Page 10 onwards.docx

by Kuntal Gorai

Submission date: 15-Nov-2022 02:47PM (UTC+0530)
Submission ID: 1954635948
File name: Page_10_onwards.docx.pdf (6.38M)
Word count: 12928
Character count: 83902

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 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.

   For our project we have made use of two deep learning models **MobileNetv2** and **EfficentNetv2** .

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. A **bluetooth module HC-05** to help transfer data to our android app and get the weights of items added to the shopping cart.

F. Finally a cloud backend - for which we have used **firebase**, which will serve as a database to store the details about the user information,the items like cost, quantity, and description

available in the supermarket, the order history consisting of orders that the customer had bought at the store.

While the current market offers shopping carts that make use of items with RFID tags we plan to avoid such tedious tasks by making use of image classification with help of deep learning models like MobileNetv2 and EfficentNetv2 to classify the item and we plan to integrate the payment section in our android app which helped customers from waiting in a long queue enabling a hassle-free shopping.

CHAPTER 2

PROBLEM DEFINITION

- With the increasing complexity and time spent on average by a person on purchasing groceries from supermarkets, the number of people coming to the supermarkets has reduced, preferring to purchase the same required items online, compromising on quality and money for time.
- With the increase in number of online customers, these online service providers tend to pay less attention to the quality of items each user is receiving at the end of the day and concentrate more on expanding their user base which is not what happens when a user has the freedom of choosing items on their own thereby not compromising on quality.
- There are three consistent questions that then arise throughout this process:

  1) How to make the process of shopping more **efficient**?

  2) How do we keep track of all **the changes in the price of items** and also the **past orders** of each user?

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

CHAPTER 3

LITERATURE SURVEY

3.1 Fruit Classification for Retail Stores Using Deep Learning

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

Year of Publication: 2020

Link: https://link.springer.com/chapter/10.1007/978-3-030-49076-8_1

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

Advantage:

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

Limitation:

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

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

Authors: Marcus Klasson, Cheng Zhang, Hedvig Kjellstrom

Year of Publication: 2019

Link: https://arxiv.org/abs/1901.00711

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.

Advantages:

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

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.

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

Authors: Ammeema Zaina, Dabeeruddin Syed

Year of Publication: 2020

Link: https://ieeexplore.ieee.org/document/9089651

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.

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.

Limitations:

- Generation of protobuf files is quite challenging.
- Versions of Python and OpenCV that are incompatible.
- Object detection for mobile devices moving at a higher speed.
- Object detection under low illumination.
- Detection of objects in images displayed on a screen or monitor.

3.4 Android Application for Grocery Ordering System

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

Year of Publication: 2021

Summary: https://www.irjet.net/archives/V8/i4/IRJET-V8I4678.pdf

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

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

Advantage:

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

Limitation:

- This application is Android-specific and does not support any iOS devices.

- The UID of all items had to be scanned.

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

Authors: Jamie Duque Domingo , Roberto Medina Aparicio, and Luis Miguel Gonzalez Rodrigo

Date of Publication: 16 February 2022

Link: https://ieeexplore.ieee.org/document/9715066

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.

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

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

CHAPTER 4

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

1. Faster and correct classification of items by our deep learning models implemented.
2. Update of the weight of item added to cart through load cell to our app via bluetooth module HC-05 through arduino.
3. User friendly app to perform hassle free shopping..

4. 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

1. User
    a. Show items to our app to detect it.
    b. Press remove item to remove the item from cart.
    c. Else PressAdd items to our cart containing load cell
    d. Verify the quantity and press add button on arduino
    e. Verify the items with items on the app
    f. Pay the final item through various payment gateways
2. Admin
    a. Make updates to the item based on stock available in mart
    b. Add users to database based on the user details sent to app

4.2.3 Operating Environment

1. Hardware Platform - Mobile device, Arduino, Bluetooth HC-05 module, Load Cell
2. Operating system - Android
3. 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 much 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 bill from firebase.
- The app is robust and is always connected to the Arduino when in use.

10

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

5.5.1 Performance Requirement

Availability

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

Integrity & Safety

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

In terms of data loss and protection, because we use 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 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 wights placed in the cart.

4.5.2 Safety Requirements

- Take bluetooth name of shopping cart from user when connecting 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 from firebase in the app.

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 experience of shopping easier. The proposed device will classify items as and when they are put into the cart and append their net price to the bill accessible via the mobile application for easy payment.

5.2 CURRENT SYSTEM

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

16

5.3 DESIGN CONSIDERATIONS

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

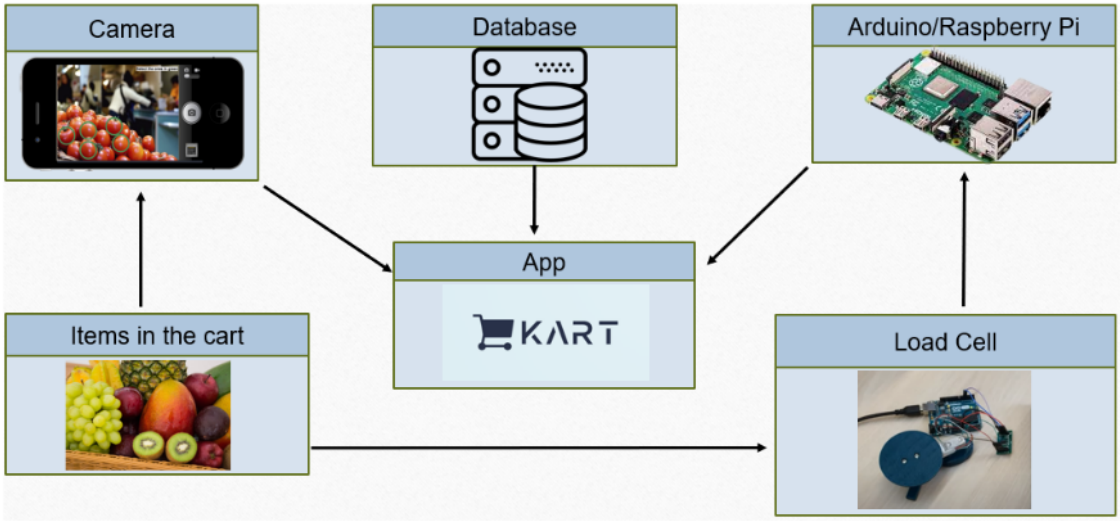


Fig 5.4.1 Data Flow Diagram

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

5.4.2 System Architecture Diagram

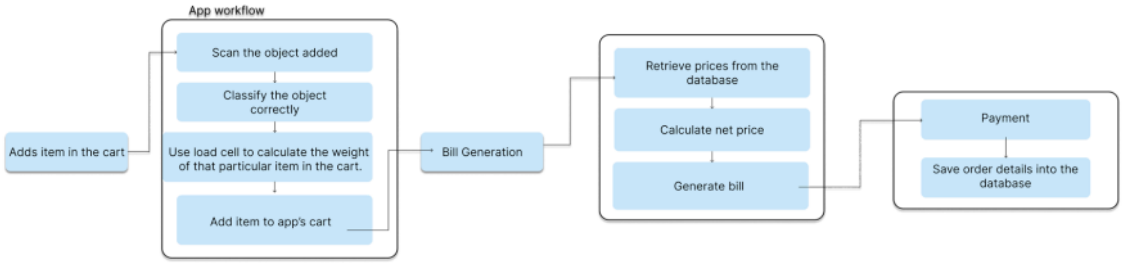


Fig 5.4.2 System Architecture Design

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:

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

5.5.2 Design Description

5.5.2.1 Master Class Diagram

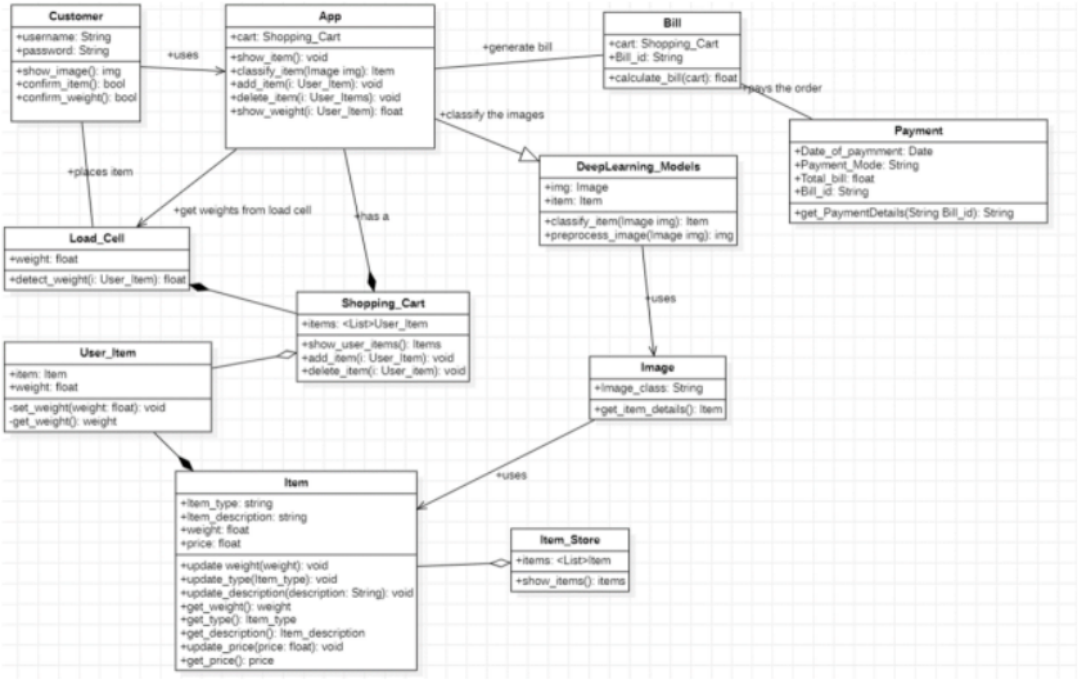


Fig 5.5.2.1: Master Class Digram

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 it to the app. It is composed of a load cell to get the weight of the item that the user is placing in the cart. For classification, the app makes use of a deep learning model which was pre-trained to classify the item. Once it is able to classify the image it links with the item class where we can get all details

about the item. Once all the items are added and the user is done with adding, the app uses the billing class to make the final payment and store the details of payment in our database.

5.5.2.2 UseCase Diagram

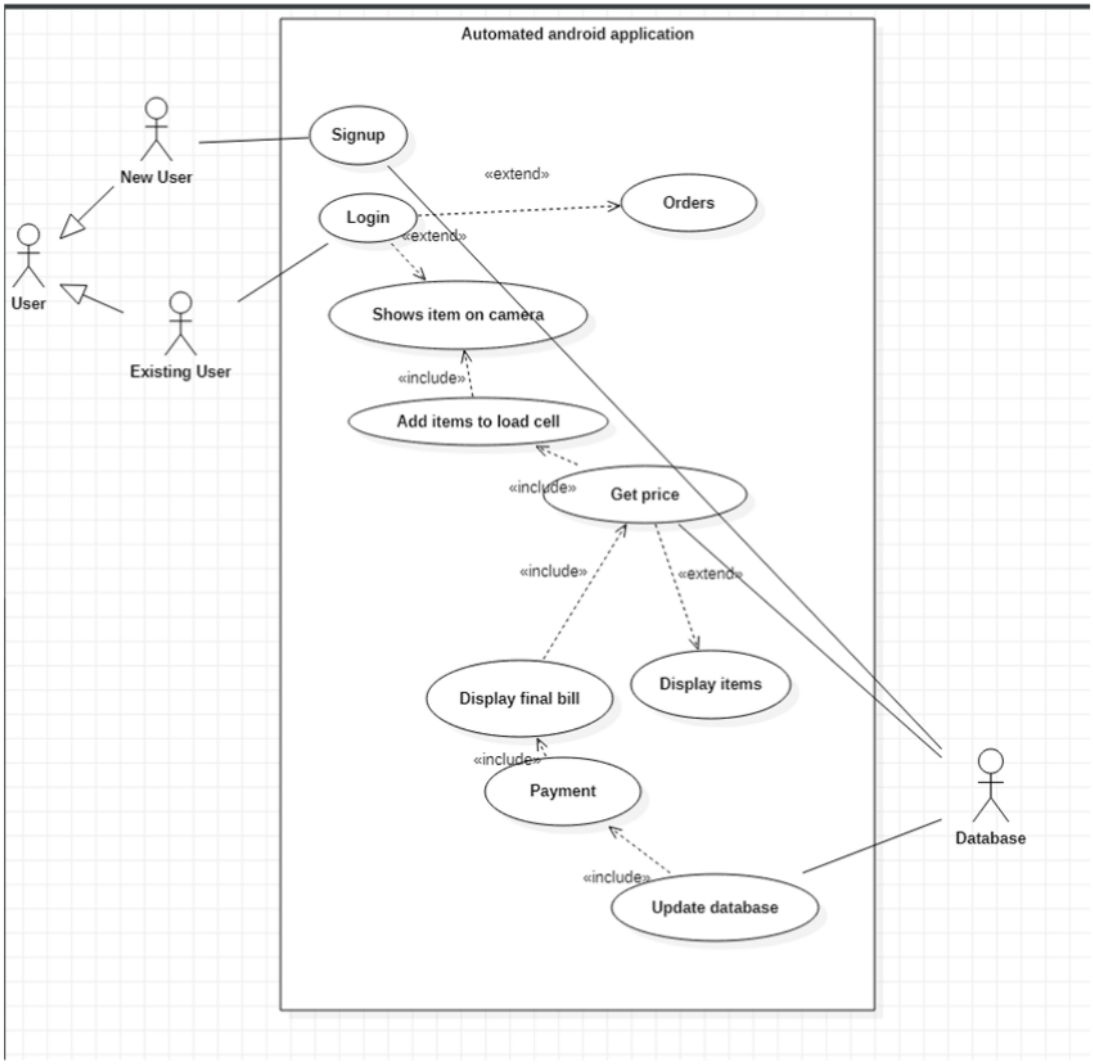


Fig 5.5.2.2: Use Case Digram

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	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

