THE KALYANI SCHOOL

ACADEMIC YEAR : 2024-25

NAME	**:**	**SUBHOJIT GHOSH**
CLASS	**:**	**XII - E**
ROLL NO	**:**	**13**
SUBJECT	**:**	**COMPUTER SCIENCE**
SUBJECT CODE	**:**	**083**

The Kalyani School

॥ अक्षयम् ज्ञानम् ॥

INDEX

THE KALYANI SCHOOL

CERTIFICATE

COMPUTER SCIENCE PROJECT

FOR AISSCE EXAM 2024-25

This is to certify that the Computer Science(083) Practical was done by

SUBHOJIT GHOSH, of Grade **XII E**, Roll No. **13**, under my guidance and supervision. It is further certified that the work is original.

Name and Signature of the teacher

Ms. Hemant Sharma

_____

Name and signature of the principal:

Ms. Nirmal Waddan

_____

Date:

School Seal

ACKNOWLEDGEMENT

Apart from the efforts of me, the success of any project depends largely on the encouragement and guidelines of many others.

I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.

I express a deep sense of gratitude to almighty God for giving me strength for the successful completion of the project.

I express my heartfelt gratitude to my parents for constant encouragement while carrying out this project.

I gratefully acknowledge the contribution of the individuals who contributed in bringing this project up to this level, who continue to look after me despite my flaws.

My sincere thanks to our computer science teacher **MR. HEMANT SHARMA**, who critically reviewed my project and helped in solving each and every problem, occurred during implementation of the project.

The guidance and support received from all the members who contributed and who are contributing to this project, was vital for the success of the project.

I am grateful for their constant support and help.

DECLARATION

I hereby declare that the project word entitled **GESTURE BASED ORDERING SYSTEM** submitted to the Department of Computer Science, The Kalyani School, Manjiri is a record of the original work done by me under the guidance of **MR. HEMANT SHARMA**. All the coding is the result of my personal efforts.

Name of the Student: **SUBHOJIT GHOSH**
Class - **XII - E**

INTRODUCTION

TheGESTURE BASED ORDERING SYSTEM is an innovative software designed to revolutionize ordering food through a touchless experience. Customers can simply use hand gestures (1, 2, or 3 fingers) to make their selections, ensuring an intuitive and hygienic interaction. This gesture-based control is particularly useful in environments where minimizing physical contact is essential.

The system's user-friendly GUI is designed for simplicity, making it accessible to all age groups. Order data is securely stored in a CSV file on the user's PC, ensuring quick access without needing an internet connection while maintaining data privacy and security.

This system is versatile, adaptable for use in restaurants, grocery stores, pharmacies, and other retail environments where streamlined, contactless ordering is crucial. It enhances the speed and accuracy of the ordering process, benefiting both younger and older generations, and reducing errors for improved customer satisfaction.

In summary, this project is a forward-thinking application that combines innovative technology with practical functionality, offering a touchless, user-friendly, and efficient solution for modern ordering needs.

HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements:

1. **Computer**: A standard desktop or laptop computer would be sufficient.

2. **Storage**: Enough storage space to store the program files and the Excel workbook (if the data is stored locally).

3. **Memory** (RAM): Adequate RAM to handle the size of data one is working with. For this program, 4 -6 of RAM should be sufficient.

4. **Webcam**: An HD webcam is required for real time video feed and to detect hand gestures.

Software Requirements:

1. **Programming Language**: Python should be installed in the PC.

