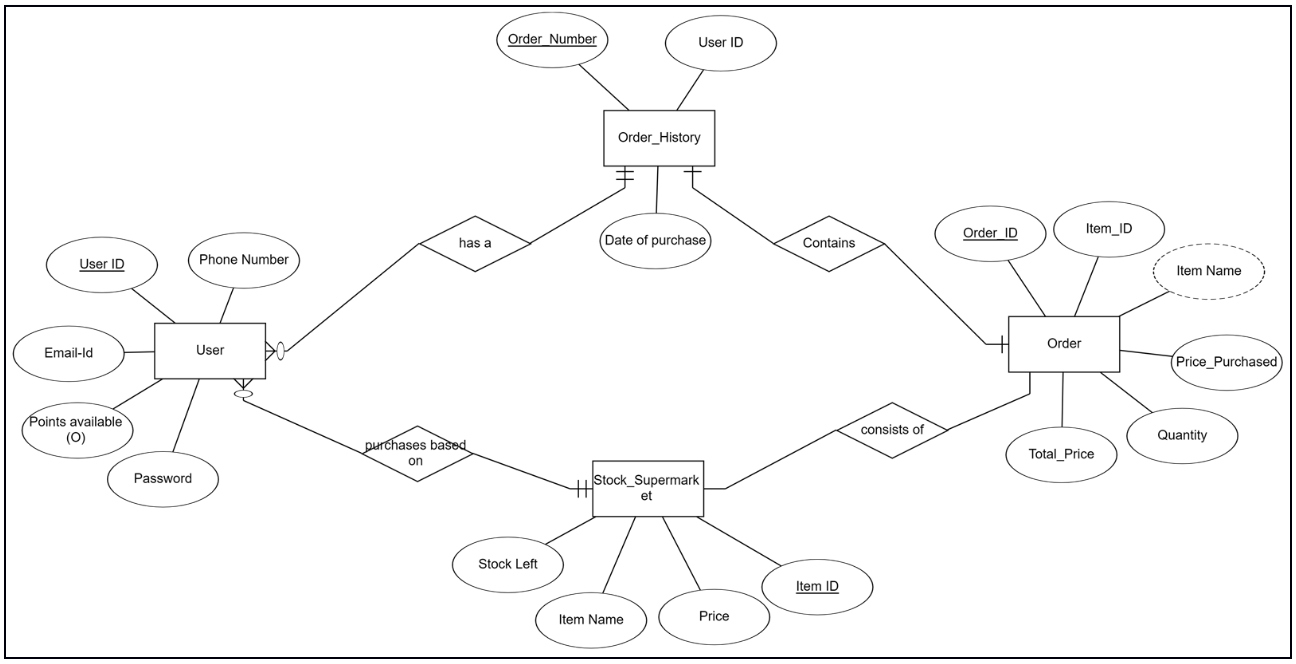


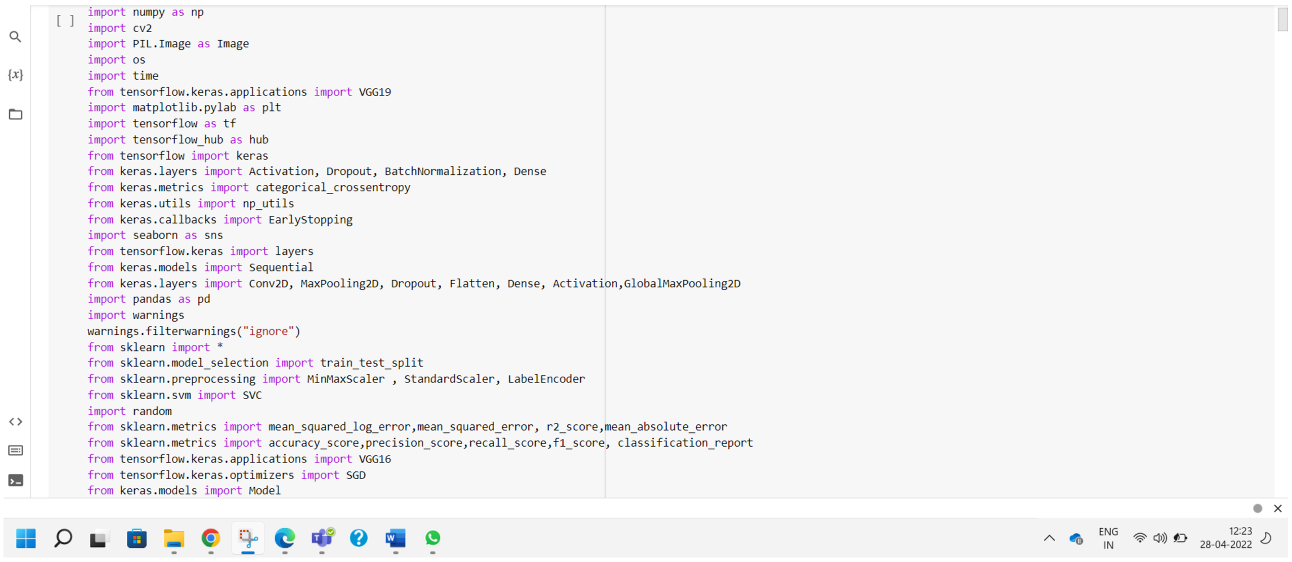
Fig 6.9 ER Diagram

CHAPTER 7

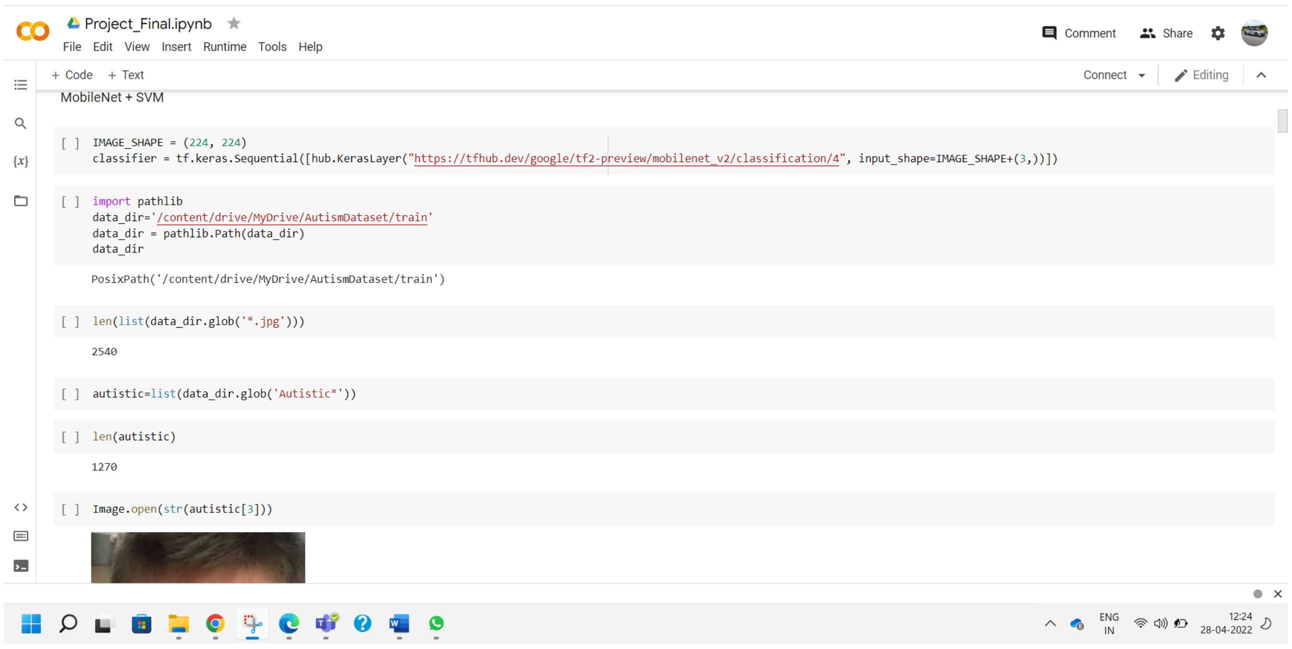
IMPLEMENTATION

To understand the working of the way transfer learning works with MobileNetV2 for classification we have developed its working only for 2 classes. The code of the same is seen below:

Fig 7: Implementation Screenshots

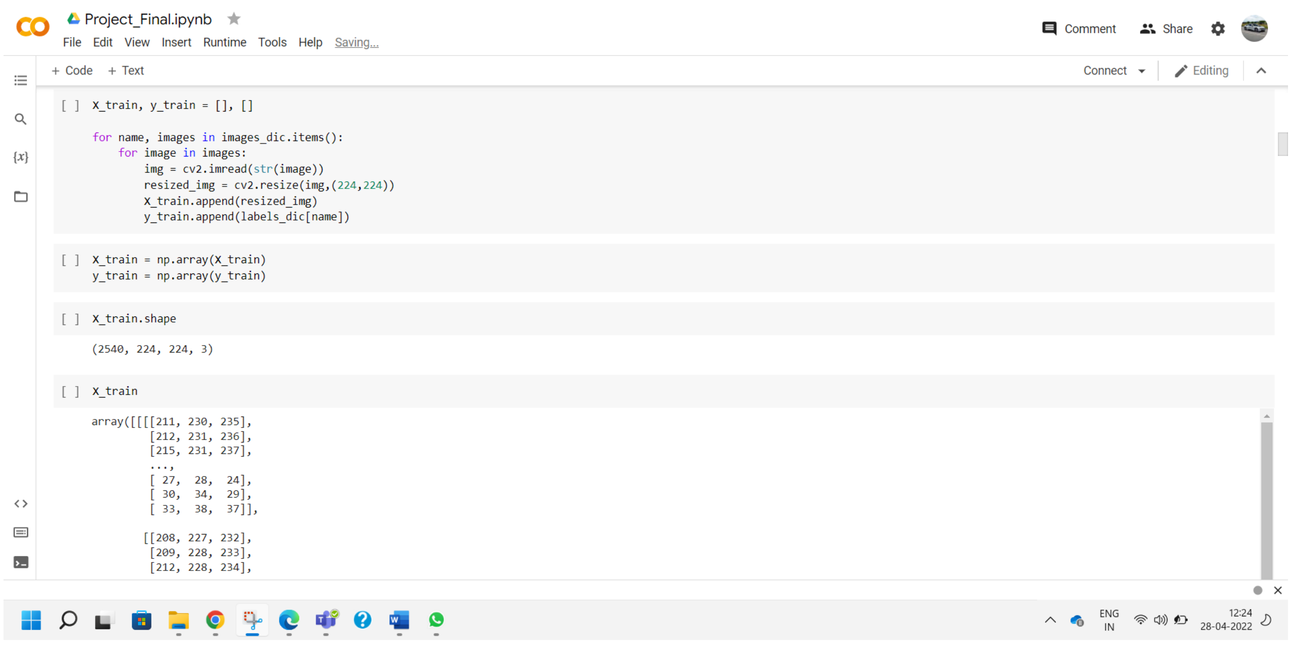


```
[ ] import numpy as np
    import cv2
    import PIL.Image as Image
    import os
    import time
    from tensorflow.keras.applications import VGG19
    import matplotlib.pylab as plt
    import tensorflow as tf
    import tensorflow_hub as hub
    from tensorflow import keras
    from keras.layers import Activation, Dropout, BatchNormalization, Dense
    from keras.metrics import categorical_crossentropy
    from keras.utils import np_utils
    from keras.callbacks import EarlyStopping
    import seaborn as sns
    from tensorflow.keras import layers
    from keras.models import Sequential
    from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense, Activation,GlobalMaxPooling2D
    import pandas as pd
    import warnings
    warnings.filterwarnings("ignore")
    from sklearn import *
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import MinMaxScaler , StandardScaler, LabelEncoder
    from sklearn.svm import SVC
    import random
    from sklearn.metrics import mean_squared_log_error,mean_squared_error, r2_score,mean_absolute_error
    from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score, classification_report
    from tensorflow.keras.applications import VGG16
    from tensorflow.keras.optimizers import SGD
    from keras.models import Model
```



MobileNet + SVM

```
[ ] IMAGE_SHAPE = (224, 224)
    classifier = tf.keras.Sequential([hub.KerasLayer("https://tfhub.dev/google/tf2-preview/mobilenet_v2/classification/4", input_shape=IMAGE_SHAPE+(3,))])

[ ] import pathlib
    data_dir='/content/drive/MyDrive/AutismDataset/train'
    data_dir = pathlib.Path(data_dir)
    data_dir

    PosixPath('/content/drive/MyDrive/AutismDataset/train')

[ ] len(list(data_dir.glob('*.jpg')))

    2540

[ ] autistic=list(data_dir.glob('Autistic*'))

[ ] len(autistic)

    1270

[ ] Image.open(str(autistic[3]))
```