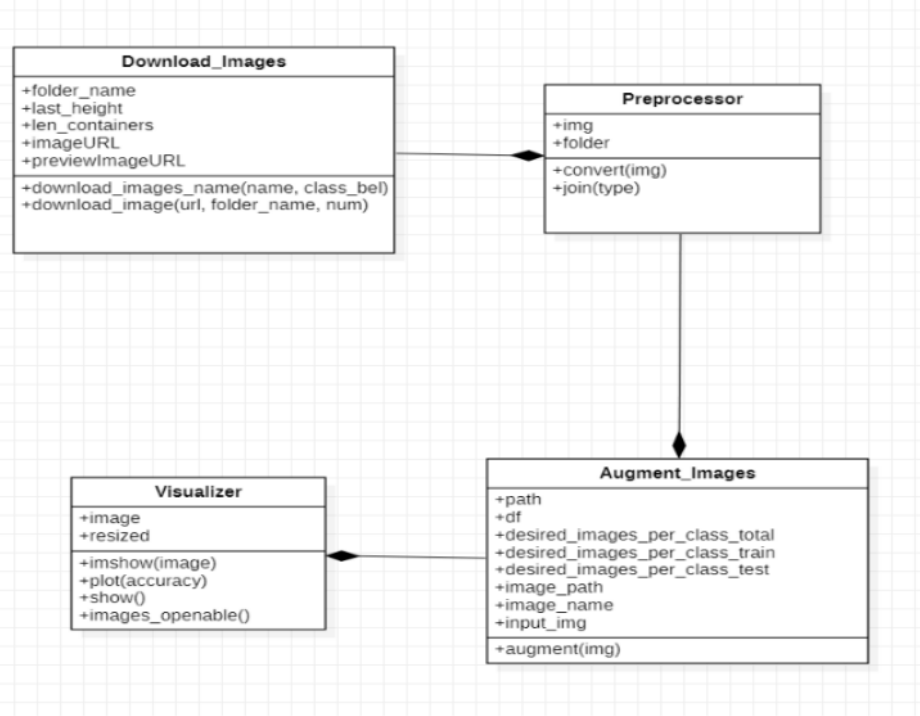


Fig 5.5.3.2: Class Diagram for Building Dataset

5.5.3.3 Class Name 1- Download_Images:

Deals with automation with 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.

i. Data Members 1

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.

ii. Method 1- download_images_name

1) 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
2) Input: Name of the class
3) Output: Saved the particular class of images
4) Parameters: images
5) Exceptions: None
6) Pseudo-code:

    Use selenium and chrome web driver to start the automation process

    Create folders for each class

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

    Download images and write to corresponding files.

iii. Method 2- download_images

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

    Make GET requests and fetch their responses.

    Write the responses converting it to jpeg file in particular folders.

5.5.3.4 Class 2 - Preprocessor

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

i. Data Members

Data Type	Data Name	Access Modifiers	Description
String	folder	public	Contains path of folder with all images
String	img	public	Contains path of image for preprocessing

ii. Method - convert

1. 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.
2. Input: Image path
3. Output: Preprocessed image
4. Parameters: image
5. Exception: None
6. 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.

i. Data Members 3:

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

ii. Method 1 - augment:

1. Purpose: To augment images, name the augmented images and store it in the corresponding folder.
2. Input: Path of the image to be augmented.
3. Output: Augmented image stored.
4. Parameters: image
5. Exceptions: None
6. 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.

i. Data Members 4

Data Type	Data Name	Access Modifiers	Description
String	image	public	Contains path of image before preprocessing

String	resized	public	Contains path of image for after preprocessing

ii. Method 1 - imshow

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

iii. Method 2- plot

1. Purpose: To plot the line graph accuracy of models helping us to find the visual difference in the accuracies of images
2. Input: Accuracy of the models
3. Output: Graph with the accuracy plotted
4. Parameters: Accuracy
5. Exceptions: None
6. Pseudo-code:

Accuracies are given and plotted on the line graph.

iv. Method 3 - images_openable

1. Purpose: Method to find out the number of images that can be opened and displayed
2. Input: file path
3. Output: Number obtained after counting the number of openable images.
4. Parameters: folder path

5. Exceptions: None
6. Pseudo-code: Counter counts the number of images that can be opened through looping until end of file

v. Method 4 - show

1. Purpose: Enables the image to be displayed
2. Input: Image loaded to the plt.imshow method.
3. Output: Image displayed
4. Parameters: image
5. Exceptions: Only openable images can be displayed.
6. Pseudo-code:

   Image is first loaded to the imshow method.

   The loaded image is then displayed.

5.5.4 Module 2- Model Development

5.5.4.1 Description

This Module makes use of the dataset formed containing 153k images of more than 50 classes which is thereby used to predict the class of the image it belongs to. For this we made use of python and various pre-trained Deep Learning Models and used transfer learning to set it to our dataset. Then finally we used ensemble learning to get better accuracy.

5.5.4.2 Class Diagram

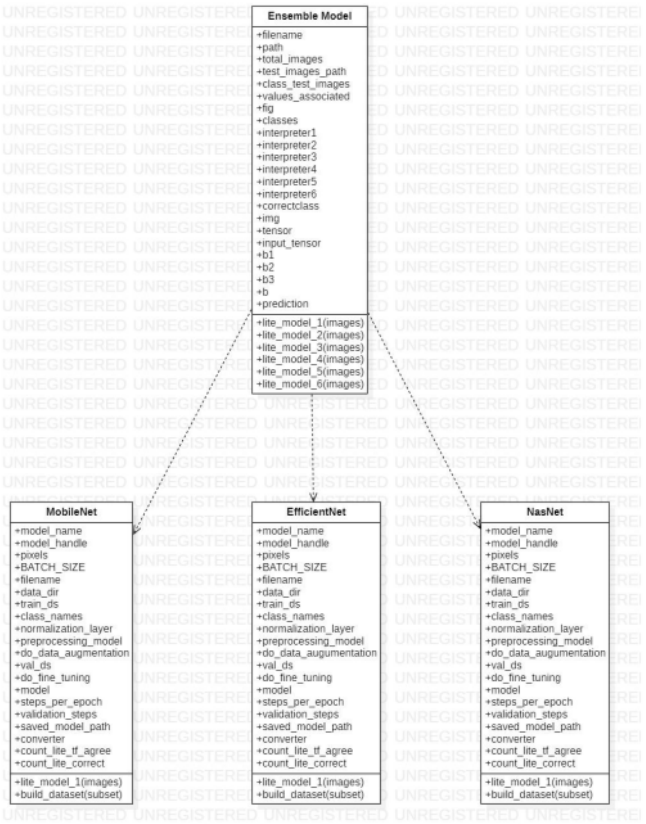


Fig 5.5.4.2: Class Diagram forModel Devolopment

5.5.4.3 Class Name 1 - Ensemble Model

This class Predicts the final class an image clicked by user finally belongs to.

i. Class Description

This Class is used to test the accuracy of each of the models converted into tflite format which can be compatible with our Android Studio App. This model depends on the other 3 classes and then tests the accuracy of each model on the test part of the dataset. Based on that we have 6 different models whose accuracy is obtained and the best 2 models have been chosen for Ensemble Learning. In Ensemble Learning we use the top 2 models thereby giving us better accuracy compared to each of the individual models which is what is used finally to obtain the correct classification of the image

ii. Data members 1.

Data Type	Data Name	Access Modifiers	Description
String	filename	public	Contains path of Zip file of all images
String	path	public	Contains path of Zip file of test images
Integer	total_images	public	Counter to find total test images
Dictionary	test_images_path	public	Contains a list of all the pathnames of each image of a class and the images per class
Integer	class_test_images	public	Counter to keep track of number of images per class

List	values_associated	public	List of all the images of all classes
Image	fig	public	Used to plot the histogram for data visualization
Tuple	classes	public	Contains all the classes which are taken into consideration
Interpreter Object	interpreter1	public	TFLITE Model saved into drive is loaded back here to test accuracy of EfficientNETV2 Non Augmented
Interpreter Object	interpreter2	public	TFLITE Model saved into drive is loaded back here to test accuracy of EfficientNETV2 Augmented
Interpreter Object	interpreter3	public	TFLITE Model saved into drive is loaded back here to test accuracy of MobileNETV2 Non Augmented
Interpreter Object	interpreter4	public	TFLITE Model saved into drive is loaded back here to test accuracy of MobileNETV2 Augmented

Interpreter Object	interpreter5	public	TFLITE Model saved into drive is loaded back here to test accuracy of NasNet Non Augmented
Interpreter Object	interpreter6	public	TFLITE Model saved into drive is loaded back here to test accuracy of NasNet Augmented
Integer	correctclass	public	Counter to see how many images of the test images are correctly classified
Tensor	img	public	Loads each image into a tensor
Tensor	tensor	public	Decodes the input tensor into the 3 RGB pixel
Tensor	input_tensor	public	Resizes and expands the tensor according to shape of input image for trained model
Numpy Array	b1	public	Array of prediction by each the best model
Numpy Array	b2	public	Array of prediction by each the second best model

Numpy Array	b3	public	Array of prediction by each the third best model
Numpy Array	b	public	Array of prediction by combining the best 3 models
String	prediction	public	Final Prediction from ensemble Learning

iii. Method 1- lite_model_1

1) Purpose: Method to reinitialize the Non Augmented Efficient NETV2 Model which has been saved into tflite format for it to be able to predict again
2) Input: Tensor of the input image whose predictions have to be obtained.
3) Output: Return an numpy array of the prediction score it says that the image belongs to a particular class
4) Parameters: images
5) Exceptions: None
6) Pseudo-code:

    Allocate Tensors so that the model works well.

    After Allocating Tensors we can set the value of that tensor to that of the image which can be used for prediction.

    Invoke the loaded Interpreter

    Return the returned predictions.

iv. Method 2- lite_model_2

1) Purpose: Method to reinitialize the Augmented Efficient NETV2 Model which has been saved into tflite format for it to be able to predict again
2) Input: Tensor of the input image whose predictions have to be obtained.
3) Output: Return an numpy array of the prediction score it says that the image belongs to a particular class
4) Parameters: images.
5) Exceptions: None.
6) Pseudo-code:

Allocate Tensors so that the model works well.
After Allocating Tensors we can set the value of that tensor to that of the image which can be used for prediction.
Invoke the loaded Interpreter
Return the returned predictions.

v. Method 3- lite_model_3

1) Purpose: Method to reinitialize the Non Augmented MobileNeV2 Model which has been saved into tflite format for it to be able to predict again.
2) Input: Tensor of the input image whose predictions have to be obtained.
3) Output: Return an numpy array of the prediction score it says that the image belongs to a particular class.
4) Parameters: images.
5) Exceptions: None.

6) Pseudo-code:

Allocate Tensors so that the model works well.

After Allocating Tensors we can set the value of that tensor to that of the image which can be used for prediction.
Invoke the loaded Interpreter
Return the returned predictions.

vi. Method 4- lite_model_4

1) Purpose: Method to reinitialize the Augmented MobileNetV2 Model which has been saved into tflite format for it to be able to predict again.
2) Input: Tensor of the input image whose predictions have to be obtained.
3) Output: Return an numpy array of the prediction score it says that the image belongs to a particular class.
4) Parameters: images.
5) Exceptions: None.
6) Pseudo-code:

Allocate Tensors so that the model works well.
After Allocating Tensors we can set the value of that tensor to that of the image which can be used for prediction.
Invoke the loaded Interpreter
Return the returned predictions.

vii. Method 5- lite_model_5

1) Purpose: Method to reinitialize the Non Augmented NasNet Model which has been saved into tflite format for it to be able to predict again.

