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

UE19CS390B – Capstone Project Phase - 2

Submitted by:

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

Under the guidance of

Dr. Prajwala T R
Assistant Professor
PES University

June - Nov 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**
SVSC Santosh	**PES2UG19CS346**
Skanda S	**PES2UG19CS391**
Vijay Murugan A S	**PES2UG19CS454**

In partial fulfilment for the completion of seventh semester Capstone Project Phase - 2 (UE19CS390B) in the Program of Study -Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period June 2022 – Nov. 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 7th semester academic requirements in respect of project work.

Dr. Prajwala T R Assistant Professor	**Dr. Sandesh B J** Chairperson	**Dr. B K Keshavan** Dean of Faculty

External Viva

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

DECLARATION

We hereby declare that the Capstone Project Phase - 2 entitled **"Automated Shopping Cart"** has been carried out by us under the guidance of Dr. Prajwala T.R, Assistant Professor, and submitted in partial fulfilment of the course requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester June – Nov. 2022. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

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

ACKNOWLEDGEMENT

I would like to express my gratitude to Prof. Dr. Prajwala T.R, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE19CS390B -Capstone Project Phase – 2.

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

I take this opportunity to thank Dr. Sandesh B J, Chairperson, 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 me with various opportunities and enlightenment every step of the way. Finally, this 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.

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 centres 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

LIST OF TABLES

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 is to avoid the long queues that a customer has to wait for payment for the items they have added to the shopping cart. While the current market offers shopping carts that make use of RFID tags, we want to avoid such tedious tasks by using image classification to classify the item and integrate the payment section in an android app, which helps customers from waiting in a long queue enabling hassle-free shopping.

The primary features of our application include

- An **efficient deep learning model** which can classify the items with very minimal error and faster classification.
    - For this project we have made use of two deep learning models **MobileNet-v2** and **EfficentNet-v2.**
- An **Android app** that has 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.
- 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.
- An **Arduino board** to send the weights detected by the load cell with the android app.
- A **Bluetooth module HC-05** to help transfer data to our android app and get the weights of items added to the shopping cart.
- Finally, a cloud backend - using **firebase**, will serve as a database to store the details about the user information, cost, quantity, and description of the products available in the supermarket and the order history of each use

CHAPTER 2

PROBLEM DEFINITION

Shopping malls have become a necessary component of urban life. These marts are frequented by many commuters for groceries and everyday necessities. When shopping, standing in long lines is a common problem that is both time-consuming and tedious.

This project seeks to create a system that uses Deep Learning to systematically identify and classify the products added to the shopping cart. This system will update the cart whenever a new item is scanned and placed inside. When the customer is done shopping, the bill is already generated, and the customer can pay immediately without needing to stand in a long queue.

Objectives of this project would include:

- Connect to a shopping cart via Bluetooth swiftly.
- Scan and classify products quickly and efficiently.
- Perform cost lookup for the product in the database.
- Measure the weight of the product placed in the shopping cart accurately.
- Add the item to the cart on the app with the calculated amount. Generate a final bill as soon as the customer is done shopping.

CHAPTER 3

LITERATURE SURVEY

3.1 Cross-Validation Voting for Improving CNN Classification in Grocery Products.

3.1.1 Summary:

A CVV (Cross-Validation-Voting) technique for grocery product classification is provided in this work. This method enhances a number of independent state-of-the-art classifiers without integrating them, and it avoids overfitting concerns with respect to the training data. The training dataset is split into distinct training and validation slots to train the ensemble model using all of the data. Each model uses a part of the general training dataset for training and a part of the dataset for validation. This technique was used to evaluate the improvements in three models, namely ResNeXt-101, EfficientNet B7, and Wide ResNet-101. When the ensemble model was combined with soft voting, the models showed the greatest improvement. The probabilities of each class are accumulated across the several models in soft voting, and the model with the highest probability wins.

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

3.1.3 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.

3.2 Fruit Classification for Retail Stores Using Deep Learning.

3.2.1 Summary: This paper talks about developing a simple lightweight CNN-based model for classifying the 3 types of 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.

3.2.2 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.

3.3.3 Limitation:

- The project had only 3 classes of fruits, although the number of classes in real life exceeds 100.
- Using the RGB histogram, it was discovered that backdrop color plays a significant influence in determining item class. Hence the background color must be consistent.

3.3 A Hierarchical Grocery Store Image Dataset with Visual and Semantic Labels.

3.3.1 Summary:

This research intends to assist people with vision impairments in grocery shopping by giving assistance. This project made use of both generative and deterministic deep neural networks along with a simple linear classification model like the 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.

3.3.2 Advantages:

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

3.3.3 Limitations:

- Because of the vast number of parameters involved, high GPU and CPU capacity, as well as good hardware, are required to execute rapid and accurate classification.
- The time needed for developing the model by training on images is quite a lot.

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

3.4.1 Summary:

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. They used darknet with CUDA, dark flow, and OpenCV after the model was pre-trained to make the detection happen in real-time. The pre-trained model's initial weights are saved 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 makes 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.

3.4.2 Advantages:

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

3.4.3 Limitations:

- Generation of protobuf files is quite challenging.
- Detection of objects in images displayed on a screen or monitor.

3.5 Android Application for Grocery Ordering System.

3.5.1 Summary:

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.

3.5.2 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.

3.5.3 Limitation:

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

CHAPTER 4

PROJECT REQUIREMENT SPECIFICATION

4.1 PROJECT SCOPE

Develop a fully working application which 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.

4.2 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 the customers in queues and make the whole shopping experience better.

4.2.1 PRODUCT FEATURES

- Faster and correct classification of items by our deep learning models implemented.
- Update of the weight of item added to cart through load cell to our app via bluetooth module HC-05 through arduino.
- User friendly app to perform hassle free shopping..
- A cloud based database like firebase to store various details of items, users and previous orders details of a user.

4.2.2 User Classes and Characteristics

- User
    - Show items to our app to detect it.
    - Press remove item to remove the item from cart.
    - Else Press 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
- Admin
    - Make updates to the item based on stock available in mart
    - Add users to database based on the user details sent to app

4.2.3 Operating Environment

- Hardware Platform - Mobile device, Arduino, Bluetooth HC-05 module, Load Cell
- Operating system - Android
- Software Components - Inputs include user real-time image capturing and weight from load cell. Outputs include updation of cart items and bill generation.

4.2.4 General Constraints, Assumptions and Dependencies

4.2.4.1 Hardware constraints:

- Ability of device to run the model,
- Inaccuracy of load cell,
- Training requires powerful GPUs.
- Misuse of hardware present in the shopping cart.

4.2.4.2 Software constraints:

- 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.

4.2.4.3 Assumptions:

- Objects are placed in a transparent bag
- Final Payment is handled by some 3rd party interface.
- Customer has an android phone with minimum computational resources to use the app.
- Items being shown to the camera by the user are either fruits or vegetables and nothing else.
- Items being removed by the user are the same as the one given as input to remove in the app by the user and the user removes the entire item weight.
- Dependencies: Keras, TensorFlow, OpenCV, DarkFlow, Firebase,

4.2.5 Risks

- Model Using up too many computational resources causing the app to crash
- Failure of Hardware
- Disconnection with Bluetooth

4.3 Functional Requirements

- Ability of the system to classify items shown by the user accurately.
- Fetch item details and add items to the bill from firebase.
- The app is robust and is always connected to Arduino when in use.

4.4 External Interface Requirements

4.4.1 User Interfaces

- The UI is an android application.
- Functionality to scan and classify objects.
- Functionality to the weight of the objects placed in the cart.
- Screen to view all the items in the cart.
- Screen to view the final computed bill and payment feature.

4.4.2 Hardware Requirements

- Android platform for the app
- Precise load-cell
- Reliable microcontroller(Arduino)
- Bluetooth module hc-05
- Shopping cart

4.4.3 Software Requirements

- Android Studio, Tensorflow , Keras, Firebase

4.4.4 Communication Interfaces

- Bluetooth/USB connection to Arduino.
- Wired communication between Arduino and the load cell.

4.5 Non-Functional Requirements

4.5.1 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 Firebase, 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 3 seconds to ensure that the user doesn't have to wait for a very long time and 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 and the weights placed in the cart.

4.5.2 Safety Requirements

- Take the Bluetooth name of the shopping cart from the user when connecting the microcontroller to his device
- Ensuring Bluetooth is always connected once items start getting placed in the cart till the final bill generation.

4.5.3 Security Requirements

- A login authorization system using an email id and password.
- Ensure the database can be modified by the authorized user/admin.
- Misuse of hardware components in the shopping cart.

4.6 Feasibility Study

Upon referring to a few research papers in the field, YOLO(You Only Look Once), VGG19, RESNET, and ALEXNET, DenseNET are some available options, however, they require a lot of computing power, and optimizing them will be time-consuming because of the various parameters they come with. Reflecting on the changes made to the cart on the app would be a tedious task. Our model doesn't take into consideration those items already packed having a barcode on them.

4.7 Other Requirements

- Need a battery to power up the Arduino and weight cell.
- Need the use of the internet for logging in and fetching the details of items and orders.

CHAPTER 5

SYSTEM DESIGN (DETAILED)

5.1 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 shopping experience 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.

5.2 Current System

Existing systems use RFID tags and barcodes to bill items which are 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 fruit and vegetables being a waste of time and effort.

5.3 Design Constraints

5.3.1 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.

5.4 High-Level System Design

5.4.1 Data Flow Diagram

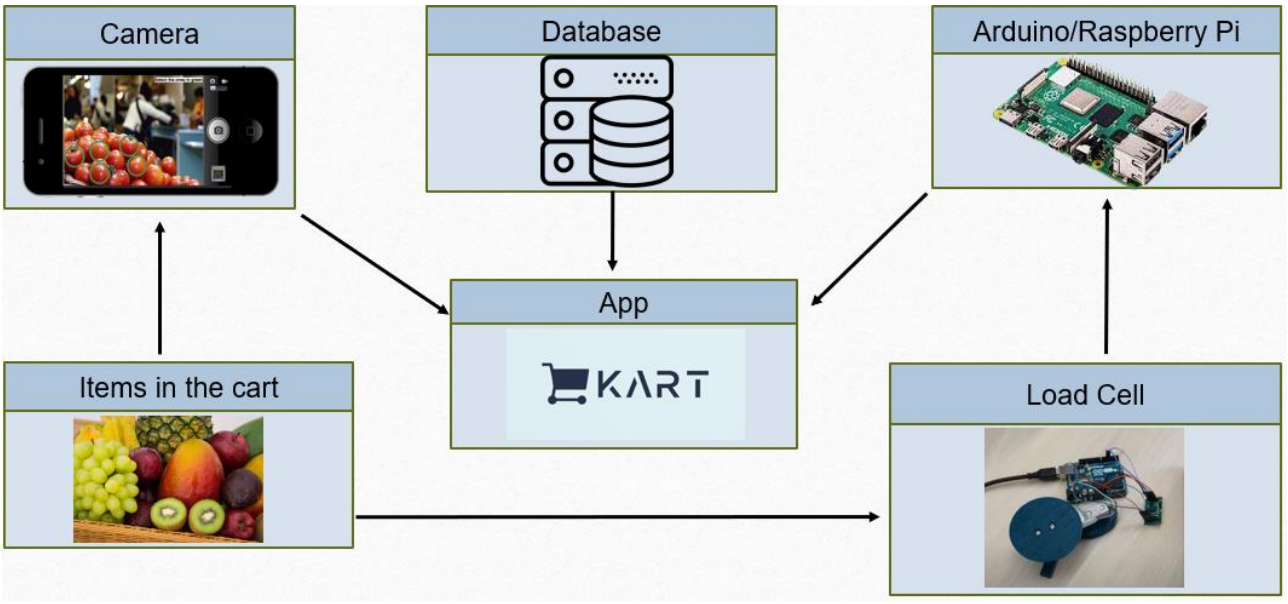

Fig 5.4.1 Data Flow Diagram

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

5.4.2 System Architecture Diagram

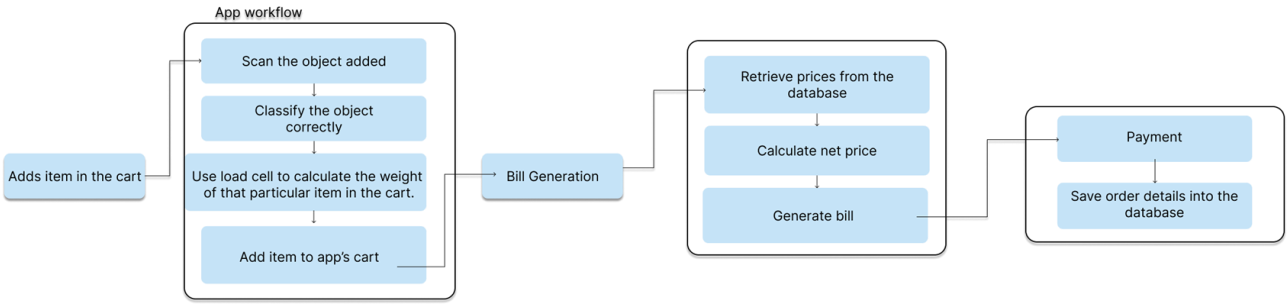

Fig 5.4.2: System Architecture Design

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.

5.5 Low-Level System Design

5.5.1 Overview

The low-level design document provides an in-depth view of each of the sub-component present in the high-level design. We followed a bottom-up design where we tried to solve the subproblems found in our existing system. Once we solve each subproblem we then try to integrate it to make our final workable system. The sub-problems that we found are:

- Setting up of weight sensor to find out weights placed on it
- Development of an android app with various pages and adding dependencies between each other.
- Image classification of classes based on deep learning models.
- Integration of the android app with the image classification using tflite module.
- Integration of android app with weight sensor using android.

5.5.2 Design Description

5.5.2.1 Master Class Diagram

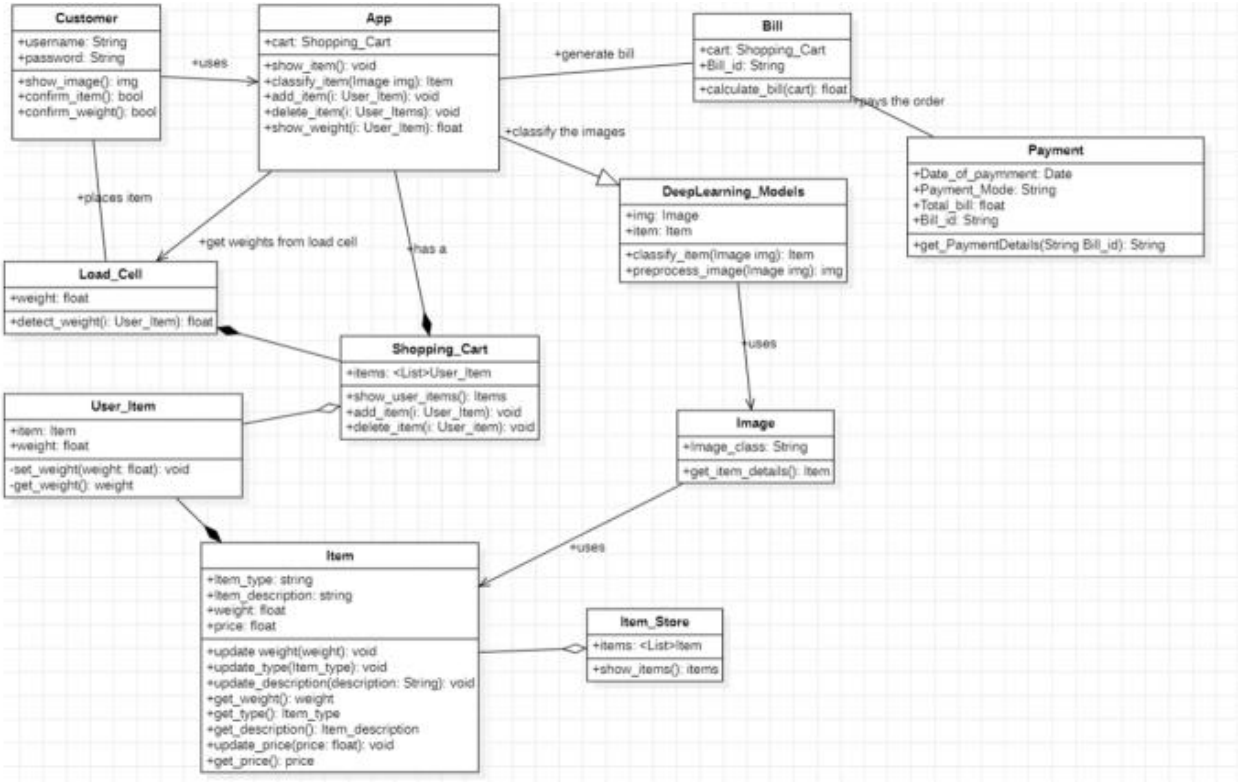

Fig 5.5.2.1: Master Class Diagram describing the functionalities of the app

Our app consists 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 them 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 can 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.