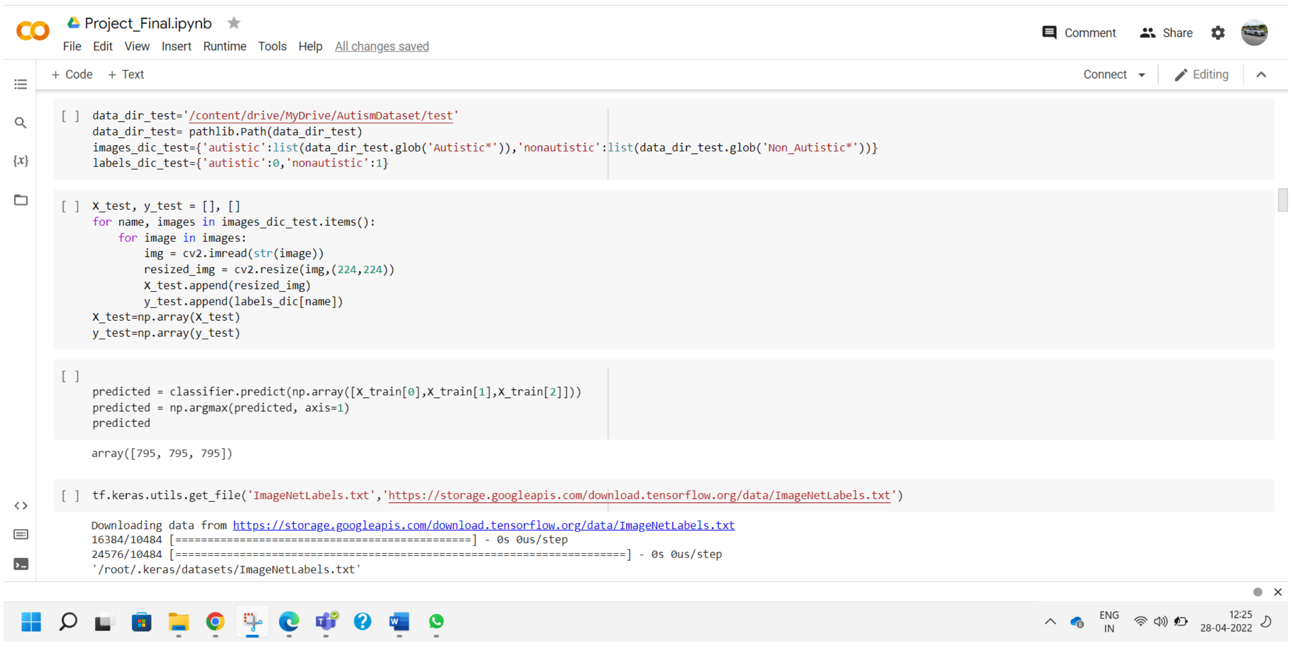


```
[ ] X_train, y_train = [], []

    for name, images in images_dic.items():
        for image in images:
            img = cv2.imread(str(image))
            resized_img = cv2.resize(img,(224,224))
            X_train.append(resized_img)
            y_train.append(labels_dic[name])
```

```
[ ] X_train = np.array(X_train)
    y_train = np.array(y_train)
```

```
[ ] X_train.shape

    (2540, 224, 224, 3)
```

```
[ ] X_train

    array([[[[211, 230, 235],
             [212, 231, 236],
             [215, 231, 237],
             ...,
             [ 27,  28,  24],
             [ 30,  34,  29],
             [ 33,  38,  37]],

            [[208, 227, 232],
             [209, 228, 233],
             [212, 228, 234],
```

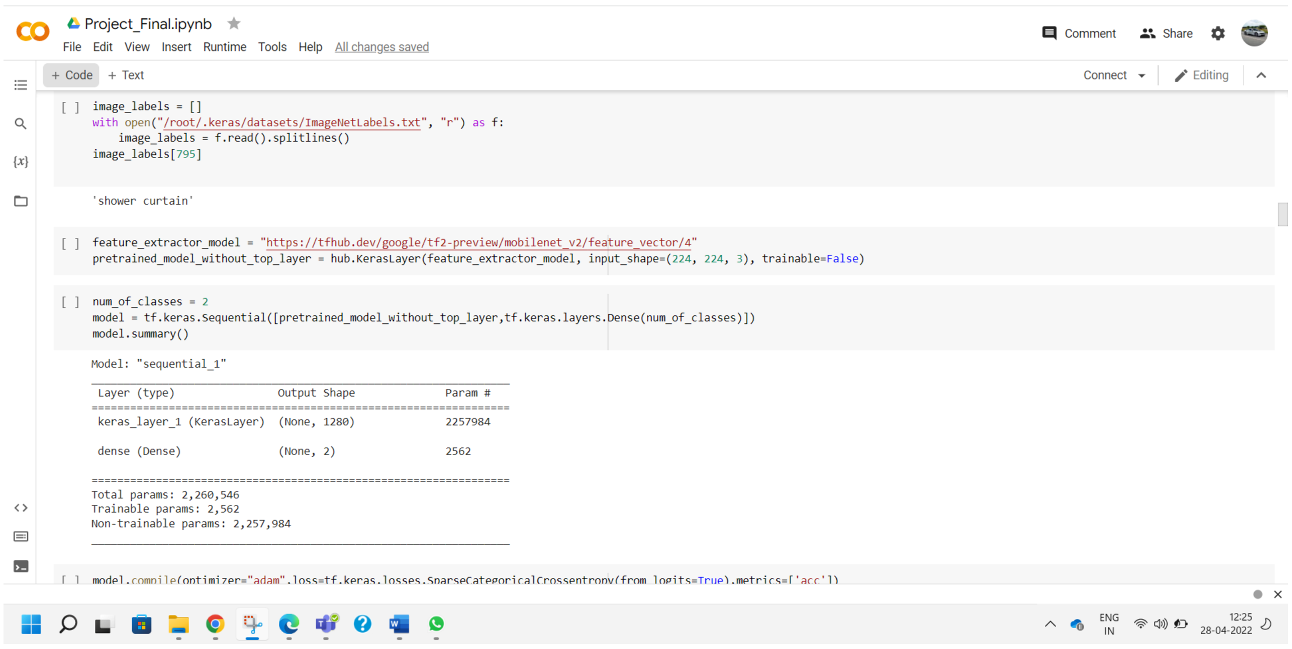


```
[ ] data_dir_test='/content/drive/MyDrive/AutismDataset/test'
    data_dir_test= pathlib.Path(data_dir_test)
    images_dic_test={'autistic':list(data_dir_test.glob('Autistic*')),'nonautistic':list(data_dir_test.glob('Non_Autistic*'))}
    labels_dic_test={'autistic':0,'nonautistic':1}
```

```
[ ] X_test, y_test = [], []
    for name, images in images_dic_test.items():
        for image in images:
            img = cv2.imread(str(image))
            resized_img = cv2.resize(img,(224,224))
            X_test.append(resized_img)
            y_test.append(labels_dic[name])
    X_test=np.array(X_test)
    y_test=np.array(y_test)
```

```
[ ]
    predicted = classifier.predict(np.array([X_train[0],X_train[1],X_train[2]]))
    predicted = np.argmax(predicted, axis=1)
    predicted

    array([795, 795, 795])
```

```
[ ] tf.keras.utils.get_file('ImageNetLabels.txt','https://storage.googleapis.com/download.tensorflow.org/data/ImageNetLabels.txt')

    Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/ImageNetLabels.txt
    16384/10484 [==============================================] - 0s 0us/step
    24576/10484 [==============================================================] - 0s 0us/step
    '/root/.keras/datasets/ImageNetLabels.txt'
```