2) Input: Tensor of the input image whose predictions have to be obtained.
3) Output: Return an numpy array of the prediction score it says that the image belongs to a particular class.
4) Parameters: images.
5) Exceptions: None.
6) Pseudo-code:

Allocate Tensors so that the model works well.
After Allocating Tensors we can set the value of that tensor to that of the image which can be used for prediction.
Invoke the loaded Interpreter
Return the returned predictions.

viii. Method 6- lite_model_6

1) Purpose: Method to reinitialize the Augmented NasNet Model which has been saved into tflite format for it to be able to predict again.
2) Input: Tensor of the input image whose predictions have to be obtained.
3) Output: Return an numpy array of the prediction score it says that the image belongs to a particular class.
4) Parameters: images.
5) Exceptions: None.
6) Pseudo-code:

Allocate Tensors so that the model works well.
After Allocating Tensors we can set the value of that tensor to that of the image which can be used for prediction.
Invoke the loaded Interpreter
Return the returned predictions.

5.5.4.4 Class Name 2- MobileNet

Use Transfer Learning and retrain the pretrained MobileNet to make it fit to our dataset of our choice.

i. Class Description 2

Use Transfer Learning and Develop MobileNet V2 model using tensorflow. Later the model is saved as it is into our drive desired location. This is then used to convert the model to tflite format which is compatible to be used with our App.

ii. Data Members 2

Data Type	Data Name	Access Modifiers	Description
String	model_name	public	Has name of model being retrained
String	model_handle	public	Link to the pretrained Model available on tfhub
Integer	pixels	public	Input Image pixels
Tuple	BATCH_SIZE	public	Number of training images to be used in one iteration of training.
String	filename	public	Path to the zip file of all images to be extracted to local runtime

String	data_dir	public	Path to the zip file of all tran images
tf.data.Dataset	train_ds	public	Dataset of all images converted into dictionary
Tuple	class_names	public	Represents all the classes used which is obtained from the names of folder in the dataset
tf.keras.Model	normalization_layer	public	Create an normalization layer to convert to values between 0,1
tf.keras.Model	preprocessing_model	public	Add a Normalization Layer as one of the layers in the CNN.
Boolean	do_data_augumentation	public	Indicates if the dataset has to be augmented by some changes
tf.data.Dataset	val_ds	public	Validation Dataset Used for validation
Boolean	do_fine_tuning	public	Used to determine if some fine tuning has to be done on the model

tf.keras.Model	model	public	Adds all the necessary other layers thereby defining the final model used for prediction
Integer	steps_per_epoch	public	Number of iterations to perform to finish one epoch for training
Integer	validation_steps	public	Number of iterations to perform to finish one epoch for Validation
Integer	saved_model_path	public	Path of where the Tensor Flow Model has to be saved to
TensorFlow Lite Model	converter	public	Stores the converted Tensorflow model to tensorflow lite model
Integer	count_lite_tf_agree	public	Number of images agreed by tflite model as with the original tf model
Integer	count_lite_correct	public	Number of correct predictions by the tflite model with respect to the original class.

iii. Method 1- build_dataset(subset)

1) Purpose: Method to Generate Either the Training or Validation set as required from the entire images present in the subfolder of our dataset into a dictionary containing the same.
2) Input: String representing if its the training or validation dataset
3) Output: An tf.data.Dataset based dictionary like object Which is output of conversion of all images into dictionary.
4) Parameters: subset.
5) Exceptions: None.
6) Pseudo-code:

    Call image dataset from dictionary function available in tensorflow

iv. Method 2- lite_model_2

1) Purpose: Method to reinitialize the MobileNet V2 Model which has been saved into tflite format for it to be able to predict again.
2) Input: Tensor of the input image whose predictions have to be obtained.
3) Output: Return an numpy array of the prediction score it says that the image belongs to a particular class.
4) Parameters: images.
5) Exceptions: None.
6) Pseudo-code:

    Allocate Tensors so that the model works well.

After Allocating Tensors we can set the value of that tensor to that of the image which can be used for prediction.
Invoke the loaded Interpreter
Return the returned predictions.

5.5.4.5 Class Name 3- EfficientNet

Use Transfer Learning and retrain the pre-trained EfficientNet to make it fit to our dataset of our choice.

i. Class Description 3

Use Transfer Learning and Develop EfficientNetV2 model using tensorflow. Later the model is saved as it is into our drive desired location. This is then used to convert the model to tflite format which is compatible to be used with our App.

ii. Data Members 3

Data Type	Data Name	Access Modifiers	Description
String	model_name	public	Has name of model being retrained
String	model_handle	public	Link to the pretrained Model available on tfhub
Integer	pixels	public	Input Image pixels
Tuple	BATCH_SIZE	public	Number of training images to be used in one iteration of training.

String	filename	public	Path to the zip file of all images to be extracted to local runtime
String	data_dir	public	Path to the zip file of all tran images
tf.data.Dataset	train_ds	public	Dataset of all images converted into dictionary
Tuple	class_names	public	Represents all the classes used which is obtained from the names of folder in the dataset
tf.keras.Model	normalization_layer	public	Create an normalization layer to convert to values between 0,1
tf.keras.Model	preprocessing_model	public	Add a Normalization Layer as one of the layers in the CNN.
Boolean	do_data_augumentation	public	Indicates if the dataset has to be augmented by some changes
tf.data.Dataset	val_ds	public	Validation Dataset Used for validation

Boolean	do_fine_tuning	public	Used to determine if some fine tuning has to be done on the model
tf.keras.Model	model	public	Adds all the necessary other layers thereby defining the final model used for prediction
Integer	steps_per_epoch	public	Number of iterations to perform to finish one epoch for training
Integer	validation_steps	public	Number of iterations to perform to finish one epoch for Validation
Integer	saved_model_path	public	Path of where the Tensor Flow Model has to be saved to
TensorFlow Lite Model	converter	public	Stores the converted Tensorflow model to tensorflow lite model
Integer	count_lite_tf_agree	public	Number of images agreed by tflite model as with the original tf model

Integer	count_lite_correct	public	Number of correct predictions by the tflite model with respect to the original class.

iii. Method 1- build_dataset(subset)

1) Purpose: Method to Generate Either the Training or Validation set as required from the entire images present in the subfolder of our dataset into a dictionary containing the same.
2) Input: String representing if its the training or validation dataset.
3) Output: An tf.data.Dataset based dictionary like object Which is output of conversion of all images into dictionary.
4) Parameters: subset.
5) Exceptions: None.
6) Pseudo-code:

Call image dataset from dictionary function available in tensorflow

iv. Method 2- lite_model_2

1) Purpose: Method to reinitialize the EfficientNet V2 Model which has been saved into tflite format for it to be able to predict again.
2) Input: Tensor of the input image whose predictions have to be obtained.
3) Output: Return an numpy array of the prediction score it says that the image belongs to a particular class.
4) Parameters: images.

5) Exceptions: None.

6) Pseudo-code:

Allocate Tensors so that the model works well.

After Allocating Tensors we can set the value of that tensor to that of the image which can be used for prediction.

Invoke the loaded Interpreter

Return the returned predictions.

5.5.4.6 Class Name 4- NasNet

Use Transfer Learning and retrain the pretrained NasNet to make it fit to our dataset of our choice.

i. Class Description 4

Use Transfer Learning and Develop NasNet V2 model using tensorflow. Later the model is saved as it is into our drive desired location. This is then used to convert the model to tflite format which is compatible to be used with our App.

ii. Data Members 4

Data Type	Data Name	Access Modifiers	Description
String	model_name	public	Has name of model being retrained
String	model_handle	public	Link to the pretrained Model available on tfhub
Integer	pixels	public	Input Image pixels

Tuple	BATCH_SIZE	public	Number of training images to be used in one iteration of training.
String	filename	public	Path to the zip file of all images to be extracted to local runtime
String	data_dir	public	Path to the zip file of all tran images
tf.data.Dataset	train_ds	public	Dataset of all images converted into dictionary
Tuple	class_names	public	Represents all the classes used which is obtained from the names of folder in the dataset
tf.keras.Model	normalization_layer	public	Create an normalization layer to convert to values between 0,1
tf.keras.Model	preprocessing_model	public	Add a Normalization Layer as one of the layers in the CNN.

Boolean	do_data_augumentation	public	Indicates if the dataset has to be augmented by some changes
tf.data.Dataset	val_ds	public	Validation Dataset Used for validation
Boolean	do_fine_tuning	public	Used to determine if some fine tuning has to be done on the model
tf.keras.Model	model	public	Adds all the necessary other layers thereby defining the final model used for prediction
Integer	steps_per_epoch	public	Number of iterations to perform to finish one epoch for training
Integer	validation_steps	public	Number of iterations to perform to finish one epoch for Validation
Integer	saved_model_path	public	Path of where the Tensor Flow Model has to be saved to

TensorFlow Lite Model	converter	public	Stores the converted Tensorflow model to tensorflow lite model
Integer	count_lite_tf_agree	public	Number of images agreed by tflite model as with the original tf model
Integer	count_lite_correct	public	Number of correct predictions by the tflite model with respect to the original class.

iii. Method 1- build_dataset(subset)

1) Purpose: Method to Generate Either the Training or Validation set as required from the entire images present in the subfolder of our dataset into a dictionary containing the same.
2) Input: String representing if its the training or validation dataset.
3) Output: An tf.data.Dataset based dictionary like object Which is output of conversion of all images into dictionary.
4) Parameters: subset.
5) Exceptions: None.
6) Pseudo-code: Call image dataset from dictionary function available in tensorflow

iv. Method 2- lite_model_2

1) Purpose: Method to reinitialize the MobileNet V2 Model which has been saved into tflite format for it to be able to predict again.
2) Input: Tensor of the input image whose predictions have to be obtained.
3) Output: Return an numpy array of the prediction score it says that the image belongs to a particular class.
4) Parameters: images.
5) Exceptions: None.
6) Pseudo-code:

Allocate Tensors so that the model works well.
After Allocating Tensors we can set the value of that tensor to that of the image which can be used for prediction.
Invoke the loaded Interpreter
Return the returned predictions.

5.5.5 Module 3- App Development

5.5.5.1 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 as a whole. The details about the class diagram and various components of this module are explained below.

5.5.5.2 Class Diagram

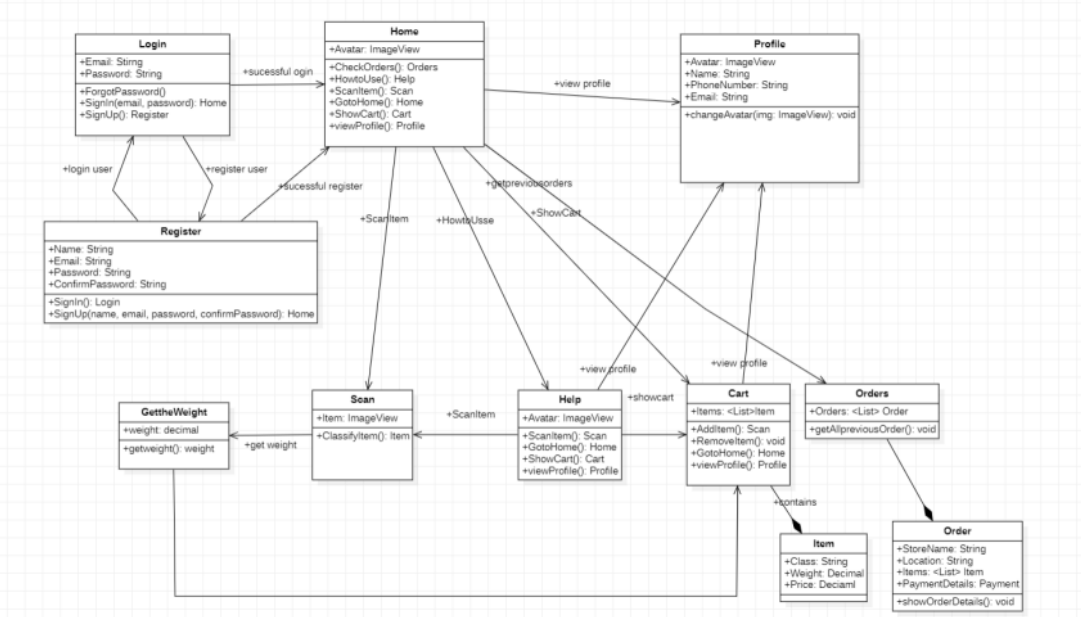


Fig 5.5.3.2: Class Diagram for App Development

5.5.4.3 Class name 1-SignIn

i. Class Description 1

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

ii. Data members 1

Data Type	Data Name	Access Modifiers	Initial Value	Description

String	Email	Private	"	Email input taken from user
String	Password	Private	"	Password taken from user

iii. 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 email id and new password
Parameters	Email
Exceptions	None
Pseudocode	If user doesn't remember password: Add a new password after sending email to user.

iv. Method 2 - SignIn(email,password)

Purpose	It logins user with email and password provide by the user
Input	Email and password of registered user
Output	Opens the homepage of the app with user data.
Parameters	Email and password
Exceptions	None
Pseudocode	If user present in database nd password match: Move to home page Else: Display error message that email or password is incorrect.

v. Method 3- SignUp()

Purpose	It moves the user to sign up page if user selects sign up button
Input	Click on Register button

Output	Opens the register page of the app
Parameters	Register Button
Exceptions	None
Pseudocode	If OnClick of sign up button: Move to register page

5.5.5.4 Class name 2: Register

i. Class Description 2

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

ii. Data members 2

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	“	confirmpassword given by the user

iii. Method 1 - SignIn()

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

iv. Method 2 - SignUp(email,password)

Purpose	It registers user with email and password provided by the user
Input	Email and password entered by the user

Output	successful Registration opens the home page of the app with user data.
Parameters	Email and password
Exceptions	None
Pseudocode	If emailid follows correct format and password and confirmpassword matches: Register the user Move to home page Else: Display error message that email or password is invalid.