5.5.2.2 Use Case Diagram

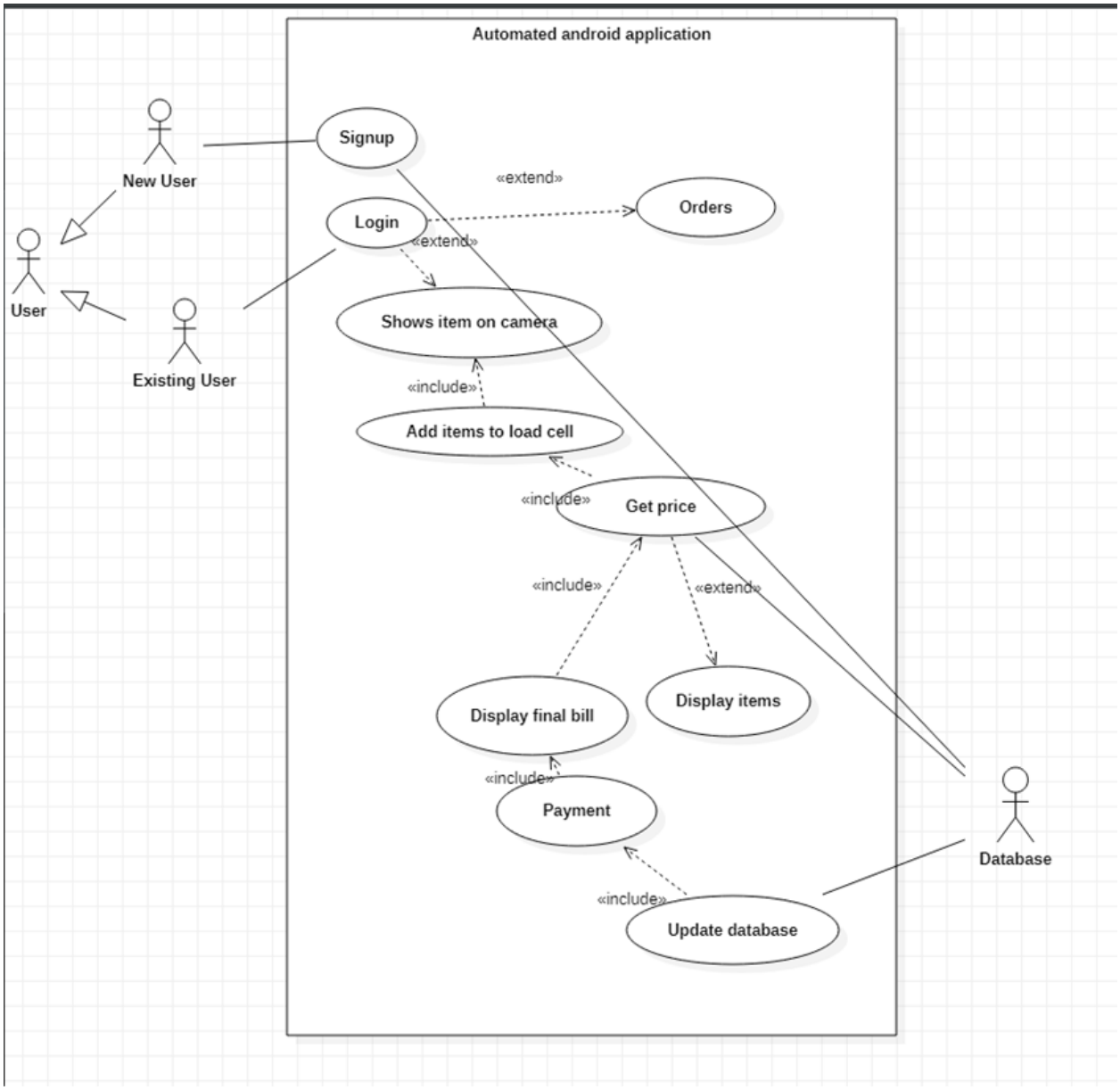

Fig 5.5.2.2: Use Case Diagram of the app

Table 5.2.2.2: Use Case items and their descriptions

Use Case Item	Description
SignUp	Sign up a new user to our app storing the details in our database.
Login	Logs an existing user to our app.
Orders	Logged-in users can track their past orders.
Shows item on camera	The user shows the item on camera to help classify which class that item belongs to.
Add items to the load cell	The quantity of the item added is weighed and updated on the app.
Get Price	Fetches the price of the item from the database and records on the app.
Display final bill	The final bill is computed and shown on the app.
Display items	Displays all the items in the cart along with the quantity.
Payment	Integrate with third-party applications to process payment with the sellers
Update Database	Take care of all operations regarding the users, items, and all grocery-related operations.

5.5.3 Module 1- Building Dataset

5.5.3.1 Description

This module deals with the building of the dataset and classifying them. A lot of automation has gone into scraping the images from the web and collecting various classes of images from several data sources. The scraped images were later augmented to have a good number of images. We ended up having over 3000 training images and 51 classes resulting in over 153,000 images and built the final dataset.

5.5.3.2 Class Diagram

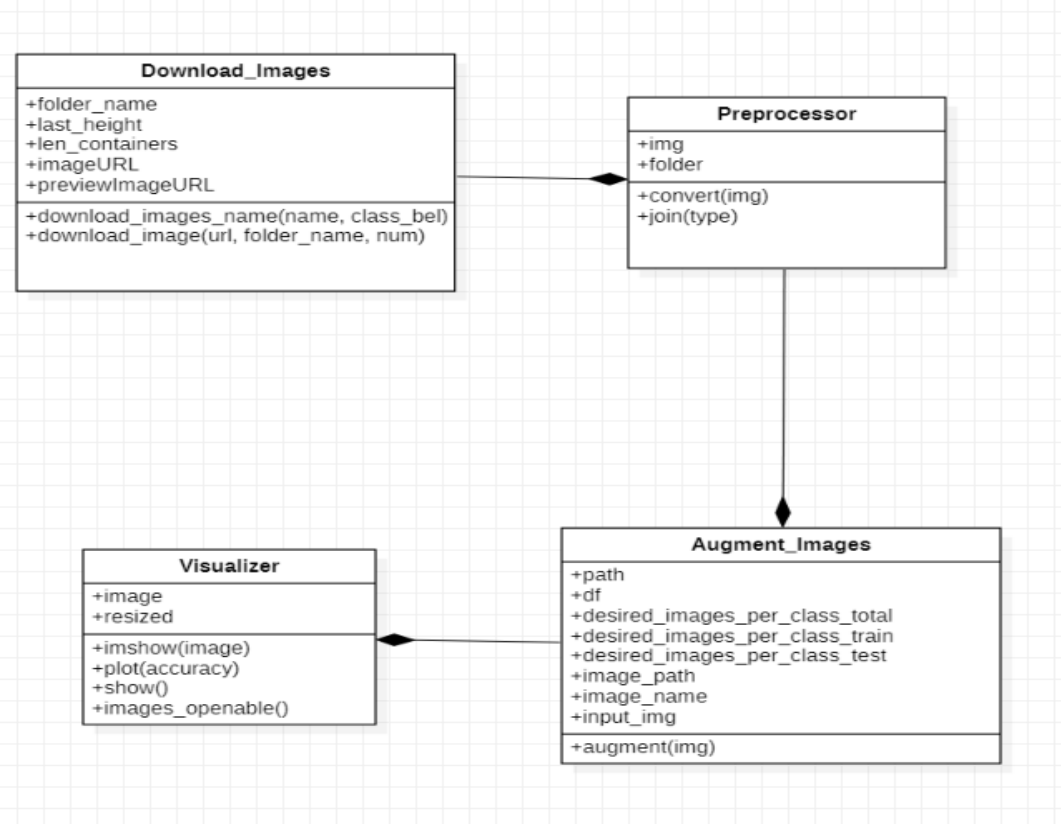

Fig 5.5.3.2: Class Diagram for Building Dataset

5.5.3.3 Class Name 1- Download_Images:

Deals with automation by downloading images from the web using web scraping with the help of automation tools like selenium and beautifulSoup. The dataset is also structured here as per the needs.

Table 5.5.3.3: Data Members and their descriptions

Data Type	Data Name	Access Modifiers	Description
String	folder_name	public	Contains path of Zip file of all images.
Integer	last_height	public	Contains path of Zip file of test images.
Integer	len_containers	public	Counter to find total number of containers.
String	imageURL	public	Stores the URL of the image to be scrapped.
String	previewImageURL	public	Stores the preview URL so that we get high-quality images.

i. Method 1- download_images_name

- Purpose: Method to download images as per their corresponding names from a yaml file which has the list of classes we aim to build our dataset with
- Input: Name of the class
- Output: Saved the class of images
- Parameters: images
- Exceptions: None
- Pseudo-code:

Use selenium and chrome web driver to start the automation process

Create folders for each class

Automate google search and scroll until the end of the page to load the contents.

Download images and write to corresponding files.

ii. Method 2- download_images

- Purpose: Fetch responses and get requests. Write the fetched responses converting them to image format in their corresponding folders.
- Input: URL, folder name, number of images
- Output: Saves the images in folders
- Parameters: Image URLs
- Exceptions: None
- Pseudo-code:

  Make GET requests and fetch their responses.

  Write the responses converting them to jpeg files in particular folders.

5.5.3.4 Class 2 - Preprocessor

Deals with the pre-processing of images where resizing are done to RGB format. Later if needed the downloaded images are converted to jpeg to maintain the same format throughout.

Table 5.5.3.4: Data Members and their descriptions

Data Type	Data Name	Access Modifiers	Description
String	folder	public	Contains path of the folder with all images
String	img	public	Contains path of image for pre-processing

i. Method - convert

- Purpose: To convert all images to jpeg format and resize all images to RGB size maintaining all images in the same size and format uniformly.
- Input: Image path
- Output: Pre-processed image
- Parameters: image
- Exception: None
- Pseudo-code:
    - Read the image from the path
    - Convert type to jpeg if not in jpeg
    - Resize image to RGB format
    - Write back to original location

5.5.3.5 Class 3 - Augment_Images

Method to augment images and expand the dataset. Used various augmentation techniques to build the dataset. Augmented images of each and every class having 3000 training images and 500 testing images resulting in a total of 153,000 images across all classes.

Table 5.5.3.5: Data Members and their descriptions

Data Type	Data Name	Access Modifiers	Description
String	path	public	Contains path of the folder with all images
String	image_path	public	Contains path of image for after pre-processing
String	image_name	public	Used to name the augmented image

String	input_img	public	Contains the path of the input image to be augmented
Integer	desired_images_per_class_total	public	Contains the number of images required to be augmented per class
Integer	desired_images_per_class_train	public	Contains the number of training images required to be augmented per class
Integer	desired_images_per_class_test	public	Contains the number of test images required to be augmented per class

i. Method 1 - augment:

- Purpose: To augment images, name the augmented images and store it in the corresponding folder.
- Input: Path of the image to be augmented.
- Output: Augmented image stored.
- Parameters: image
- Exceptions: None
- Pseudo-code:
    - Open the image to be augmented.
    - Run various techniques to augment images.
    - Store the augmented image.

5.5.3.6 Class 4 - Visualizer

Method to visualize the data and various components of the dataset with several visualization techniques.

Table 5.5.3.6: Data Members and their descriptions

Data Type	Data Name	Access Modifiers	Description
String	image	public	Contains path of image before pre-processing
String	resized	public	Contains path of image for after pre-processing

i. Method 1 - imshow

- Purpose: To display the particular image
- Input: Image path
- Output: Image displayed
- Parameters: Image
- Exceptions: None
- Pseudo-code: The image path is given to the show function which displays the image in that path.

ii. Method 2- plot

- Purpose: To plot the line graph accuracy of models helping us to find the visual difference in the accuracies of images
- Input: Accuracy of the models
- Output: Graph with the accuracy plotted
- Parameters: Accuracy
- Exceptions: None

- Pseudo-code: Accuracies are given and plotted on the line graph.

iii. Method 3 - images_openable

- Purpose: Method to find out the number of images that can be opened and displayed
- Input: the file path.
- Output: Number obtained after counting the number of openable images.
- Parameters: folder path
- Exceptions: None
- Pseudo-code: Counter counts the number of images that can be opened through looping until end of the file.

iv. Method 4 - show

- Purpose: Enables the image to be displayed
- Input: Image loaded to the plt.imshow method.
- Output: Image displayed.
- Parameters: image
- Exceptions: Only openable images can be displayed.
- Pseudo-code:
    - Image is first loaded to the imshow method.
    - The loaded image is then displayed.

5.5.4 Module 2- App Development

5.5.4.1 Description

This Module talks about how various pages of an app (each page having a class with attributes and methods) are linked to form a workable system. The details about the class diagram and various components of this module are explained below.

5.5.4.2 Class Diagram

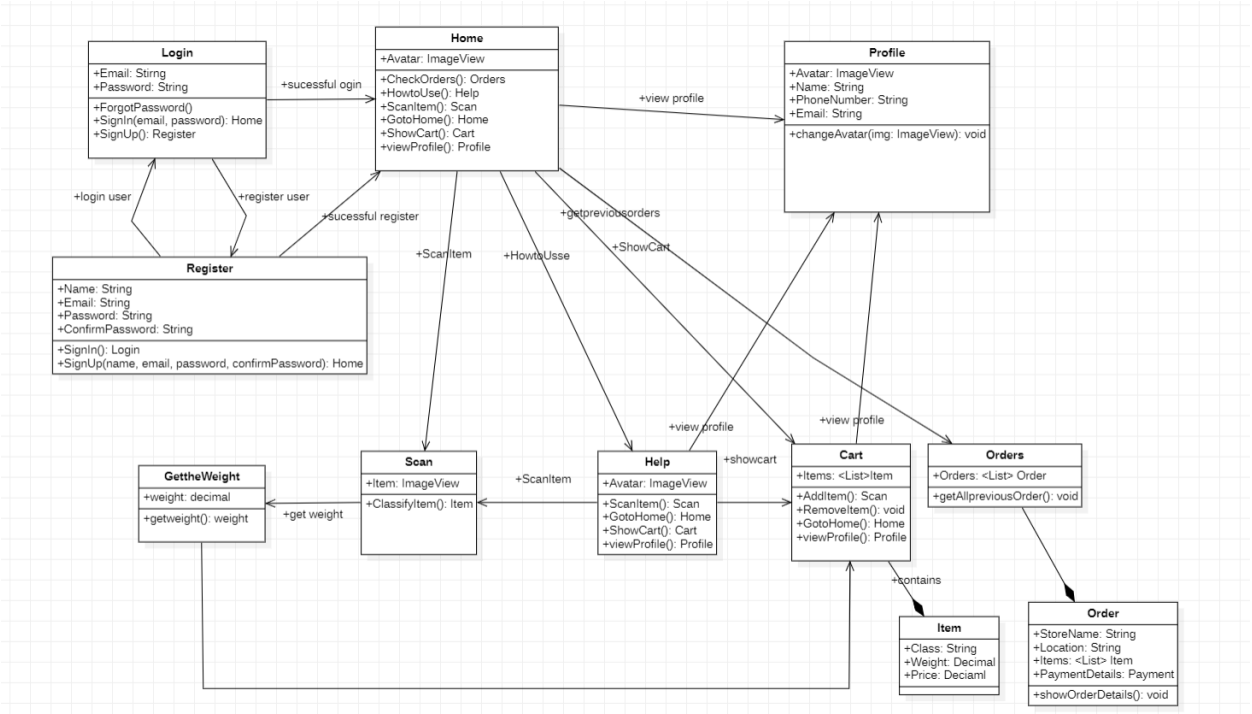

Fig 5.5.4.2: Class Diagram for App Development.

5.5.4.3 Class name 1-SignIn

i. Class Description 1

Performs a simple login based on a valid email id and password present in the database.

Table 5.5.4.3: Data members for SignIn class.

Data Type	Data Name	Access Modifiers	Initial Value	Description
String	Email	Private	“	Email input is taken from the user.
String	Password	Private	“	Password is taken by the user.

ii. Method 1 - ForgotPassword()

- Purpose: It is used to reset the password in case the user has forgotten the password for his account.
- Input: Email registered by the user.
- Output: Creates a user with an email id and new password.
- Parameters: Email
- Exceptions: None
- Pseudocode:

  Add a new password after sending an email to the user.

iii. Method 2 - SignIn(email,password)

- Purpose: It logins the user in with the email and password provided by the user.
- Input: Email and password of the registered user.

- Output: Opens the homepage of the app with user data.
- Parameters: Email and password.
- Exceptions: None
- Pseudocode:

```
If the user is present in the database and the password match:
    Move to the home page.
Else:
    Display an error message that the email or password is incorrect.
```

iv. Method 3- SignUp()

- Purpose: It moves the user to the signup page if the user selects the signup button.
- Input: On press of the Register button.
- Output: Opens the register of the app.
- Parameters: None.
- Exceptions: None
- Pseudocode:

```
If OnClick of sign up button:
    Move to register page.
```

5.5.4.4 Class name 2: Register

i. Class Description 2: Registers the user to use the app,

Table 5.5.4.4: Data members for the Register Class

Data Type	Data Name	Access Modifiers	Initial Value	Description
String	Email	Private	“	Email input taken from user.
String	Password	Private	“	Password taken from user.
String	name	Private	“	username given by the user.
String	ConfirmPassword	Private	“	Confirm password given by the user.

ii. Method 1 - SignIn():

- Purpose: It moves the user to the login page.
- Input: Click on signin button.
- Output: Move to the login page of the app.
- Parameters: None.
- Exceptions: None
- Pseudocode:

If Onclick of the sign-in button:

Move the user to the login page of the app.

iii. Method 2 - SignUp(email,password)

- Purpose: It registers the user with the email and password provided by the user.
- Input: Email and password entered by the user.
- Output: Successful registration opens the homepage of the app with user data.
- Parameters: Email and password.
- Exceptions: None
- Pseudocode:

If the email is valid and password and confirm password matches:

Register the user Move to the home page

Else:

Display an error message that the email or password is invalid.

