CHRIST
(DEEMED TO BE UNIVERSITY)
BANGALORE · INDIA

REAL-TIME CROWD DETECTION ANALYTICS IN TRADING OUTLETS

by

PULKIT KHANDELWAL(2048017)
ASHRITHA K (2048029)

Under the guidance of Dr. SARAVANAN K S

A Project report submitted in partial fulfillment of the requirements for the award of degree of Master of Science (Data Science) of CHRIST (Deemed to be University)

November - 2021

CHRIST
(DEEMED TO BE UNIVERSITY)
BANGALORE · INDIA

CERTIFICATE

This is to certify that the report titled ***Real-Time Crowd detection analytics in Trading Outlets*** *is a bona fide record of work done by* ***Pulkit Khandelwal (2048017) and Ashritha k - 2048029*** *of CHRIST (Deemed to be University), Bengaluru, in partial fulfillment of the requirements of III Semester M.Sc (Data Science) during the year 2020.*

Head of the Department **Project Guide**

Valued-by:

	Name	:
1.	Register Number	:
	Examination Centre	: CHRIST (Deemed to be University)
2.	Date of Exam	:

ACKNOWLEDGEMENTS

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely fortunate to have got this all along the completion of our project work. Whatever we have done is only due to such guidance and assistance and we would not forget to thank them.

First and foremost we would like to thank god almighty for giving us good health and for helping us complete this project in time.
We respect and thank our department head Prof. Joy Paulose and our coordinator Dr.Senthilnathan T for giving us an opportunity to develop this project work. We owe our profound gratitude to our project guide Dr. Sarvanan K S, who took keen interest on our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good project.

We would not forget to remember Dr. Saleema J and Dr. Cecil of the computer department for the suggestion and guidance during the project presentation.

We heartily thank our Alumni evaluator, ----------for his guidance and suggestions for this project work during the alumni project presentation.
We are thankful to and fortunate enough to get constant encouragement, support and motivation from our class teacher Dr.Sreeja which helped us in successfully completing our project work.

Last but not the least, we would also like to thank our family and friends for their constant support and guidance for the smooth development of this project.

Pulkit Khandelwal
Ashritha

ABSTRACT

The pandemic has made everyone aware of social and physical distancing; whether it is Malls, Restaurants, or Shopping Centers, all are crowded hotspots during weekends. Now, as outlets are reopening, people are more conscious about visiting overcrowded places. Currently, people lack a way to know the real-time crowd count at these trading outlets.

The idea of Crowd Count estimation will pave the way to find the solution to many such problems. This project helps customers with queries regarding the real-time crowd in some of the socially active trading outlets, answered through websites or web apps. This application provides a live crowd count of the outlets, which will help the customer to get to know the real-time count of the people inside the outlet, which will lead to the customer's choice whether to visit it now or schedule it later in case of overcrowding. This will save the customer's time as time is precious. Amidst the pandemic, this will also help people avoid social interactions, and crowd management will be easier for the outlets. The tools and technology used for the project include machine learning applications like Computer vision(object detection), Web Technology(front end and back end) - front end technology like HTML, CSS, JS, Bootstrap. Back-end technology like Django framework and Database technology like My SQL.

TABLE OF CONTENTS

Contents

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

The crowd is a source of transmission in the COVID-19 spread. Prevention and alleviation measures have focused on reducing people's mass gatherings. When it comes to the Covid-19 pandemic, communities are still unsure how to resume a "new normal" existence while the virus is still circulating among the people, despite an initial lock-down phase. The use of a vaccine at a global level represents a massive challenge for humanity, and it is not likely to achieve even within months. In the meantime, we still need some mechanism to allow the people integration into their regular routines reducing the risk of infection. When we talk about trading outlets, like shopping malls, restaurants, etc., many people are gathering, thus increasing the risk of covid-19. To avoid such risk, we can use the concept of "Counting" and let people know the real-time Crowd Count of these outlets. Crowd Count will help you know how many people are present in outlets leading to effective monitoring and management of crowd levels.

In this digital age, many crowd counting systems still rely on old-fashioned approaches like keeping registers, using people counters, and using sensors to count people at the door. These strategies are ineffective in situations where people's movement is fully random, extremely unpredictable, and dynamic.

As a result, crowd-counting technologies based on CCTV video feeds have evolved. The main advantage of adopting video feed counting methods is that the dynamic of people's movement cannot be accounted for in any of the previous methods of crowd counting. This necessitates a contemporary approach to the problem. An accurate crowd counting method can help in emergency circumstances like fires, earthquakes, and other disasters. In these circumstances, a crowd estimate would help the involved authorities to make the best decisions possible about resource supplies.

1.1 PROBLEM DESCRIPTION

The crowd is a source of transmission in the COVID-19 spread. Prevention and alleviation measures are focused on reducing mass gatherings. To help to prevent the spread of COVID-19, social distancing, also known as physical separation, has evolved as a new public standard. People will have narrower social circles post-COVID, be more aware of strangers, and have less desire to be in locations where there are many people. In this scenario, Crowd Detection and Crowd Count may assist. Outlets can use this technology to detect crowd count and provide this information to the public or customers. Crowd Count will also help outlets to manage and regulate occupancy and detect overcrowding. Because social separation has no exceptions, crowd detection can ensure that social distancing follows. As an Objective of the Project, Crowd Counting technology can be implemented in cameras at entrances and exits to assist in the enforcement of live crowd count and Web Technology to make it real-time available to people.

1.2 EXISTING SYSTEMS

In recent years, advances in image processing algorithms and computer technology have led to video cameras to track and count people. Different tools and technologies have made it easier to solve numerous problems on computer vision and related applications.

Many crowd counting systems are already available in different forms to monitor the people's count in a public store. They are either monitored through sensors or pre-installed video surveillance cameras, which are trained to detect objects and track them along with many other features. Each system is implemented with different computer vision methods, object detection, object tracking algorithms, and people counting algorithms.

There are few existing systems which can auto summarize the source content.

The table 1.1 gives details about the existing systems.

Table 1.1: Existing systems

Existing applications	**Features**
People Counting System Using Existing Surveillance Video Camera	OpenCV libraries and computer vision. shadow and highlight detection, and blob detection and tracking.
Crowd Counting Using End- to-End Semantic Image Segmentation	The framework was based on semantic scene segmentation using an optimized convolutional neural network. crowd counting through integrating the density maps
Convolutional-Neural Network-Based Image Crowd Counting: Categorization, Analysis, and Performance Evaluation	Crowd Counting, Counting by Detection, Counting by Regression, Counting by Density Estimation, Counting by CNN, Counting by Clustering
Near Real-time Crowd Counting using Deep Learning Approach	Deep Convolutional Neural Networks (DCNN) in its front-end with the back end of Dilated Convolutional Neural Networks. Regression-based Crowd Counting
Real-time people counting system using a single video camera	Motion Model Tracking Matrix Matching, Merging, and Splitting Module Counting

The limitations of the existing system are the crowd count tracked by the cameras in these retail stores, benefitting only trading outlets to manage their crowd. This crowd count information is not helping the public in any sense. The outlets do not provide a system where real-time people count information is displayed on the website or web application.

1.3 OBJECTIVE

a. Counting The Number of People Currently Present In The Outlet

The primary goal of this project is to provide machine learning functionality to pre- installed CCTV cameras at outlets to detect people count using Computer Vision Technology (Object Detection).

b. Real-Time Crowd Count Statistic On Website

The project also aims to provide crowd count on the website, which will help the end customers know the number of people possibly in the outlet in real time.

c. Gender And Age Estimation

Estimation of age and gender of customers getting in the outlet will help the outlet companies to examine the type of crowd that comes in their outlet and provide services accordingly.

d. Helping Outlets In Handling Crowd Management

Real-time crowd count will help the outlets to design strategies to handle the crowd in case of an everyday scenario or overcrowding scenario.

e. Helping public to make an informed decision

The project is helping to get to know the real-time count of the people inside the outlet, which will lead to the customer's choice whether to visit it now or schedule it later in case of overcrowding. This will save the customer's time as time is precious. Because of this, the public will be able to make an informed decision.

1.4 PURPOSE, SCOPE AND APPLICABILITY

1.4.1 PURPOSE

To help to prevent the spread of COVID-19, social distancing, also known as physical separation, has evolved as a new public standard. However, enforcing social distancing is difficult. The link between crowds and the transmission of the virus has been so ingrained in the public consciousness that is simply viewing a large group can increase worry, anxiety, and even fear in some people. People will have narrower social circles post-COVID, be more aware of strangers, and have less desire to be in locations where there are many people. In this scenario, Crowd Detection and Crowd Count may assist. Outlets can use this technology to detect crowd count and provide this information to the public or customers.

The project focuses on developing a web application that provides a live crowd count that will help the customer to get to know the real-time count of the people at the outlet leading to the customer's choice whether to visit it now or schedule it later in case of overcrowding.

This will save the customer's time as time is precious. Crowd Count will also help outlets to manage and regulate occupancy and detect overcrowding. Because social separation has no exceptions, crowd detection can ensure that social distancing follows. Crowd Counting technology can be implemented in cameras at entrances and exits to assist in the enforcement of live crowd count and Web Technology to make it real-time available to people. While this method does not directly address the physical separation of individuals, it is a step in the right direction.

1.4.2 SCOPE

The project covers people or the general public and outlet companies as its scope. We will be utilizing the cameras that are pre-installed in the outlets to detect the person, count the number of persons that are currently in the outlets, and, using the outlet's website to display the information to the general public.

1.4.3 APPLICABILITY

Using this project, we are able to benefit both public and outlet companies. By examining the real-time crowd count at the outlet's website, people will be able to make a decision as to when to visit the outlet and avoid it in case of overcrowding, leading to less risk in the time of the pandemic. Outlet companies will also be benefitted as they will get the actual number of people and will be able to handle the crowd accordingly as per the social distancing norms.

1.5 OVERVIEW OF THE REPORT

This report gives a detailed explanation of the working, model, specification and other necessary descriptions for this project which can help the reader understand the benefits of the application, its limitation and in short an overview of the entire project.

2. SYSTEM ANALYSIS AND REQUIREMENTS

This chapter gives the requirements and conceptual model of the proposed project.

2.1 PROBLEM DEFINITION

This project is developed to ease the process of crowd detection and helps to prevent the spread of COVID-19, social distancing, also known as physical separation, has evolved as a new public standard. People will have narrower social circles post-COVID, be more aware of strangers, and have less desire to be in locations where there are many people. In this scenario, Crowd Detection and Crowd Count may assist. Outlets can use this technology to detect crowd count and provide this information to the public or customers. Crowd Count will also help outlets to manage and regulate occupancy and detect overcrowding. Because social separation has no exceptions, crowd detection can ensure that social distancing follows. As an Objective of the Project, Crowd Counting technology can be implemented in cameras at entrances and exits to assist in the enforcement of live crowd count and Web Technology to make it real-time available to people.

2.2 REQUIREMENT SPECIFICATION

Crowd Counting will benefit both public and outlet companies. The system will functionally require the cameras pre-installed in the outlets to detect the person, count the number of persons currently in the outlets, and use the outlet's website to display the information to the general public.

Functional Requirements of the System

- Counting The Number Of People Currently Present In The Outlet

- Real-Time Crowd Count Statistic On Website

- Gender And Age Estimation

- Helping Outlets In Handling Crowd Management

Non Functional Requirements of the System

- **System Constraints** - Our system's extracted video feed is limited to CCTV entrance and exit gate footage. We will not cover any other cameras installed for crowd detection. We will directly take the live CCTV footage for people counting and not recorded videos.

- **Response time** - Our system will real-time display the website's people count statistics, leading to a fast response time.

- **Security** - The system protects in a very efficient way. No CCTV live footage will be leaked via the website and will share no customers' personal information to any third party.

- **Reusability** - The model can be reused for different trading outlets to solve the related problem. It is a generalized model that can be used in any public place where people counting is necessary.

- **Modifiability** - The website is flexible to modify according to the outlet's expectations. They wish to display the real-time count statistic and make the website more interactive with their customers.

2.3 BLOCK DIAGRAM

The figure 2.1 represents the block diagram for this project. It shows the basic components and functionalities for the proposed project model.