5.5.5.5 Class name 3: Home

i. Class Description 3

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

ii. Data members 3

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

iii. Method 1 - checkOrders()

Purpose	It send the user to order page if user clicks the check previous orders button
Input	Click on 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 orders page

iv. Method 2 - HowtoUse()

Purpose	It send the user to help page if user clicks the help icon button
Input	Click on 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 help page

v. Method 3- ScamItem()

Purpose	It opens up the camera if user clicks on the scan icon button and on taking image performs a classification of image.

Input	Click on 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 picture is clicked: Open scan page and execute classify item()

vi. Method 4- GotoHome()

Purpose	It send the user to help page if user clicks the home icon button
Input	Click on Home icon button
Output	Opens the home page of the app
Parameters	Home icon Button
Exceptions	None

Pseudocode	If OnClick of home icon button: Move to home page

vii. Method 5- ShowCart()

Purpose	It shows list of items 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 user has added to the cart

viii. Method 6- viewProfile()

Purpose	It send the user to profile page if user clicks the avatar

Input	Click on 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

5.5.5.6 Class name 4 Profile

i. Class Description 4

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

ii. Data members 4

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 allows tuser to change avatar on clicking of avatar edit button on profile page**
Input	**Click on avatar icon edit button**
Output	**Changes the avatar of the user**
Parameters	**Avatar icon edit button**
Exceptions	**None**
Pseudocode	**If OnClick of avatar icon edit button:** **Select image form file(png)** **Change avatar of the user.**

5.5.5.7 Class name 5 Help

i. Class Description 5

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

ii. Data members 5

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

iii. Method 1- ScamItem()

Purpose	It opens up the camera if user clicks on the scan icon button and on taking image performs a classification of image.
Input	Click on 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 picture is clicked: Open scan page and execute classify item()

iv. Method 2- GotoHome()

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

v. Method 3- ShowCart()

Purpose	It shows list of items 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 user has added to the cart

vi. Method 4 viewProfile()

Purpose	It send the user to profile page if user clicks the avatar
Input	Click on 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

5.5.5.8 Class name 6 Cart

i. Class Description 6

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

ii. Data members 6

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

iii. Method 1 - AddItem()

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

iv. Method 2 - RemoveItem(item)

Purpose	It removes the item from user cart after getting the weights form shopping cart.
Input	Click on remove item button
Output	remove item to the cart
Parameters	Remove item button in scan page
Exceptions	None
Pseudocode	If OnClick of removeitem button: remove item to the cart

v. Method 3- GotoHome()

Purpose	It send the user to help page if user clicks the home icon button
Input	Click on Home icon button
Output	Opens the home page of the app

Parameters	Home icon Button
Exceptions	None
Pseudocode	If OnClick of home icon button: Move to home page

vi. Method 4- viewProfile()

Purpose	It send the user to profile page if user clicks the avatar
Input	Click on 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

5.5.5.9 Class name 7 Orders

i. Class Description 7

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

ii. Data members 7

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

iii. Method 1 - getPreviousOrderDetails()

Purpose	It shows list of previous order fetched fro database for current user if user clicks the previous order button
Input	Click on previous order button
Output	Shows a list of all previous order done by the user
Parameters	Previous order button
Exceptions	None

Pseudocode	If OnClick of previous order button: Show list of previous order done by the user.

5.5.5.10 Class name 8 scan

i. Class Description 8

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

ii. Data members 8

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 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 item

5.5.5.11 Class name 9 GetTheWeight

i. Class Description 9

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

ii. Data members 9

Data Type	Data Name	Access Modifiers	Initial Value	Description
decimal	weight	Private	"	weight detected by the weight senssor

iii. Method 1 - getweight()

Purpose	It detects the weight sensed by the weight sensor.
Input	Item placed on shopping cart
Output	Weight detected by weight sensor
Parameters	None

Exceptions	None
Pseudocode	If OnChange in weight sensor: Return the weight detected by weight sensor.

5.5.5.12 Class name 10 Item

i. Class Description 10

It contains different details for item added to the shopping cart

ii. Data members 10

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 wight sensor

5.5.5.13 Class name 11 Order

i. Class Description 11

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

ii. Data members 11

Data Type	Data Name	Access Modifiers	Initial Value	Description
String	Storename	Private	“	Storename 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 form database
Parameters	None
Exceptions	None
Pseudocode	If onClick on one of previous orders: Display list of selected order

5.5.6 Sequence Diagram

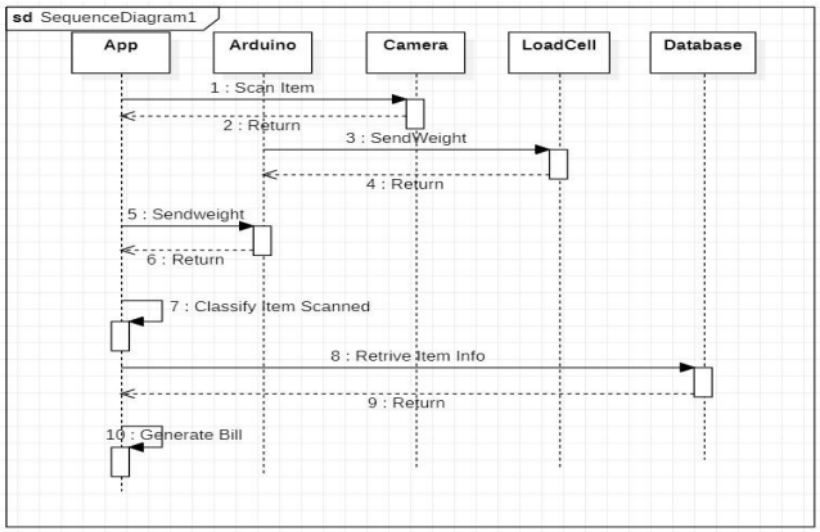


Fig 5.5.6: sequence Diagram

5.5.7 Packaging and Deployment Diagrams

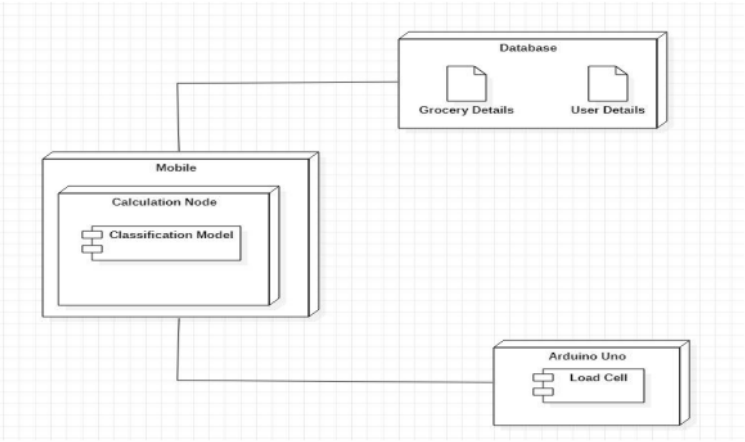


Fig 5.5.7: Deployment Digram

5.5.8 Module 4 - IoT Components

5.5.8.1 Description:

This module deals with the hardware that includes the loadcell, the arduino board and the microcontroller that acts as a bridge between the two. The wight 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.

5.5.8.2 Class Diagram:

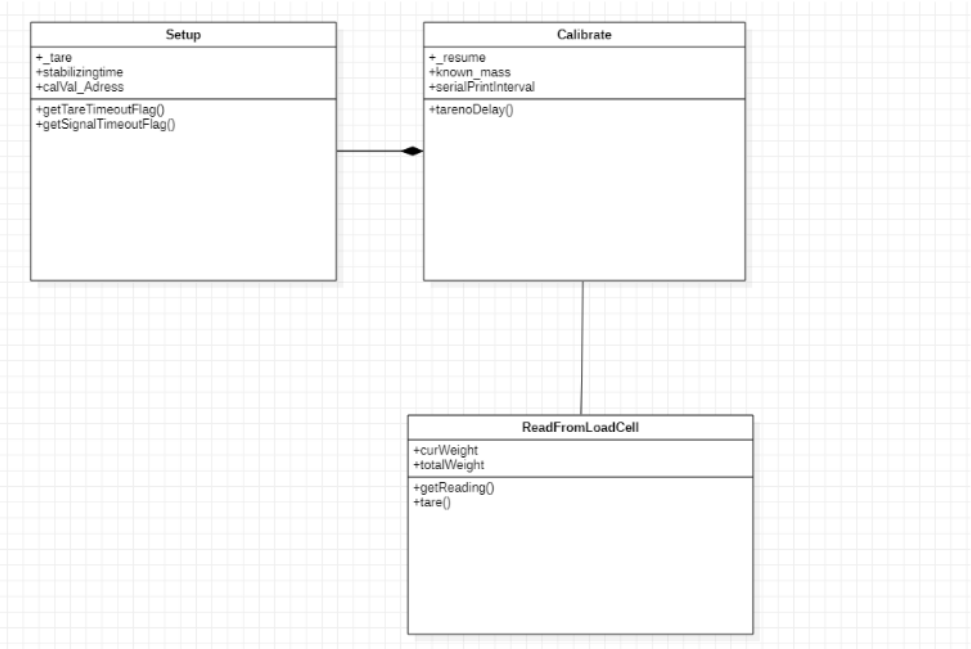


Fig 5.5.8.2: Class Diagram for IoT Components

5.5.8.3 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.

Data Type	Data Name	Access Modifiers	Initial Value	Description
Boolean	_resume	Private	“	ensures updation from loadcell
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.

5.5.8.4 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.

i. Data Members

Data Type	Data Name	Access Modifiers	Initial Value	Description
float	curWeight	Private	“0	holds most recently read value from load cell controller
float	totalWeigh t	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:

1. 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.
2. 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.
3. Once the model was trained, it was converted to a .tflite file compatible with android.
4. With the .tflite model available, we performed an ensemble technique for better classification in our android app.
5. On getting an accurate classification we moved on to making various pages of our android app.
6. We implemented the login page, home page, help page, and scan page.
7. Next we moved on to get the weights of an item classified with the help of a load cell.
8. 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.
9. 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.
10. After data was received in our Arduino IDE we moved to receive our data in our android app.

11. Once we were able to receive data we then used some timer and difference of weight > 50 gm to get the final weight of the item being placed.
12. 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.
13. The main part of the database included a user table storing the details of the user with the previous order history, and an item table storing details of an item.
14. Once database design was done we moved on to finish the remaining pages of the app i.e the profile page, the orders page, the cart page, etc.