5.5.4.5 Class name 3: Home

i. Class Description 3

It contains different components needed to form the home page of our app

Table 5.5.4.5: Data members for Home class.

Data Type	Data Name	Access Modifiers	Initial Value	Description
ImageView	avatar	Private	“	Avatar of the user.

ii. Method 1 - checkOrders()

- Purpose: It sends the user to the order page if the check previous orders button is pressed.
- Input: Click on the check previous orders button.

- Output: Opens the orders page of the app.
- Parameters: Previous order Button.
- Exceptions: None
- Pseudocode:

```
If OnClick of previous order button:
    Move to the orders page.
```

iii. Method 2 - HowtoUse()

- Purpose: It redirects the user to the help page if the help icon button is pressed.
- Input: Click on the Help icon button.
- Output: Opens the help page of the app.
- Parameters: Help icon Button.
- Exceptions: None
- Pseudocode:

```
If OnClick of help icon button:
    Move to the help page
```

iv. Method 3- ScamItem()

- Purpose: It opens the camera if the user clicks on the scan icon button and on taking the image performs a classification of the image.
- Input: Click on the Scan icon button.
- Output: Opens the camera of the app.
- Parameters: Scan icon Button.
- Exceptions: None
- Pseudocode:

```
If OnClick of Scan icon button:
    openCamera
```

If a picture is clicked:

Open the scan page and execute classify item()

v. Method 4- GotoHome()

- Purpose: It sends the user to the help page if the home icon button is pressed.
- Input: Click on the Home icon button.
- Output: Opens the homepage of the app.
- Parameters: Home icon Button.
- Exceptions: None
- Pseudocode:

If OnClick of home icon button:

Move to the home page.

vi. Method 5- ShowCart()

- Purpose: It shows a list of items the user has added to the cart.
- Input: Click on the Cart icon button.
- Output: Opens the cart page of the app.
- Parameters: Cart icon Button.
- Exceptions: None.
- Pseudocode:

If OnClick of cart icon button:

Display the list of items a user has added to the cart

vii. Method 6- viewProfile()

- Purpose: It sends the user to the profile page if the user clicks the avatar.
- Input: Click on the avatar.

- Output: Opens the profile page of the app.
- Parameters: Avatar icon.
- Exceptions: None.
- Pseudocode:

If OnClick of avatar icon button:

Move to the profile page.

5.5.4.6 Class name 4 Profile

i. Class Description 4: It contains different components needed to form the profile page of the app.

Table 5.5.4.6: Data members for Profile class.

Data Type	Data Name	Access Modifiers	Initial Value	Description
ImageView	avatar	Private	“	Avatar of the user.
String	name	Private	“	Username of the user.
String	email	Private	“	Email of the user.

iii. Method 1 - changeAvatar()

- Purpose: It sends the user to a profile page if the user clicks the avatar.
- Input: Click on the avatar.
- Output: Opens the profile page of the app.
- Parameters: Avatar icon.
- Exceptions: None.
- Pseudocode: If OnClick of avatar icon edit button:

Select an image from the file(png) and Change the avatar of the user.

5.5.4.7 Class name 5 Help

i. Class Description 5

It contains different components needed to form the help page of our app

Table 5.5.4.7: Data members for Help class.

Data Type	Data Name	Access Modifiers	Initial Value	Description
ImageView	avatar	Private	“	Avatar of the user.

ii. Method 1- ScamItem()

- Purpose: It opens up the camera if the user clicks on the scan icon button and on taking an image performs a classification of the image.
- Input: Click on the Scan icon button.
- Output: Opens the camera of the app.
- Parameters: Scan icon Button.
- Exceptions: None
- Pseudocode: If OnClick of Scan icon button:

  openCamera

  If a picture is clicked:

  Open scan page and execute classify item()

iii. Method 2- GotoHome()

- Purpose: It sends the user to the help page if the user clicks the home icon button.
- Input: Click on the Home icon button.

- Output: Opens the homepage of the app.
- Parameters: Home icon Button.
- Exceptions: None
- Pseudocode: If OnClick of home icon button:

  Move to homepage

iv. Method 3- ShowCart()

- Purpose: It shows a list of items the user has added to the cart.
- Input: Click on Cart icon button.
- Output: Opens the cart page of the app.
- Parameters: Cart Icon Button.
- Exceptions: None
- Pseudocode: If OnClick of cart icon button:

  Display the list of items the user has added to the cart.

v. Method 4 viewProfile()

- Purpose: It sends the user to the profile page if the user clicks the avatar.
- Input: Click on the avatar.
- Output: Opens the profile page of the app.
- Parameters: Avatar icon.
- Exceptions: None
- Pseudocode: If OnClick of avatar icon button:

  Move to profile page

5.5.4.8 Class name 6 Cart

i. Class Description 6: It contains different components needed to form the cart page of our app

Table 5.5.5.8: Data members for cart class.

Data Type	Data Name	Access Modifiers	Initial Value	Description
ImageView	avatar	Private	“	Avatar of the user.

ii. Method 1 - AddItem()

- Purpose: It adds the item to the user’s cart after getting the weights from the shopping cart.
- Input: Click on add item button.
- Output: Add item to the cart.
- Parameters: Add item button on scan page.
- Exceptions: None
- Pseudocode: If OnClick of additem button:

  add the item to the cart

iii. Method 2 - RemoveItem(item)

- Purpose: It removes the item from the user’s cart after getting the weights from the shopping cart.
- Input: Click on the remove item button.
- Output: Remove the item from the cart.
- Parameters: Remove item button in the scan page.
- Exceptions: None

- Pseudocode: If OnClick of removeitem button:

  Remove item from the cart

iv. Method 3- GotoHome()

- Purpose: It sends the user to the help page if the user clicks the home icon button.
- Input: Click on the Home icon button.
- Output: Opens the homepage of the app.
- Parameters: Home icon Button.
- Exceptions: None
- Pseudocode: If OnClick of home icon button:

  Move to homepage

v. Method 4- viewProfile()

- Purpose: It sends the user to the profile page if the user clicks the avatar.
- Input: Click on the avatar.
- Output: Opens the profile page of the app.
- Parameters: Avatar icon.
- Exceptions: None.
- Pseudocode: If OnClick of avatar icon button:

  Move to profile page.

5.5.4.9 Class 7 name Orders

i. Class Description

It contains different components needed to form the order page of our app.

Table 5.5.4.9: Data members for Orders class.

Data Type	Data Name	Access Modifiers	Initial Value	Description
ImageView	avatar	Private	“	Avatar of the user.
<List>Order	orders	Private	‘’	List of previous user order detail

ii. Method 1 - getPreviousOrderDetails()

- Purpose: It shows a list of previous orders fetched from the database for the current user if the user clicks the previous order button.
- Input: Click on the previous order button.
- Output: Shows a list of all previous orders done by the user.
- Parameters: Previous order button.
- Exceptions: None
- Pseudocode: If OnClick of previous order button:

  Show a list of previous orders done by the user.

5.5.4.10 Class name 8 scan:

i. Class Description 8

It contains different components needed to form the scan page of our app

Table 5.5.4.10: Data members for scan class.

Data Type	Data Name	Access Modifiers	Initial Value	Description
ImageView	Item	Private	“	Image clicked by the user.

iii. Method 1 - ClassifyItem()

- Purpose: It performs an image classification based on the image clicked by the user.
- Input: Image captured by camera.
- Output: Class of item image belongs to.
- Parameters: None
- Exceptions: None
- Pseudocode: If OnPerform of scanitem:
  Perform a classification of items.

5.5.4.11 Class name 9 GetTheWeight

i. Class Description 9

It contains different components needed to form the get the weight page of our app

Table 5.5.4.11: Data members for GetTheWeight class.

Data Type	Data Name	Access Modifiers	Initial Value	Description
decimal	weight	Private	“	Weight detected by the weight sensor.

iii. Method 1 - getweight()

- Purpose: It detects the weight sensed by the weight sensor.
- Input: Item placed in shopping cart.
- Output: Weight detected by weight sensor.
- Parameters: None
- Exceptions: None
- Pseudocode: If OnChange in weight sensor:

Return the weight detected by the weight sensor.

5.5.4.12 Class name 10 Item

i. Class Description 10

It contains different details for items added to the shopping cart.

Table 5.5.5.12: Data members for item class.

Data Type	Data Name	Access Modifiers	Initial Value	Description
String	Class	Private	“	Class of item clicked by user.
Decimal	Price	Private	‘’	Price of Item fetched by the user.
Decimal	Weight	Private	‘’	Weight detected by the weight sensor.

5.5.4.13 Class Name 11 Order

i. Class Description 11

It contains different details for orders placed by the user previously.

Table 5.5.5.13: Data members for orders class.

Data Type	Data Name	Access Modifiers	Initial Value	Description
String	Storename	Private	“	Store name where order was placed.
String	Location	Private	‘’	Location where order was placed.
<List>Item	Items	Private	‘’	List of Items placed by the user.
String	Paymentdetails	Private	‘’	Payment details for the order.

iii. Method 1 - showorderdetails()

- Purpose: It shows all details of order on clicking an order shown in previous order list.
- Input: Click on one of the previous order list.
- Output: Details of order fetched from database.
- Parameters: None
- Exceptions: None
- Pseudocode: If onClick on one of previous orders: Display list of selected order.

5.5.5. Sequence Diagram

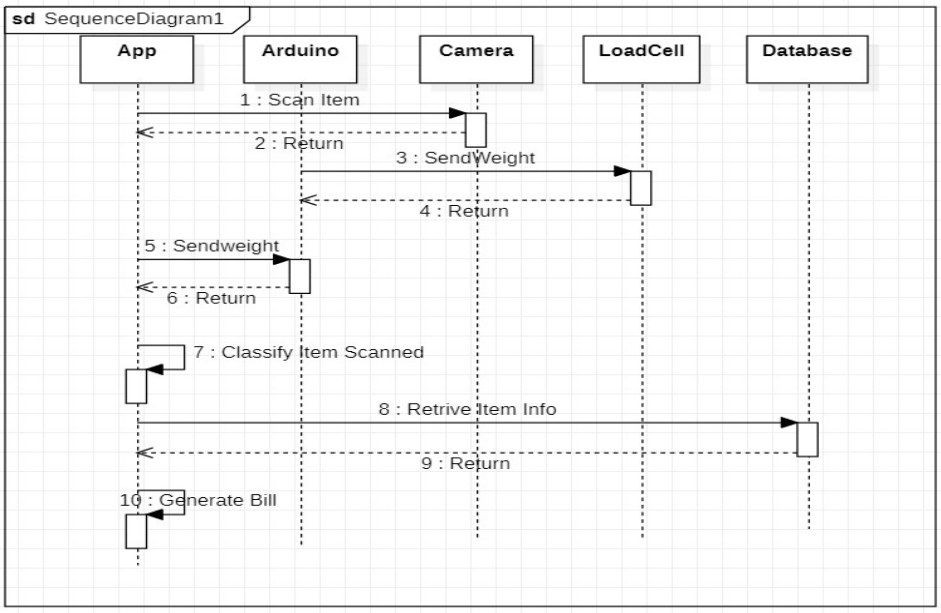

Fig 5.5.5: sequence Diagram

5.5.6 Packaging and Deployment Diagrams

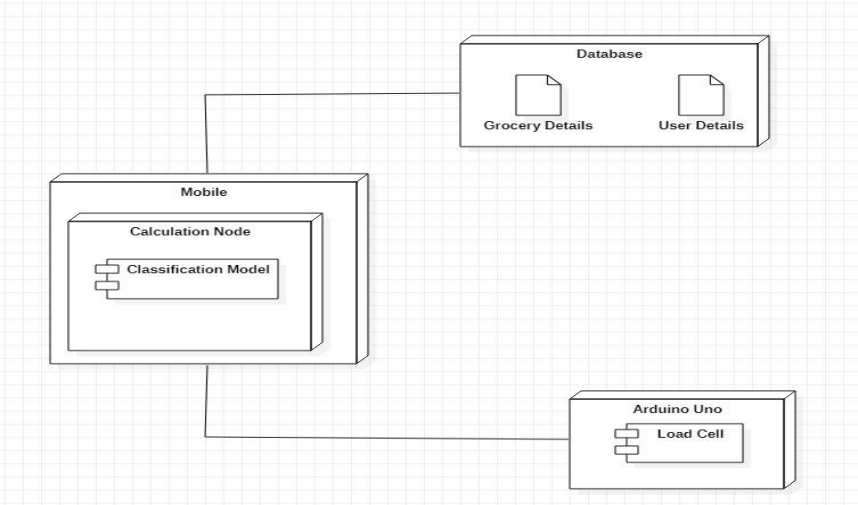

Fig 5.5.6: Deployment Diagram

5.5.7 Module 3 - IoT Components

5.5.7.1 Description:

This module deals with the hardware that includes the load cell, the Arduino board and the microcontroller that acts as a bridge between the two. The weight of the load is measured by the load cell and is transferred to the Arduino through the microcontroller. Taring the weight on the load cell is done to ensure correct readings.

5.5.7.2 Class Diagram:

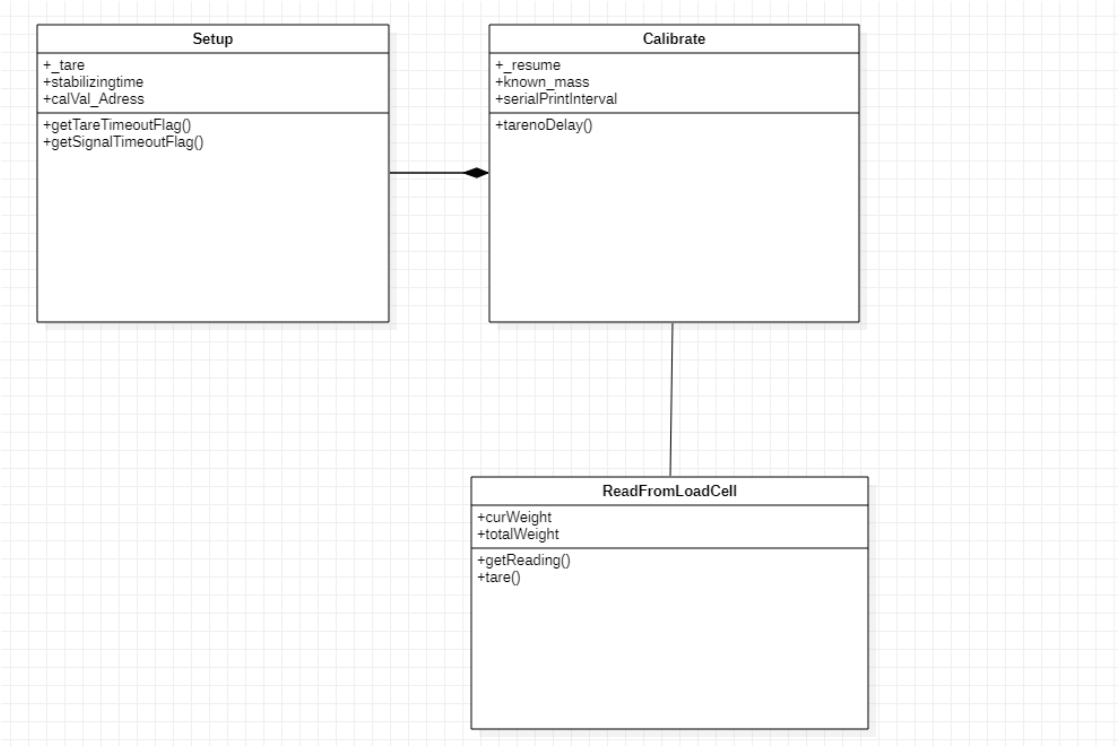

Fig 5.5.7.2: Class Diagram for IoT Components

5.5.7.3 Class 1- Calibration:

i. Class Description:

The system is tared to the weight of the plate kept on the load cells to increase surface area used to measure the weight of the products.

Table 5.5.7.3: Data members for calibration table.

Data Type	Data Name	Access Modifiers	Initial Value	Description
Boolean	_resume	Private	“	Ensures updation from load cell.
float	known_mass	Private	“	Holds current known mass reading from load cell.

ii. Method 1- tareNoDelay():

- Purpose: Method to tare the weight of the load onto the microcontroller
- Input: none
- Output: weight shows zero.
- Parameters: load.
- Exceptions: None.

5.5.7.4 Class 2- Read from load cell:

This class is used to read the weight from the load cell which is converted by the microcontroller then sent to the Arduino board.

Table 5.5.7.4: Data Members for readfromloadcell class.