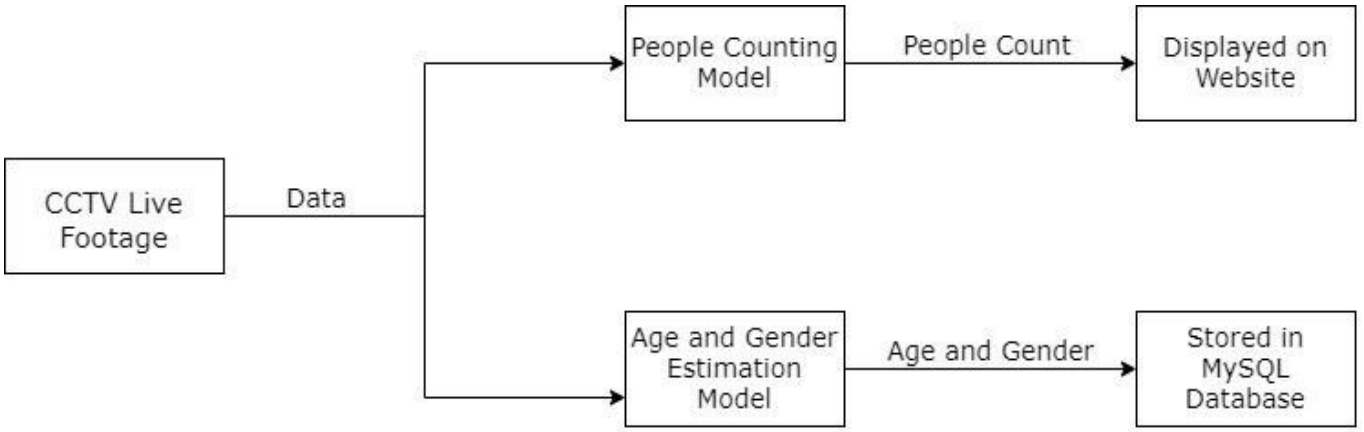

Fig 2.1 Block Diagram

2.4 SYSTEM REQUIREMENTS

The following are the system requirements:

2.4.1 USER CHARACTERISTICS:

There are mainly two users for this application. One is the end user who uses this application to generate summary for a given source file whereas the other user is admin who manages the smooth flow of this application by monitoring it on a regular basis.

2.4.2 SOFTWARE AND HARDWARE REQUIREMENTS:

Requirement needs to be considered for both deployment and development:

1. Hardware Requirements: –

- CPU -
  - Intel Core i5/i7/i9 HQ/Z
  - AMD Ryzen 5/7/9 Zen+ or later
  - RAM - 4 GB or higher
  - Preferable above 300 GB

2. Software Requirements: –

- Operating System: Windows 10–
- Presentation tier- HTML5, CSS3, JavaScript
- Logic tier- Python3

- Data tier- My SQL

3. Documentation: Microsoft Word-2010 1.1.4

The major technologies used in the project is :-

1. Computer Vision (Object Detection)

Python libraries and Algorithms for Object Detection, Object Tracking (people), Age & Gender estimation are

- Numpy
- OpenCV
- Dlib
- imutils
- Pre-trained Caffe Deep learning models

The algorithm used for Object Detection(people) is a hybrid approach MobileNet-SSDs+Linear SVM as it is a highly accurate object detection method without as much of the computational burden. The object tracking algorithm used is the centroid object tracking algorithm in order to track the object in a video stream.

2. Web Technology

The project will be using the website as the delivery mode. Crowd Count, Age & Gender of people will be stored in a database, and later, using the website, we will display the real-time information to the public.

- Front-end Technology (HTML , CSS , Bootstrap)
- Back-end Technology (Django)
- Database (MySQL)

3. SYSTEM DESIGN

The system design section gives the details regarding the system architecture, modules, tables and interface for the proposed system.

3.1 SYSTEM ARCHITECTURE

A system architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

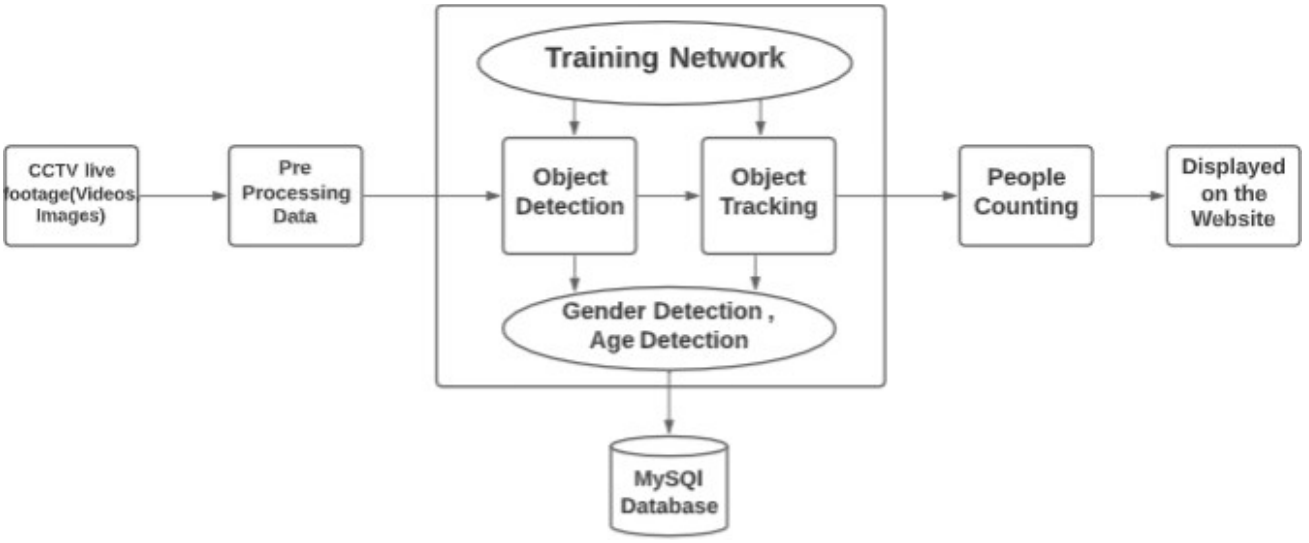

Fig 3.1 System Architecture

The proposed system architecture takes the live CCTV footage as data from the pre-installed cameras. That data will be pre processed using computer vision libraries, sent to a training network for object detection and object tracking, and then to the people counting model, where the Age & Gender is also estimated. The people Counting model sends the data, i.e., count statistics, directly to the website and displays the crowd count. The age & Gender estimation model sends the predicted age and gender data to MySQL Database for further analysis by trading outlets.

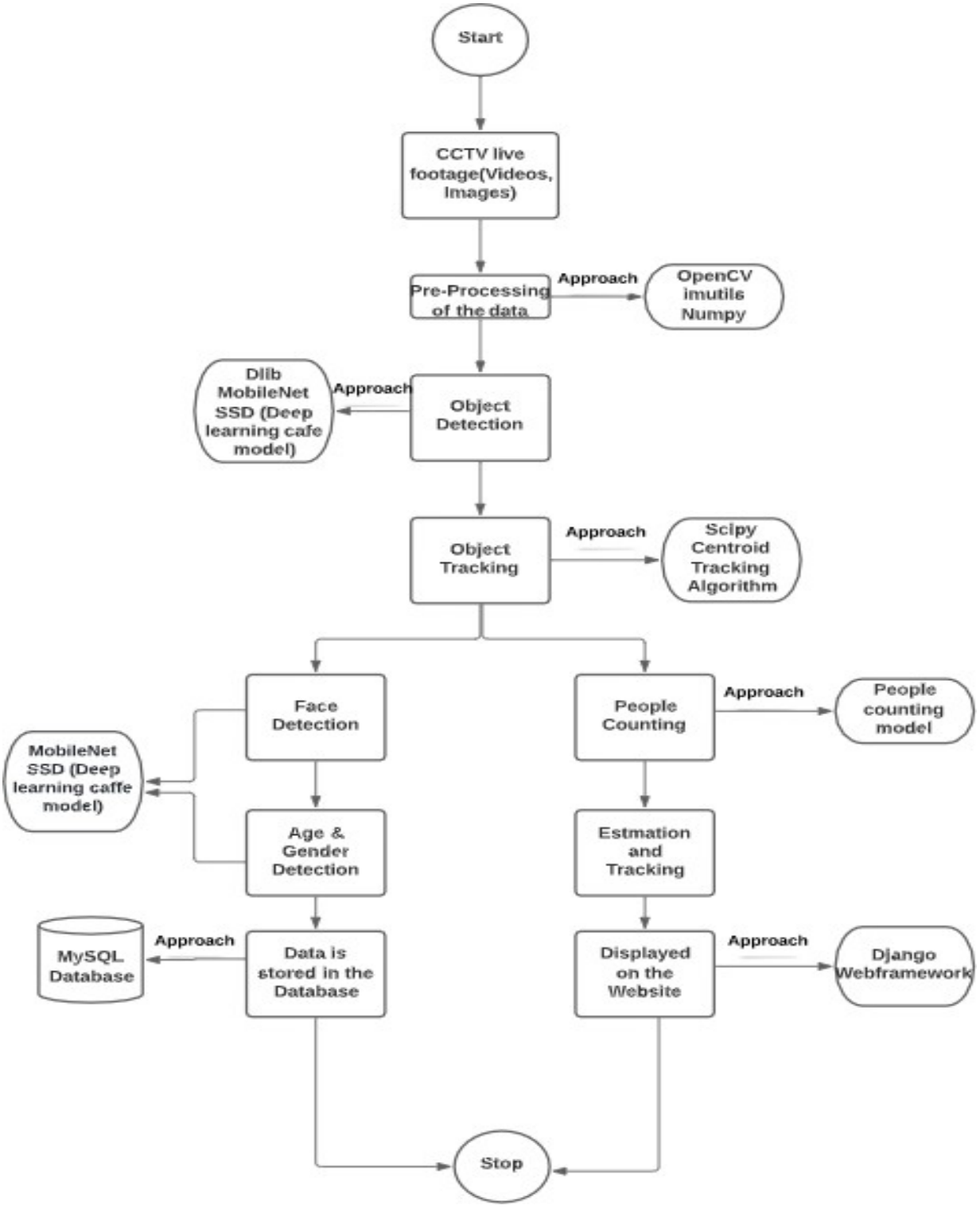

Fig 3.1.2 Flowchart of the proposed system

3.2 MODULE DESIGN

The proposed system takes the live CCTV footage as data from the pre-installed cameras, and that data will be sent to two models, i.e., the People Counting model and Age & Gender estimation model. The people counting model helps to count the number of people, while the Age & Gender estimation model helps to predict the age and gender of the people.

The people Counting model sends the data, i.e., count statistics, directly to the website and displays the crowd count. The age & Gender estimation model sends the data, i.e., predicted age and gender, to the MySQL Database for further analysis by trading outlets.

The following are the modules in the project

- CCTV live footage : It stands for Closed-Circuit Television. This is the primary module of our project. We will be extracting the real-time data from these surveillance cameras.
- These cameras are pre-installed at every outlet at entrance and exit gates.
- People Counting Model : This proposed model takes the raw video data from a video surveillance camera and then further processes it with the help of object detection and object tracking algorithms , thus tracing the count of people inside the outlet.
- Age and Gender Estimation Model - This proposed model takes the raw video data from a video surveillance camera and then further processes it with the help of Age and Gender estimation algorithms to calculate age and gender of the customer.
- Crowd Count Displayed on the Website : The website will take the count from the People Counting Model and directly display the count statistics on the outlet's website.
- Stored in MySQL Database: Data retrieved from Age and Gender Estimation Model will be stored in a Database like MySQL which will then be used for further processing by trading outlet's administration.

3.3 DATABASE DESIGN

There are three tables used for this project, namely: Customer Details, People_In_Outlet, People_Exit_Outlet in order to store the necessary information related to this project.

3.3.1 TABLES AND CONSTRAINTS

The details about the database is given in detail in Tables 3.1, 3.2 and 3.3 such as the data type which each attribute follows and the description of each attribute.

Also the screenshots Fig. 3.4, 3.5, 3.6 gives the picture of how the database looks in MySQL DB.

Table 3.1 Customer Details table

Sl_no	(integer) serial number (row wise indexing)
Gender	(String) Gender of the Customer
Minimum Age	(integer) Minimum age of the Customer in the Range
Maximum Age	(integer) Maximum age of the Customer in the Range

Table 3.2 People_In_Outlet Table

Sl_no	(integer) serial number (row wise indexing)
People Entry Time	(Timestamp) of the people entering the Outlet
People In	(integer) People inside the outlet at that timestamp

Table 3.3 People Exit Outlet table

Sl_no	(integer) serial number (row wise indexing)
People Exit Time	(Timestamp) of the people entering the Outlet
People Exit	(integer) People outside the outlet at that timestamp

DATABASE SCREENSHOTS

Fig 3.4 Customer_Details table in Database

	gender	minimum_age	maximum_age
1	Male	8	12
2	Female	25	32
3	Female	25	32
4	Female	25	32
5	Female	25	32
6	Female	25	32
7	Female	25	32
8	Female	38	43
9	Female	8	12
10	Female	8	12
11	Female	38	43
12	Female	8	12
13	Female	38	43
14	Female	8	12
15	Male		

Fig 3.5 People_In_Outlet table in Database

	people_entry_time	people_in
1	2021-10-24 22:45:08	1
2	2021-10-24 22:45:18	1
3	2021-10-24 22:45:23	1
4	2021-10-24 22:45:39	1
5	2021-10-25 12:07:51	1
6	2021-10-25 12:08:02	1
7	2021-10-25 12:08:07	1
8	2021-10-25 12:08:16	1
9	2021-10-25 12:08:25	1

Fig 3.6 People_Exit_Outlet table in Database

	people_exit_time	people_exit
1	2021-10-24 22:51:25	1
2	2021-10-24 22:51:32	1
3	2021-10-24 22:51:41	1
4	2021-10-24 22:51:49	1
5	2021-10-24 22:51:59	1
6	2021-10-25 12:09:25	1
7	2021-10-25 12:09:32	1
8	2021-10-25 12:09:40	1
9	2021-10-25 12:09:48	1

3.3.2 ER DIAGRAM

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.