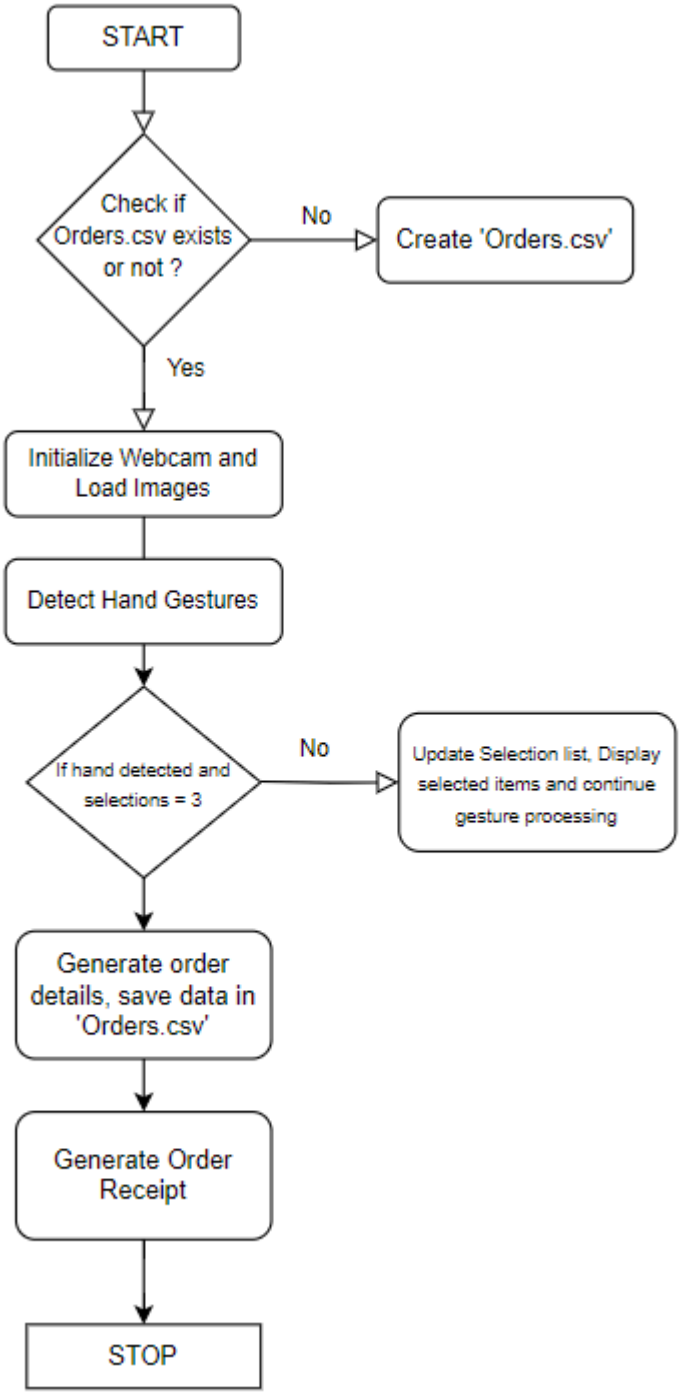
2. **Integrated Development Environment (IDE)**: - An IDE is to be installed to run the program. Examples include Visual Studio Code, PyCharm, etc.

3. **Libraries or Modules**: All the modules should be installed for the program to run efficiently and successfully.

4. **Operating System**: The program can run on Windows, macOS, or Linux.

FLOWCHART / PSEUDOCODE / ALGORITHM

Flowchart



Algorithm / Pseudocode

IMPORT necessary modules: os, csv, cvzone.HandDetector, datetime, uuid, time, tkinter, pillow, threading

FUNCTION display(order)

CREATE a GUI window with order details

Close the window after 3 seconds

RUN the GUI main loop

FUNCTION main()

CHECK if 'Orders.csv' exists

IF not, CREATE it and write headers

INITIALIZE webcam and load images for background, modes, and icons

SET variables for mode, selection, counter, etc.

CREATE HandDetector for gesture recognition

LOOP until exit:

READ webcam frame, detect hands

UPDATE background with live feed and mode selection

IF hand detected and modeType < 3:

DETECT finger gesture and set selection

IF selection confirmed, update modeType and selectionList

DISPLAY selected icons based on selectionList

IF all selections made (modeType == 3):

CREATE order with unique ID, timestamp, drink, sugar, size

SAVE order to 'Orders.csv'

DISPLAY receipt using display(), RESTART main()

SHOW updated image on screen, wait for next frame

CALL main() to start the program

MODULES AND IN-BUILT FUNCTIONS

All the modules used in the program are -

1. os
2. os.path
3. csv
4. cvzone.HandTrackingModule
5. datetime
6. uuid
7. time
8. tkinter
9. cv2
10. mediapipe
11. pillow
12. threading

DATABASE DESCRIPTION

The data in the program is stored **LOCALLY** in a CSV file named "Orders.csv". This file serves as a simple database to keep track of orders and their details.