CHAPTER 7

IMPLEMENTATION

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

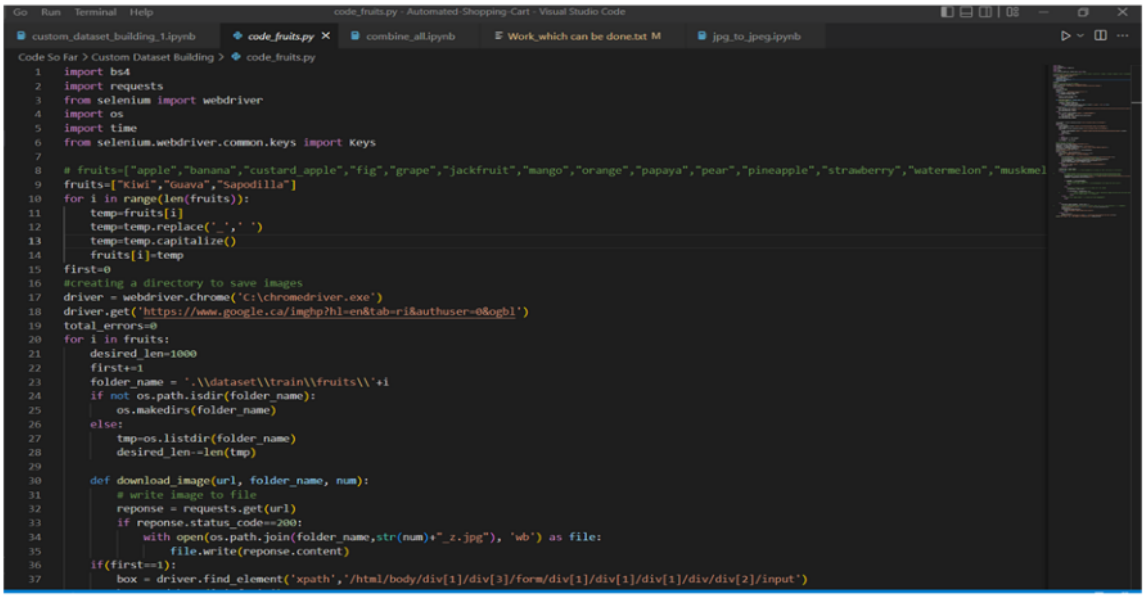


Fig.7.1: Code for building of dataset.