The Entity relationship diagram in figure 3.7 shows the relation between the different tables in the database for this project. There are mainly four tables, namely:

- People
- Outlet
- Outlet's Website
- Database

These are the four entities that are found in the proposed system. A person entering the outlet his/her Age and Gender are the two attributes of the entity People while the

outlet entity has only one attribute that is the outlet's name. The entity outlet's website

where people visit displays the crowd count of the outlet. The entity outlet stores data in the entity database which is connected to the attribute MySQL.

Attributes:

- Age: Age of the person
- Gender: Gender of the person.
- Outlet Name: Name of the outlet
- MySQL: Database of the Outlet

Entity Relationship Types

- The relationship between People and Outlet is many to one as the system can have many customers visiting one outlet.
- The relationship between People and Outlet's Website is many to one as the system can have many customers visiting the Outlet's Website.
- The relationship between Outlet and Outlet's Website is one to one as one outlet will have its own outlet's Website which displays the crowd count.
- The relationship between Outlet and Database is one to one as the system can have one Database for one which stores the data in MySQL.

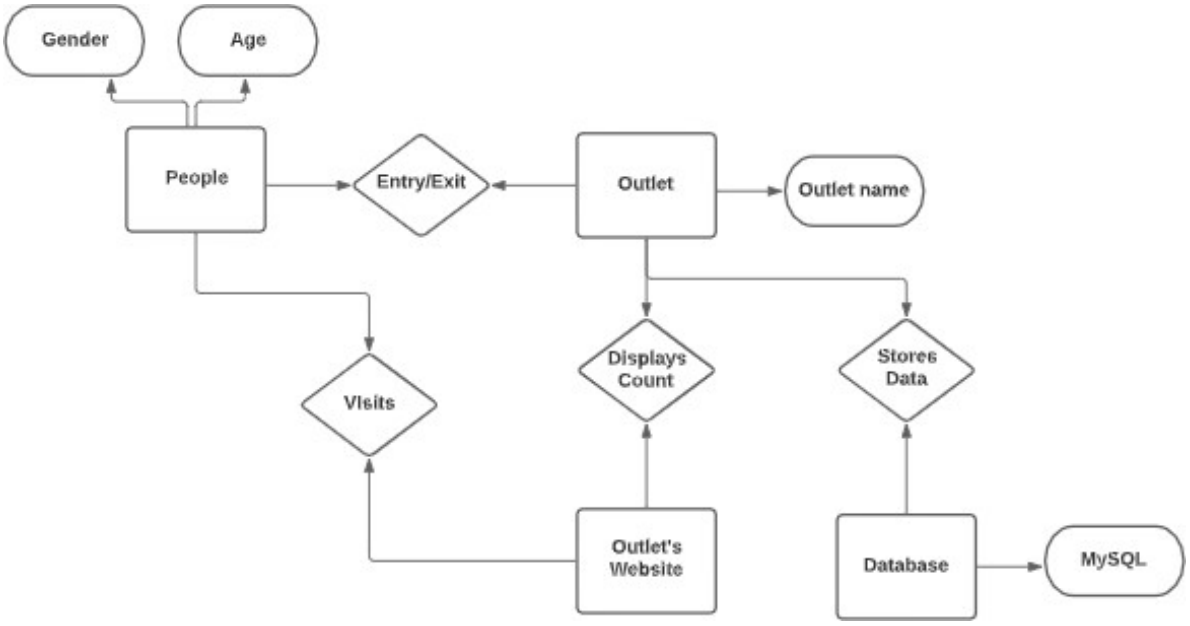

Fig 3.7 ER Diagram

3.3.3 DATAFLOW DIAGRAM

DFD level 0

Figure 3.8 is a simple representation to show the brief functionalities and the external entities in the project (people and outlet). The function shows the major task that this project focuses on which is people count.

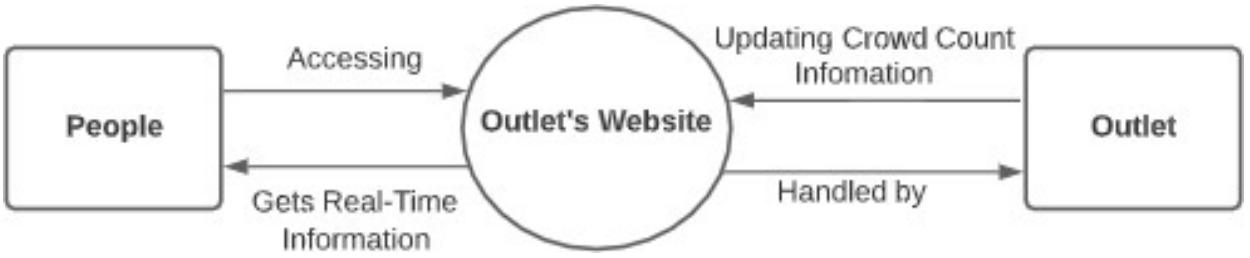

Fig 3.8 DFD level 0

DFD level 1

The level one diagram shown in Fig 3.9 gives a quick view of the divided functionalities when the application's process is examined deeply

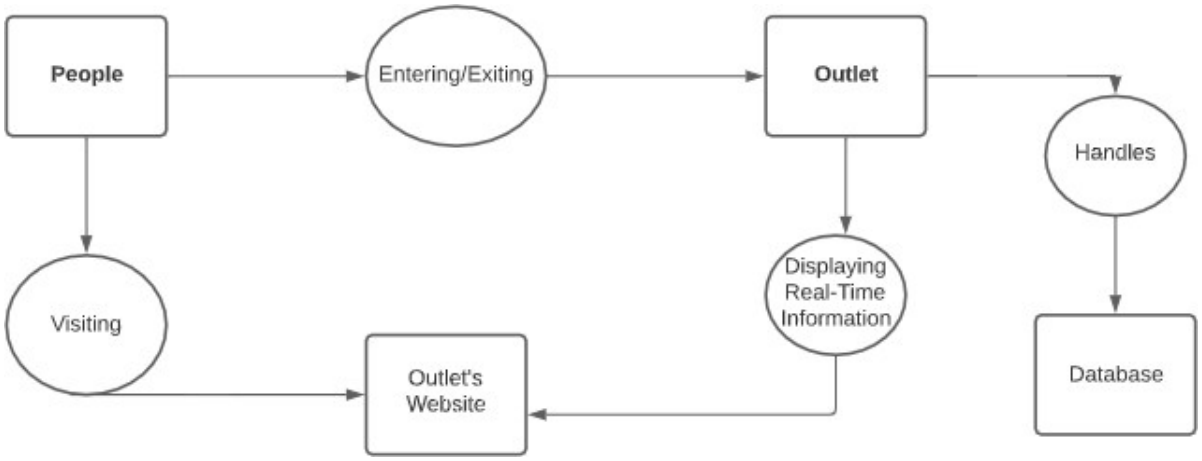

Fig 3.9 DFD level 1

DFD level 2

Figure 3.10 gives a more in depth view of how each process occurs such as the order and its respective connection to the data base.

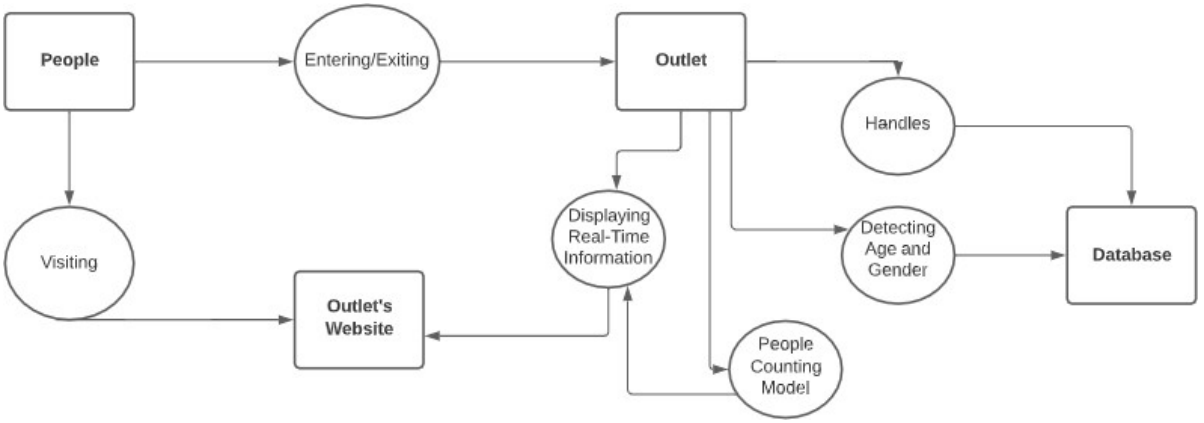

Fig 3.10 DFD level 2

3.4 INTERFACE DESIGN AND PROCEDURAL DESIGN

The following shows the interfaces and flow diagram for the proposed project.

3.4.1 USER INTERFACE DESIGN

The user interfaces for the web application of the proposed project is as follows:

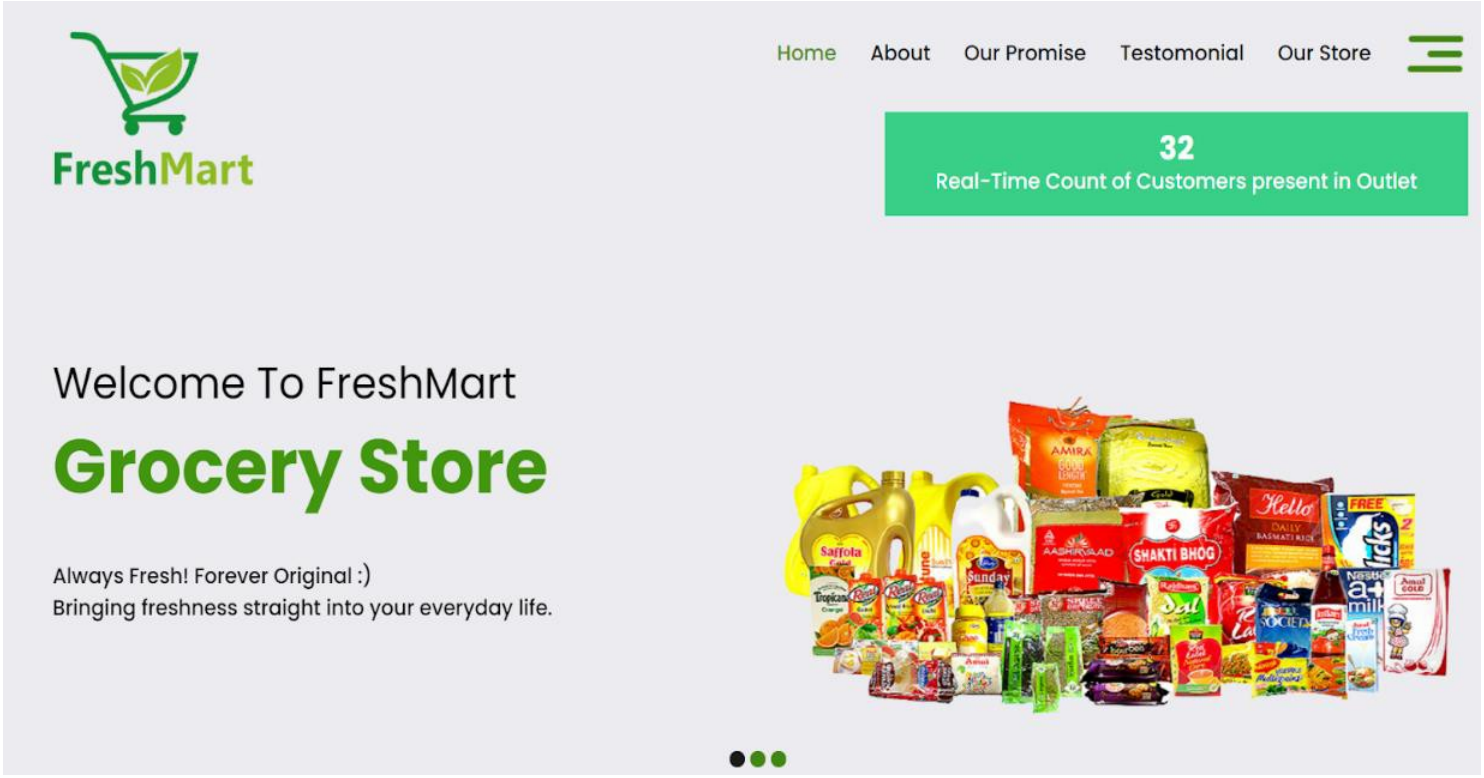

Fig 3.11 Home page

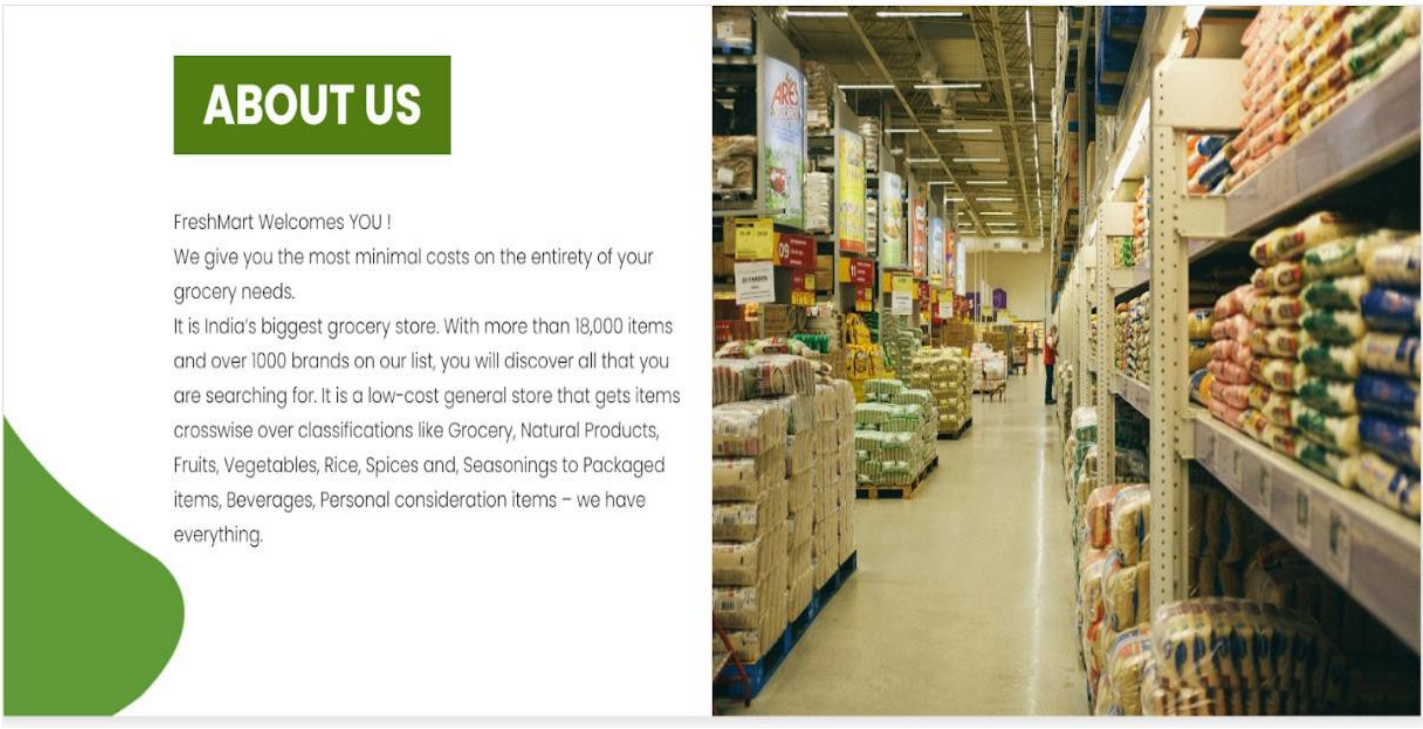

Fig 3.12 About Page

Fig 3.13 Our promise page design of the website

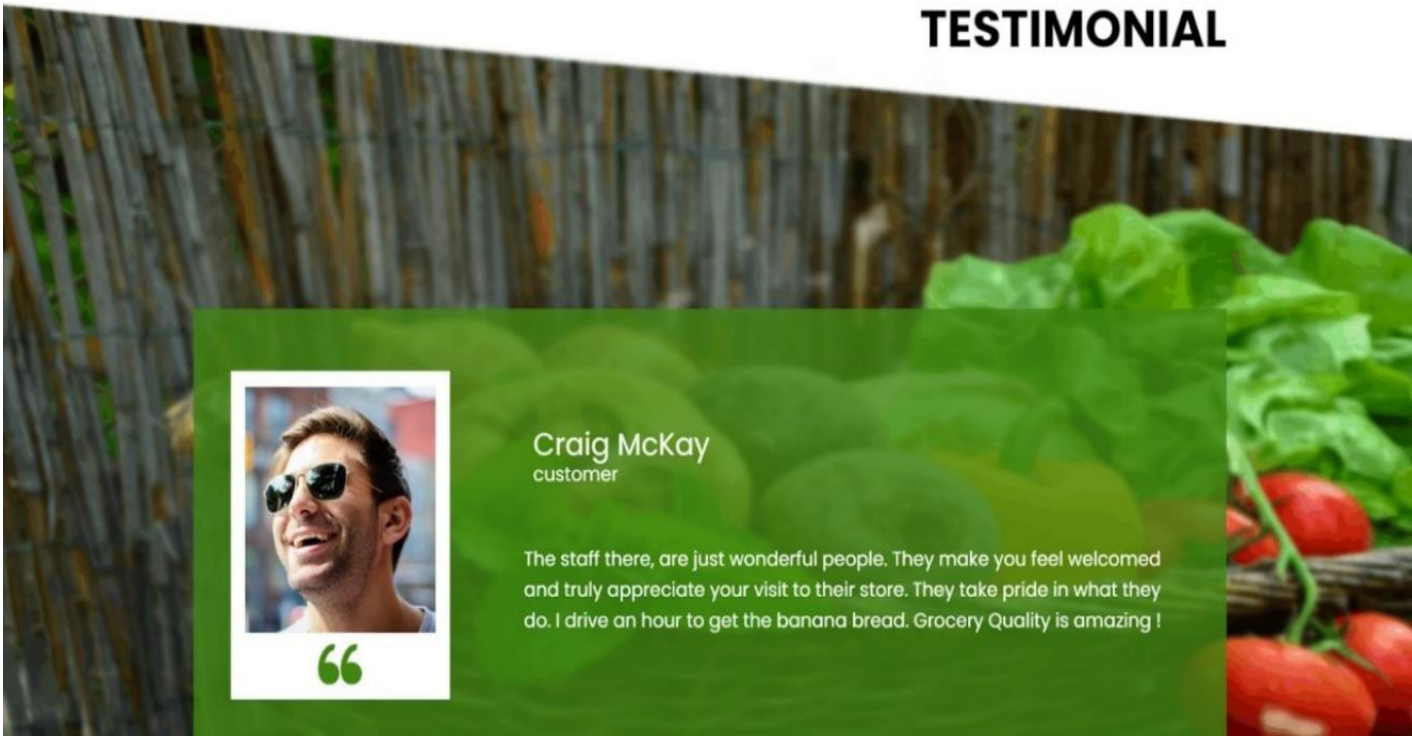

Fig 3.14 Testimonial page design of the website

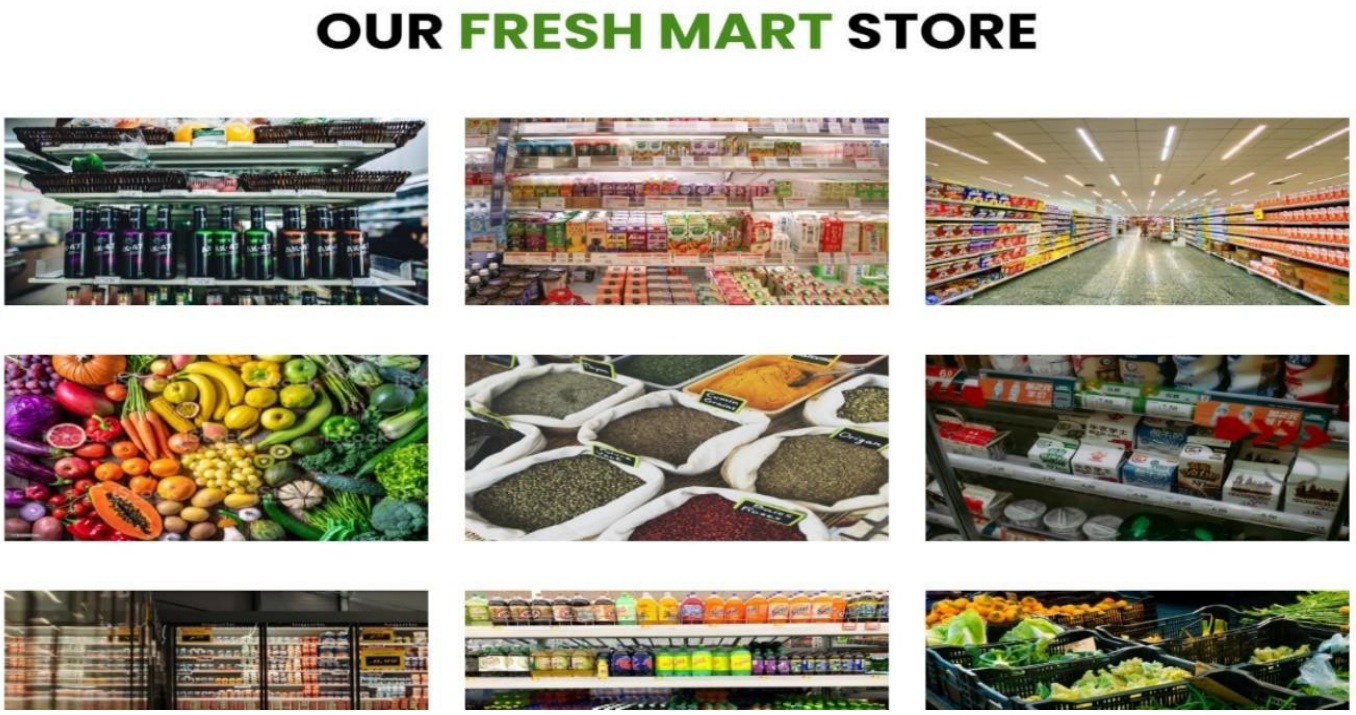

Fig 3.15 Our store page design of the website

4. IMPLEMENTATION

This section gives details regarding how the project is implemented, the algorithms, code and their respective screenshots.

4.1 IMPLEMENTATION ALGORITHMS

This project uses three main algorithms:

1. MobileNet SSD

The MobileNet-SSD model is a Single-Shot Multibox Detection (SSD) network intended to perform object detection. This model is implemented using the Caffe* framework.

SSD Object Detection extracts feature map using a base deep learning network, which is a CNN-based classifier and applies convolution filters to finally detect objects.

In the proposed system the MobileNet SSD is used for object detection, especially face, and age detection.

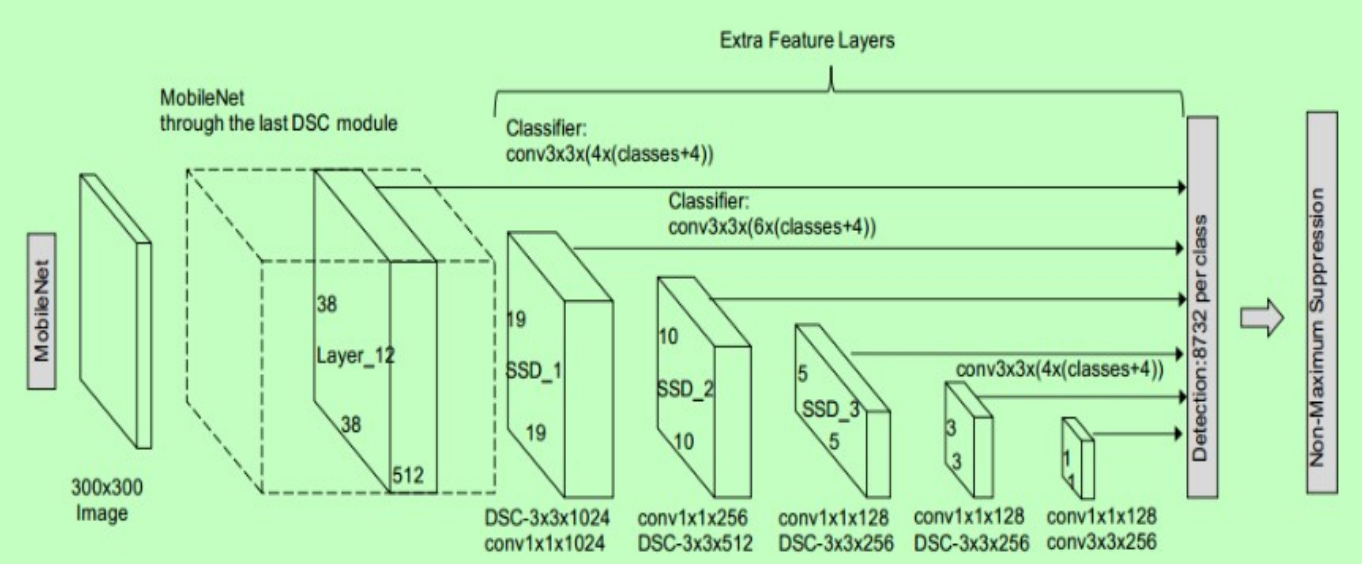

Figure 6. Mobile-Det: SSD-based detection with MobileNet as backbone, modified based on [14]

Fig 4.1 MobileNet SSD Architecture

2. Centroid Tracking Algorithm

The detected objects can be tracked by an object tracker to track the object as it moves around the frame. The object tracker should be faster and more efficient than the object detector and should continue tracking until it has reached the N-th frame and then re-run the object detector. The entire process then repeats. The centroid tracking algorithm is a multi-step process.