VARIABLES DESCRIPTION

1. order :

   ➔ Type: List
   ➔ Description: Stores the details of the current order, including order ID, date/time, drink type, sugar level, and size.

2. root:

   ➔ Type: tkinter.Tk object
   ➔ Description: Represents the main window for displaying the order receipt.

3. file_exists:

   ➔ Type: Boolean
   ➔ Description: Checks if the 'Orders.csv' file exists to determine if headers need to be written.

4. cap:

   ➔ Type: cv2.VideoCapture object
   ➔ Description: Captures video from the webcam for real-time hand gesture detection.

5. imgBackground:

   ➔ Type: Numpy array
   ➔ Description: Holds the background image that is used as the canvas for displaying the live feed and other graphical elements.

6. folderPathModes and folderPathIcons:

➔ Type: String
➔ Description: File paths to directories containing mode images and icon images, respectively.

7. listImgModesPath and listImgIconsPath:

➔ Type: List of Strings
➔ Description: Contains file names of mode and icon images loaded from the respective directories.

8. listImgModes and listImgIcons:

➔ Type: List of Numpy arrays
➔ Description: Stores the mode images and icon images loaded into memory for quick access during display.

9. modeType:

➔ Type: Integer
➔ Description: Tracks the current selection mode (e.g., drink type, sugar level, size) the user is interacting with.

10. selection:

➔ Type: Integer
➔ Description: Holds the current selection made by the user based on hand gestures.

11. counter:

➔ Type: Integer
➔ Description: Counts the frames to determine how long a gesture is held, used to confirm the user's selection.

12. selectionSpeed:

➔ Type: Integer
➔ Description: Controls the speed at which the selection is confirmed based on how long a gesture is maintained.

13. detector:

➔ Type: HandDetector object
➔ Description: Used for detecting and tracking hand gestures in the video feed.

14. modePositions:

➔ Type: List of Tuples
➔ Description: Coordinates for drawing selection indicators on the screen based on the user's choice.

15. counterPause:

➔ Type: Integer
➔ Description: Controls the pause between mode transitions to avoid accidental selections.

16. selectionList:

➔ Type: List of Integers
➔ Description: Stores the final selections made by the user for drink type, sugar level, and size.

17. success:

➔ Type: Boolean
➔ Description: Indicates whether the webcam frame was successfully read.

18. img:

➔ Type: Numpy array
➔ Description: Contains the current frame captured from the webcam.

19. hands:

➔ Type: List
➔ Description: Contains data about detected hands, such as position and finger states.

20. fingers1:

➔ Type: List of Integers
➔ Description: Represents the state of each finger (up or down) on the detected hand, used to identify gestures.

21. r:

➔ Type: List
➔ Description: Contains the order details such as order ID, date/time, drink type, sugar level, and size, which will be written to the CSV file.

22. f:

➔ Type: File object
➔ Description: Used to open the 'Orders.csv' file for appending new order data.

23. cw:

➔ Type: csv.writer object
➔ Description: Used to write order data into the CSV file.

24. imgModePath and imgIconsPath:

➔ Type: String
➔ Description: File paths for mode and icon images during their loading.

25. hand1:

➔ Type: Dictionary
➔ Description: Contains data about the first detected hand.

26. fingers1:

➔ Type: List of Integers
➔ Description: Represents the state of each finger on the detected hand.

27. l1 ,l2, l3, l4, l5

➔ Type: Tkinter Label Widget Objects
➔ Description: Display Text.

28. i

➔ Type: Tkinter Image Widget Objects
➔ Description: Display Image.

29. t

➔ Type: Thread Timer Object
➔ Description: To set a timer of 3 seconds.