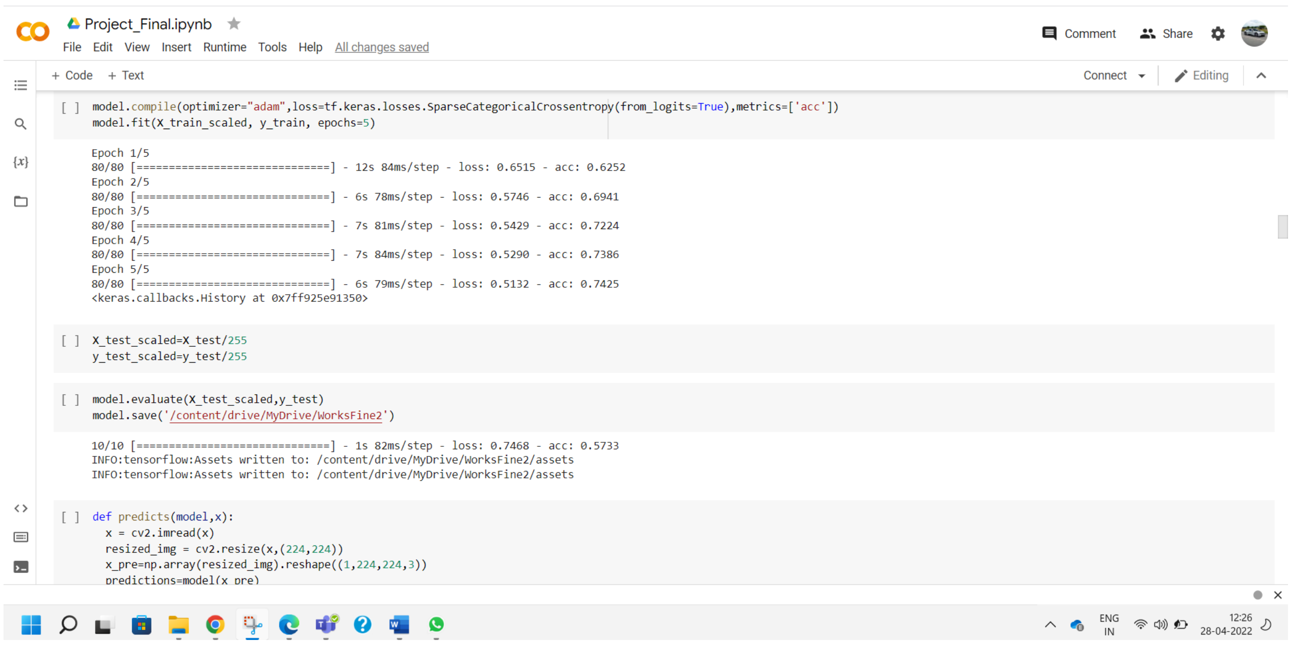


```
image_labels = []
with open("/root/.keras/datasets/ImageNetLabels.txt", "r") as f:
    image_labels = f.read().splitlines()
image_labels[795]

'shower curtain'
```

```
feature_extractor_model = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4"
pretrained_model_without_top_layer = hub.KerasLayer(feature_extractor_model, input_shape=(224, 224, 3), trainable=False)
```

```
num_of_classes = 2
model = tf.keras.Sequential([pretrained_model_without_top_layer,tf.keras.layers.Dense(num_of_classes)])
model.summary()

Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer_1 (KerasLayer)  (None, 1280)              2257984

 dense (Dense)               (None, 2)                 2562

=================================================================
Total params: 2,260,546
Trainable params: 2,562
Non-trainable params: 2,257,984
_________________________________________________________________
```

```
model.compile(optimizer="adam",loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),metrics=['acc'])
```

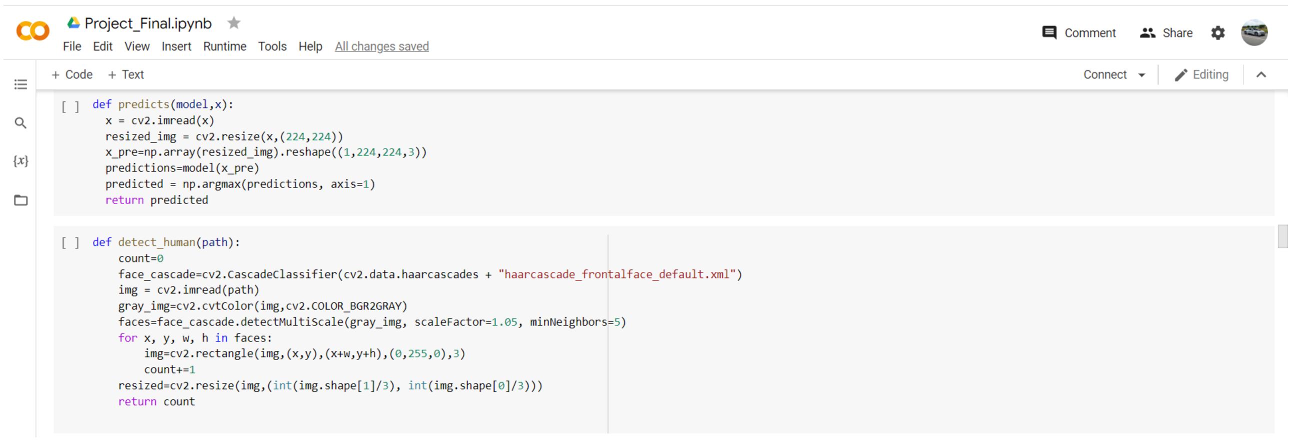


```
model.compile(optimizer="adam",loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),metrics=['acc'])
model.fit(X_train_scaled, y_train, epochs=5)

Epoch 1/5
80/80 [==============================] - 12s 84ms/step - loss: 0.6515 - acc: 0.6252
Epoch 2/5
80/80 [==============================] - 6s 78ms/step - loss: 0.5746 - acc: 0.6941
Epoch 3/5
80/80 [==============================] - 7s 81ms/step - loss: 0.5429 - acc: 0.7224
Epoch 4/5
80/80 [==============================] - 7s 84ms/step - loss: 0.5290 - acc: 0.7386
Epoch 5/5
80/80 [==============================] - 6s 79ms/step - loss: 0.5132 - acc: 0.7425
<keras.callbacks.History at 0x7ff925e91350>
```

```
X_test_scaled=X_test/255
y_test_scaled=y_test/255
```

```
model.evaluate(X_test_scaled,y_test)
model.save('/content/drive/MyDrive/WorksFine2')

10/10 [==============================] - 1s 82ms/step - loss: 0.7468 - acc: 0.5733
INFO:tensorflow:Assets written to: /content/drive/MyDrive/WorksFine2/assets
INFO:tensorflow:Assets written to: /content/drive/MyDrive/WorksFine2/assets
```