Data Type	Data Name	Access Modifiers	Initial Value	Description
float	curWeight	Private	“0	Holds most recent read value from load cell controller.
float	totalWeight	Private	“0	Holds total mass present on load cell from load cell.

ii. Method 1 - GetReading():

- Purpose: Method to read weight of the load from the load cell microcontroller
- Input: none
- Output: weight from microcontroller.
- Parameters: none.
- Exceptions: None.

iii. Method 2 - tare():

- Purpose: Method to tare weight of the load in the microcontroller
- Input: none
- Output: weight from microcontroller.
- Parameters: none.
- Exceptions: None.

CHAPTER 6

PROPOSED METHODOLOGY

The proposed methodology for our project is as follows:
Generation of the dataset using web scraping and then using augmentation and fine-tuning to have an equal set of images for each class for training.
Once the dataset is generated the next step is to develop several deep-learning models. So we trained two pre-trained models found in the TensorFlow hub for our dataset. The two models that we trained were mobile net and efficient. After the model was trained, it was converted to a. tflite file compatible with android. With the. tflite model available, we performed an ensemble technique for better classification in our android app.

On getting an accurate classification we moved on to making various pages of our android app. Next, we moved on to get the weights of an item classified with the help of a load cell. We made use of 4 load cells connected to four corners of the cardboard base. These load cells were connected to the Arduino which would send the data to our Arduino IDE connected to our laptop.

Once data was received, we tried wireless transmission of data so we made use of the Bluetooth module HC-05 to send and receive data. After data was received in our Arduino IDE, we moved to receive our data in our android app. Based on the received data, we then used a timer and a difference of weight > 50 gm to get the final weight of the item being placed.
On successful completion of getting the weight and classifying the item our next part was to finish the database design and filling up of data.

The main part of the database includes a user table storing the details of the user with the previous order history, and an item table storing details of an item. Once database design was done we moved on to finish the remaining pages of the app

CHAPTER 7

IMPLEMENTATION

- Implementation of custom dataset by fine tuning and augmentation with the help of python libraries.

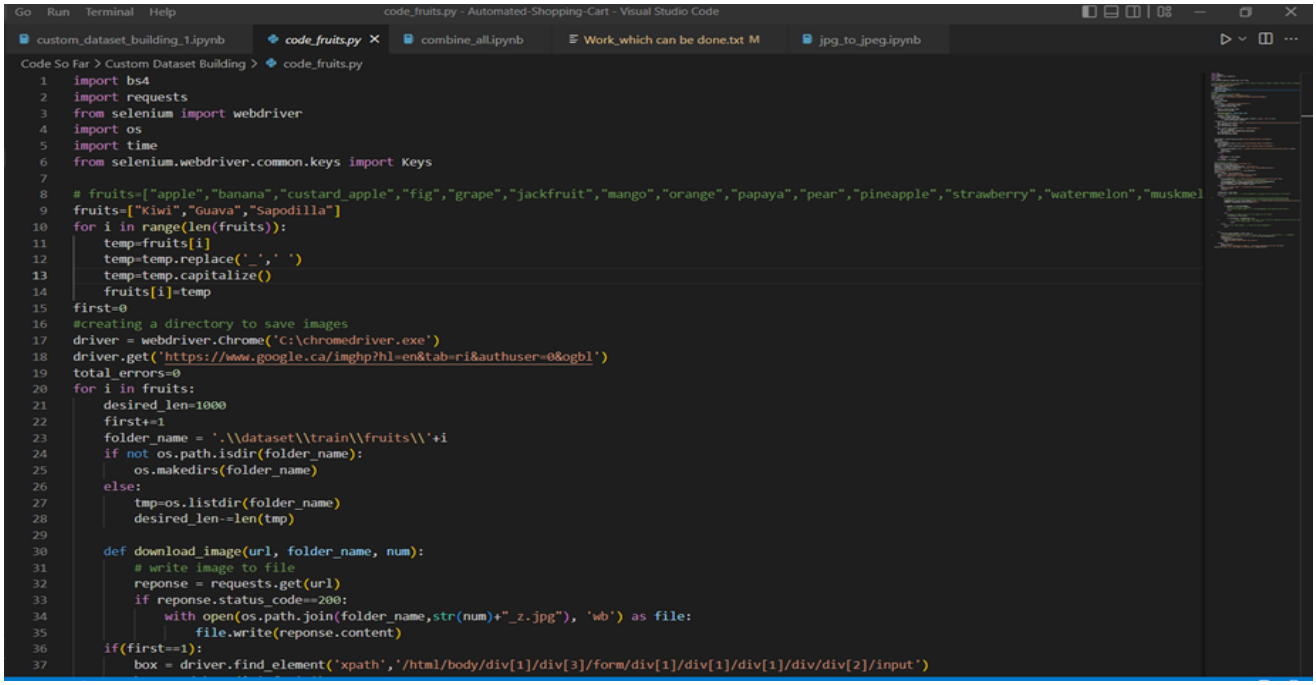

Fig.7.1: Code for building of dataset.

- Implementation of deep learning model using our custom dataset with the help of predefined model available in tensorflow hub.

```
print("Building model with", model_handle)
model = tf.keras.Sequential([
    # Explicitly define the input shape so the model can be properly
    # loaded by the TFLiteConverter
    tf.keras.layers.InputLayer(input_shape=IMAGE_SIZE + (3,)),
    hub.KerasLayer(model_handle, trainable=do_fine_tuning),
    tf.keras.layers.Dropout(rate=0.2),
    tf.keras.layers.Dense(len(class_names),
                          kernel_regularizer=tf.keras.regularizers.l2(0.0001))
])
model.build((None,)+IMAGE_SIZE+(3,))
model.summary()
```

```
Building model with https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_b0/feature_vector/2
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer (KerasLayer)    (None, 1280)              5919312

 dropout (Dropout)           (None, 1280)              0

 dense (Dense)               (None, 21)                26901

=================================================================
Total params: 5,946,213
Trainable params: 26,901
Non-trainable params: 5,919,312
_________________________________________________________________
```

```
model.compile(
  optimizer=tf.keras.optimizers.SGD(learning_rate=0.005, momentum=0.9),
  loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True, label_smoothing=0.1),
  metrics=['accuracy'])
```

```
steps_per_epoch = train_size // BATCH_SIZE
validation_steps = valid_size // BATCH_SIZE
hist = model.fit(
    train_ds,
    epochs=5, steps_per_epoch=steps_per_epoch,
    validation_data=val_ds,
    validation_steps=validation_steps).history
```

Fig 7.2: Code for building efficentnet_v2 model with the custom dataset.

```
[ ]  print("Building model with", model_handle)
     model = tf.keras.Sequential([
         # Explicitly define the input shape so the model can be properly
         # loaded by the TFLiteConverter
         tf.keras.layers.InputLayer(input_shape=IMAGE_SIZE + (3,)),
         hub.KerasLayer(model_handle, trainable=do_fine_tuning),
         tf.keras.layers.Dropout(rate=0.2),
         tf.keras.layers.Dense(len(class_names),
                               kernel_regularizer=tf.keras.regularizers.l2(0.0001))
     ])
     model.build((None,)+IMAGE_SIZE+(3,))
     model.summary()
```

```
Building model with https://tfhub.dev/google/imagenet/mobilenet_v3_large_100_224/feature_vector/5
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer (KerasLayer)    (None, 1280)              4226432

 dropout (Dropout)           (None, 1280)              0

 dense (Dense)               (None, 21)                26901

=================================================================
Total params: 4,253,333
Trainable params: 26,901
Non-trainable params: 4,226,432
_________________________________________________________________
```

```
[ ]  model.compile(
       optimizer=tf.keras.optimizers.SGD(learning_rate=0.005, momentum=0.9),
       loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True, label_smoothing=0.1),
       metrics=['accuracy'])
```

```
[ ]  steps_per_epoch = train_size // BATCH_SIZE
     validation_steps = valid_size // BATCH_SIZE
     hist = model.fit(
         train_ds,
         epochs=5, steps_per_epoch=steps_per_epoch,
         validation_data=val_ds,
         validation_steps=validation_steps).history
```

Fig 7.3: Code for building mobilenet_v2 model with the custom dataset built.

- Implementation of connection of Arduino with load cell and Bluetooth for wireless communication with the app.

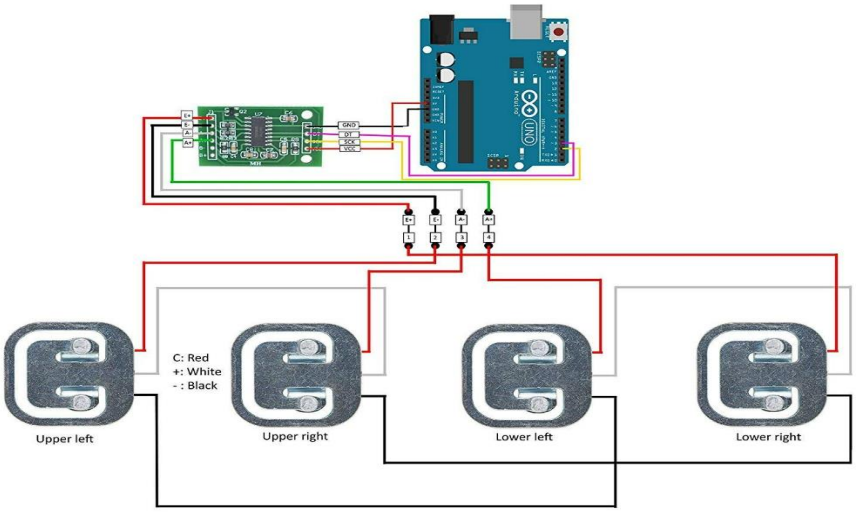

Fig 7.4: IoT components connection

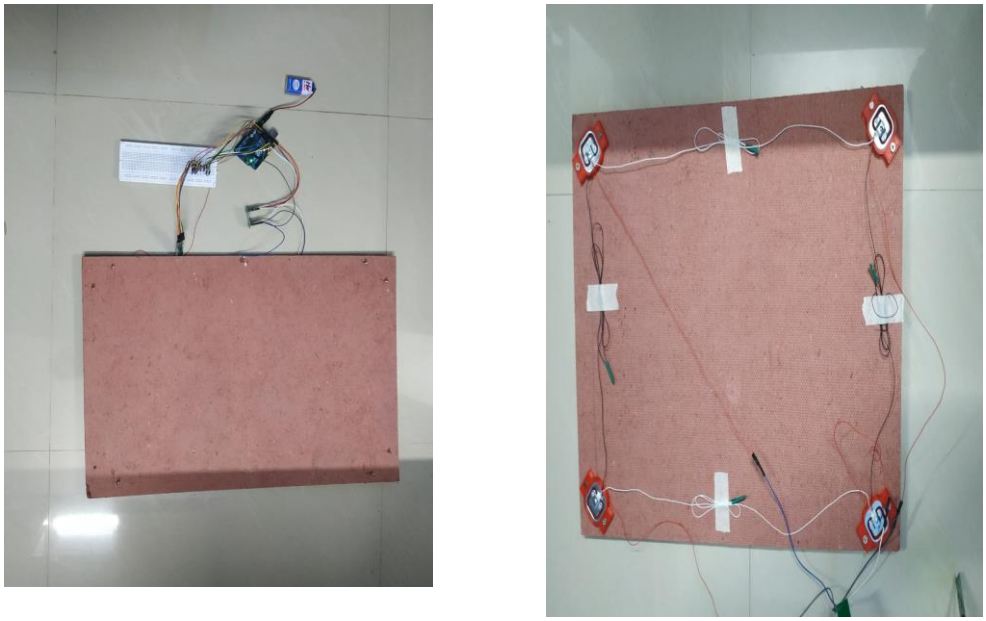

Fig 7.5: Actual implementation of load cell in a rigid surface (front & back view)

- Implementation of various pages of our app

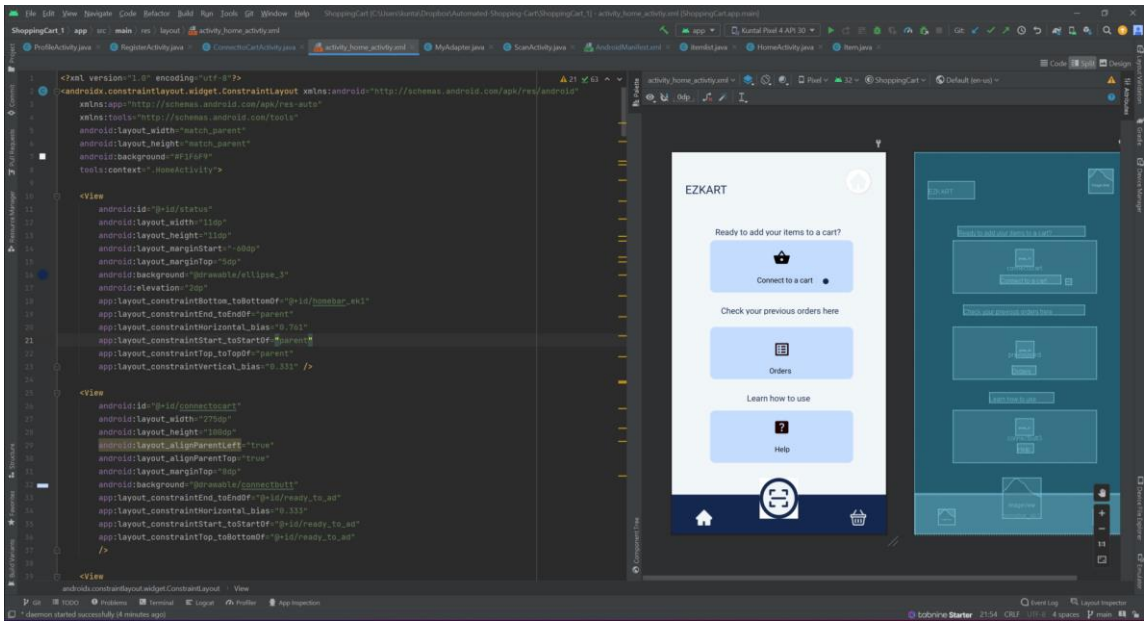

Fig 7.6 Home Page.

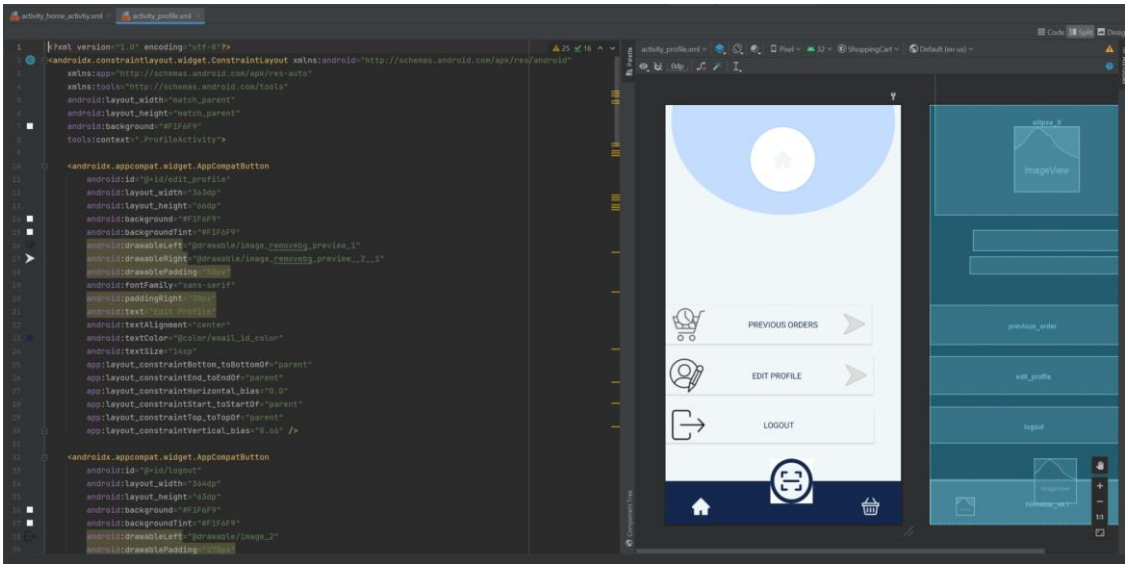

Fig 7.7 Profile Page.