SOURCE CODE

```
import os
import os.path
import csv
from cvzone.HandTrackingModule import HandDetector
import datetime
import uuid
import time
import tkinter as tk
from PIL import ImageTk, Image
from threading import Timer
def display(r):


    def d():
        root.destroy()


    order = r
    root = tk.Tk()
    root.title("Order Successfull !")
    root.geometry("320x360")
    root.resizable(0, 0)
    root.configure(background='#23064d')


    img = ImageTk.PhotoImage(Image.open("Resources/success.png"))
    i = tk.Label(root, image=img)
```

```
    l1 = tk.Label(root, text="Order ID - " + str(order[0]), width=100,
font=('calibre', 10, 'bold'),

                  background="#ffff00", anchor="w")

    l2 = tk.Label(root, text="Datetime - " + str(order[1]), width=100,
font=('calibre', 10, 'bold'), background="#fff",

                  anchor="w")

    l3 = tk.Label(root, text="Dish - " + str(order[2]), width=100,
font=('calibre', 10, 'bold'), background="#fff",

                  anchor="w")

    l4 = tk.Label(root, text="Choice - " + str(order[3]), width=100,
font=('calibre', 10, 'bold'), background="#fff",

                  anchor="w")

    l5 = tk.Label(root, text="Serving Size - " + str(order[4]),
width=100, font=('calibre', 10, 'bold'), background="#fff",

                  anchor="w")

    i.pack(pady=10)

    l1.pack(pady=10)

    l2.pack()

    l3.pack()

    l4.pack()

    l5.pack()


    t = Timer(3.0, d)

    t.start()


    root.mainloop()


def main():

    file_exists = os.path.exists('Orders.csv')

    if not file_exists:

        f = open("Orders.csv", "w", newline="")
```

```
        cw = csv.writer(f)
        cw.writerow(["Order ID", "Datetime", "Dish", "Choice", "Serving Size"])
        f.close()
    else:
        print("File already Present !")


    import cv2


    cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
    cap.set(3, 640)
    cap.set(4, 480)


    imgBackground = cv2.imread("Resources/Background.png")


    folderPathModes = "Resources/Modes"
    listImgModesPath = os.listdir(folderPathModes)
    listImgModes = []
    for imgModePath in listImgModesPath:
        listImgModes.append(cv2.imread(os.path.join(folderPathModes, imgModePath)))


    folderPathIcons = "Resources/Icons"
    listImgIconsPath = os.listdir(folderPathIcons)
    listImgIcons = []
    for imgIconsPath in listImgIconsPath:
        listImgIcons.append(cv2.imread(os.path.join(folderPathIcons, imgIconsPath)))


    modeType = 0
```