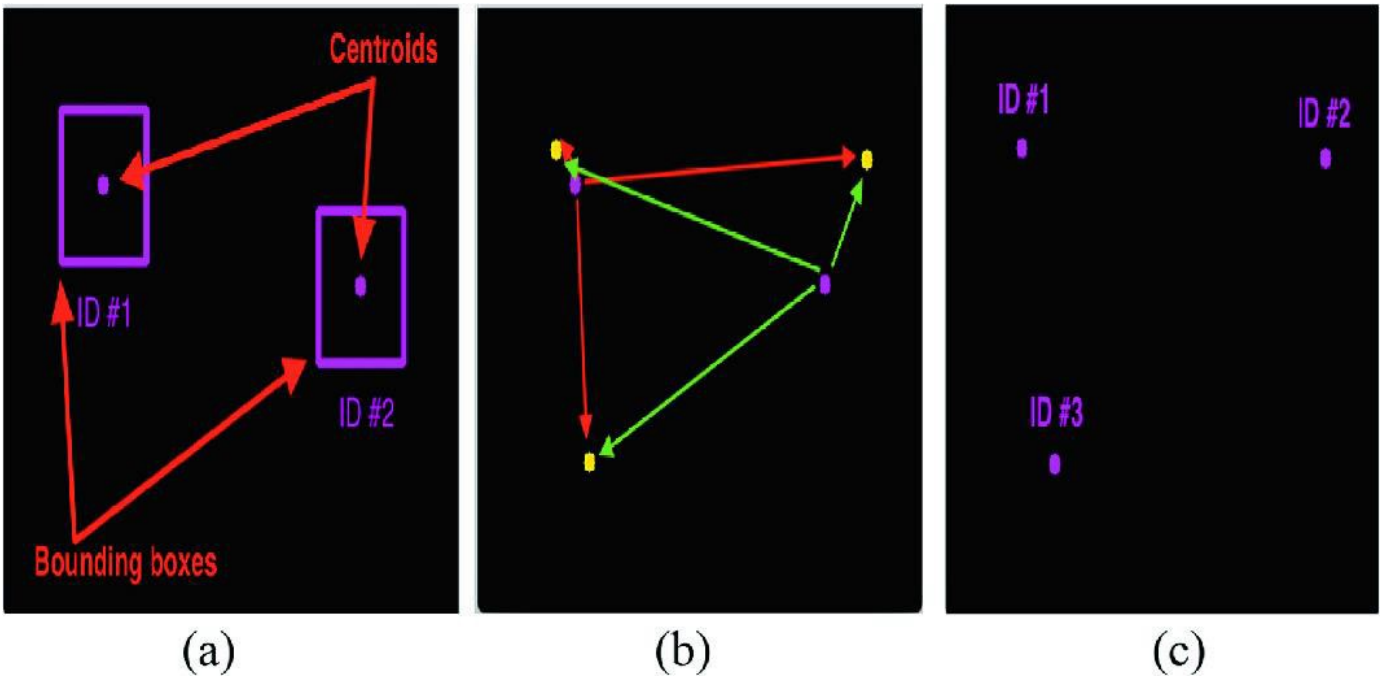

Fig 4.2 Centroid Tracking Algorithm steps

a) Accept bounding box coordinates and compute centroids

The centroid tracking algorithm assumes that the person is passing in a set of the bounding box (x, y)-coordinates for each detected object in every single frame. These bounding boxes can be produced by any type of object detector, provided that they are computed for every frame in the video.

b) Compute Euclidean distance between new bounding boxes and existing objects

Three objects are present in this image for simple object tracking with Python and OpenCV. We need to compute the Euclidean distances between each pair of original centroids (red) and new centroids (green).

c) Update (x, y)-coordinates of existing objects and Register new objects

The centroid object tracking method has associated objects with minimized object distances. The primary assumption of the centroid tracking algorithm is that a given object will potentially move in between subsequent frames, but the distance between the centroids for frames F_t and F_{t+1} will be smaller than all other distances between objects. a new object that wasn't matched with an existing object, so it is registered as object ID

3. People Counting Algorithm

The method to figure out if the object (person) is moving in or out is by grabbing the y-coordinate value for all previous centroid locations for the given object and computing the direction by taking the difference between the current centroid location and the mean of all previous centroid locations.

Therefore, by taking the mean, the people's counter can be more accurate. If the TrackableObject has not been counted then in order to determine if it's ready to be counted yet by:

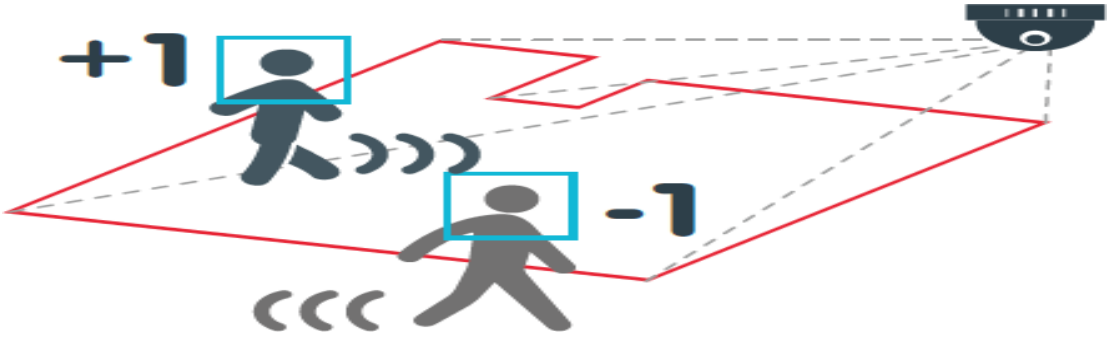

Fig 4.3 People counter system

- Checking if the direction is negative (indicating the object is moving IN) AND the centroid is Above the centerline. In this case, we increment the totalIN.
- Or checking if the direction is positive (indicating the object is moving OUT) AND the centroid is below the centerline. If this is true, we increment totalOUT.

4.2 CODING DETAILS

Few of the main codes used in this project are:

4.2.1 People Entering Outlet

people_counter_in.py

```
# import the necessary packages
from pyimagesearch.centroidtracker import CentroidTracker
from pyimagesearch.trackableobject import TrackableObject
from imutils.video import VideoStream
from imutils.video import FPS
import numpy as np
import argparse
import imutils
import dlib
import time
import cv2
from datetime import datetime
import mysql.connector

mydb = mysql.connector.connect(
  host="127.0.0.1",
  user="root",
  password="root",
  database="freshmartstoredb"
)

mycursor = mydb.cursor()

counter=1

#Command to run
#python people_counter_in.py --prototxt mobilenet_ssd/MobileNetSSD_deploy.prototxt --model mobilenet_ssd/MobileNetSSD_deploy.caffemodel --inp


# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
    help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
    help="path to Caffe pre-trained model")
```

Fig 4.4 People Entering Outlet Code

people_counter_in.py

```
# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
    help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
    help="path to Caffe pre-trained model")
ap.add_argument("-i", "--input", type=str,
    help="path to optional input video file")
ap.add_argument("-o", "--output", type=str,
    help="path to optional output video file")
ap.add_argument("-c", "--confidence", type=float, default=0.8,
    help="minimum probability to filter weak detections")
ap.add_argument("-s", "--skip-frames", type=int, default=12,
    help="# of skip frames between detections")
args = vars(ap.parse_args())

# initialize the list of class labels MobileNet SSD was trained to
# detect
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
    "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
    "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
    "sofa", "train", "tvmonitor"]

# load our serialized model from disk
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])

# if a video path was not supplied, grab a reference to the webcam
if not args.get("input", False):
    print("[INFO] starting video stream...")
    vs = VideoStream(src=0).start()
    time.sleep(2.0)

# otherwise, grab a reference to the video file
```

Fig 4.5 People Entering Outlet Code

people_counter_in.py

```
# otherwise, grab a reference to the video file
else:
    print("[INFO] opening video file...")
    vs = cv2.VideoCapture(args["input"])

# initialize the video writer (we'll instantiate later if need be)
writer = None

# initialize the frame dimensions (we'll set them as soon as we read
# the first frame from the video)
W = None
H = None

# instantiate our centroid tracker, then initialize a list to store
# each of our dlib correlation trackers, followed by a dictionary to
# map each unique object ID to a TrackableObject
ct = CentroidTracker(maxDisappeared=40, maxDistance=50)
trackers = []
trackableObjects = {}

# initialize the total number of frames processed thus far, along
# with the total number of objects that have moved either up or down
totalFrames = 0
totalDown = 0
totalUp = 0

# start the frames per second throughput estimator
print('People entering in outlet at time :-')
fps = FPS().start()

# loop over frames from the video stream
while True:
    # grab the next frame and handle if we are reading from either
    # VideoCapture or VideoStream
```

Fig 4.6 People Entering Outlet Code

people_counter_in.py

```
# loop over frames from the video stream
while True:
    # grab the next frame and handle if we are reading from either
    # VideoCapture or VideoStream
    frame = vs.read()
    frame = frame[1] if args.get("input", False) else frame

    # if we are viewing a video and we did not grab a frame then we
    # have reached the end of the video
    if args["input"] is not None and frame is None:
        break

    # resize the frame to have a maximum width of 500 pixels (the
    # less data we have, the faster we can process it), then convert
    # the frame from BGR to RGB for dlib
    frame = imutils.resize(frame, width=500)
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # if the frame dimensions are empty, set them
    if W is None or H is None:
        (H, W) = frame.shape[:2]

    # if we are supposed to be writing a video to disk, initialize
    # the writer
    if args["output"] is not None and writer is None:
        fourcc = cv2.VideoWriter_fourcc(*"MJPG")
        writer = cv2.VideoWriter(args["output"], fourcc, 30,
            (W, H), True)

    # initialize the current status along with our list of bounding
    # box rectangles returned by either (1) our object detector or
    # (2) the correlation trackers
    status = "Waiting"
    rects = []
```

Fig 4.7 People Entering Outlet Code

people_counter_in.py

```
# check to see if we should run a more computationally expensive
# object detection method to aid our tracker
if totalFrames % args["skip_frames"] == 0:
    # set the status and initialize our new set of object trackers
    status = "Detecting"
    trackers = []

    # convert the frame to a blob and pass the blob through the
    # network and obtain the detections
    blob = cv2.dnn.blobFromImage(frame, 0.007843, (W, H), 127.5)
    net.setInput(blob)
    detections = net.forward()

    # loop over the detections
    for i in np.arange(0, detections.shape[2]):
        # extract the confidence (i.e., probability) associated
        # with the prediction
        confidence = detections[0, 0, i, 2]

        # filter out weak detections by requiring a minimum
        # confidence
        if confidence > args["confidence"]:
            # extract the index of the class label from the
            # detections list
            idx = int(detections[0, 0, i, 1])

            # if the class label is not a person, ignore it
            if CLASSES[idx] != "person":
                continue

            # compute the (x, y)-coordinates of the bounding box
            # for the object
            box = detections[0, 0, i, 3:7] * np.array([W, H, W, H])
            (startX, startY, endX, endY) = box.astype("int")

            # construct a dlib rectangle object from the bounding
            # box coordinates and then start the dlib correlation
```

Fig 4.8 People Entering Outlet Code

people_counter_in.py

```
                # box coordinates and then start the dlib correlation
                # tracker
                tracker = dlib.correlation_tracker()
                rect = dlib.rectangle(startX, startY, endX, endY)
                tracker.start_track(rgb, rect)

                # add the tracker to our list of trackers so we can
                # utilize it during skip frames
                trackers.append(tracker)

    # otherwise, we should utilize our object *trackers* rather than
    # object *detectors* to obtain a higher frame processing throughput
    else:
        # loop over the trackers
        for tracker in trackers:
            # set the status of our system to be 'tracking' rather
            # than 'waiting' or 'detecting'
            status = "Tracking"

            # update the tracker and grab the updated position
            tracker.update(rgb)
            pos = tracker.get_position()

            # unpack the position object
            startX = int(pos.left())
            startY = int(pos.top())
            endX = int(pos.right())
            endY = int(pos.bottom())

            # add the bounding box coordinates to the rectangles list
            rects.append((startX, startY, endX, endY))

    # draw a horizontal line in the center of the frame -- once an
    # object crosses this line we will determine whether they were
    # moving 'up' or 'down'
    cv2.line(frame, (0, H // 2), (W , H // 2), (0, 255, 255), 2)
```