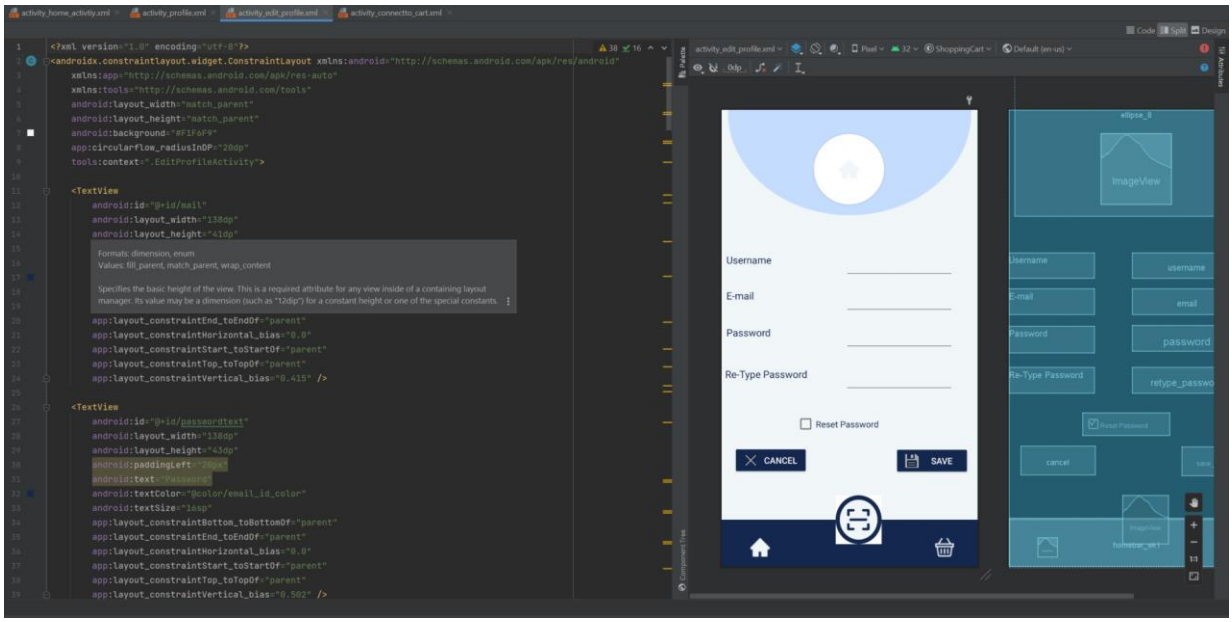

Fig 7.8 Edit profile Page.

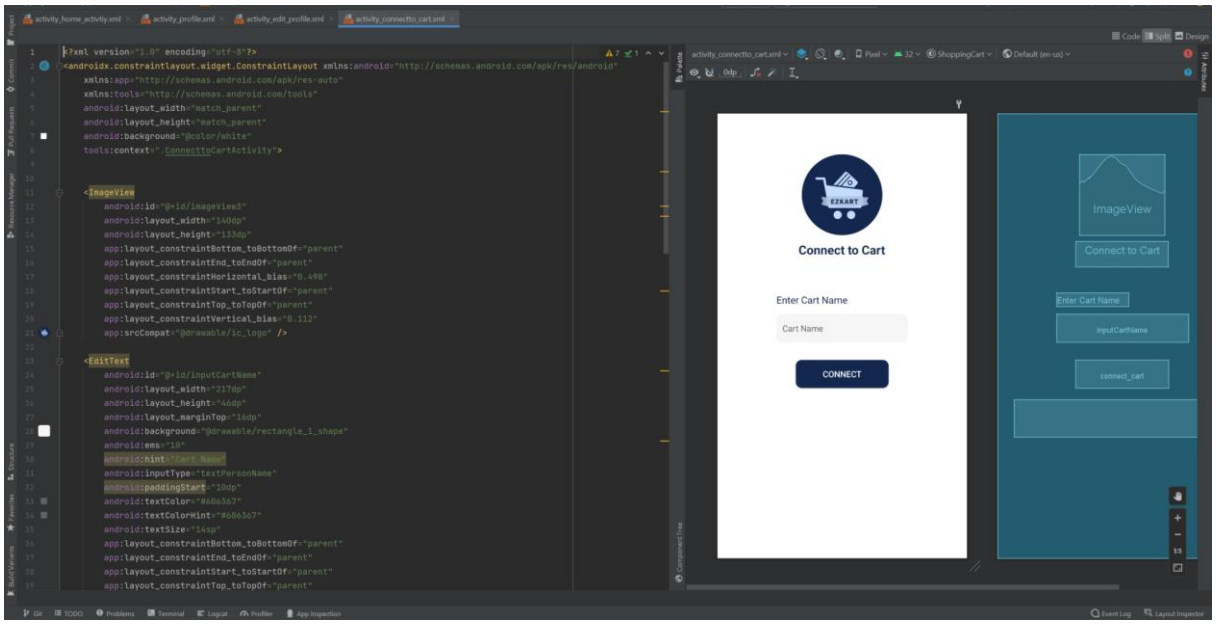

Fig 7.9 Connect to Cart Page.

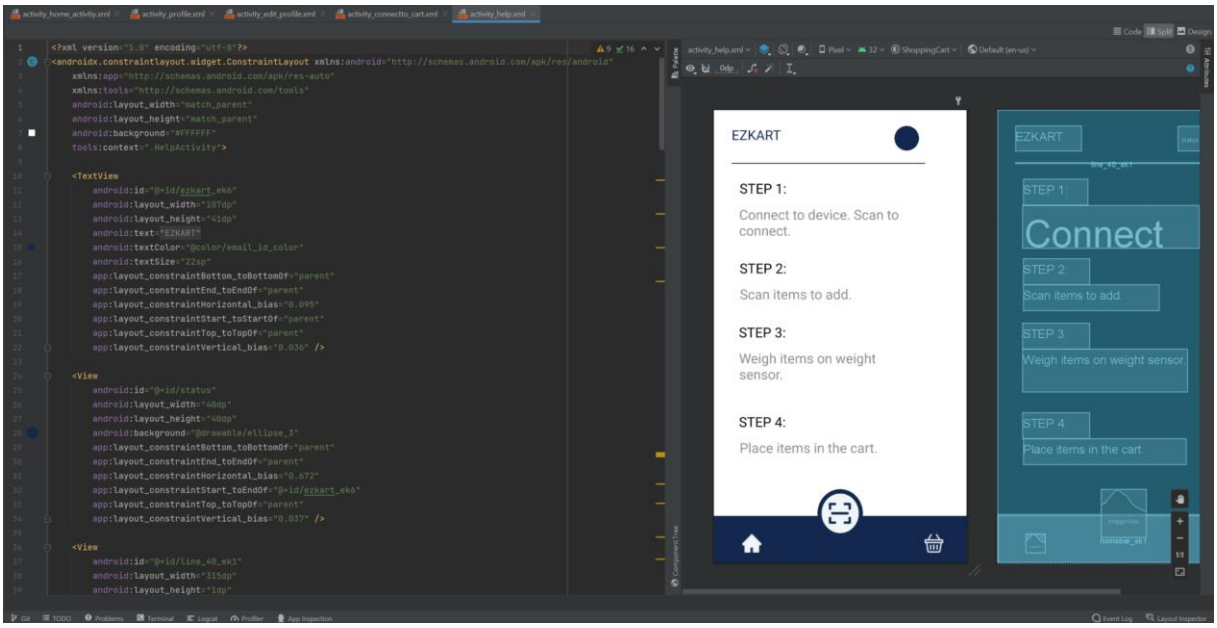

Fig 7.10 Help Page

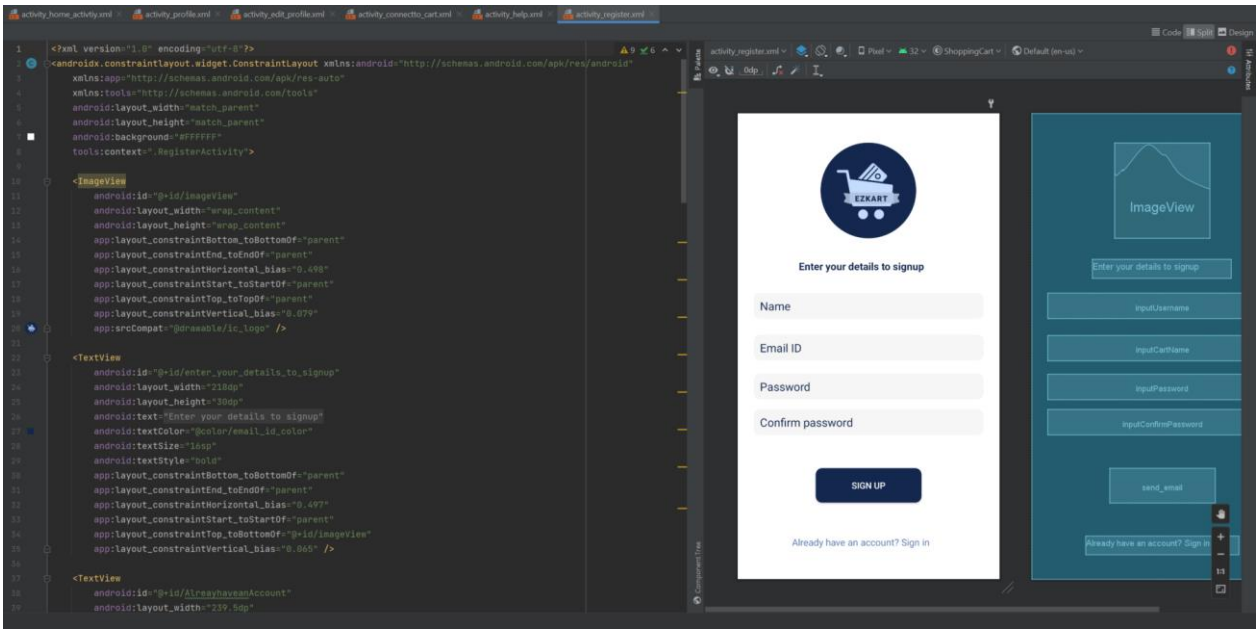

Fig 7.11 Register Page

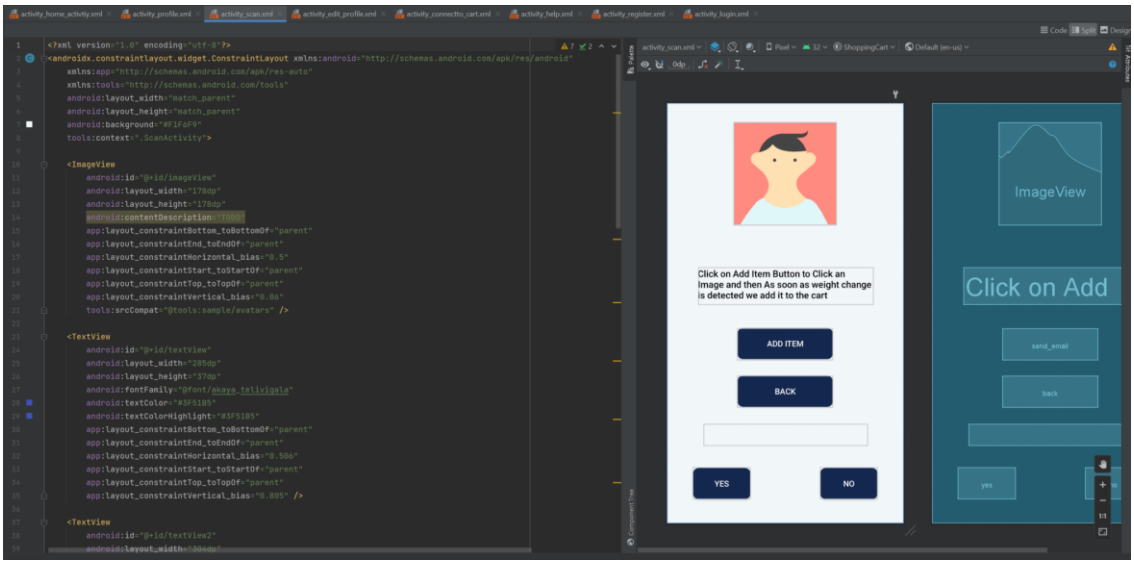

Fig 7.12 Scan Page

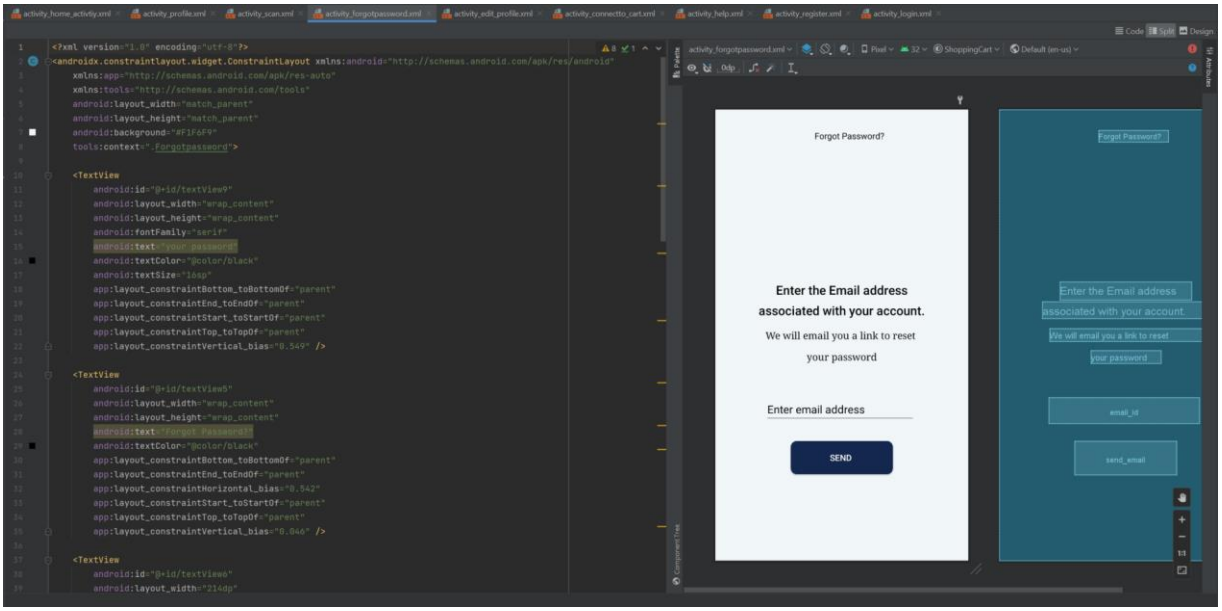

Fig 7.13 Forgot Password Page

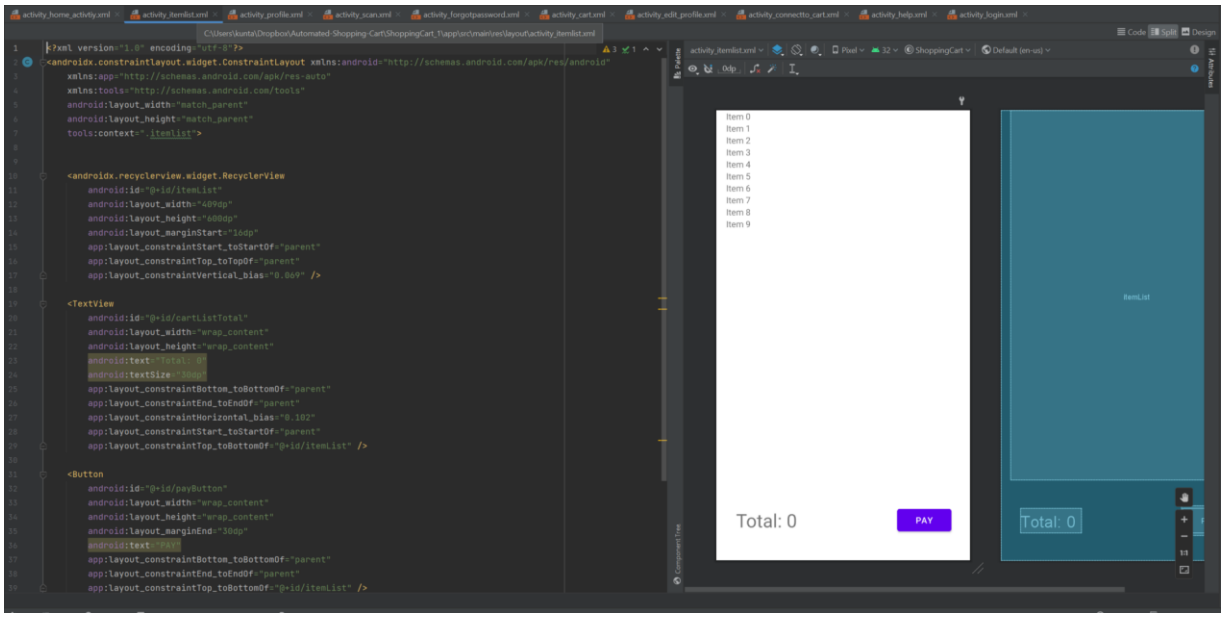

Fig 7.14 Item List Page

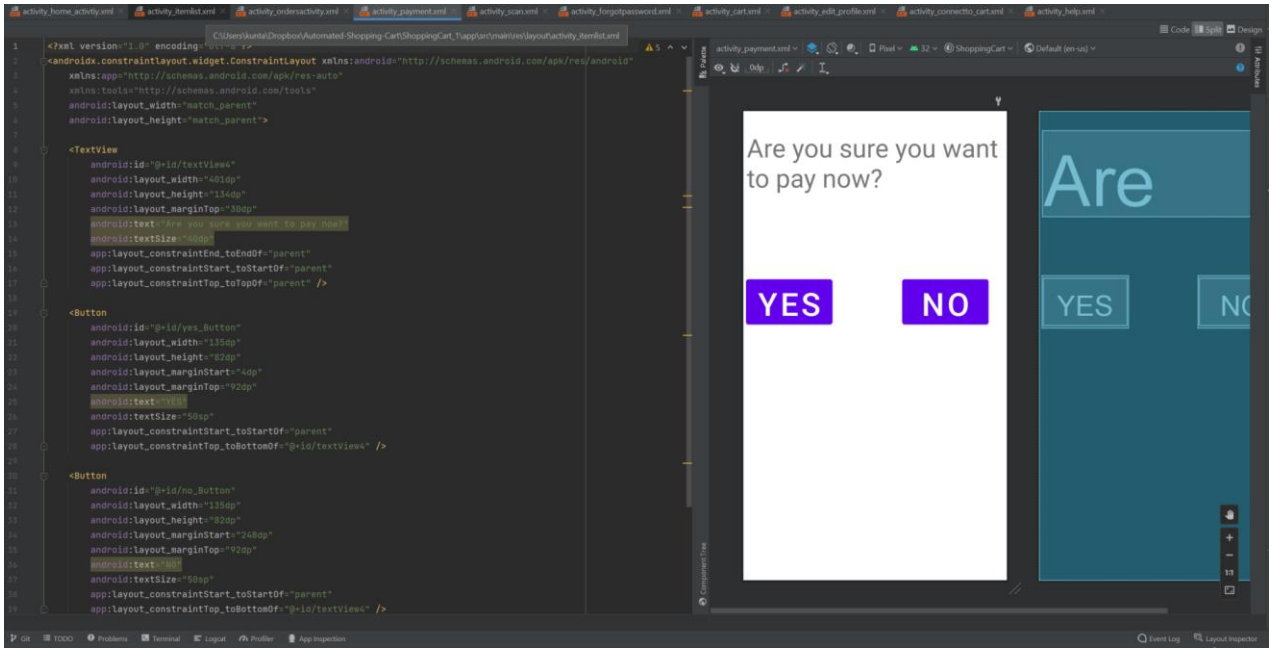

Fig 7.15: Payments Page

CHAPTER 8

RESULTS AND DISCUSSION

- **Accuracy achieved on training the model to the custom dataset.**

  On trying various models like inceptionv3,nasnet, resnet, mobilenet, efficentnet with and without augmentation we found that the model mobilenet and efficentnet gave an accuracy greater than 95 when the data were augmented. Hence we planned on ensembling both the models to achieve more accuracy. Final testing accuracy that was achieved on ensemble method was 97%. Below are a few screenshots to show the same.