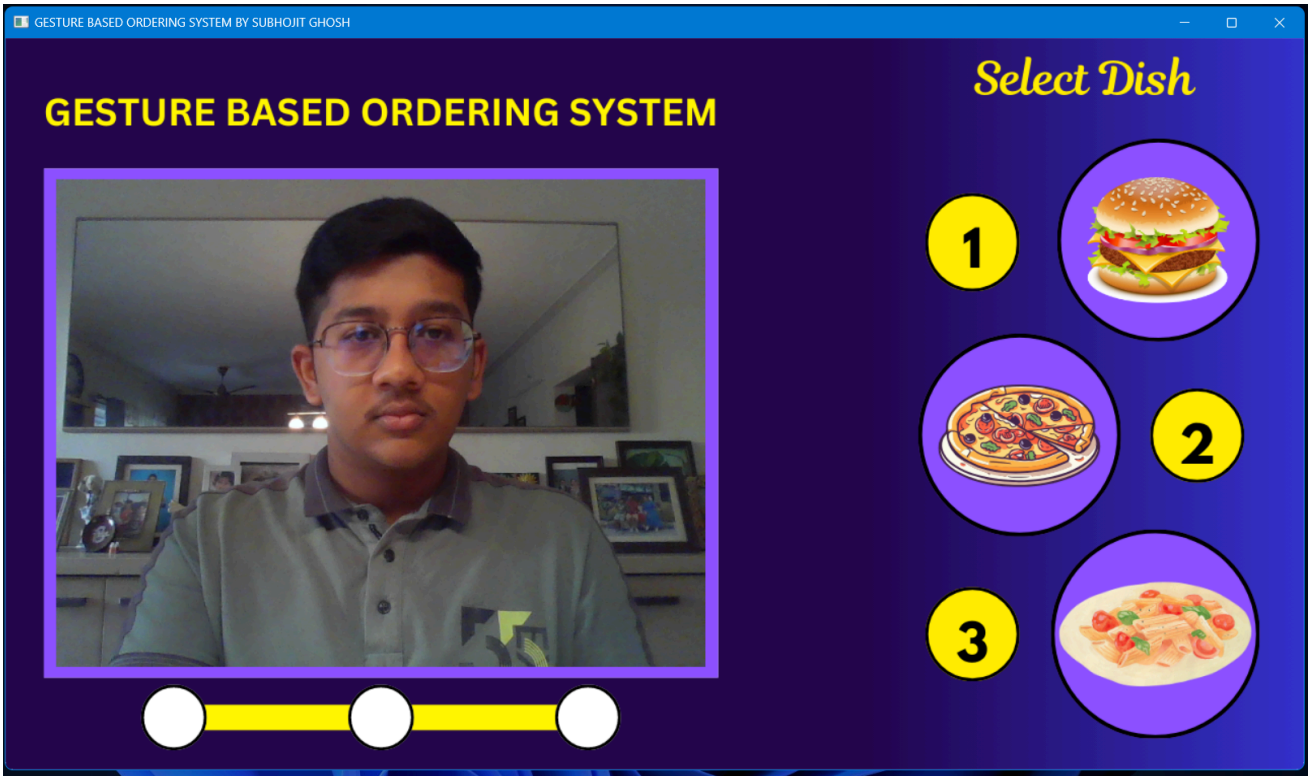
```
selection = -1
counter = 0
selectionSpeed = 7
detector = HandDetector(detectionCon=0.8, maxHands=1)
modePositions = [(1136, 196), (1000, 384), (1136, 581)]
counterPause = 0
selectionList = [-1, -1, -1]

while True:
    success, img = cap.read()
    # img = cv2.flip(img, 1)
    hands, img = detector.findHands(img, flipType=True)

    imgBackground[139:139 + 480, 50:50 + 640] = img
    imgBackground[0:720, 847: 1280] = listImgModes[modeType]

    if hands and counterPause == 0 and modeType < 3:

        hand1 = hands[0]
        fingers1 = detector.fingersUp(hand1)

        if fingers1 == [0, 1, 0, 0, 0]:
            if selection != 1:
                counter = 1
            selection = 1
        elif fingers1 == [0, 1, 1, 0, 0]:
            if selection != 2:
                counter = 1
            selection = 2
```