Fig 4.9 People Entering Outlet Code

people_counter_in.py

```
    # use the centroid tracker to associate the (1) old object
    # centroids with (2) the newly computed object centroids
    objects = ct.update(rects)

    # loop over the tracked objects
    for (objectID, centroid) in objects.items():
        # check to see if a trackable object exists for the current
        # object ID
        to = trackableObjects.get(objectID, None)

        # if there is no existing trackable object, create one
        if to is None:
            to = TrackableObject(objectID, centroid)

        # otherwise, there is a trackable object so we can utilize it
        # to determine direction
        else:
            # the difference between the y-coordinate of the *current*
            # centroid and the mean of *previous* centroids will tell
            # us in which direction the object is moving (negative for
            # 'up' and positive for 'down')
            y = [c[1] for c in to.centroids]
            direction = centroid[1] - np.mean(y)
            to.centroids.append(centroid)

            # check to see if the object has been counted or not
            if not to.counted:
                # if the direction is negative (indicating the object
                # is moving up) AND the centroid is above the center
                # line, count the object
                if direction < 0 and centroid[1] < H // 2:
                    totalUp += 1
                    to.counted = True

                # if the direction is positive (indicating the object
                # is moving down) AND the centroid is below the
                # center line, count the object
```

Fig 4.10 People Entering Outlet Code

people_counter_in.py

```
                # center line, count the object
                elif direction > 0 and centroid[1] > H // 2:
                    totalDown += 1
                    to.counted = True

            # store the trackable object in our dictionary
            trackableObjects[objectID] = to

            # draw both the ID of the object and the centroid of the
            # object on the output frame
            text = "ID {}".format(objectID)
            cv2.putText(frame, text, (centroid[0] - 10, centroid[1] - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
            cv2.circle(frame, (centroid[0], centroid[1]), 4, (0, 255, 0), -1)

        # construct a tuple of information we will be displaying on the
        # frame

        info = [
            ("IN", totalDown),
            ("Status", status),
        ]
        # while counter == totalDown:
        #   print(datetime.now())
        #   sql = "INSERT INTO people_in_outlet (people_entry_time, people_in) VALUES (%s, %s)"
        #   val = (datetime.now(), 1)
        #   mycursor.execute(sql, val)
        #   mydb.commit()
        #   counter+=1
        #   break

        # loop over the info tuples and draw them on our frame
        for (i, (k, v)) in enumerate(info):
            text = "{}: {}".format(k, v)
            cv2.putText(frame, text, (300, H - ((i * 20) + 20)),
                cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)

        # check to see if we should write the frame to disk
```

Fig 4.11 People Entering Outlet Code

people_counter_in.py

```
        # check to see if we should write the frame to disk
        if writer is not None:
            writer.write(frame)

        # show the output frame
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF

        # if the `q` key was pressed, break from the loop
        if key == ord("q"):
            break

        # increment the total number of frames processed thus far and
        # then update the FPS counter
        totalFrames += 1
        fps.update()

    # stop the timer and display FPS information
    fps.stop()
    print(f'People inside the Outlet - {totalDown}')


    # check to see if we need to release the video writer pointer
    if writer is not None:
        writer.release()

    # if we are not using a video file, stop the camera video stream
    if not args.get("input", False):
        vs.stop()

    # otherwise, release the video file pointer
    else:
        vs.release()

    # close any open windows
    cv2.destroyAllWindows()
```

Fig 4.12 People Entering Outlet Code

i. People Exiting Outlet

people_counter_out.py

```
# import the necessary packages
from pyimagesearch.centroidtracker import CentroidTracker
from pyimagesearch.trackableobject import TrackableObject
from imutils.video import VideoStream
from imutils.video import FPS
import numpy as np
import argparse
import imutils
import time
import dlib
import cv2
from datetime import datetime
import mysql.connector


# #Connecting to database
# mydb = mysql.connector.connect(
#   host="127.0.0.1",
#   user="root",
#   password="root",
#   database="freshmartstoredb"
# )

# #Creating cursor for database
# mycursor = mydb.cursor()


# counter=1

#Command to run
#python people_counter_out.py --prototxt mobilenet_ssd/MobileNetSSD_deploy.prototxt --model mobilenet_ssd/MobileNetSSD_deploy.caffemodel --input vide


# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
    help="path to Caffe 'deploy' prototxt file")
```

Fig 4.13 People Exiting Outlet Code

people_counter_out.py

```
#Command to run
#python people_counter_out.py --prototxt mobilenet_ssd/MobileNetSSD_deploy.prototxt --model mobilenet_ssd/MobileNetSSD_deploy.caffemodel --input vide


# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
    help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
    help="path to Caffe pre-trained model")
ap.add_argument("-i", "--input", type=str,
    help="path to optional input video file")
ap.add_argument("-o", "--output", type=str,
    help="path to optional output video file")
ap.add_argument("-c", "--confidence", type=float, default=0.8,
    help="minimum probability to filter weak detections")
ap.add_argument("-s", "--skip-frames", type=int, default=12,
    help="# of skip frames between detections")
args = vars(ap.parse_args())

# initialize the list of class labels MobileNet SSD was trained to
# detect
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
    "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
    "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
    "sofa", "train", "tvmonitor"]

# load our serialized model from disk
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])

# if a video path was not supplied, grab a reference to the webcam
if not args.get("input", False):
    print("[INFO] starting video stream...")
    vs = VideoStream(src=0).start()
    time.sleep(2.0)
```

Fig 4.14 People Exiting Outlet Code

people_counter_out.py

```
# otherwise, grab a reference to the video file
else:
    print("[INFO] opening video file...")
    vs = cv2.VideoCapture(args["input"])

# initialize the video writer (we'll instantiate later if need be)
writer = None

# initialize the frame dimensions (we'll set them as soon as we read
# the first frame from the video)
W = None
H = None

# instantiate our centroid tracker, then initialize a list to store
# each of our dlib correlation trackers, followed by a dictionary to
# map each unique object ID to a TrackableObject
ct = CentroidTracker(maxDisappeared=40, maxDistance=50)
trackers = []
trackableObjects = {}

# initialize the total number of frames processed thus far, along
# with the total number of objects that have moved either up or down
totalFrames = 0
totalDown = 0
totalUp = 0

# start the frames per second throughput estimator
fps = FPS().start()

# loop over frames from the video stream
while True:
    # grab the next frame and handle if we are reading from either
    # VideoCapture or VideoStream
    frame = vs.read()
    frame = frame[1] if args.get("input", False) else frame

    # if we are viewing a video and we did not grab a frame then we
```

Fig 4.15 People Exiting Outlet Code

people_counter_out.py

```
    frame = vs.read()
    frame = frame[1] if args.get("input", False) else frame

    # if we are viewing a video and we did not grab a frame then we
    # have reached the end of the video
    if args["input"] is not None and frame is None:
        break

    # resize the frame to have a maximum width of 500 pixels (the
    # less data we have, the faster we can process it), then convert
    # the frame from BGR to RGB for dlib
    frame = imutils.resize(frame, width=500)
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # if the frame dimensions are empty, set them
    if W is None or H is None:
        (H, W) = frame.shape[:2]

    # if we are supposed to be writing a video to disk, initialize
    # the writer
    if args["output"] is not None and writer is None:
        fourcc = cv2.VideoWriter_fourcc(*"MJPG")
        writer = cv2.VideoWriter(args["output"], fourcc, 30,
            (W, H), True)

    # initialize the current status along with our list of bounding
    # box rectangles returned by either (1) our object detector or
    # (2) the correlation trackers
    status = "Waiting"
    rects = []

    # check to see if we should run a more computationally expensive
    # object detection method to aid our tracker
    if totalFrames % args["skip_frames"] == 0:
        # set the status and initialize our new set of object trackers
        status = "Detecting"
        trackers = []
```

Fig 4.16 People Exiting Outlet Code

people_counter_out.py

```
            # convert the frame to a blob and pass the blob through the
            # network and obtain the detections
            blob = cv2.dnn.blobFromImage(frame, 0.007843, (W, H), 127.5)
            net.setInput(blob)
            detections = net.forward()

            # loop over the detections
            for i in np.arange(0, detections.shape[2]):
                # extract the confidence (i.e., probability) associated
                # with the prediction
                confidence = detections[0, 0, i, 2]

                # filter out weak detections by requiring a minimum
                # confidence
                if confidence > args["confidence"]:
                    # extract the index of the class label from the
                    # detections list
                    idx = int(detections[0, 0, i, 1])

                    # if the class label is not a person, ignore it
                    if CLASSES[idx] != "person":
                        continue

                    # compute the (x, y)-coordinates of the bounding box
                    # for the object
                    box = detections[0, 0, i, 3:7] * np.array([W, H, W, H])
                    (startX, startY, endX, endY) = box.astype("int")

                    # construct a dlib rectangle object from the bounding
                    # box coordinates and then start the dlib correlation
                    # tracker
                    tracker = dlib.correlation_tracker()
                    rect = dlib.rectangle(startX, startY, endX, endY)
                    tracker.start_track(rgb, rect)

                    # add the tracker to our list of trackers so we can
                    # utilize it during skip frames
```

Fig 4.17 People Exiting Outlet Code

people_counter_out.py

```
                    # utilize it during skip frames
                    trackers.append(tracker)

        # otherwise, we should utilize our object *trackers* rather than
        # object *detectors* to obtain a higher frame processing throughput
        else:
            # loop over the trackers
            for tracker in trackers:
                # set the status of our system to be 'tracking' rather
                # than 'waiting' or 'detecting'
                status = "Tracking"

                # update the tracker and grab the updated position
                tracker.update(rgb)
                pos = tracker.get_position()

                # unpack the position object
                startX = int(pos.left())
                startY = int(pos.top())
                endX = int(pos.right())
                endY = int(pos.bottom())

                # add the bounding box coordinates to the rectangles list
                rects.append((startX, startY, endX, endY))

        # draw a horizontal line in the center of the frame -- once an
        # object crosses this line we will determine whether they were
        # moving 'up' or 'down'
        cv2.line(frame, (0, H // 2), (W , H // 2), (0, 255, 255), 2)

        # use the centroid tracker to associate the (1) old object
        # centroids with (2) the newly computed object centroids
        objects = ct.update(rects)

        # loop over the tracked objects
        for (objectID, centroid) in objects.items():
            # check to see if a trackable object exists for the current
```

Fig 4.18 People Exiting Outlet Code

people_counter_out.py

```
    objects = ct.update(rects)

    # loop over the tracked objects
    for (objectID, centroid) in objects.items():
        # check to see if a trackable object exists for the current
        # object ID
        to = trackableObjects.get(objectID, None)

        # if there is no existing trackable object, create one
        if to is None:
            to = TrackableObject(objectID, centroid)

        # otherwise, there is a trackable object so we can utilize it
        # to determine direction
        else:
            # the difference between the y-coordinate of the *current*
            # centroid and the mean of *previous* centroids will tell
            # us in which direction the object is moving (negative for
            # 'up' and positive for 'down')
            y = [c[1] for c in to.centroids]
            direction = centroid[1] - np.mean(y)
            to.centroids.append(centroid)

            # check to see if the object has been counted or not
            if not to.counted:
                # if the direction is negative (indicating the object
                # is moving up) AND the centroid is above the center
                # line, count the object
                if direction < 0 and centroid[1] < H // 2:
                    totalUp += 1
                    to.counted = True

                # if the direction is positive (indicating the object
                # is moving down) AND the centroid is below the
                # center line, count the object
                elif direction > 0 and centroid[1] > H // 2:
                    totalDown += 1
```

Fig 4.19 People Exiting Outlet Code

people_counter_out.py

```
            # store the trackable object in our dictionary
            trackableObjects[objectID] = to

            # draw both the ID of the object and the centroid of the
            # object on the output frame
            text = "ID {}".format(objectID)
            cv2.putText(frame, text, (centroid[0] - 10, centroid[1] - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
            cv2.circle(frame, (centroid[0], centroid[1]), 4, (0, 255, 0), -1)

    # construct a tuple of information we will be displaying on the
    # frame
    info = [
        ("OUT", totalDown),
        ("Status", status),
    ]

    # while counter == totalDown:
    #   print(datetime.now())
    #   sql = "INSERT INTO people_exit_outlet (people_exit_time, people_exit) VALUES (%s, %s)"
    #   val = (datetime.now(), 1)
    #   mycursor.execute(sql, val)
    #   mydb.commit()
    #   counter+=1
    #   break

    # loop over the info tuples and draw them on our frame
    for (i, (k, v)) in enumerate(info):
        text = "{}: {}".format(k, v)
        cv2.putText(frame, text, (10, H - ((i * 20) + 20)),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)

    # check to see if we should write the frame to disk
    if writer is not None:
        writer.write(frame)
```