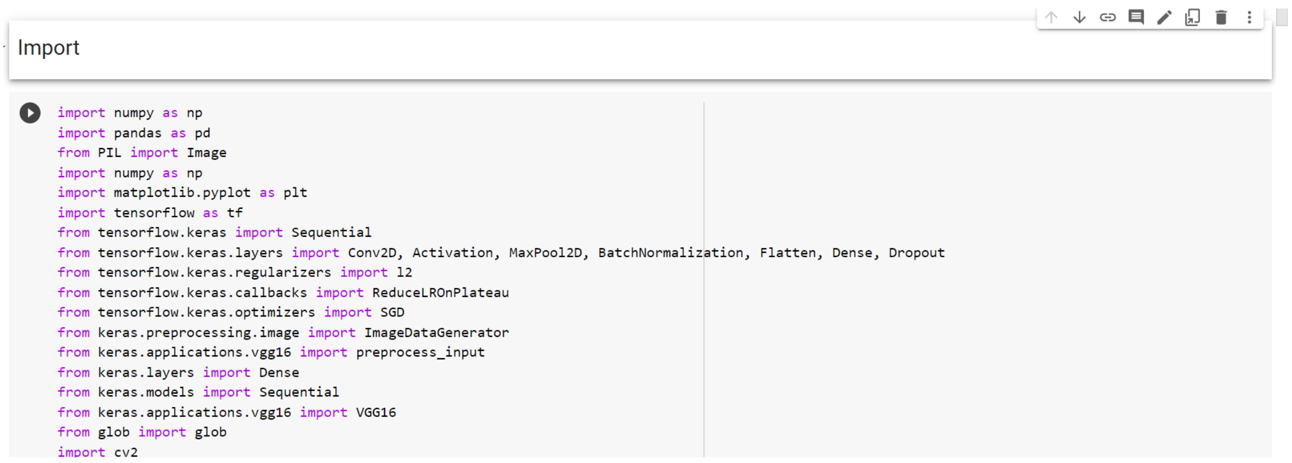
```
def predicts(model,x):
  x = cv2.imread(x)
  resized_img = cv2.resize(x,(224,224))
  x_pre=np.array(resized_img).reshape((1,224,224,3))
  predictions=model(x_pre)
```



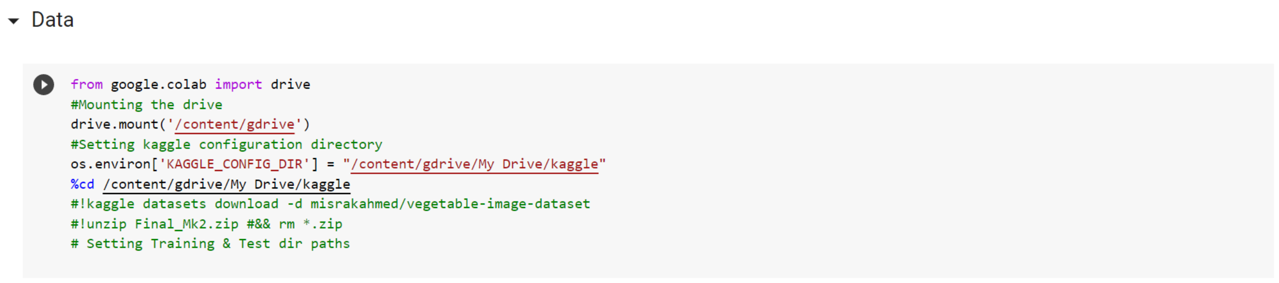
```
[ ] def predicts(model,x):
      x = cv2.imread(x)
      resized_img = cv2.resize(x,(224,224))
      x_pre=np.array(resized_img).reshape((1,224,224,3))
      predictions=model(x_pre)
      predicted = np.argmax(predictions, axis=1)
      return predicted
```

```
[ ] def detect_human(path):
        count=0
        face_cascade=cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
        img = cv2.imread(path)
        gray_img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
        faces=face_cascade.detectMultiScale(gray_img, scaleFactor=1.05, minNeighbors=5)
        for x, y, w, h in faces:
            img=cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),3)
            count+=1
        resized=cv2.resize(img,(int(img.shape[1]/3), int(img.shape[0]/3)))
        return count
```

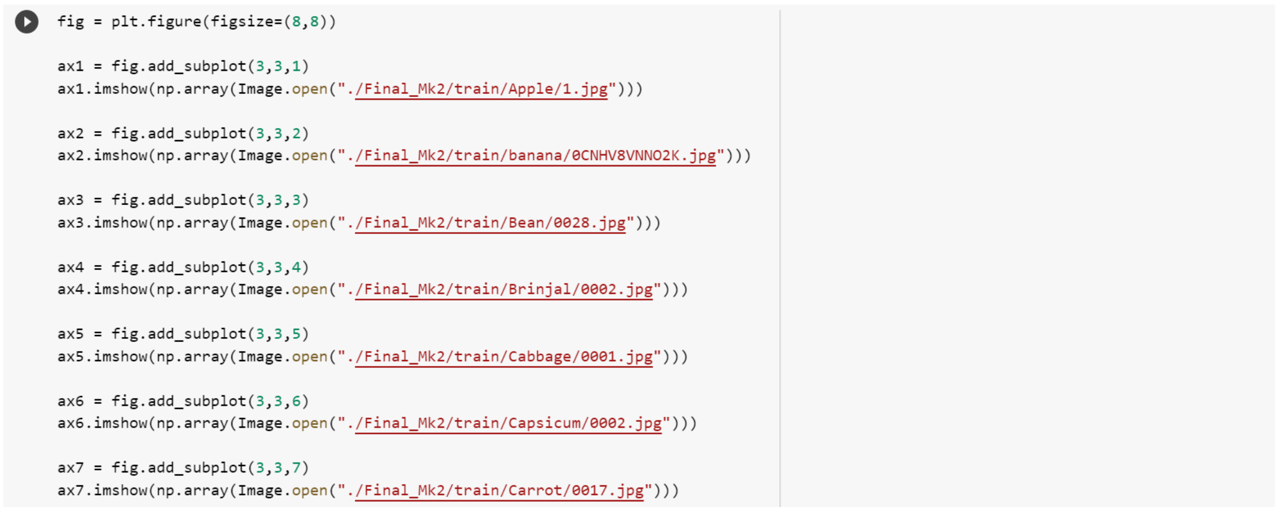
We have also added tried implementing the classification technique on vegetable classification by using VGG16 model the code of which can be seen below.



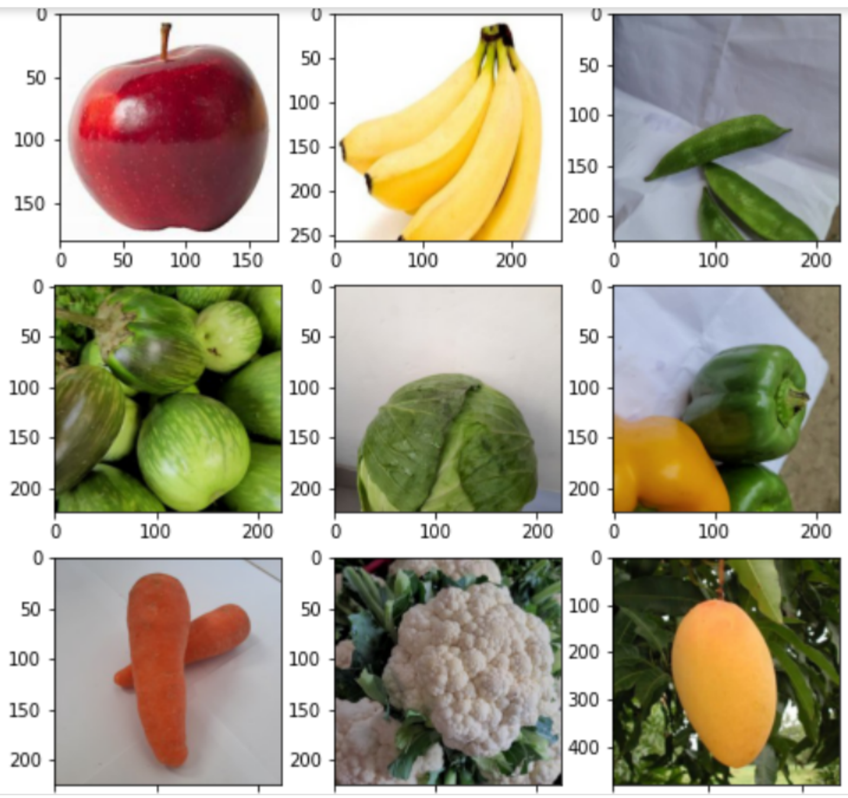
```
import numpy as np
import pandas as pd
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D, Activation, MaxPool2D, BatchNormalization, Flatten, Dense, Dropout
from tensorflow.keras.regularizers import l2
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.optimizers import SGD
from keras.preprocessing.image import ImageDataGenerator
from keras.applications.vgg16 import preprocess_input
from keras.layers import Dense
from keras.models import Sequential
from keras.applications.vgg16 import VGG16
from glob import glob
import cv2
```