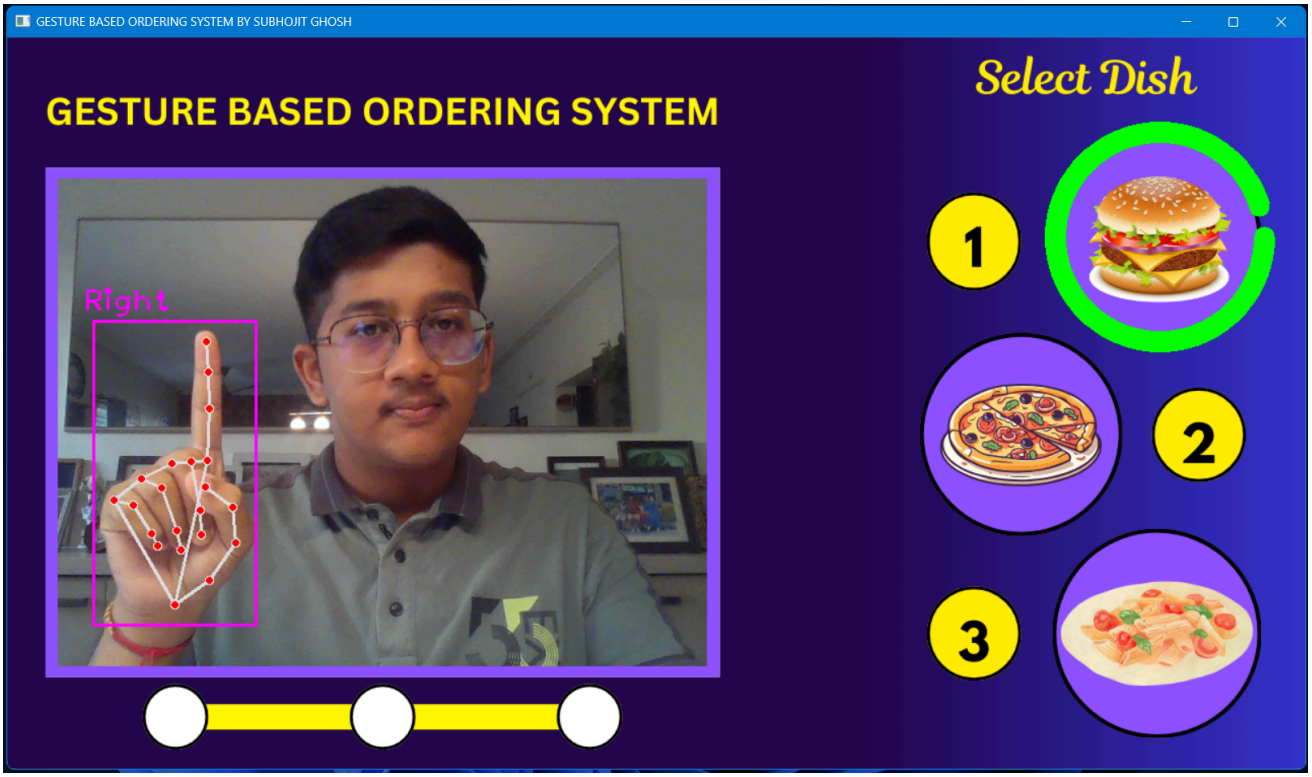
```
            elif fingers1 == [0, 1, 1, 1, 0]:
                if selection != 3:
                    counter = 1
                selection = 3
            else:
                selection = -1
                counter = 0

            if counter > 0:
                counter += 1
                print(counter)

                cv2.ellipse(imgBackground, modePositions[selection - 1],
(103, 103), 0, 0,
                            counter * selectionSpeed, (0, 255, 0), 20)
                if counter * selectionSpeed > 360:
                    selectionList[modeType] = selection
                    modeType += 1
                    counter = 0
                    selection = -1
                    counterPause = 1

        if counterPause > 0:
            counterPause += 1
            if counterPause > 20:
                counterPause = 0

        if selectionList[0] != -1:
```