Fig 4.20 People Exiting Outlet Code

people_counter_out.py

```
    # check to see if we should write the frame to disk
    if writer is not None:
        writer.write(frame)

    # show the output frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break

    # increment the total number of frames processed thus far and
    # then update the FPS counter
    totalFrames += 1
    fps.update()

# stop the timer and display FPS information
fps.stop()
print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
print(f'People went out of the Outlet = {totalDown}')

# check to see if we need to release the video writer pointer
if writer is not None:
    writer.release()

# if we are not using a video file, stop the camera video stream
if not args.get("input", False):
    vs.stop()

# otherwise, release the video file pointer
else:
    vs.release()

# close any open windows
cv2.destroyAllWindows()
```

Fig 4.21 People Exiting Outlet Code

ii. Age and Gender Detection

video_age_detection.py

```
# ----------------------------
#   USAGE
# ----------------------------
# python video_age_detection.py --face face_detector --age age_detector --gender gender_detector

# ----------------------------
#   IMPORTS
# ----------------------------
# Import the necessary packages
from imutils.video import VideoStream,FileVideoStream
import numpy as np
import argparse
import imutils
import time
import cv2
import re
import os
import mysql.connector

cnx = mysql.connector.connect(user='root', password='root',
                              host='127.0.0.1',
                              database='freshmartstoredb')

cursor = cnx.cursor()


# ----------------------------
#   FUNCTIONS
# ----------------------------
def detect_and_predict_age(frame, faceNet, ageNet, genderNet, minConf=0.5):
    """
        Age detection and prediction function.
        :param frame: Input image frame
        :param faceNet: Face detection model
        :param ageNet: Age detection model
        :param genderNet: Gender detection model
        :param minConf: Minimum confidence probability
        :return: results with predictions
```

Fig 4.22 Age and Gender Detection Code

video_age_detection.py

```
        :return: results with predictions
    """
    # Define the list of age buckets that the age predictor will predict
    AGE_BUCKETS = ["(0-2)", "(4-6)", "(8-12)", "(15-20)", "(25-32)", "(38-43)", "(48-53)", "(60-100)"]
    #List of genders
    gender_list = ['Male', 'Female']
    # Initialize the results list
    results = []
    # Grab the dimensions of the frame and then construct a blob from it
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (300, 300), (104.0, 177.0, 123.0))
    # Pass the blob through the network and obtain the face detections
    faceNet.setInput(blob)
    detections = faceNet.forward()
    # Loop over the detections
    for i in range(0, detections.shape[2]):
        # Extract the confidence (i.e, probability) associated with the prediction
        confidence = detections[0, 0, i, 2]
        # Filter out weak detections by ensuring the confidence is greater than the minimum confidence probability
        if confidence > minConf:
            # Compute the (x, y) coordinates of the bounding box for the object
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")
            # Extract the ROI of the frame
            face = frame[startY:endY, startX:endX]
            # Ensure that the ROI is sufficiently large
            if face.shape[0] < 20 or face.shape[1] < 20:
                continue
            # Construct a blob from *just* the face ROI
            faceBlob = cv2.dnn.blobFromImage(face, 1.0, (227, 227), (78.4263377603, 87.7689143744, 114.895847746),
                                             swapRB=False)
            # Make predictions on the age and find the age bucket with the largest corresponding probability
            ageNet.setInput(faceBlob)
            preds = ageNet.forward()
            i = preds[0].argmax()
            age = AGE_BUCKETS[i]
            ageConfidence = preds[0][i]
```

Fig 4.23 Age and Gender Detection Code

video_age_detection.py

```
                # Make predictions on the gender
                genderNet.setInput(faceBlob)
                gender_preds = genderNet.forward()
                i = gender_preds[0].argmax()
                gender = gender_list[i]
                genderConfidence = gender_preds[0][i]

                age = re.sub(r'[()]','',age)
                min_age = age.split('-')[0]
                max_age = age.split('-')[1]

                # details = ("INSERT INTO customer_details "
                #     "(gender,minimum_age,maximum_age) "
                #     "VALUES (%s, %s, %s)")

                # record = (gender,min_age,max_age)
                # cursor.execute(details, record)
                # cnx.commit()

                print('data inserted')


                # Construct a dictionary consisting of both the face bounding box location along with the age prediction
                # then update the results list
                d = {
                    "loc": (startX, startY, endX, endY),
                    "age": (age),
                    "gender":(gender),
                }
                results.append(d)
        # Return results to the calling function
        return results


    # Construct the argument parser and parse the arguments
    ap = argparse.ArgumentParser()
    ap.add_argument("-f", "--face", required=True, help="Path to face detector model directory")
```

Fig 4.24 Age and Gender Detection Code

video_age_detection.py

```
ap = argparse.ArgumentParser()
ap.add_argument("-f", "--face", required=True, help="Path to face detector model directory")
ap.add_argument("-a", "--age", required=True, help="Path to age detector model directory")
ap.add_argument("-g", "--gender", required=True, help="Path to gender detector model directory")
ap.add_argument("-c", "--confidence", type=float, default=0.5, help="Minimum probability to filter weak detections")
args = vars(ap.parse_args())

# Load the serialized face detector model from disk
print("[INFO] Loading face detector model...")
prototxtPath = os.path.sep.join([args["face"], "deploy.prototxt"])
weightsPath = os.path.sep.join([args["face"], "res10_300x300_ssd_iter_140000.caffemodel"])
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# Load the serialized age detector model from disk
print("[INFO] Loading age detector model...")
prototxtPath = os.path.sep.join([args["age"], "age_deploy.prototxt"])
weightsPath = os.path.sep.join([args["age"], "age_net.caffemodel"])
ageNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# Load the serialized gender detector model from disk
print("[INFO] Loading gender detector model...")
prototxtPath = os.path.sep.join([args["gender"], "deploy_gender.prototxt"])
weightsPath = os.path.sep.join([args["gender"], "gender_net.caffemodel"])
genderNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# Initialize the video stream and allow the camera sensor to warm up
print("[INFO] Starting video stream...")

#If video file, uncomment this
vs = FileVideoStream(path='store.mp4').start()

#If directly want to stream web camera, refer this line
# vs = VideoStream(0).start()

time.sleep(2.0)

# Loop over the frames from the video stream
while True:
```

Fig 4.25 Age and Gender Detection Code

video_age_detection.py

```
vs = FileVideoStream(path='store.mp4').start()

#If directly want to stream web camera, refer this line
# vs = VideoStream(0).start()

time.sleep(2.0)

# Loop over the frames from the video stream
while True:
    # Grab the frame from the threaded video stream and resize it to have a maximum width of 400 pixels
    frame = vs.read()
    frame = imutils.resize(frame, width=600)
    # Detect the faces in the frame and for each face in the frame predict the age
    results = detect_and_predict_age(frame, faceNet, ageNet, genderNet, minConf=args["confidence"])
    # Loop over the face age detection results
    for r in results:
        # Draw the bounding box of the face along with the associated age
        (startX, startY, endX, endY) = r["loc"]
        y = startY - 10 if startY - 10 > 10 else startY + 10
        cv2.rectangle(frame, (startX, startY), (endX, endY), (0, 0, 255), 2)
        #cv2.putText(frame, text, (startX, y), cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255), 2)
        overlay_text = "%s" % (r["gender"])
        cv2.putText(frame, overlay_text, (startX, y), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 255), 2, cv2.LINE_AA)
    # Show the output frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF
    # If the 'q' key was pressed, break from the loop
    if key == ord("q"):
        break

# Do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

Fig 4.26 Age and Gender Detection Code

iii. Website Code (Index.html)

index.html

```
{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>
  <!-- basic -->
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <!-- mobile metas -->
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="viewport" content="initial-scale=1, maximum-scale=1">
  <!-- site metas -->
  <title>FreshMart Store</title>
  <meta name="keywords" content="">
  <meta name="description" content="">
  <meta name="author" content="">
  <!-- fevicon -->
  <link rel="icon" href="{% static 'images/fevicon.png' %}" type="image/gif" />
  <!-- bootstrap css -->
  <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">
  <!-- style css -->
  <link rel="stylesheet" href="{% static 'css/style.css' %}">
  <!-- Responsive-->
  <link rel="stylesheet" href="{% static 'css/responsive.css' %}">
  <!-- Scrollbar Custom CSS -->
  <link rel="stylesheet" href="{% static 'css/jquery.mCustomScrollbar.min.css' %}">
  <!-- Tweaks for older IEs-->
```

Fig 4.27 Index.html

index.html

```
    <!-- header inner -->
    <div class="header-top">
      <div class="header">
        <div class="container-fluid">
          <div class="row">
            <div class="col-xl-2 col-lg-4 col-md-4 col-sm-3 col logo_section">
              <div class="full">
                <div class="center-desk">
                  <div class="logo">
                    <a href=""><img src="{% static 'images/logo.png' %}" alt="#" /></a>
                  </div>
                </div>
              </div>
            </div>
            <div class="col-xl-10 col-lg-8 col-md-8 col-sm-9">
              <div class="menu-area">
                <div class="limit-box">
                  <nav class="main-menu ">
                    <ul class="menu-area-main">
                      <li class="active"><a href="">Home</a></li>
                      <li> <a href="#about">About</a> </li>
                      <li> <a href="#promise">Our Promise</a> </li>
                      <li> <a href="#testimonial">Testomonial</a> </li>
                      <li> <a href="#store">Our Store</a> </li>

                      <li> <a href="#"><img src="{% static 'icon/icon_b.png' %}" alt="#" /></a></li>
                    </ul>
                    <br>
                    <div class="row">
```

Fig 4.28 Index.html

index.html

```
96      </div>
97
98      <!-- end header -->
99      <section class="slider_section">
100      <div id="myCarousel" class="carousel slide" data-ride="carousel">
101        <ol class="carousel-indicators">
102          <li data-target="#myCarousel" data-slide-to="0" class="active"></li>
103          <li data-target="#myCarousel" data-slide-to="1"></li>
104          <li data-target="#myCarousel" data-slide-to="2"></li>
105        </ol>
106        <div class="carousel-inner">
107          <div class="carousel-item active">
108
109            <div class="container-fluid padding_dd">
110              <div class="carousel-caption">
111                <div class="row">
112                  <div class="col-xl-6 col-lg-6 col-md-6 col-sm-12">
113                    <div class="text-bg">
114                     <span>Welcome To FreshMart</span>
115                      <h1>Grocery Store</h1>
116                      <p>Always Fresh! Forever Original :) <br>
117                      Bringing freshness straight into your everyday life.
118                      </p>
119                    </div>
120                  </div>
121                  <div class="col-xl-6 col-lg-6 col-md-6 col-sm-12">
122                    <div class="images_box">
123                      <figure><img src="{% static 'images/slider1.png' %}"></figure>
124                    </div>
```

Fig 4.29 Index.html

index.html

```
156          <div class="carousel-item">
157            <div class="container-fluid padding_dd">
158              <div class="carousel-caption ">
159                <div class="row">
160                  <div class="col-xl-6 col-lg-6 col-md-6 col-sm-12">
161                    <div class="text-bg">
162                      <span>Welcome To FreshMart</span>
163                      <h1>Grocery Store</h1>
164                      <p>Always Fresh! Forever Original :) <br>
165                      Bringing freshness straight into your everyday life.
166                      </p>
167                    </div>
168                  </div>
169                  <div class="col-xl-6 col-lg-6 col-md-6 col-sm-12">
170                    <div class="images_box">
171                      <figure><img src="{% static 'images/slider3.png' %}"></figure>
172                    </div>
173                  </div>
174                </div>
175              </div>
176            </div>
177          </div>
178        </div>
179      </div>
180    </div>
181    <a class="carousel-control-prev" href="#myCarousel" role="button" data-slide="prev">
182      <span class="carousel-control-prev-icon" aria-hidden="true"></span>
183      <span class="sr-only">Previous</span>
184    </a>
```

Fig 4.30 Index.html

index.html

```
186         <span class="carousel-control-next-icon" aria-hidden="true"></span>
187         <span class="sr-only">Next</span>
188       </a>
189     </div>
190
191 </section>
192 </div>
193 </header>
194
195
196
197 <!-- about  -->
198 <div id="about" class="about">
199   <div class="container-fluid">
200     <div class="row">
201       <div class="col-xl-6 col-lg-6 col-md-6 col-sm-12">
202         <div class="about-box">
203           <h2>About us</h2>
204           <p>FreshMart Welcomes YOU ! <br>
205           We give you the most minimal costs on the entirety of your grocery needs.<br>
206           It is India's biggest grocery store. With more than 18,000 items and over 1000 brands on our list, you
              discover all that you are searching for.
207            It is a low-cost general store that gets items crosswise over classifications like Grocery, Natural
               Products, Fruits, Vegetables, Rice, Spices and, Seasonings to Packaged items, Beverages, Personal
               consideration items - we have everything.
208           </p>
209         </div>
210       </div>
211       <div class="col-xl-6 col-lg-6 col-md-6 col-sm-12 padding_rl">
```

Fig 4.31 Index.html

index.html