2. 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 7.2.1: Code for building efficentnet_v2 model with the custom dataset built.

```
[ ] 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 7.2.1: Code for building mobilenet_v2 model with the custom dataset built.

3. Implementation of connection of arduino with load cell and bluetooth for wireless communication with the app.

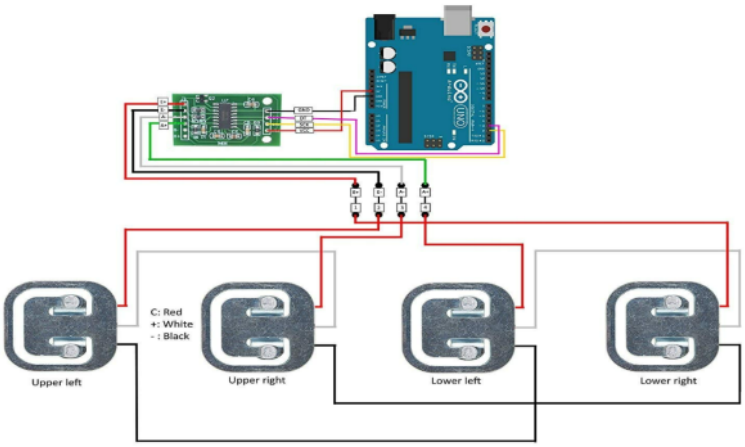


Fig 7.3.1: Iot components connection

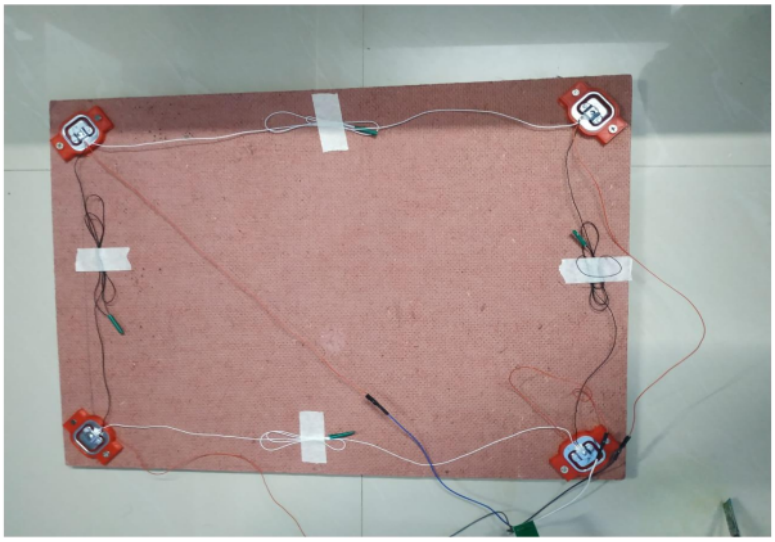
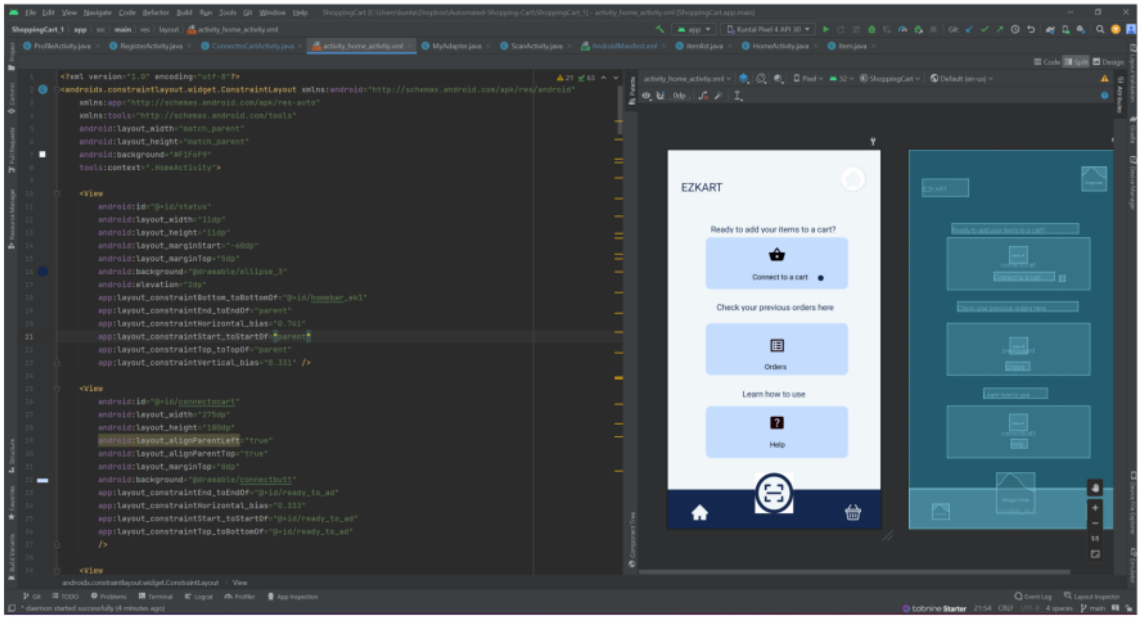


Fig 7.2.2: Actual implementation of load cell in a rigid cardboard(back view)

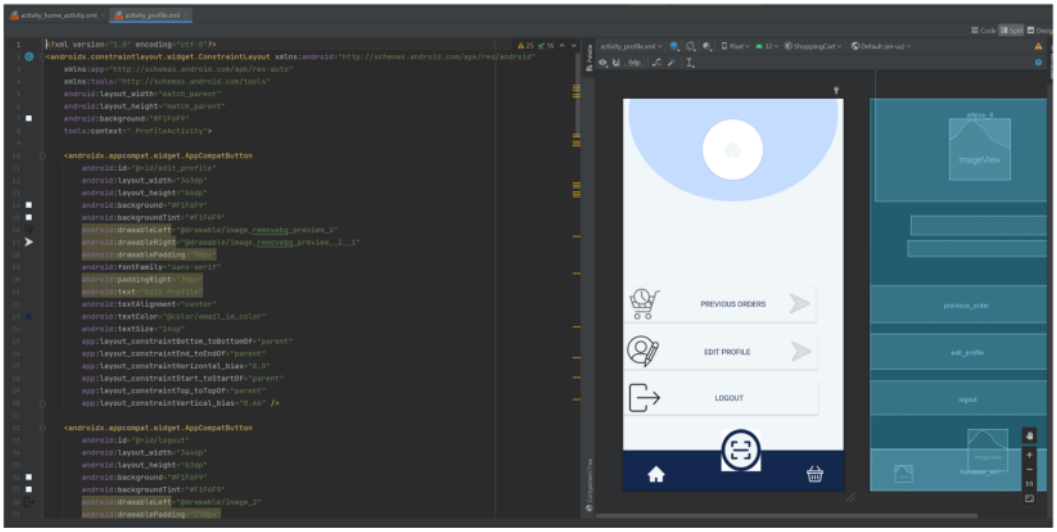
Fig 7.2.3: Arduino with bluetooth and load cell connection powered up by battery

Fig 7.2.4: Front view of the weight weight sensor base.

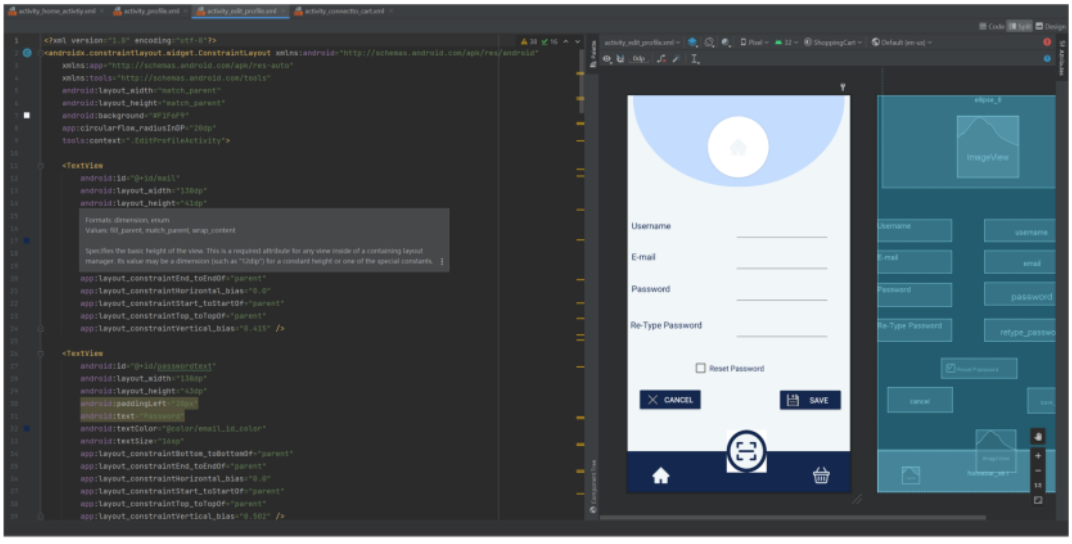
4. Implementation of various pages of our app
    a. Home Page



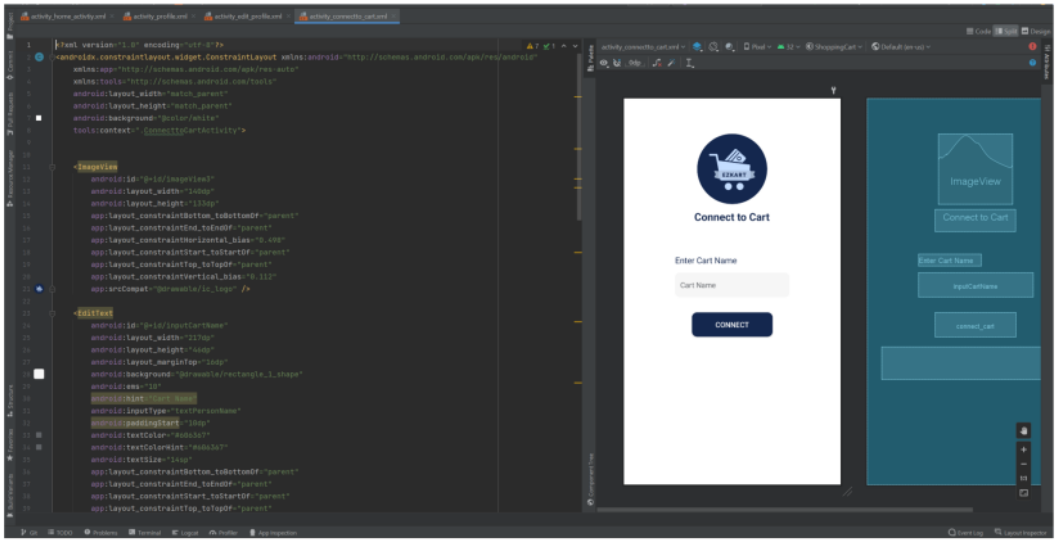
b. Profile Page



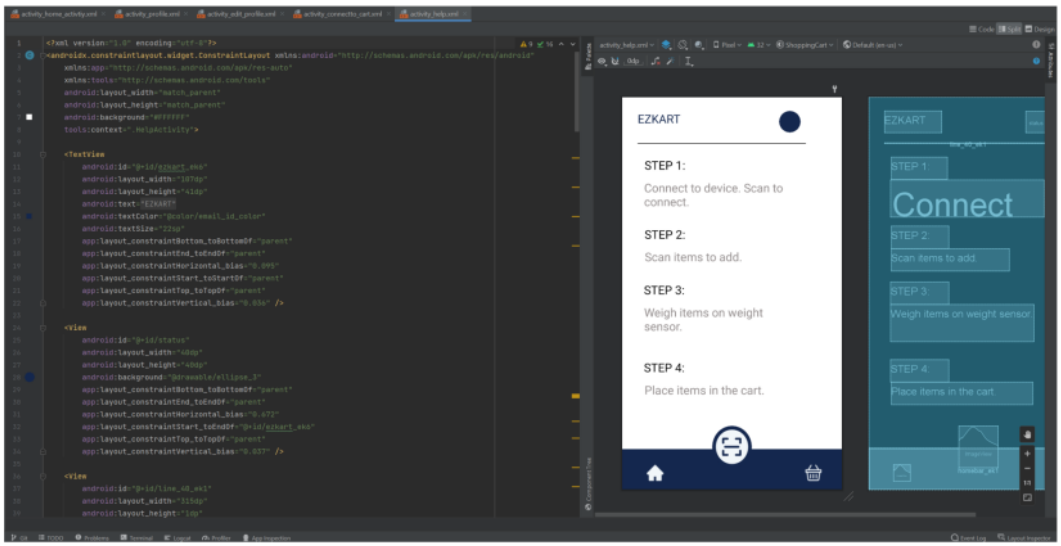
c. EditprofilePage



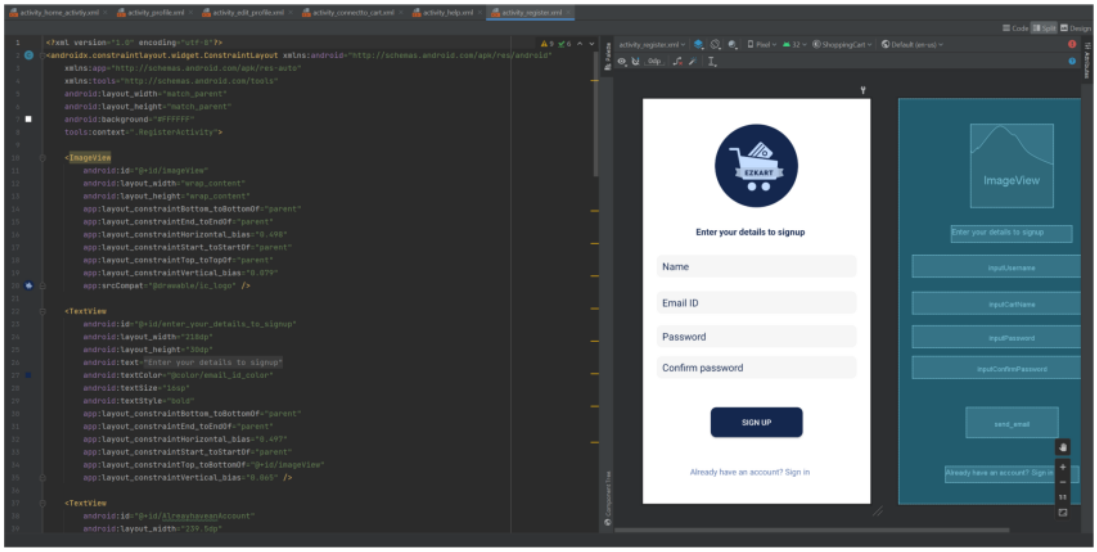
d. ConnecttoCartPage



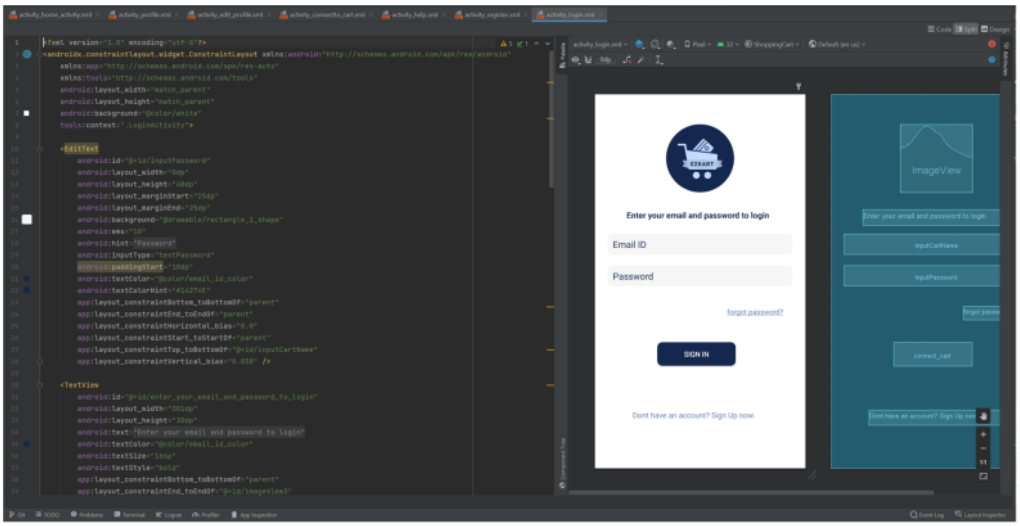
e. HelpPage



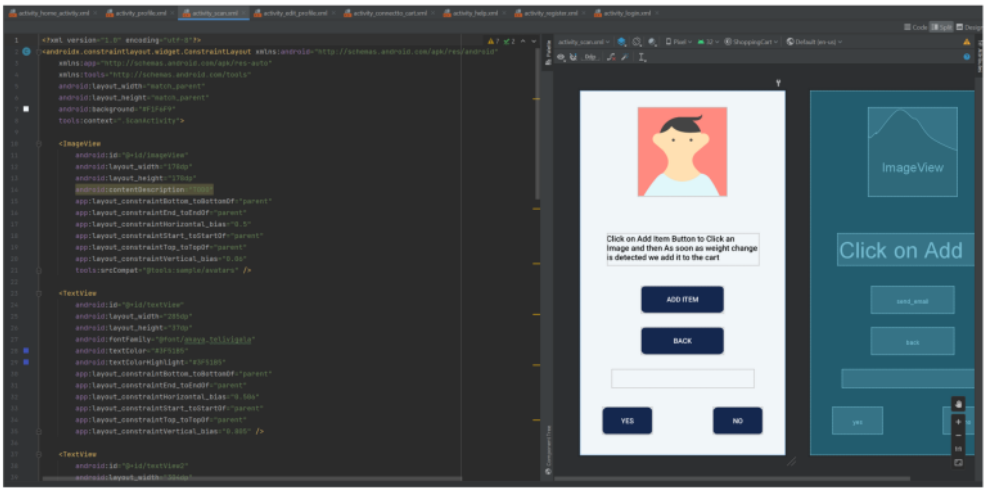
f. RegisterPage



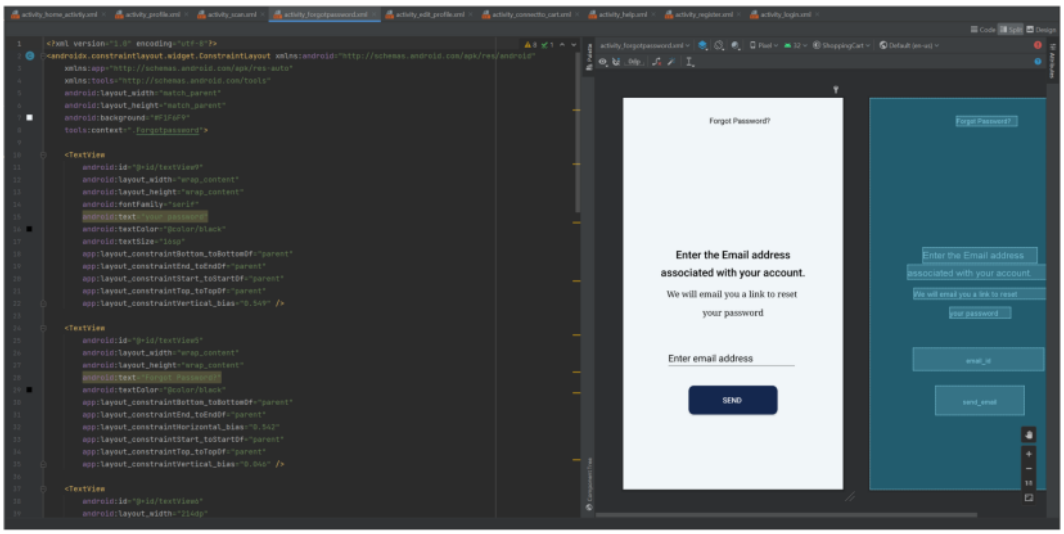
g. LoginPage



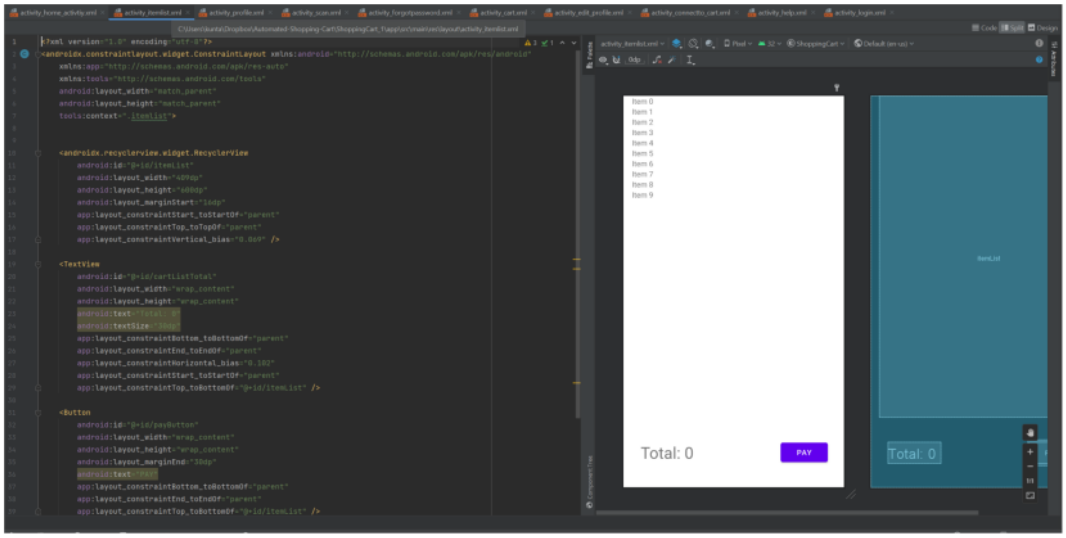
h. ScanPage



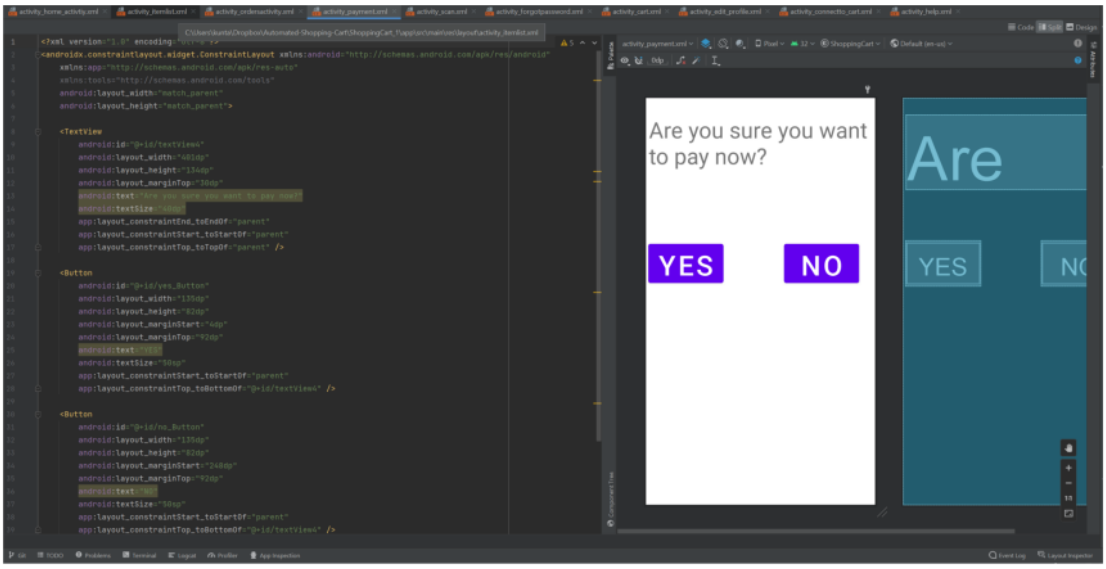
i. ForgotPasswordPage



j. ItemListPage



k. Payments Page



CHAPTER 8

RESULTS AND DISCUSSION

1. 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.

Test_All_Models.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
#----------------------------EFFICIENTNETV2 AUGMENTATION(4TH)-----------------------
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_5(input_tensor)[0])]
      if(prediction==i):
        correctclass+=1
    except:
      # print(j,"Causes Error")
      pass
print("EfficientNet(Augmented) Number of Correct Predictions is ",correctclass," out of ",total_images)
print("EfficientNet(Augmented) Accuracy of testing is ",correctclass/total_images)
```

```
EfficientNet(Augmented) Number of Correct Predictions is  17727  out of  18383
EfficientNet(Augmented) Accuracy of testing is  0.9643148561170647
```