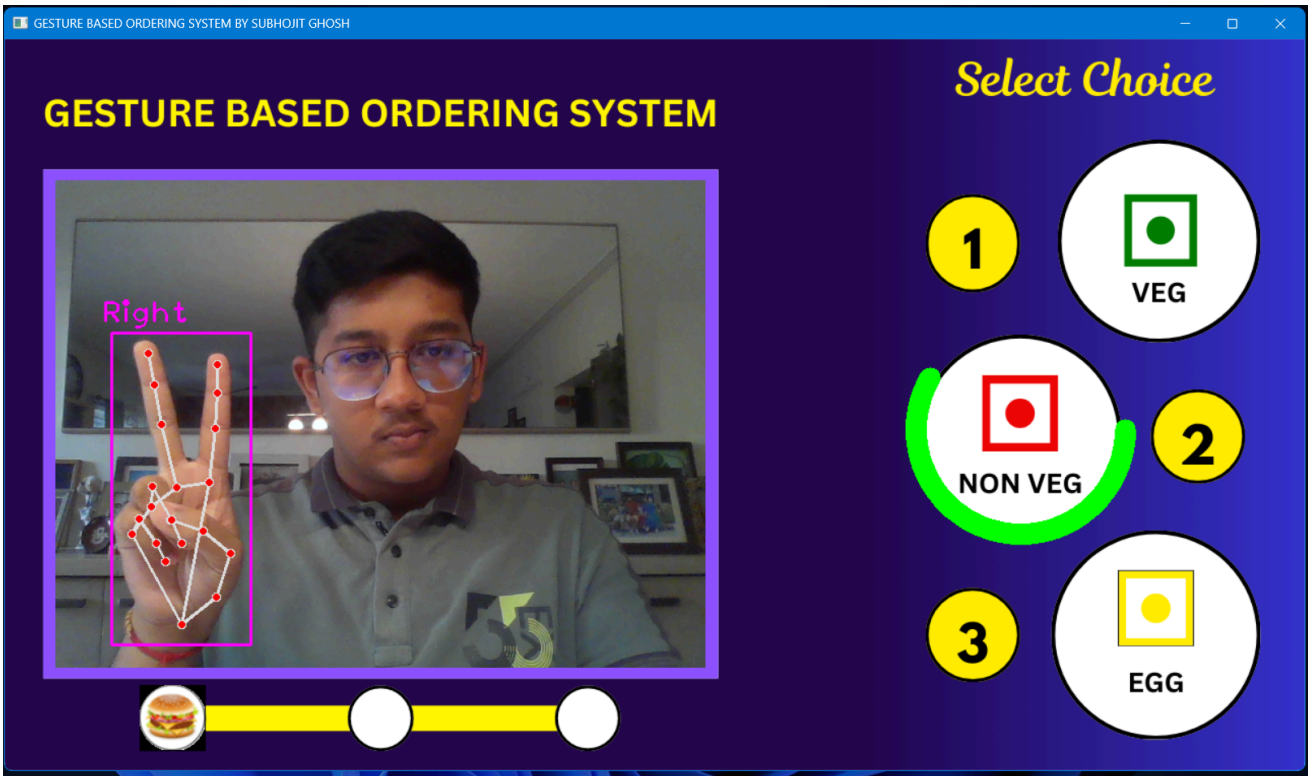
```
            imgBackground[636:636 + 65, 133:133 + 65] =
listImgIcons[selectionList[0] - 1]

        if selectionList[1] != -1:

            imgBackground[636:636 + 65, 340:340 + 65] = listImgIcons[2 +
selectionList[1]]

        if selectionList[2] != -1:

            imgBackground[636:636 + 65, 542:542 + 65] = listImgIcons[5 +
selectionList[2]]


        if modeType == 3:

            imgBackground[0:720, 847: 1280] = listImgModes[3]

            print("Order Placed !")

            r = [uuid.uuid4(), datetime.datetime.now().strftime("%c")]


            if selectionList[0] == 1:

                r.append("Burger")

            if selectionList[0] == 2:

                r.append("Pizza")

            if selectionList[0] == 3:

                r.append("Pasta")


            if selectionList[1] == 1:

                r.append("VEG")

            if selectionList[1] == 2:

                r.append("NON VEG")

            if selectionList[1] == 3:

                r.append("EGG")


            if selectionList[2] == 1:

                r.append("Small")
```