```
222
223
224 <!-- vegetable -->
225 <div id="promise" class="vegetable">
226   <div class="container">
227     <div class="row">
228       <div class="col-md-12">
229         <div  class="titlepage">
230           <h2>Freshness - The<strong class="llow"> promise</strong> of Quality </h2>
231         </div>
232       </div>
233     </div>
234     <div class="row">
235       <div class="col-xl-5 col-lg-5 col-md-5 col-sm-12 ">
236         <div class="vegetable_shop">
237           <h3>Fresh Vegetable in our Store</h3>
238           <p>The freshness of the vegetable being delivered to you is unquestionable. All these vegetables are
              from verified and reputed vendors, giving no room to worry about quality.
239           Striving to sustain the nutritive benefits of the vegetables, FreshMart takes its quality very seriou
              abides by a very strict quality standard.</p>
240         </div>
241       </div>
242       <div class="col-xl-7 col-lg-7 col-md-7 col-sm-12 ">
243         <div class="vegetable_img">
244          <figure><img src="{% static 'images/v1.jpg' %}" alt="#"/></figure>
245         </div>
246       </div>
247       <div class="col-xl-8 col-lg-8 col-md-8 col-sm-12 ">
248         <div class="vegetable_img margin_top">
```

Fig 4.32 Index.html

index.html

```
280                 </ul>
281                 <!-- The slideshow -->
282                 <div class="carousel-inner">
283                     <div class="carousel-item">
284                         <div class="testomonial_section">
285
286                             <div class="full testimonial_cont">
287                                 <div class="row">
288                                     <div class="col-xl-3 col-lg-3 col-md-3 col-sm-12 pa_right">
289                                         <div class="testomonial_img">
290                                             <figure><img src="{% static 'images/review1.jpg' %}" alt="#"/></fig
291                                             <i><img src="{% static 'images/test_con.png' %}" alt="#"/></i>
292                                         </div>
293                                     </div>
294                                     <div class="col-xl-9 col-lg-9 col-md-9 col-sm-12 pa_left">
295                                         <div class="cross_inner">
296                                             <h3>Craig McKay<br><strong class="ornage_color">customer</strong></
297                                             <p>The staff there, are just wonderful people. They make you feel w
                                            and truly appreciate your visit to their store. They take pride in
                                            they do. I drive an hour to get the banana bread. Grocery Quality i
                                            amazing !
298                                             </p>
299
300                                         </div>
301                                     </div>
302                                 </div>
303                             </div>
304                         </div>
305                     </div>
```

Fig 4.33 Index.html

index.html

```
316                                             <figure><img src="{% static 'images/review2.jpg' %}" alt="#"/></fig
317                                             <i><img src="{% static 'images/test_con.png' %}" alt="#"/></i>
318                                         </div>
319                                     </div>
320                                     <div class="col-xl-9 col-lg-9 col-md-9 col-sm-12 pa_left">
321                                         <div class="cross_inner">
322                                             <h3>Stefan Ferrero<br><strong class="ornage_color">customer</strong
323                                             <p>Had wonderful experience shopping at FreshMart today around 9:30
                                            The store was clean, well stocked and in great order. It was great
                                            shopping experience to be able to find the things I needed in an ex
                                            manner. I really enjoyed shopping there!!
324                                             </p>
325
326                                         </div>
327                                     </div>
328                                 </div>
329                             </div>
330                         </div>
331
332                     </div>
333
334                     <div class="carousel-item">
335
336                         <div id="testomonial" class="testomonial_section">
337                             <div class="full center">
338                             </div>
339                             <div class="full testimonial_cont ">
340                                 <div class="row">
341                                     <div class="col-xl-3 col-lg-3 col-md-3 col-sm-12 pa_right">
```

Fig 4.34 Index.html

index.html

```
<div id="store">
<div class="container">
<div  class="titlepage">
  <h2> Our <strong class="llow">fresh mart </strong>Store </h2>
</div>
<div class="row">
 <div class="col-md-4">
   <div class="thumbnail">
      <img src="{% static 'images/1.jpg' %}" alt="Lights" style="width:100%">
   </div>
 </div>
 <div class="col-md-4">
   <div class="thumbnail">
      <img src="{% static 'images/2.jpg' %}" alt="Nature" style="width:100%">
   </div>
 </div>
 <div class="col-md-4">
   <div class="thumbnail">
      <img src="{% static 'images/3.jpg' %}" alt="Fjords" style="width:100%">
   </div>
 </div>
</div>
<br><br>
<div class="row">
 <div class="col-md-4">
   <div class="thumbnail">
      <img src="{% static 'images/4.jpg' %}" alt="Lights" style="width:100%">
   </div>
```

Fig 4.35 Index.html

index.html

```
</div>



<!--  footer -->
<footr>
  <div class="copyright">
    <div class="container">
      <p>Copyright © 2021 FreshMart</p>
    </div>
  </div>
      </footr>
      <!-- end footer -->
      <!-- Javascript files-->
      <script src="{% static 'js/jquery.min.js' %}"></script>
      <script src="{% static 'js/popper.min.js' %}"></script>
      <script src="{% static 'js/bootstrap.bundle.min.js' %}"></script>
      <script src="{% static 'js/jquery-3.0.0.min.js' %}"></script>
      <script src="{% static 'js/plugin.js' %}"></script>
      <!-- sidebar -->
      <script src="{% static 'js/jquery.mCustomScrollbar.concat.min.js' %}"></script>
      <script src="{% static 'js/custom.js' %}"></script>
      <script src="https:cdnjs.cloudflare.com/ajax/libs/fancybox/2.1.5/jquery.fancybox.min.js"></script>



</body>
```

Fig 4.36 Index.html

iv. Website Code (Django Code)

settings.py

```
"""
Django settings for freshmartstore project.

Generated by 'django-admin startproject' using Django 3.2.5.

For more information on this file, see
https://docs.djangoproject.com/en/3.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.2/ref/settings/
"""

from pathlib import Path

import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-2=*r%=4x$^-yocw4m=+(4ze1h^xms9)ph^x2)mu6&bjh&ej1$s'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
```

Fig 4.37 Django Code

settings.py

```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'freshmartstore.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': ['freshmartstore/templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

Fig 4.38 Django Code

settings.py

```
WSGI_APPLICATION = 'freshmartstore.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/3.2/topics/i18n/

LANGUAGE_CODE = 'en-us'
```

Fig 4.39 Django Code

settings.py

```
# Internationalization
# https://docs.djangoproject.com/en/3.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-files/
STATIC_URL = '/static/'

#This will create the static folder and will access everything from this directory
STATIC_ROOT = os.path.join(BASE_DIR,'static')

#where to look for all our staticfiles(here we have keep data --  photo)
STATICFILES_DIRS= [
    os.path.join(BASE_DIR,'freshmartstore/static')
]


# Default primary key field type
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

Fig 4.40 Django Code

v. SQL Script

SQL File 1 freshmart

Limit to 1000 rows

```
#creating database
create database freshmartstoredb;
use freshmartstoredb;

CREATE TABLE customer_details (
gender varchar(6),
minimum_age int,
maximum_age int
);

select * from customer_details;
```

Fig 4.41 SQL Script Code

```
CREATE TABLE people_in_outlet (
people_entry_time timestamp,
people_in int
);

select * from people_in_outlet;
```

Fig 4.42 SQL Script Code

```
CREATE TABLE people_exit_outlet (
people_exit_time timestamp,
people_exit int
);

select * from people_exit_outlet;
```

Fig 4.43 SQL Script Code

vi. Requirements

requirements.txt

```
asgiref==3.4.1
Django==3.2.7
pytz==2021.1
sqlparse==0.4.2
dlib==19.19.0
imutils==0.5.4
numpy==1.21.2
opencv-python==4.5.3.56
pip==21.2.4
scipy==1.7.1
setuptools==57.4.0
wheel==0.36.2
```

5. TESTING

5.1 TEST CASES AND REPORTS

In this project, the web application is tested by using a series of different tests whose sole purpose is to exercise the full application. It is an investigation conducted to provide stakeholders with information about the quality of the product or the services under test. It also describes the scope of testing, testing techniques to be used, resources required for testing and the schedule of intended test activities. The scope helps in identifying test items and the features to be tested. It also contains details of who will perform a given task.

Fig 5.1 shows the different test cases and their reports.

Table 5.1 Test Cases and Reports

Test case id	**Test case Objective**	**Steps**	**Output**	**Status**
TC_01	Insertion of values in Database	1. Counting of people in outlet 2. People count, getting inserted into database 3. commit of database	Real Time people count getting inserted into the Database	Display of successful insertion into database on terminal

TC_02	Insertion of Values in Database	1. Detecting the Age and Gender of the people entering the outlet 2. Age and gender estimation inserted into the database 3. commit of Database	Real Time Age and gender of people getting inserted into the Database	Display of successful insertion into database on terminal

6. CONCLUSION

6.1 DESIGN AND IMPLEMENTATION ISSUES

This work presents an approach that is very effective in displaying the real-time crowd count on the website. This information is beneficial to the customers who are visiting the trading outlets to save their time and avoid overcrowding places. The Python libraries , Algorithms and Pre-trained Caffe Deep learning models for Object Detection, Object Tracking (people), Age & Gender estimation were effective in tracing the people by processing the real-time video feed taken through a camera from the entry and exit points of the outlets. The proposed system performs admirably in situations where manual counting is simply not possible and the outlets have to manage the crowd amidst the pandemic. The project has a large future scope and is expandable in terms of scale. It can be developed to track or research crowd movement, which might be useful in managing many different social outlets, restaurants etc.

6.2 ADVANTAGES AND LIMITATIONS

The following are the advantages of the proposed system:

- The proposed system is featured with a website to display the crowd count of the people.
- This system will benefit the customers to make informed decisions about visiting the store if it is overcrowded, as they can schedule their visits accordingly to avoid the crowd.
- This system will also benefit trading outlets to manage their crowd and further analyze the crowd by the age & gender of the customers.

The limitations are as follows:

- The limitations of the existing system are the crowd count tracked by the cameras in these retail stores, benefitting only trading outlets to manage their crowd.
- This crowd count information is not helping the public in any sense.
- The outlets do not provide a system where real-time information is displayed on the website or web application.

7. FUTURE WORK

In future, we would like to upgrade our project using Cloud Services

There are many cloud services available for Object Detection and Video Intelligence in AWS and GCP.

We would like to integrate them in our project and use their functions in a more optimized manner and thus making our project more scalable and efficient.

REFERENCES

[1] N. Pouw, Caspar AS, et al. "Monitoring physical distancing for crowd management: Real-time trajectory and group analysis." *PloS one* 15.10 (2020): e0240963..

[2] Conte, Donatello, et al. "A method for counting moving people in video surveillance videos." *EURASIP Journal on Advances in Signal Processing* 2010 (2010): 1-10.

[3] J. Sreenu, G., and MA Saleem Durai. "Intelligent video surveillance: a review through deep learning techniques for crowd analysis." *Journal of Big Data* 6.1 (2019): 1-27.

[4] S. Singh, Dushyant Kumar, et al. "Human crowd detection for city wide surveillance." *Procedia Computer Science* 171 (2020): 350-359.

[5] Polanco, Liliana Duran, and Mario Siller. "Crowd managementCOVID-19." *Annual Reviews in Control* (2021).

[6] Bhangale, Ujwala, et al. "Near Real-time Crowd Counting using Deep Learning Approach." Procedia Computer Science 171 (2020): 770-779.

[7] Raghavachari, Chakravartula, et al. "A comparative study of vision based human detection techniques in people counting applications." *Procedia Computer Science* 58 (2015): 461-469.

[8] Liu, Zhi, et al. "Crowd counting method based on convolutional neural network with global density feature." *IEEE Access* 7 (2019): 88789-88798.

[9] Schlögl, Thomas, et al. *Evaluation of people counting systems*. na, 2001.

[10] Cahyadi, Nanang, and Budi Rahardjo. "Literature Review of People Counting." *2021 International Conference on Artificial Intelligence and*

Mechatronics Systems (AIMS). IEEE, 2021.

[11] Myint, Ei Phyu, and Myint Myint Sein. "People Detecting and Counting System." 2021 IEEE 3rd Global Conference on Life Sciences and Technologies (LifeTech). IEEE, 2021.

[12] Khan, Javed Ali, et al. "Crowd intelligence in requirements engineering: Current status and future directions." *International working conference On requirements engineering: Foundation for software quality*. Springer, Cham,2019

[13] Boukerche, Azzedine, and Rodolfo WL Coutinho. "Crowd management: The overlooked component of smart transportation systems." *IEEE Communications Magazine* 57.4 (2019): 48-53.

[14] Bendali-Braham, Mounir, et al. "Recent trends in crowd analysis: A review." *Machine Learning with Applications* (2021): 100023.

[15] Stojmenovic, Milos, et al. "Health Informatics: Applications of Mobile and Wireless Technologies." (2019