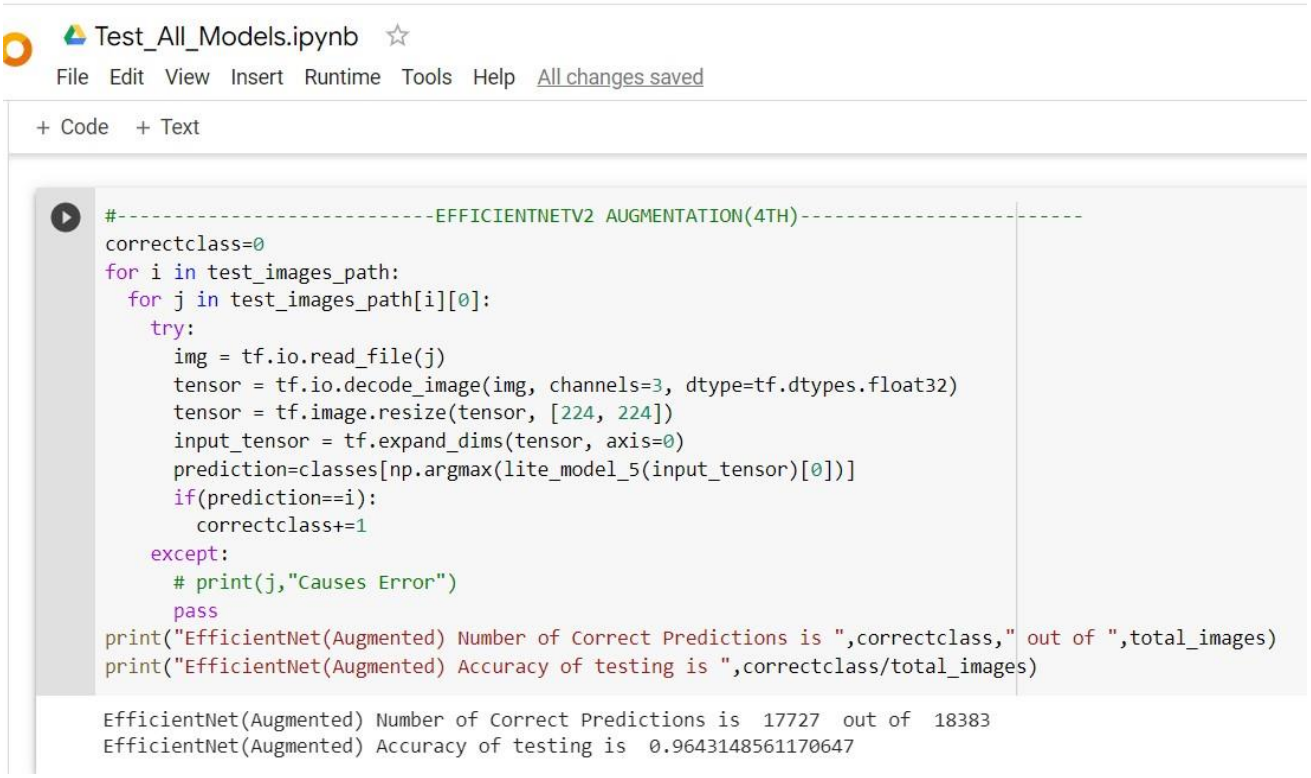

Fig 8.1: EfficientNet Accuracy

- 96% Accuracy achieved on random test data for efficentnet with augmentation model.

Test_All_Models.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ ] #-------------------------------MOBILENET AUGMENTATION(6TH)-----------------------------
    correctclass=0
    for i in test_images_path:
      for j in test_images_path[i][0]:
        try:
          img = tf.io.read_file(j)
          # print(img)
          tensor = tf.io.decode_image(img, channels=3, dtype=tf.dtypes.float32)
          tensor = tf.image.resize(tensor, [224, 224])
          input_tensor = tf.expand_dims(tensor, axis=0)
          prediction=classes[np.argmax(lite_model_4(input_tensor)[0])]
          # print("prediction is",prediction,"class is",i )
          if(prediction==i):
            correctclass+=1
        except:
          # print(j,"Causes Error")
          pass
    print("MobileNetV2(Augmented) Number of Correct Predictions is ",correctclass," out of ",total_images)
    print("MobileNetV2(Augmented) Accuracy of testing is ",correctclass/total_images)
```

```
MobileNetV2(Augmented) Number of Correct Predictions is  17747  out of  18383
MobileNetV2(Augmented) Accuracy of testing is  0.9654028178208127
```

Fig 8.2: MobileNet-v2 with augmentation model accuracy .

- 96% Accuracy achieved on random test data for mobilenet-v2 with augmentation model.

Test_All_Models.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ ] #----------------------------MOBILENETV2 AUGMENTATION(2ND)--------------------
    correctclass=0
    for i in test_images_path:
      for j in test_images_path[i][0]:
        try:
          img = tf.io.read_file(j)
          tensor = tf.io.decode_image(img, channels=3, dtype=tf.dtypes.float32)
          tensor = tf.image.resize(tensor, [224, 224])
          input_tensor = tf.expand_dims(tensor, axis=0)
          prediction=classes[np.argmax(lite_model_2(input_tensor)[0])]
          if(prediction==i):
            correctclass+=1
        except:
          print(j,"Causes Error")
          pass
    print("MobileNetV2(Non Augmented) Number of Correct Predictions is ",correctclass," out of ",total_images)
    print("MobileNetV2(Non Augmented) Accuracy of testing is ",correctclass/total_images)
```

```
MobileNetV2(Non Augmented) Number of Correct Predictions is  15326  out of  18383
MobileNetV2(Non Augmented) Accuracy of testing is  0.8337050535821139
```

Fig 8.3: MobileNet-v2 Accuracy with augmentation model.

- 83% accuracy achieved on random test data for mobilenetv2 without augmentation model.

Test_All_Models.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ ]  #--------------------------------ENSEMBLE MODEL-----------------------------------
     correctclass=0
     for i in test_images_path:
       for j in test_images_path[i][0]:
         try:
             img = tf.io.read_file(j)
             tensor = tf.io.decode_image(img, channels=3, dtype=tf.dtypes.float32)
             tensor = tf.image.resize(tensor, [224, 224])
             input_tensor = tf.expand_dims(tensor, axis=0)
             #print(lite_model_6(input_tensor)[0])
             # prediction=classes[np.argmax(lite_model_6(input_tensor)[0])]

             # b1 = lite_model_2(input_tensor)[0]
             b2 = lite_model_4(input_tensor)[0]
             b3 = lite_model_5(input_tensor)[0]
             # prediction_1 = classes[np.argmax(b1)]
             # prediction_2 = classes[np.argmax(b2)]
             # prediction_3 = classes[np.argmax(b3)]
             # print(b3)
             # b1 = 2.*(b1 - np.min(b1))/np.ptp(b1)-1
             b2 = 2.*(b2 - np.min(b2))/np.ptp(b2)-1
             b3 = 2.*(b3 - np.min(b3))/np.ptp(b3)-1
             b = 0.50*b2+0.50*b3
             prediction = classes[np.argmax(b)]
             if(prediction==i):
                     correctclass+=1
             # print(prediction_1,prediction_2,prediction_3,prediction_4)
         except:
           # print(j,"Causes Error")
           pass
     print("Number of Correct Predictions is ",correctclass," out of ",total_images)
     print("Accuracy of testing is ",correctclass/total_images)