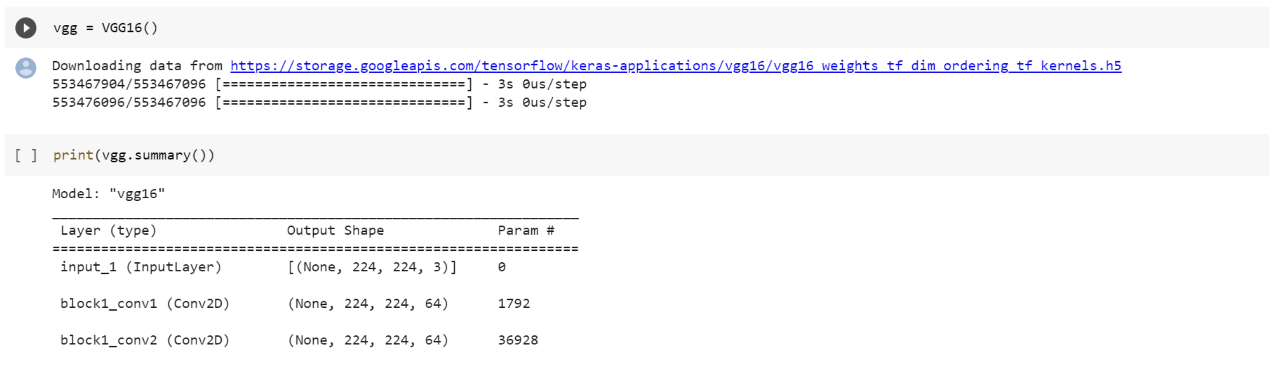
```
from google.colab import drive
#Mounting the drive
drive.mount('/content/gdrive')
#Setting kaggle configuration directory
os.environ['KAGGLE_CONFIG_DIR'] = "/content/gdrive/My Drive/kaggle"
%cd /content/gdrive/My Drive/kaggle
#!kaggle datasets download -d misrakahmed/vegetable-image-dataset
#!unzip Final_Mk2.zip #&& rm *.zip
# Setting Training & Test dir paths
```



```
fig = plt.figure(figsize=(8,8))

ax1 = fig.add_subplot(3,3,1)
ax1.imshow(np.array(Image.open("./Final_Mk2/train/Apple/1.jpg")))

ax2 = fig.add_subplot(3,3,2)
ax2.imshow(np.array(Image.open("./Final_Mk2/train/banana/0CNHV8VNNO2K.jpg")))

ax3 = fig.add_subplot(3,3,3)
ax3.imshow(np.array(Image.open("./Final_Mk2/train/Bean/0028.jpg")))

ax4 = fig.add_subplot(3,3,4)
ax4.imshow(np.array(Image.open("./Final_Mk2/train/Brinjal/0002.jpg")))

ax5 = fig.add_subplot(3,3,5)
ax5.imshow(np.array(Image.open("./Final_Mk2/train/Cabbage/0001.jpg")))

ax6 = fig.add_subplot(3,3,6)
ax6.imshow(np.array(Image.open("./Final_Mk2/train/Capsicum/0002.jpg")))

ax7 = fig.add_subplot(3,3,7)
ax7.imshow(np.array(Image.open("./Final_Mk2/train/Carrot/0017.jpg")))
```



▾ Implementation

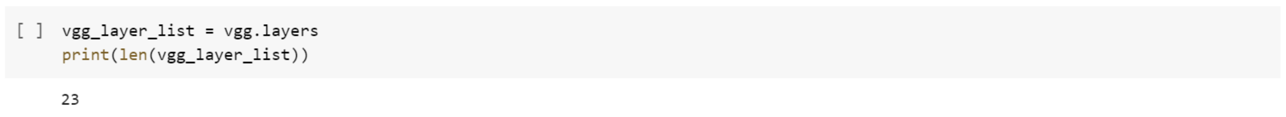


```
vgg = VGG16()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467904/553467096 [==============================] - 3s 0us/step
553476096/553467096 [==============================] - 3s 0us/step
```

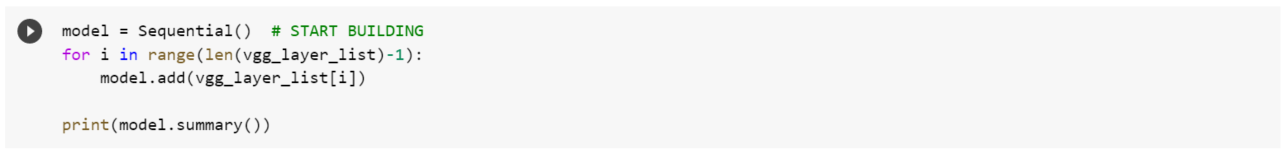
```
[ ] print(vgg.summary())
```

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928
```

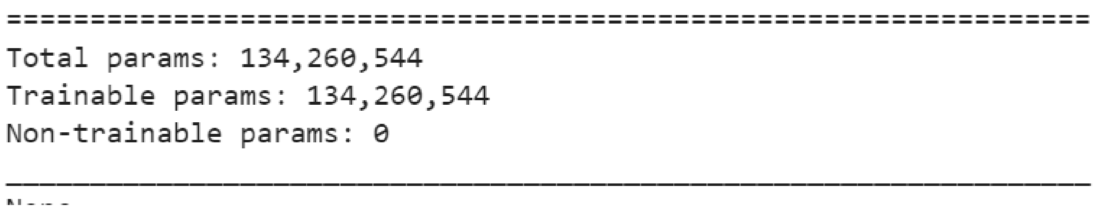