Fig 8.1.1: 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

```
[ ] #-----------------------EFFICENTNETV2 NON AUGMENTATION (1ST)----------------------------
    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_1(input_tensor)[0])]
          if(prediction==i):
            correctclass+=1
        except:
          print(j,"Causes Error")
          pass
    print("EfficientNetV2(Non Augmented) Number of Correct Predictions is ",correctclass," out of ",total_images)
    print("EfficientNetV2(Non Augmented) Accuracy of testing is ",correctclass/total_images)
```

```
EfficientNetV2(Non Augmented) Number of Correct Predictions is  14427  out of  18383
EfficientNetV2(Non Augmented) Accuracy of testing is  0.78480117499864
```

Fig 8.1.2: 78% Accuracy achieved on random test data for efficentnet with non 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 8.1.3: 96% Accuracy achieved on random test data for mobilenetv2 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 8.1.4: 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

```
[ ] #-------------------------------NASNET WITHOUT AUGMENTATION(5TH)------------------
    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_3(input_tensor)[0])]
          # print("prediction is",prediction,"class is",i )
          if(prediction==i):
            correctclass+=1
        except:
          # print(j,"Causes Error")
          pass
    print("NasNetV2(Non Augmented) Number of Correct Predictions is ",correctclass," out of ",total_images)
    print("NasNetV2(Non Augmented) Accuracy of testing is ",correctclass/total_images)
```

```
NasNetV2(Non Augmented) Number of Correct Predictions is  13071  out of  18383
NasNetV2(Non Augmented) Accuracy of testing is  0.7110373714845237
```

Fig 8.1.5: 71% Accuracy achieved on random test data for nasnet 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 8.1.6: 98% Accuracy achieved on random test data for ensemble of top 3 accurate models.

2. Execution of the various pages of the app.

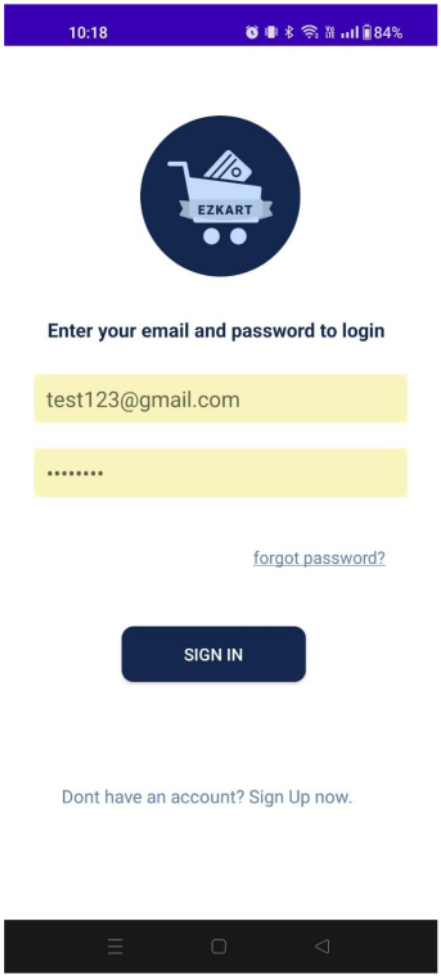


Fig 8.2.1: Login with email & Password

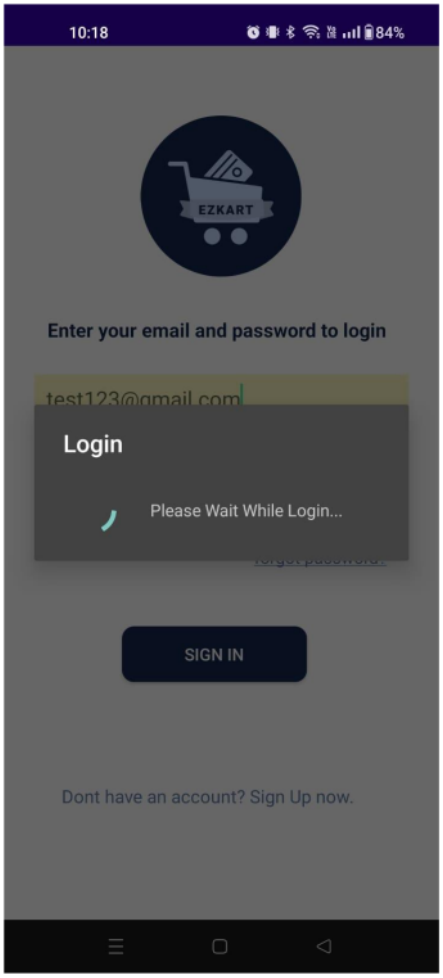


Fig 8.2.2: Logging In

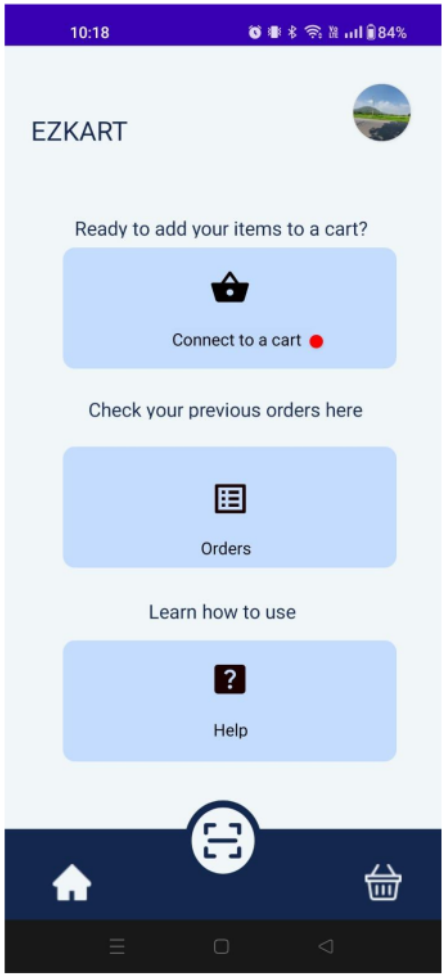


Fig. 8.2.3: Home Page with bluetooth not connected shown by red circle

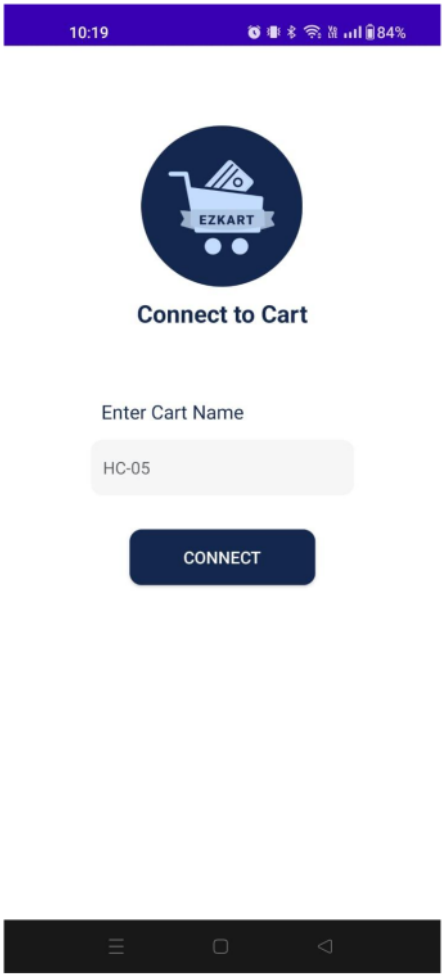


Fig 8.2.4: Bluetooth connection Page

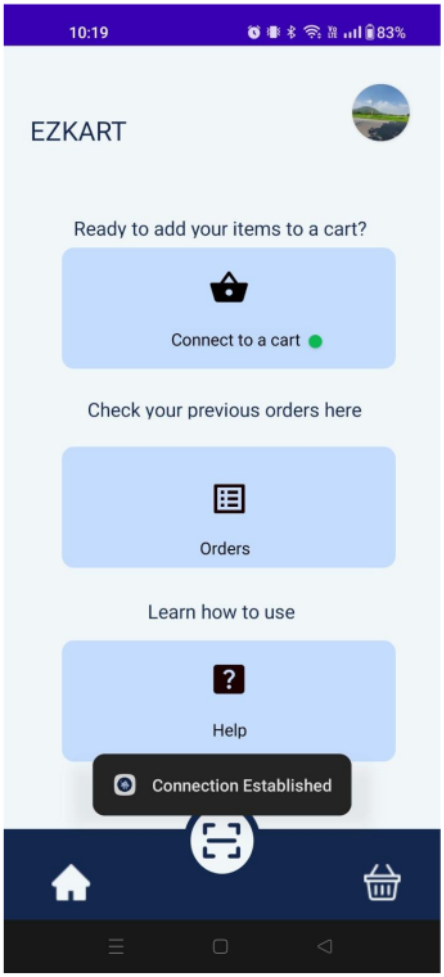


Fig. 8.2.5: Home Page with successful

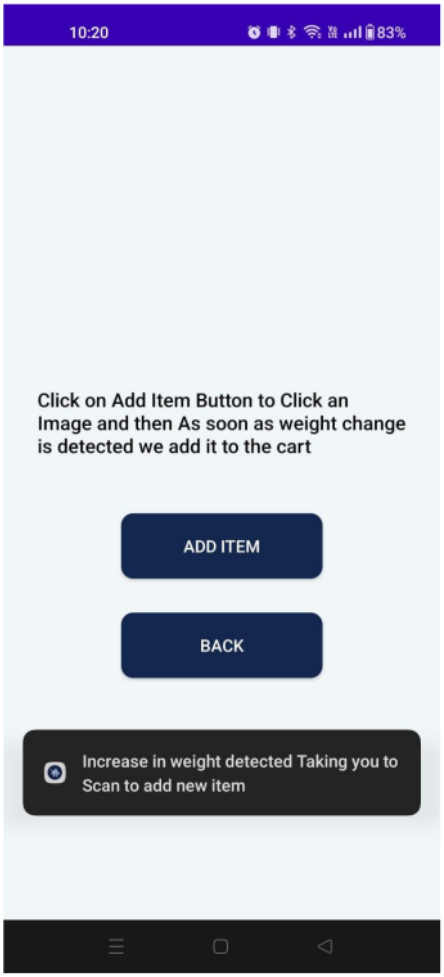


Fig. 8.2.6: Scan Page redirection because of

connection of bluetooth.

change in weight detected by load cell.

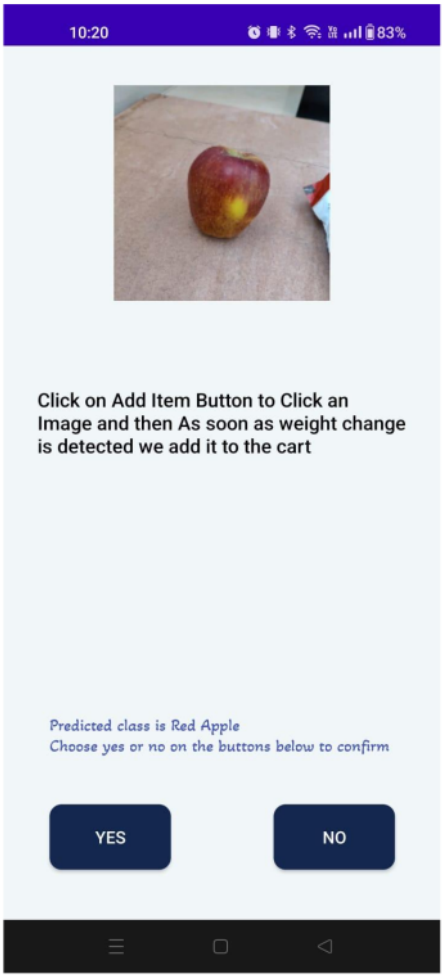


Fig 8.2.7: Prediction page after scanning the item with a confirmation option.

Fig 8.2.8: Cart Page addition of item and weight after confirming yes.