     Number of Correct Predictions is  17956  out of  18383
     Accuracy of testing is  0.9767720176249796
```

Fig 8.4: Accuracy for the best models.

- 98% accuracy achieved on random test data for ensemble of top 3 accurate models.

- Execution of the various pages of the app.

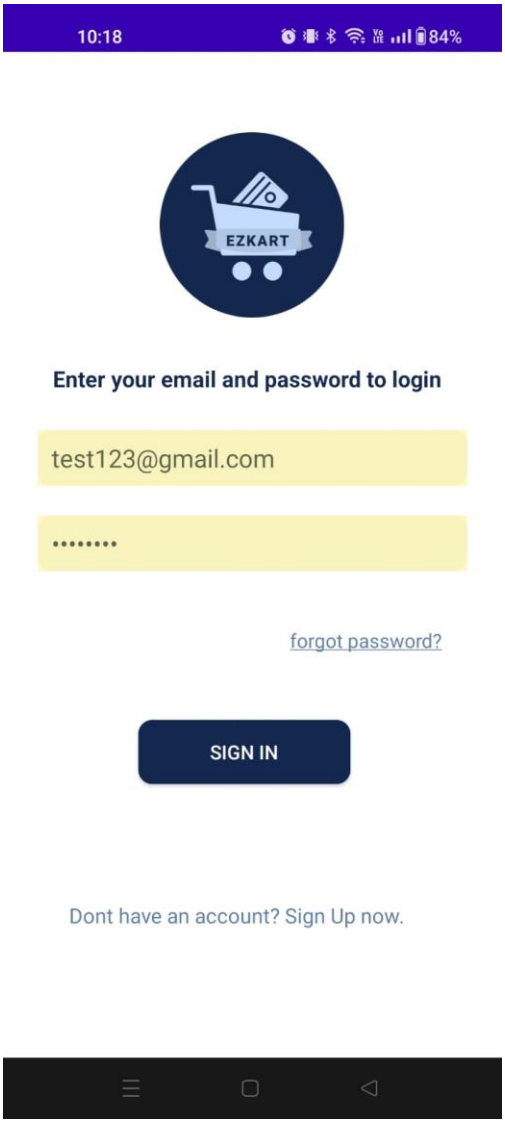

Fig 8.5: Login with email & Password.

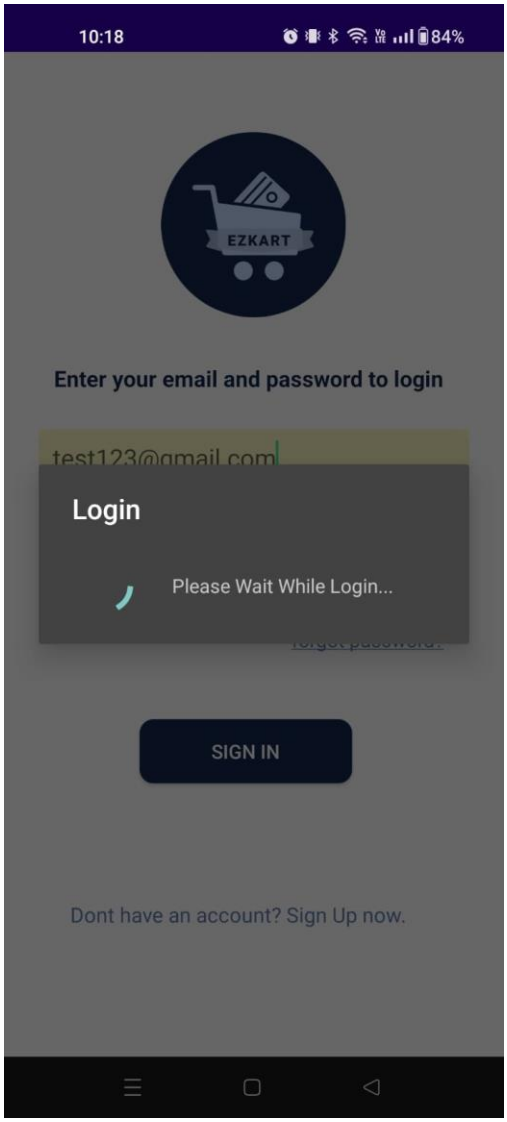

Fig 8.6: Logging In

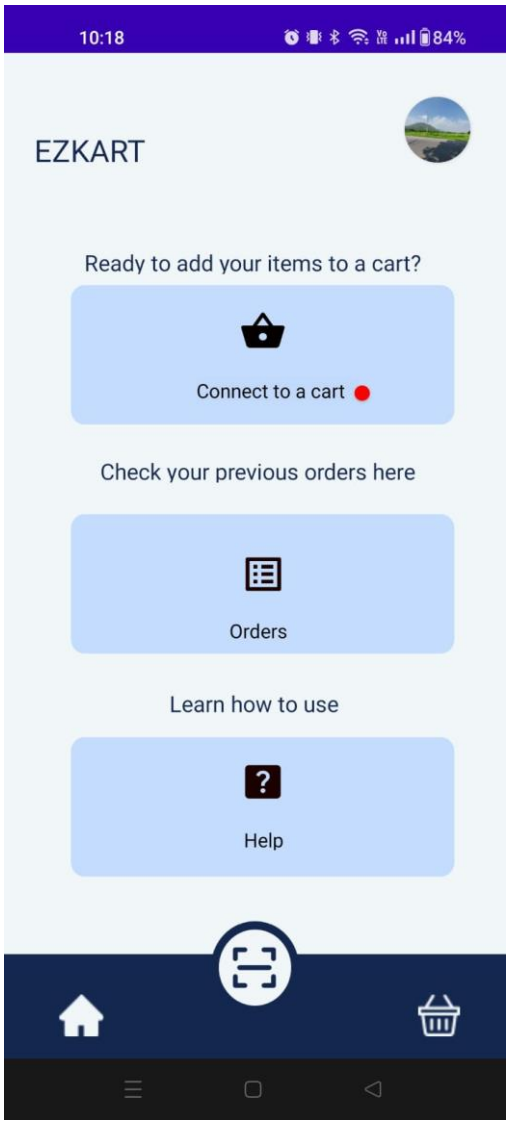

Fig. 8.7: Home Page with bluetooth disconnected

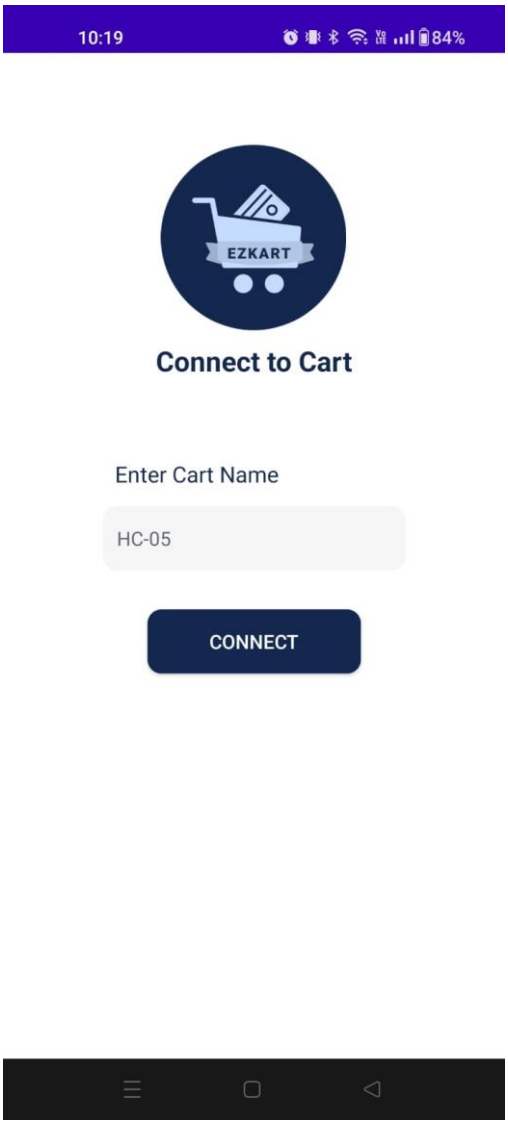

Fig 8.8: Bluetooth connection Page

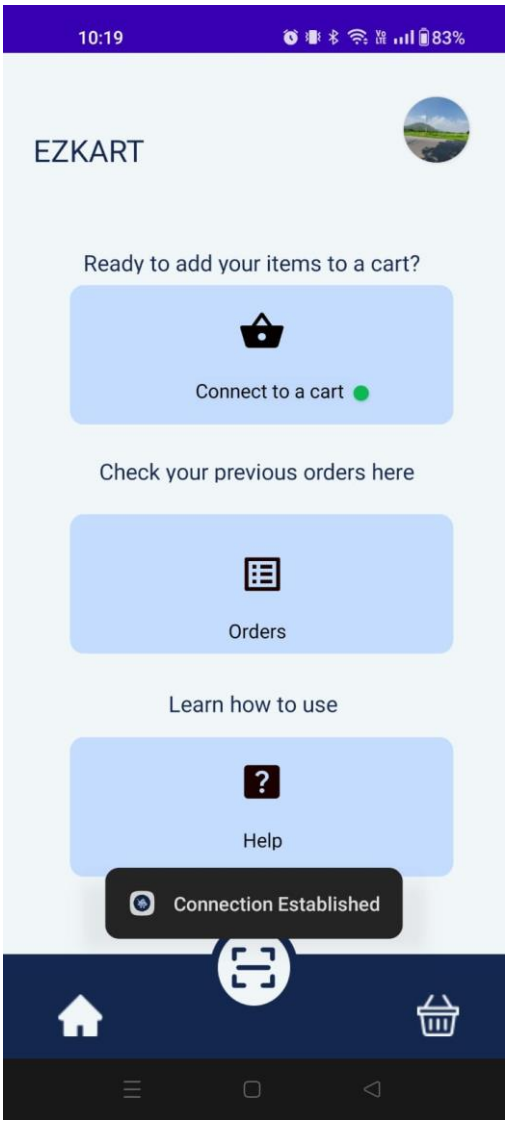

Fig. 8.9: Home Page with Bluetooth connected

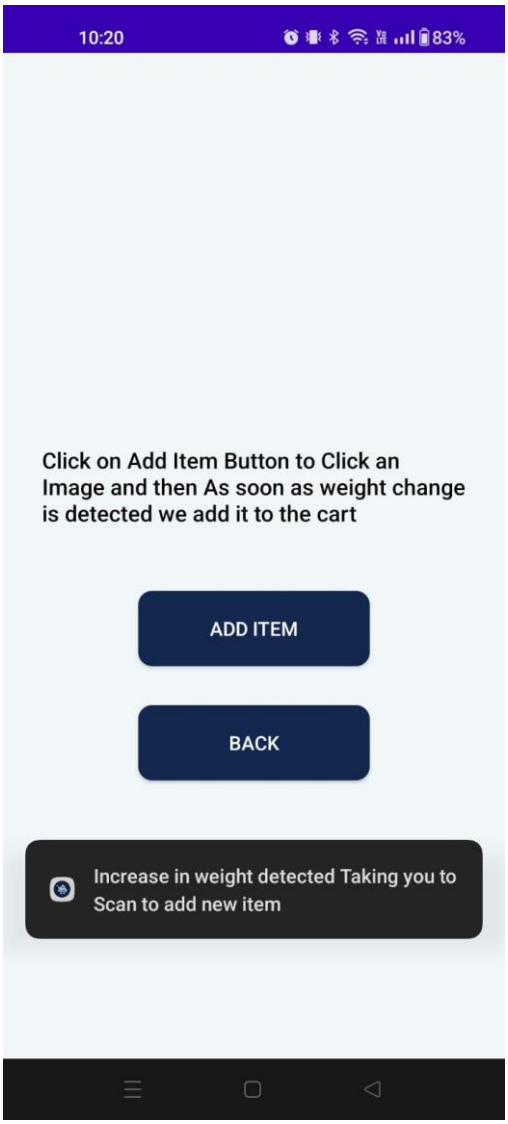

Fig. 8.10: Scan Page redirection

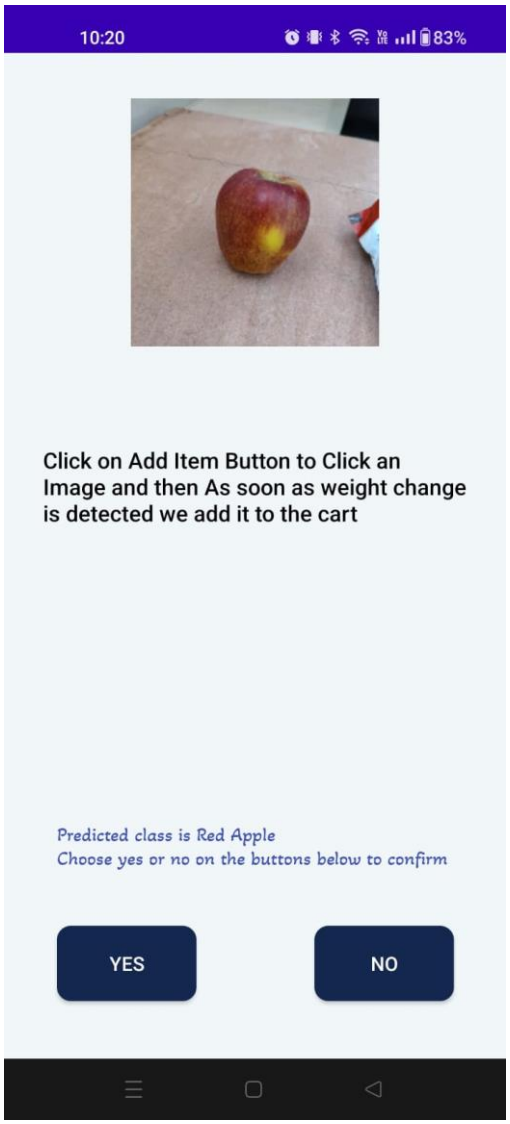

Fig 8.11: Prediction page after scanning.

Fig 8.12: Cart Page addition of item.

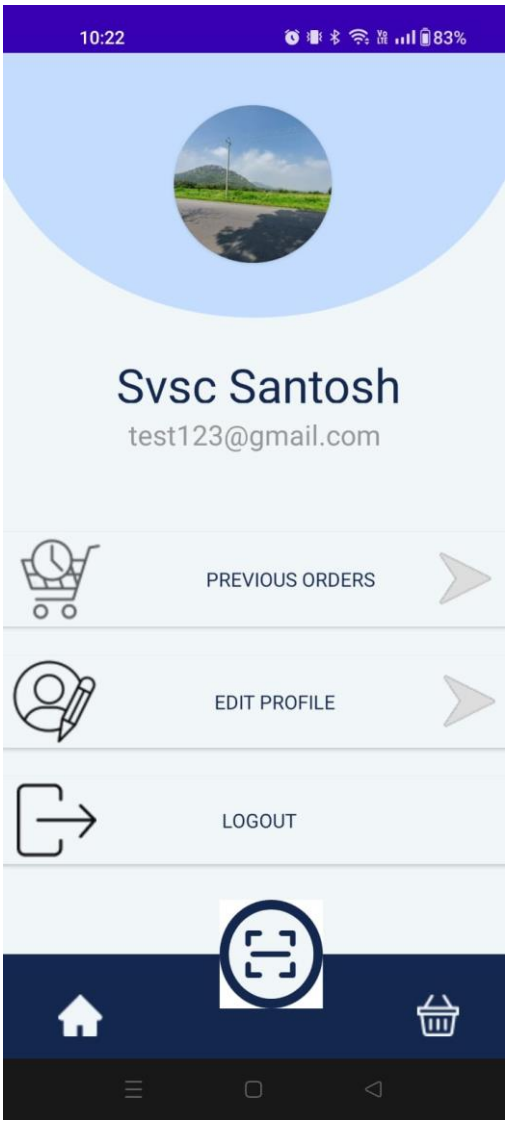

Fig 8.13: Profile Page

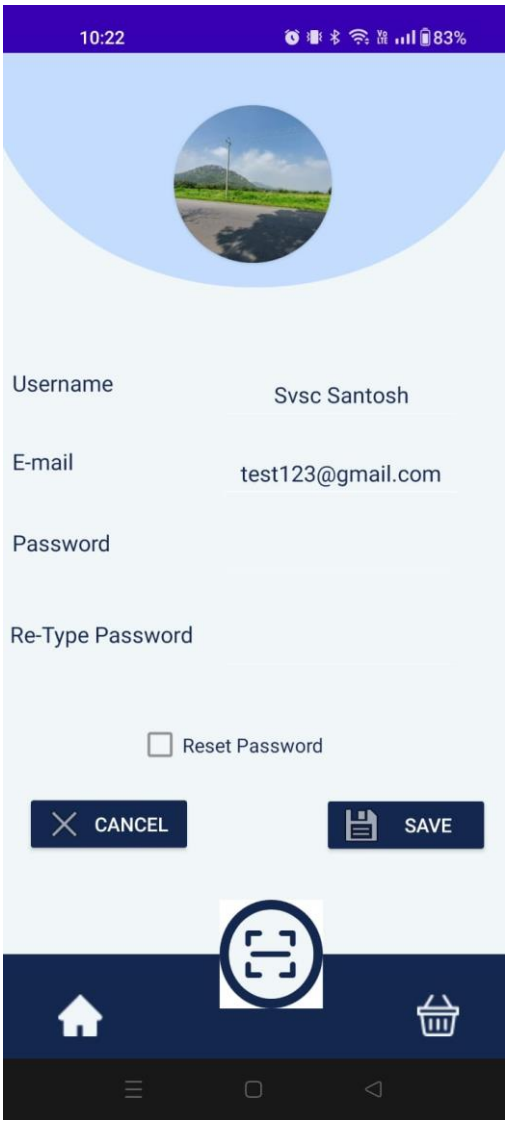

Fig 8.14. Edit profile Page

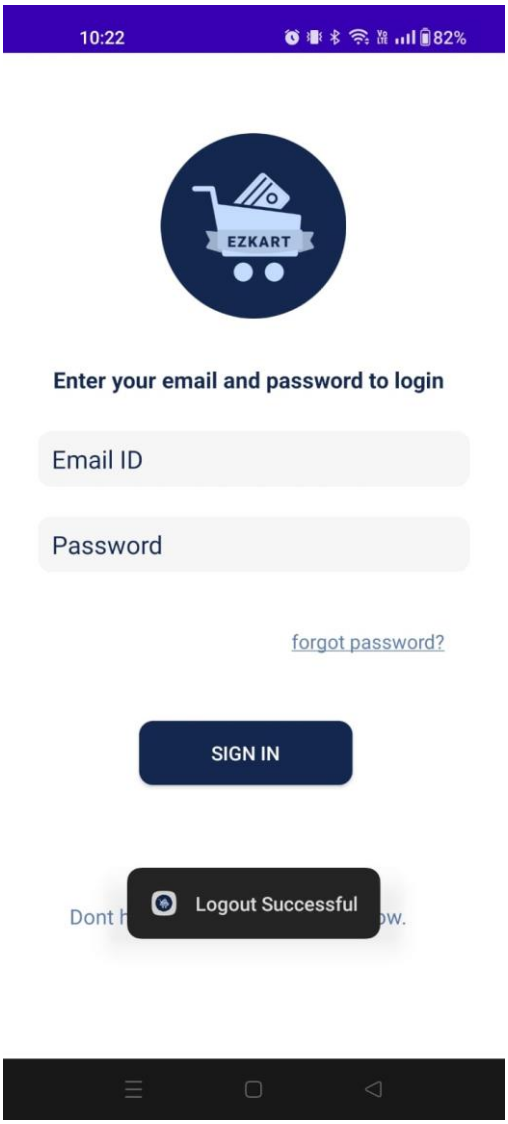

Fig 8.15: Successful Logout

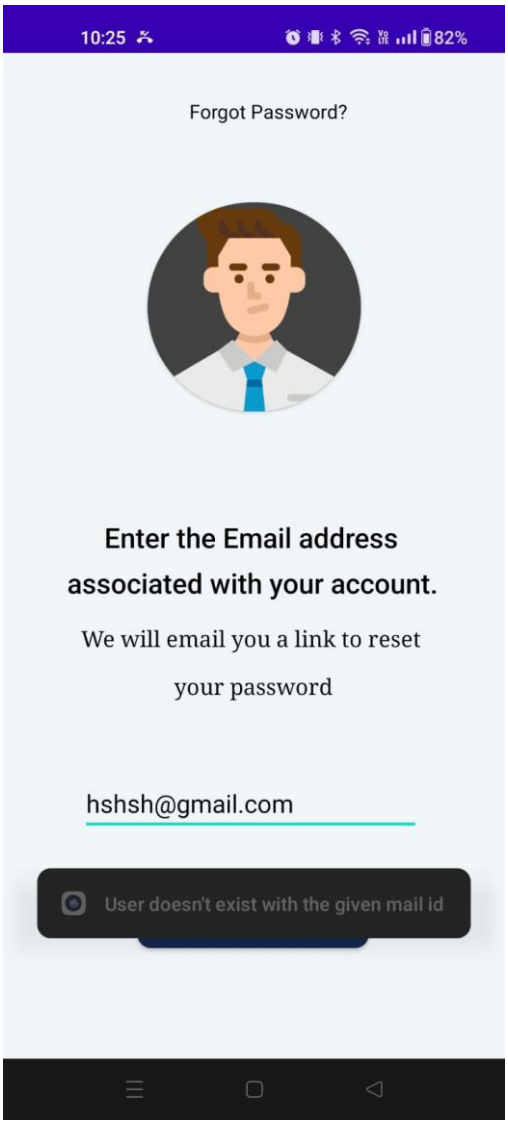

Fig 8.16. Forgot password Page

CHAPTER 9

CONCLUSION AND FUTURE WORK

The conclusions of the Capstone Project Phase – 2 are the following:

- Low-Level Design Document Completed
- Building our own custom dataset with the help of web scraping and augmentation.
- Training the pre-trained model for the custom dataset created.
- Implementation of IoT components connection with the app.
- Implementation of pages of the app discussed in low-level design completed.
- Testing of the app with various scenarios completed.

Future work that needs to be done is:

- Improvements over classifying the certain items that were not able to classify properly.
- Updating from typing the cart name to connect to the Bluetooth to a QR code for connection.
- Customizing the dataset beyond fruits and vegetables with various items present in the shopping cart.
- Customizing the shopping cart like the addition of an AI camera to see the items so that there is no misuse of product, improvement over the accuracy of weights can be done.
- Writing the research paper on the work that has been done by now and publishing to various conferences.

REFERENCES

[1] Jaime Duque Domingo, Roberto Medina Aparicio, and Luis Miguel Rodrigo. “Cross Validation Voting for Improving CNN Classification in Grocery Products.” *IEEE Access* 10 (2022): 20913–25.

[2] Shubham B Dubey, Gaurav M Kadam, and Omkar Angane. “International Research Journal of Engineering and Technology (IRJET).” *Android Application for Grocery Ordering System*, 2021.

[3] Marcus Klasson, Cheng Zhang, and Hedvig Kjellstrom. “A Hierarchical Grocery Store Image Dataset with Visual and Semantic Labels.” *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[4] Jose Luis Rojas-Aranda, Jose Ignacio Nunez-Varela, J. C. Cuevas-Tello, and Gabriela Rangel-Ramirez. “Fruit Classification for Retail Stores Using Deep Learning.” *Lecture Notes in Computer Science*, 2020, 3–13.

[5] Ameema Zainab, and Dabeeruddin Syed. “Deployment of Deep Learning Models on Resource-Deficient Devices for Object Detection.” *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, 2020.

BIBLIOGRAPHY

- https://ieeexplore.ieee.org/abstract/document/9089651
- https://www.javatpoint.com/uml-class-diagram
- https://www.uml-diagrams.org/deployment-diagrams-overview.html
- https://medium.com/@mobindustry/how-to-integrate-machine-learning-into-an-android-app-best-image-recognition-tools-1ba837c27903
- https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/blob/master/Raspberry_Pi_Guide.md
- https://www.semanticscholar.org/paper/Implementation-of-Smart-Shopping-Cart-using-Object-Oh-Chun/f5ff8985a06e02715dbbbbf4f2b4470289e003f0
- https://analyticsindiamag.com/deploying-machine-learning-models-in-android-apps-using-python/
- https://eloquentarduino.github.io/2020/01/image-recognition-with-esp32-and-arduino/
- 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.
- https://forum.arduino.cc/t/password-protect-your-project/494551
- https://github.com/antonnifo/fruits-360/blob/master/Fruit%20Classifer.ipynb
- https://images.cv/category
- https://link.springer.com/article/10.1007/s12652-020-01865-8
- https://ieeexplore.ieee.org/abstract/document/9089651
- https://www.javatpoint.com/uml-class-diagram
- https://www.uml-diagrams.org/deployment-diagrams-overview.html
- https://medium.com/@mobindustry/how-to-integrate-machine-learning-into-an-android-app-best-image-recognition-tools-1ba837c27903
- https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS

Definitions:

- 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.
- 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.
- Arduino: A microcontroller board that can be used to control the communication among all the hardware components having multiple input and output pins
- MobileNetV2: A pre trained CNN which can be used for image classification trained on a very large dataset for better classification accuracy.

Abbreviations:

- DL: Deep Learning
- ML: Machine Learning
- VAE: Variational Autoencoder
- CNN: Convolutional Neural Network
- GPU: Graphical Processing Unit
- UI: User Interface
- RFID: Radio Frequency identification
- YOLO: You Only Look Once
- VGG: Visual Geometry Group
- ResNet: Residual Neural Network

Components Used:

- Arduino UNO (1 Unit): A microcontroller to receive and transmit signals to the app.
- Load Cell (4 Units): The sensors to detect weight based on the resistance change in a Wheatstone-bridge network.
- HC05 Bluetooth module (1 Unit): The module which allows users to connect to the cart through the Android app.
- HX711 Amplifier (1 Unit): A board to easily read load cells to measure weight.

Annexure I

Automated Shopping Cart

1st Kuntal Gorai
Department of Computer Science
PES University
Bengaluru, India
kuntal0901@gmail.com

2nd S V S C Santosh
Department of Computer Science
PES University
Bengaluru, India
santoshsvsc2002@gmail.com

3rd Skanda S
Department of Computer Science
PES University
Bengaluru, India
skanda2594@gmail.com

4th Vijay Murugan A S
Department of Computer Science
PES University
Bengaluru, India
vijaymurugan1457@gmail.com

***Abstract*—The automated shopping cart aims to reduce the manual labor required for weighing and billing the items a user wishes to buy. In this paper, we build a shopping cart with weight sensors and an android application that identifies the item by scanning it with the help of a smartphone's camera and detecting the weight, making the shopping experience easier and simpler.LaTeX.**

***Index Terms*—Deep Neural Networks, Database, Automation, Weight Sensors, Shopping**

I. 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 they have 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 use transfer learning with MobileNetV2, which makes it lightweight, enabling it to run on resource-deficient devices and enabling the model to be trained again without the pre-trained saved model. MobileNetV2 has a deficient 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 back-end - 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 RF-ID 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.

Identify applicable funding agency here. If none, delete this.

II. Literature Survey

Up to five research papers were reviewed in context of different models and their architecture, viable hardware and IoT and are summarized in the following paragraphs.

Paper [1] by Jose Luis Rojas-Aranda(B) and others 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. Mobile Net 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. K means was used to get the color of the object. the model showed an accuracy of 93%. The model took into account plastic bags and was computationally inexpensive. The shortcomings include not considering varieties of groceries and needing to have a constant background.

Paper [2] intends to assist people with vision impairments in grocery shopping by giving assistance. This project made use

of both generative and deterministic deep neural networks along with a simple linear classification model like the 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. It used a small data set of real life images to train. Because of the vast number of parameters involved, high GPU and CPU capacity, as well as good hardware, are required to execute rapid and accurate classification.

Paper [3] researches faster deployment of deep learning model in resource deficit devices like 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. They used darknet with CUDA, dark flow, and OpenCV after the model was pre-trained to make the detection happen in real-time. The pre-trained model's initial weights are saved 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 makes 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.The object detection model was able to detect and classify object in mobile device within fraction of seconds. But generation of protobuf file is quite challenging. Incompatibilities with versions of python and OpenCV. Moving a huge neural network to mobile devices and making device understand the deep layers of network is very difficult.

Paper [4] 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. Based on the commands received, the shopping cart was able to move in the preferred directions. At no point during the process did the application crash. There is 100% success rate in each direction of the motion control. However This application is not developed for iOS. The UID of all the items must be scanned.