```
[ ] vgg_layer_list = vgg.layers
    print(len(vgg_layer_list))
```

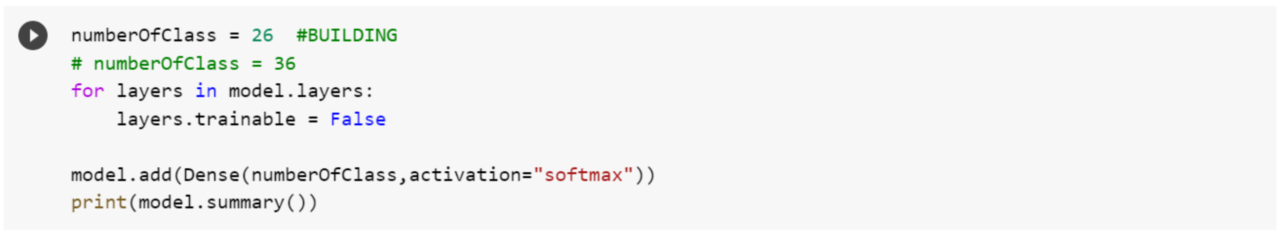
```
23
```



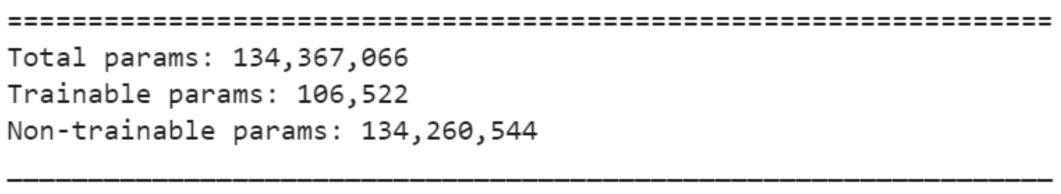
```
model = Sequential()  # START BUILDING
for i in range(len(vgg_layer_list)-1):
    model.add(vgg_layer_list[i])

print(model.summary())
```



```
=================================================================
Total params: 134,260,544
Trainable params: 134,260,544
Non-trainable params: 0
_________________________________________________________________
```



```
numberOfClass = 26  #BUILDING
# numberOfClass = 36
for layers in model.layers:
    layers.trainable = False

model.add(Dense(numberOfClass,activation="softmax"))
print(model.summary())
```



```
=================================================================
Total params: 134,367,066
Trainable params: 106,522
Non-trainable params: 134,260,544
_________________________________________________________________
```

```
model_layer_list = model.layers
print(len(model_layer_list))
```

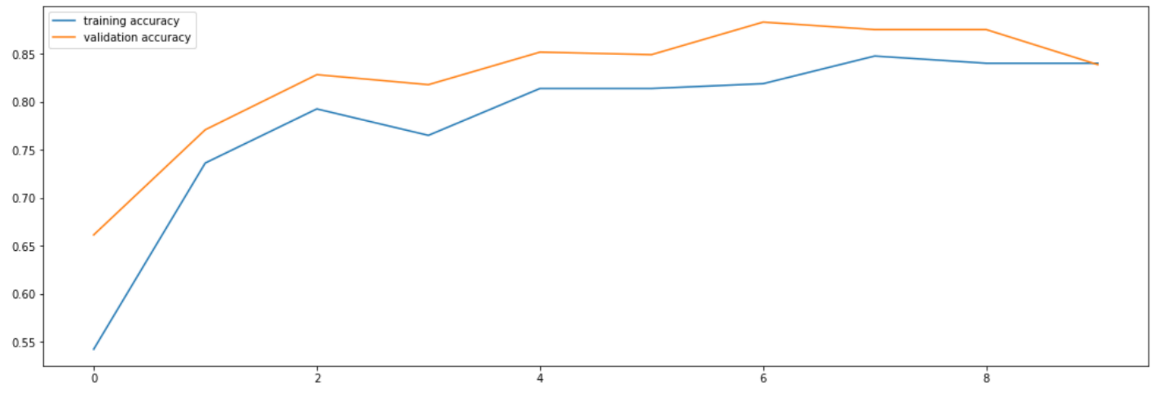
```
22
```

```
model.compile(loss="categorical_crossentropy", # FINAL BUILDING
              optimizer="rmsprop",
              metrics=["accuracy"])

train_data = ImageDataGenerator().flow_from_directory(training_dir,target_size=(224,224))

test_data = ImageDataGenerator().flow_from_directory(validation_dir,target_size=(224,224))
```

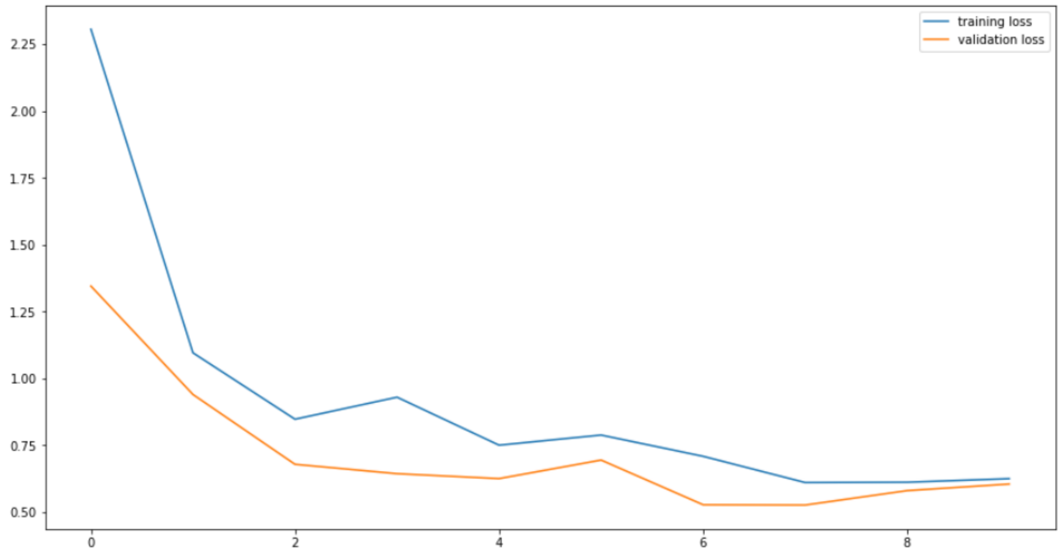
```
Found 22670 images belonging to 26 classes.
Found 4305 images belonging to 26 classes.
```

```
from tensorflow import keras
# model = keras.models.load_model("./model_saves/4/")
# keras.models.load_weights("fruits_model.h5")
batch_size=64
hist = model.fit_generator(train_data,
                           steps_per_epoch=1600//batch_size,
                           epochs=10,
                           validation_data=test_data,
                           validation_steps=800//batch_size)
```

```
plt.figure(figsize=(18,6))
plt.plot(hist.history["accuracy"],label="training accuracy")
plt.plot(hist.history["val_accuracy"],label="validation accuracy")
plt.legend()
plt.show();
```



```
plt.figure(figsize=(15,8))
plt.plot(hist.history["loss"],label="training loss")
plt.plot(hist.history["val_loss"],label="validation loss")
plt.legend()
plt.show();
```



```
print("Model accuracy is: ",hist.history["accuracy"][-1])
print("Model validation accuracy is: ",hist.history["val_accuracy"][-1])
```