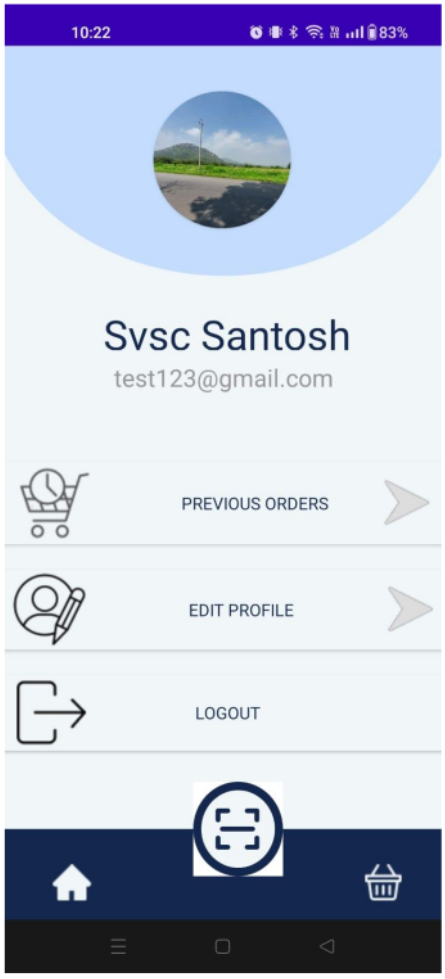


Fig 8.2.9: Profile Page

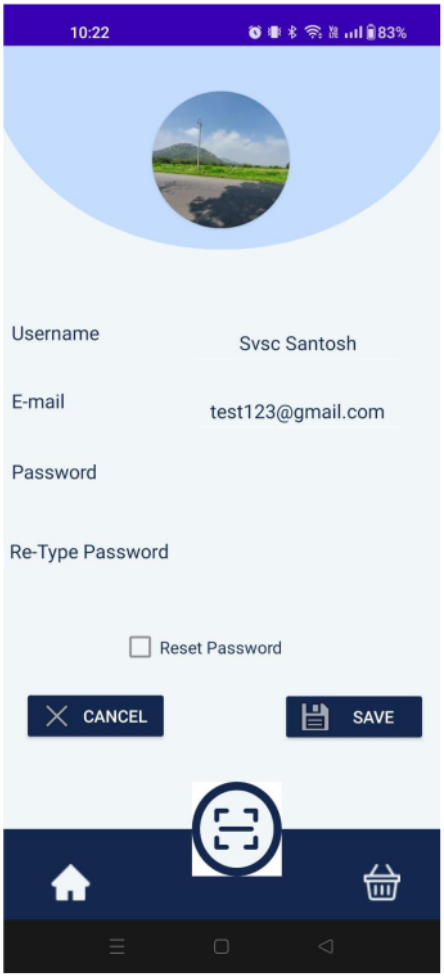


Fig 8.2.10. Edit profile Page

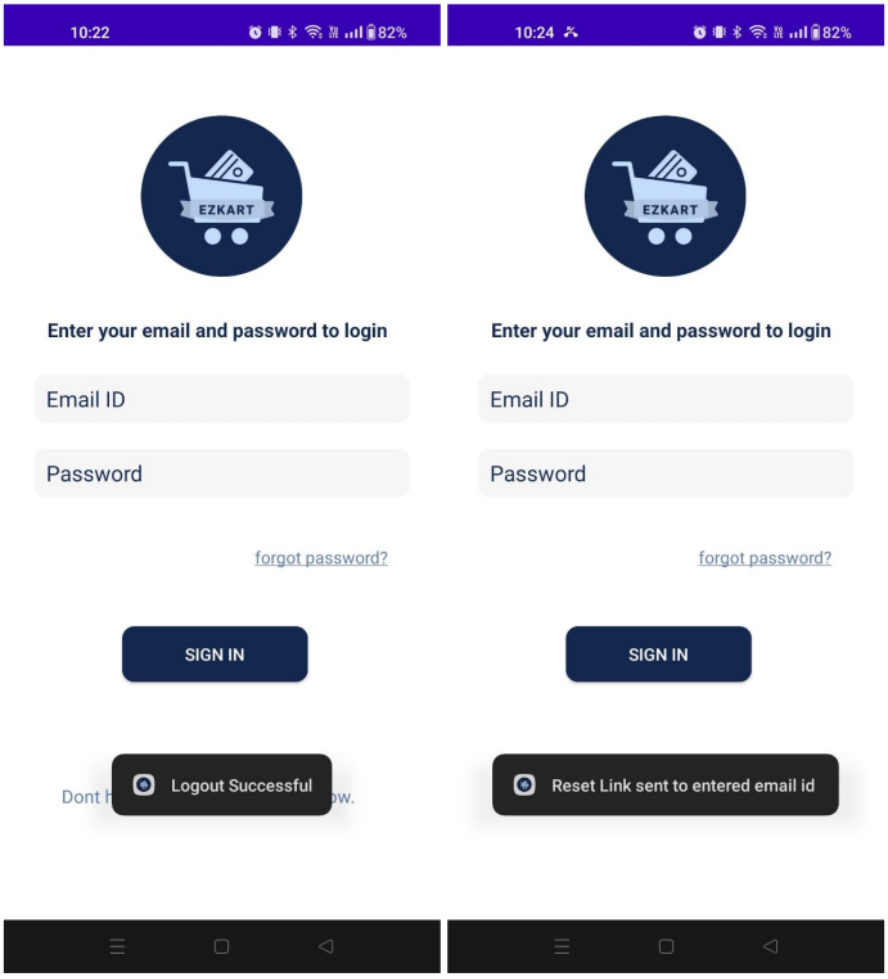


Fig 8.2.11: Successful Logout

Fig 8.2.10. Reset link for forgot password.

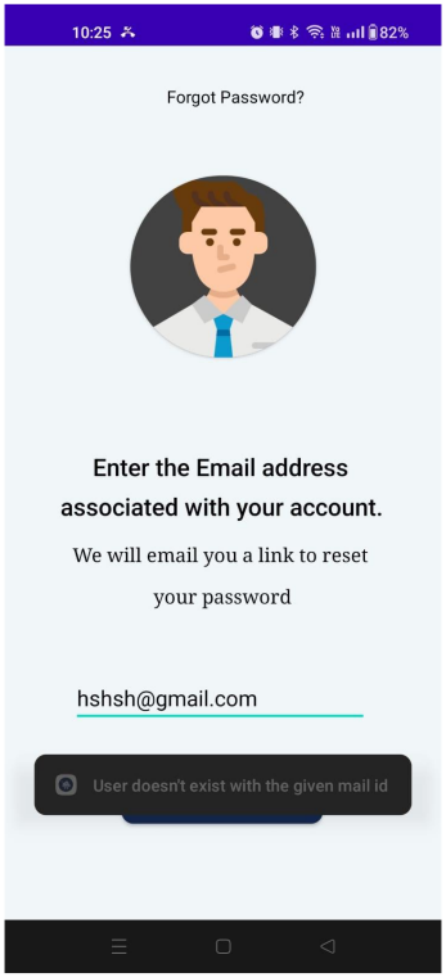

10:25
82%
Forgot Password?
Enter the Email address associated with your account.
We will email you a link to reset your password
hshsh@gmail.com
User doesn't exist with the given mail id

Fig 8.2.13. 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 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 are

- Improvements over classifying the certain items that were not able to classify properly.
- 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/BIBLIOGRAPHY

1) **A Hierarchical Grocery Store Image Dataset with Visual and Semantic Labels**

Authors : Marcus Klasson, Cheng Zhang, Hedvig Kjellstrom

Link: https://arxiv.org/abs/1901.00711

Year: 2019

2) **Fruit Classification for Retail Stores Using Deep Learning**

Authors : Jose Luis RojasAranda(B), Jose Ignacio Nunez-Varela, J. C. Cuevas-Tello, and Gabriela RangelRamirez

Link: https://link.springer.com/chapter/10.1007/978-3-030-49076-8_1

Year : 2020

3) **Deployment of Deep Learning Models on Resource-Deficient Devices for Object Detection**

Authors: Ameena Zaina , Dabeeruddin Syed

Link: https://ieeexplore.ieee.org/document/9089651

Year: 2020

4) **Android Application for Grocery Ordering System**

Authors: Shubham B Dubey1, Gaurav M Kadam2, Omkar Angane3

Link: https://www.irjet.net/archives/V8/i4/IRJET-V8I4678.pdf

Year: 2021

5) **Cross Validation Voting for Improving CNN Classification in Grocery Products**

Authors: Jamie Duque Domingo , Roberto Medina Aparicio, and Luis Miguel Gonzalez Rodrigo

Link: https://ieeexplore.ieee.org/document/9715066

Year: 2022

6) https://ieeexplore.ieee.org/abstract/document/9089651
7) https://www.javatpoint.com/uml-class-diagram
8) https://www.uml-diagrams.org/deployment-diagrams-overview.html
9) https://medium.com/@mobindustry/how-to-integrate-machine-learning-into-an-android-app-best-image-recognition-tools-1ba837c27903
10) https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/blob/master/Raspberry_Pi_Guide.md
11) https://www.semanticscholar.org/paper/Implementation-of-Smart-Shopping-Cart-using-Object-Oh-Chun/f5ff8985a06e02715dbbbbf4f2b4470289e003f0
12) https://analyticsindiamag.com/deploying-machine-learning-models-in-android-apps-using-python/
13) https://eloquentarduino.github.io/2020/01/image-recognition-with-esp32-and-arduino/
14) 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.
15) https://forum.arduino.cc/t/password-protect-your-project/494551
16) https://github.com/antonnifo/fruits-360/blob/master/Fruit%20Classifer.ipynb
17) https://images.cv/category
18) https://link.springer.com/article/10.1007/s12652-020-01865-8
19) https://ieeexplore.ieee.org/abstract/document/9089651

20)https://www.javatpoint.com/uml-class-diagram

21)https://www.uml-diagrams.org/deployment-diagrams-overview.html

22)https://medium.com/@mobindustry/how-to-integrate-machine-learning-into-an-android-app-best-image-recognition-tools-1ba837c27903

23)https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/blob/master/Raspberry_Pi_Guide.md

24)https://www.semanticscholar.org/paper/Implementation-of-Smart-Shopping-Cart-using-Obje ct-Oh-Chun/f5ff8985a06e02715dbbbbf4f2b4470289e003f0

25)https://analyticsindiamag.com/deploying-machine-learning-models-in-android-apps-using-python/

26)https://eloquentarduino.github.io/2020/01/image-recognition-with-esp32-and-arduino/

27)https://firebase.google.com/docs/ml#:~:text=With%20Firebase%20ML%20and%20AutoML,to%20recognize%20concepts%20in%20photographs.&te

APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS

Definitions:

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

Abbreviations:

1) DL: Deep Learning
2) ML: Machine Learning
3) VAE: Variational AutoEncoder
4) CNN: Convolutional Neural Network
5) GPU: Graphical Processing Unit
6) UI: User Interface
7) RFID: Radio Frequency identification

8) YOLO: You Only Look Once

9) VGG: Visual Geometry Group

10) ResNet: Residual Neural Network

Page 10 onwards.docx

ORIGINALITY REPORT

3%	2%	1%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Jaime Duque Domingo, Roberto Medina Aparicio, Luis Miguel Gonzalez Rodrigo. "Cross Validation Voting for Improving CNN Classification in Grocery Products", IEEE Access, 2022 Publication	1%
2	Submitted to International University - VNUHCM Student Paper	<1%
3	Submitted to Southampton Solent University Student Paper	<1%
4	dokumen.pub Internet Source	<1%
5	export.arxiv.org Internet Source	<1%
6	Submitted to King's Own Institute Student Paper	<1%
7	researchr.org Internet Source	<1%

8	www.jatit.org Internet Source	<1%
9	Submitted to University of Warwick Student Paper	<1%
10	csis.pace.edu Internet Source	<1%
11	ijarcce.com Internet Source	<1%
12	www.coursehero.com Internet Source	<1%
13	www.researchgate.net Internet Source	<1%
14	Gopal S. Tandel, Ashish Tiwari, O.G. Kakde. "Performance optimisation of deep learning models using majority voting algorithm for brain tumour classification", Computers in Biology and Medicine, 2021 Publication	<1%
15	cors.archive.org Internet Source	<1%
16	baadalsg.inflibnet.ac.in Internet Source	<1%
17	cradpdf.drdc-rddc.gc.ca Internet Source	<1%

18	quod.lib.umich.edu Internet Source	<1%
19	www.senescyt.gob.ec Internet Source	<1%
20	Andrea Renda. "Chapter 10 The Age of Foodtech: Optimizing the Agri-Food Chain with Digital Technologies", Springer Science and Business Media LLC, 2019 Publication	<1%

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		