Paper [5] Automatic Load Detector Design to Determine the Strength of Pedestrian Bridges Using Load Cell Sensor Based on Arduino developed a system that can test both static and dynamic loads on a surface using load cells. The load cell hx711 was used along with an arduino uno. This system The system was measured to give outputs with up to 8.5% error, which is not negligible.

III. Motivation/Scope

Shopping malls are visited by a large number of people every day, and this causes huge queues for billing. To avoid this systems such as bar codes and RF ID tags have been used to reduce billing times. This system has shown a reduction in billing times, but the problem prevails. We use an image classification model trained on a custom data set to classify and bill grocery items like fruits and vegetables as and when they are added to the cart such that the final bill can be generated at the click of a button and hence eliminates billing time. Customers' purchases are processed by cashiers or self-service checkout equipment at retail establishments. The checkout process has already been sped up because most items have scannable barcodes. Fruits and vegetables, however, are frequently treated in diverse ways. The consumer or the cashier must manually determine the category of the item being purchased and search for it in the system. In order to assess its viability for such an application, we provide a first strategy for fruit and vegetable categorization with that use in mind.

IV. Proposed Methodology

A complete collection of images of fruits and vegetables was not readily available and was constructed. Multiple sources were referred and only the best were chosen to be included. Selenium was used to script and to scrape relevant images from websites. Various data augmentation techniques such as rotation of images by some degree, zoom in were used to expand the data set for better results. The downloaded images had different formats some of which might not be acceptable to be taken in as input to the models or did not have RGB pixels for all images so necessary conversions to .jpeg, .jpg, .png had to be done in addition to filter junk images coming out of google image search. A data set of 50 classes consisting of a total of 153k images was constructed.

After extensive research on what models to use, the ones giving the best accuracy were selected and trained with augmented and non-augmented data. EfficientNet, NASNet, and MobileNet-v2 were the models giving better accuracies and were selected. Ensemble learning was applied to the mentioned models.

Reading the input weights of the food placed in the cart is necessary. A Load Cell was used along with an Arduino-UNO to read the weight of the items in the cart. Each load cell was connected to one corner of the cart and calibrated using the micro-controller to be able to read the weights. The weights read by the load cell is sent to the Android app. Users can connect to the cart using Bluetooth and start shopping with the feature "Connect to Cart" provided in the main menu of our Android application.

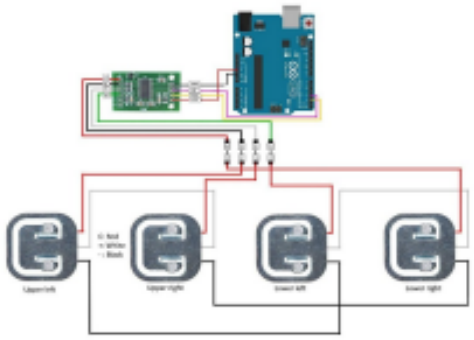

Fig. 1. Circuit Diagram

When a force, such as tension, compression, pressure, or torque, is transformed into a load cell, an electrical signal that can be measured and standardised is generated. Specifically, a force transducer. According to the force on the load cell, the electrical signal changes. It operates under the theory of a "Wheatstone Bridge." The sensor's body is often made of stainless steel or aluminium, which gives it two vital qualities: (1) the capacity to withstand enormous loads; and (2) the flexibility to gently bend and then return to its original shape when the force is removed.

By gently deforming when force (tension or compression) is applied, the metal body acts as a "spring." It returns to its original form unless it is overloaded. Due to the strain gage's changing shape and the flexure's deformation, a Wheatstone Bridge circuit produces a differential voltage fluctuation and electrical resistance. As a result, the voltage change and the physical force applied to the flexure are inversely related.

The database was setup on Firebase. Login authentication was setup on Firebase and database connections was setup and linked to android studio where the android app was developed. The prices of items are stored in the database which is fetched once a day in-case the prices vary. Once the item is detected, the price is calculated based on the weight fetched by the weight sensor. Once the user is starts shopping, they can scan and add items to cart by clicking on the scan icon

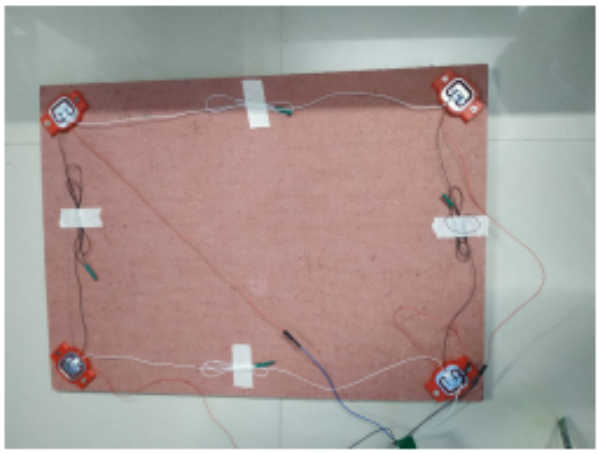

Fig. 2. View of the load cells

Fig. 3. Top View connected with the microcontroller

provided. Upon completion, they can checkout and complete the payment.

Various pages are given on the app which include "User Profile Page" where users can edit their profiles. Instructions on how to use the cart and the app is given in the "Instructions Page". Users can view their past orders in their profile page which shows all the previous orders placed.

V. Experimental Results and observations

For measuring the performance and narrowing down on the ones that gave the best accuracy, graphs were plotted against training steps and accuracy and against loss and training steps. As the training steps increased accuracy of training and validation sets converged to the higher accuracy. The models which gave the best accuracy were EfficientNet, NASNet, and MobileNet-v2. The observations made while training are shared below

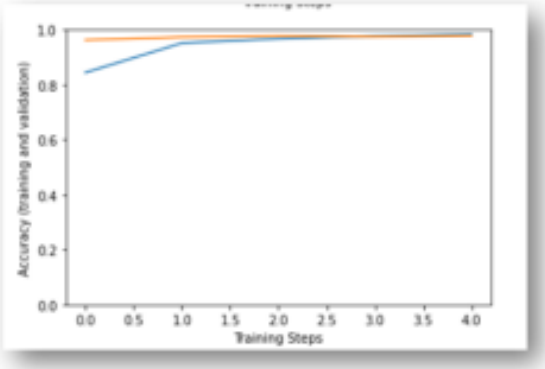

Fig. 4. EfficientNet V2 Training and Validation accuracy

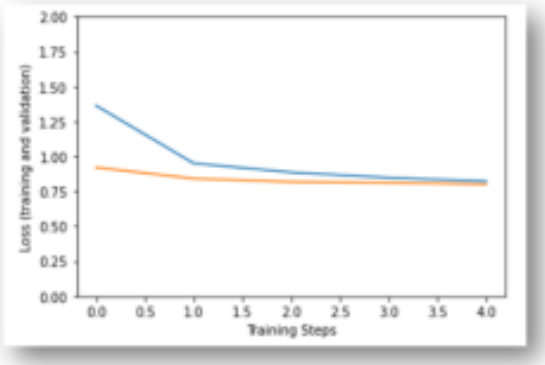

Fig. 5. EfficientNet V2 Training and Validation loss

For better accuracy, models trained over augmented data were also considered and the best three were selected to proceed with Ensemble learning. Each model's accuracy doubles as the weight of its prediction and the final prediction is the weighted average of outputs of all three models.

VI. Conclusion

This paper just put out one way of building a model from scratch using pretrained models for classifying groceries using image recognition. The final ensemble model uses EfficientNet

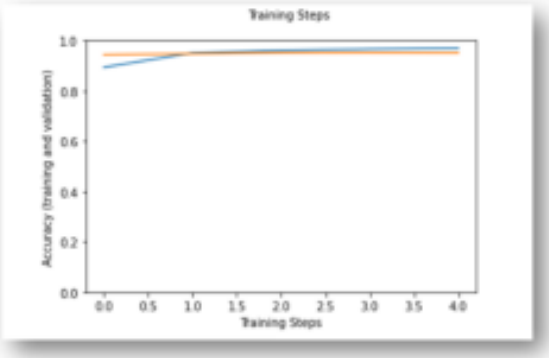

Fig. 6. MobileNet V2 Training and Validation accuracy

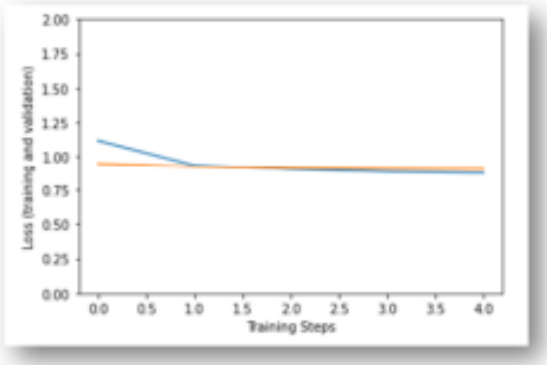

Fig. 7. MobileNet V2 Training and Validation loss

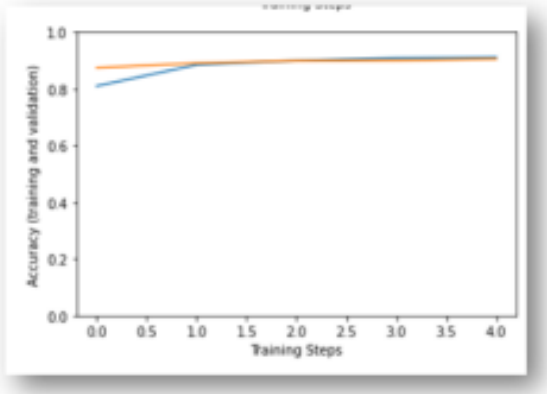

Fig. 8. NasNet Training and Validation accuracy

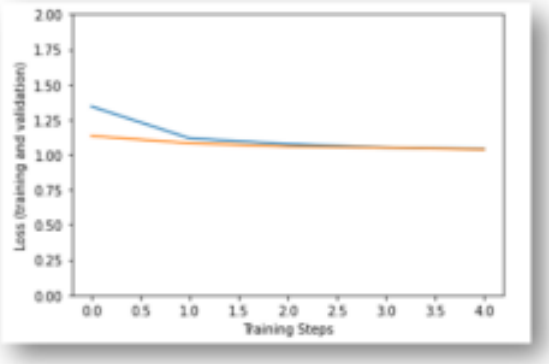

Fig. 9. NasNet Training and Validation loss

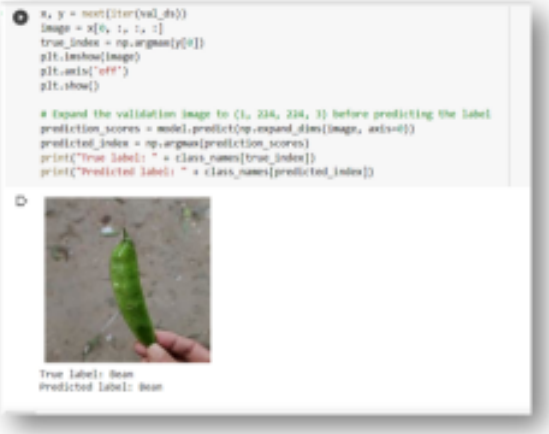

Fig. 10. An example prediction

with augmentation, MobileNet V2 with augmentation and NasNet. MobileNet V2 shows an accuracy of 96.5, EfficientNet shows an accuracy of 97.67 and NasNet shows an accuracy of 96.4. The overall model takes weighted average of out of all three models. With the help of the app built, the shopping experience of the users is seamless, fast and hassle free. The best way to check if the model is working as expected is to give the user option to challenge the output given by the model to ensure that they are correct.

REFERENCES

[1] JL. Rojas-Aranda, JI. Nunez-Varela," Fruit Classification for Retail Stores Using Deep Learning ", Springer, June 2020

[2] M. Klasson, C. Zhang, H. Kjellstrom, "A Hierarchical Grocery Store Image Dataset with Visual and Semantic Labels" Arxiv, January 2020.

[3] A. Zaina, D. Syed," Deployment of Deep Learning Models on Resource-Deficient Devices for Object Detection ",Department of Electrical Computer Engineering Texas AM University College Station, Texas, U.S.A., IEEE, May 2020.

[4] SB. Dubey, GM. Kadam, O. Angane," Android Application for Grocery Ordering System", Irjet, April 2021

[5] K. Riyanti, I. Kakaravada, A.Ahmed, " An Automatic Load Detector Design to Determine the Strength of Pedestrian Bridges Using Load Cell Sensor Based on Arduino ", IJEEEMI, February 2022

Annexure 2

Group No: 75	Title: Automated Shopping Cart Domain: ML in IoT

Abstract: The objective of this project is to make shopping easier by expediting the billing procedure for the products a customer wants to purchase. We design a shopping cart with weight sensors and an android application that automatically recognizes the item by scanning it with a smartphone's camera and determining its weight. A Bluetooth module connected to the Arduino board enables a user to connect to the shopping cart. To measure the product's weight, we utilize a load cell and an Arduino board. The app's machine learning (ML) models trained on a custom dataset with 50 classes and over 150k images are used to classify items. Once classification and weight information are available, a database search is done and the final product price is added to the bill. This would simplify people's life by removing the need to have each item separately scanned and billed at stores and supermarkets, among other things.

Team:

Kuntal Gorai
PES2UG19CS198

S V S C Santosh
PES2UG19CS346

Skanda S
PES2UG19CS391

Vijay Murugan A S
PES2UG19CS454

Supervisor:

Dr.Prajwala T R

Architecture/flow diagram