```
Model accuracy is:  0.8399999737739563
Model validation accuracy is:  0.8385416865348816
```

Prediction

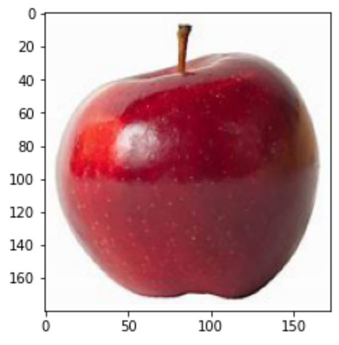
```
[ ] idx_to_name = {x:i for (x,i) in enumerate(train_data.class_indices)}

    def predict(img):
        to_predict = np.zeros(shape=train_data[0][0].shape)
        to_predict[0] = img

        return idx_to_name[np.argmax(model(to_predict)[0])]
```

```
%cd content/gdrive/MyDrive
%ls
```

```
/content/gdrive/MyDrive/kaggle/content/gdrive/MyDrive
model1/
```

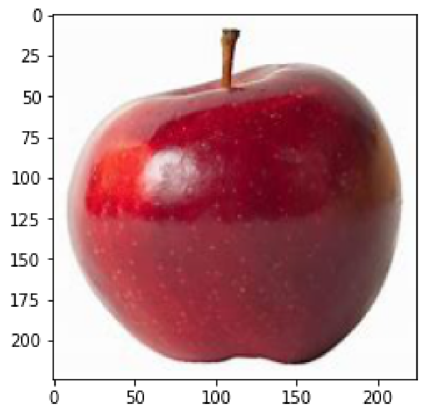
```
[ ] image = plt.imread("./Final_Mk2/train/Apple/1.jpg")
    plt.imshow(image)
```

```
<matplotlib.image.AxesImage at 0x7fc98e9e2350>
```



```
resized = cv2.resize(image, (224,224), interpolation = cv2.INTER_AREA)
plt.imshow(resized)
```

```
<matplotlib.image.AxesImage at 0x7fc98e435310>
```



```
[ ] predict(resized)
```

```
'Apple'
```

CHAPTER 8

CONCLUSION OF CAPSTONE PROJECT PHASE- 1

The Conclusion of the Capstone Project Phase – 1 are the following:

- Designing and understanding the Problem Statement.
- Understanding the Project Deliverables.
- System Requirement Specifications Completed
- Product Requirement Specifications Completed
- Literature Survey based on past Research papers and getting an idea of the way in which we can start off with our project.
- Scraping images from different website to build a dataset for training the model.
- Implemented code based on our model for classification task to understand its working.
- Constructed High Level Design for our project.
- We put together a proper design for the implementation of our solution.

CHAPTER 9

PLAN OF WORK FOR CAPSTONE PROJECT PHASE-2

The implementation of our project in the form of an Android application with necessary hardware components as required will be carried out during phase 2 in accordance with the following Gantt chart -

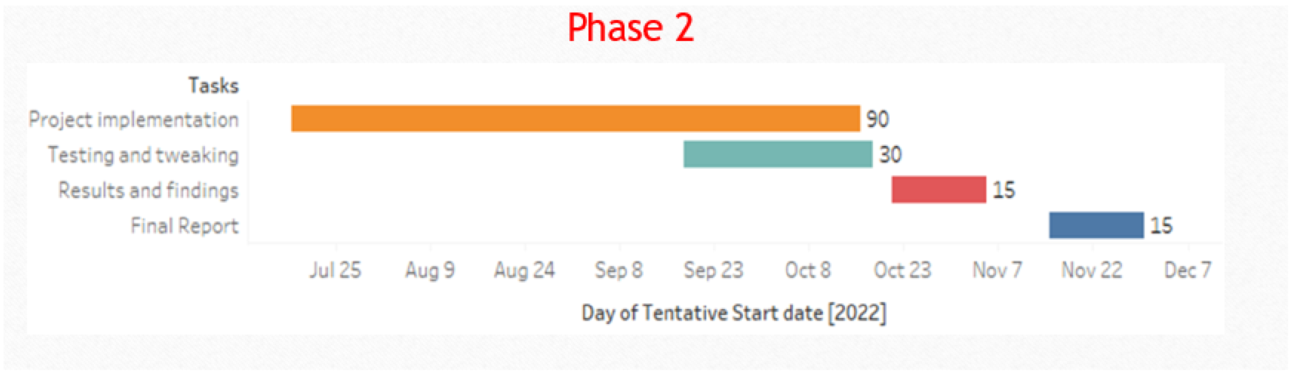


Fig 9: Capstone Phase 2 Timeline

We plan to have the core functionalities of the application developed and tested by October. We will then proceed to integrate it with the AWS cloud environment and design the UI elements to have a functioning alpha build by the end of November.

REFERENCES/BIBLIOGRAPHY

1) https://ieeexplore.ieee.org/abstract/document/9089651
2) https://www.irjet.net/archives/V8/i4/IRJET-V8I4678.pdf
3) https://arxiv.org/pdf/1901.00711.pdf
4) https://link.springer.com/chapter/10.1007/978-3-030-49076-8_1
5) https://www.javatpoint.com/uml-class-diagram
6) https://www.uml-diagrams.org/deployment-diagrams-overview.html
7) https://medium.com/@mobindustry/how-to-integrate-machine-learning-into-an-android-app-best-image-recognition-tools-1ba837c27903
8) https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/blob/master/Raspberry_Pi_Guide.md
9) https://www.semanticscholar.org/paper/Implementation-of-Smart-Shopping-Cart-using-Object-Oh-Chun/f5ff8985a06e02715dbbbbf4f2b4470289e003f0
10) https://analyticsindiamag.com/deploying-machine-learning-models-in-android-apps-using-python/
11) https://eloquentarduino.github.io/2020/01/image-recognition-with-esp32-and-arduino/
12) https://firebase.google.com/docs/ml#:~:text=With%20Firebase%20ML%20and%20AutoML,to%20recognize%20concepts%20in%20photographs.&text=These%20APIs%20leverage%20the%20power,the%20highest%20level%20of%20accuracy.
13) https://forum.arduino.cc/t/password-protect-your-project/494551
14) https://github.com/antonnifo/fruits-360/blob/master/Fruit%20Classifer.ipynb
15) https://images.cv/category
16) https://link.springer.com/article/10.1007/s12652-020-01865-8

APPENDIX

APPENDIX A: Definitions, Acronyms, and Abbreviations

Definitions:

1) Transfer Learning: Transfer learning is a machine learning-based research problem that focuses on storing knowledge gained while solving one problem and applying it to another but related problem.
2) Load Cell: It is a force sensor that converts the force or weight of items placed in the cart are converted and sent as electrical signals.
3) Arduino: A microcontroller board that can be used to control the communication among all the hardware components having multiple input and output pins
4) MobileNetV2: A pertained CNN which can be used for image classification trained on a very large dataset for better classification accuracy.

Abbreviations:

1) DL: Deep Learning
2) ML: Machine Learning
3) VAE: Variational Auto Encoder
4) CNN: Convolutional Neural Network
5) GPU: Graphical Processing Unit
6) AWS: Amazon Web Services
7) UI: User Interface