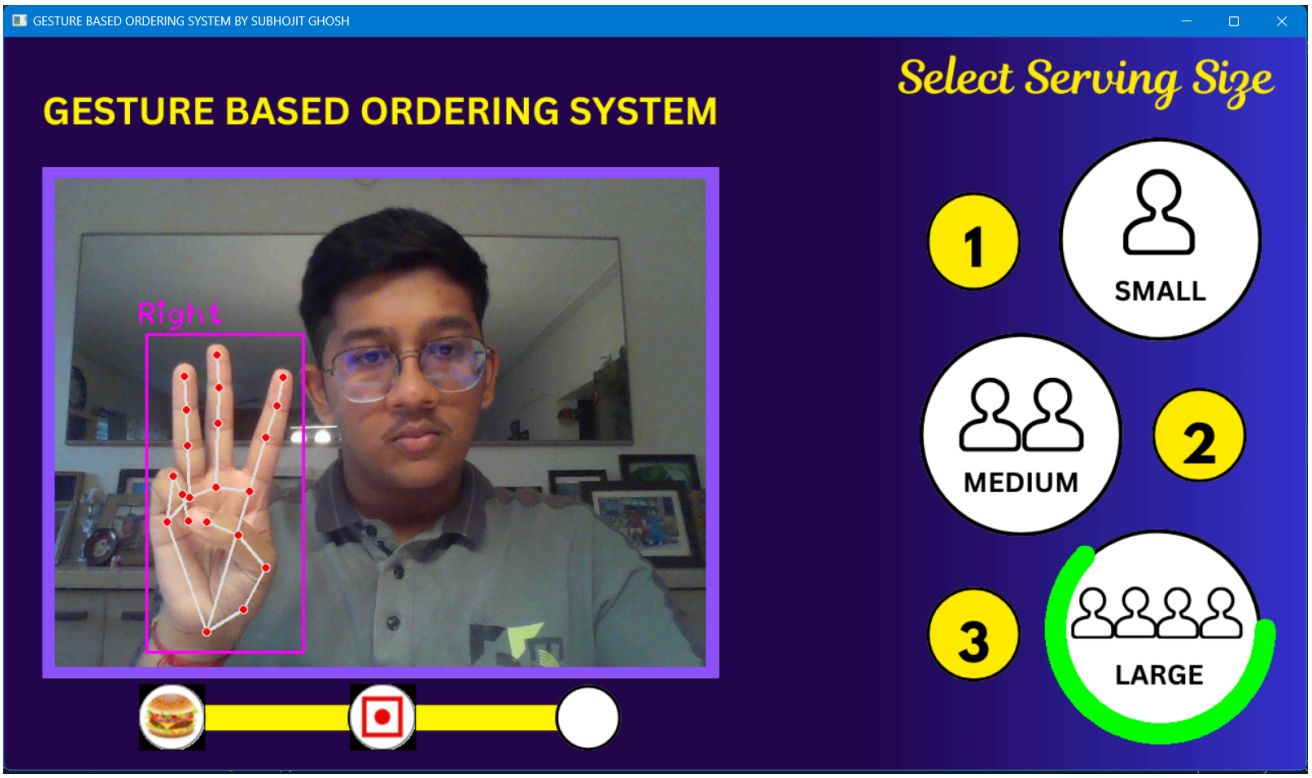
```
            if selectionList[2] == 2:
                r.append("Medium")
            if selectionList[2] == 3:
                r.append("Large")

            f = open("Orders.csv", "a", newline="")
            cw = csv.writer(f)
            cw.writerow(r)
            f.close()

            if f.closed:
                display(r)
                main()

        cv2.imshow("GESTURE BASED ORDERING SYSTEM BY SUBHOJIT GHOSH",
imgBackground)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

        cv2.waitKey(1)



main()
```

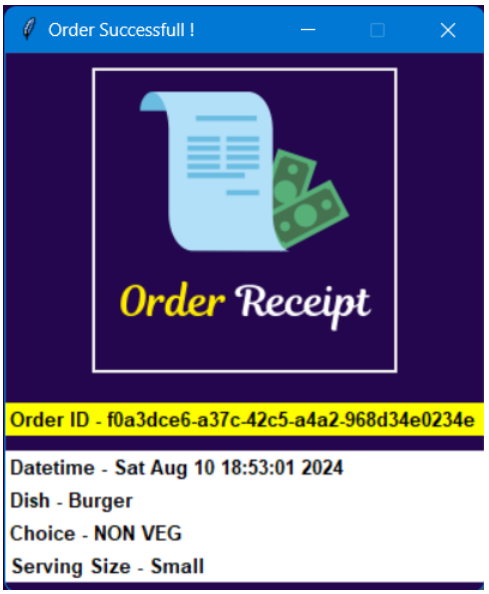
OUTPUT SCREENSHOTS

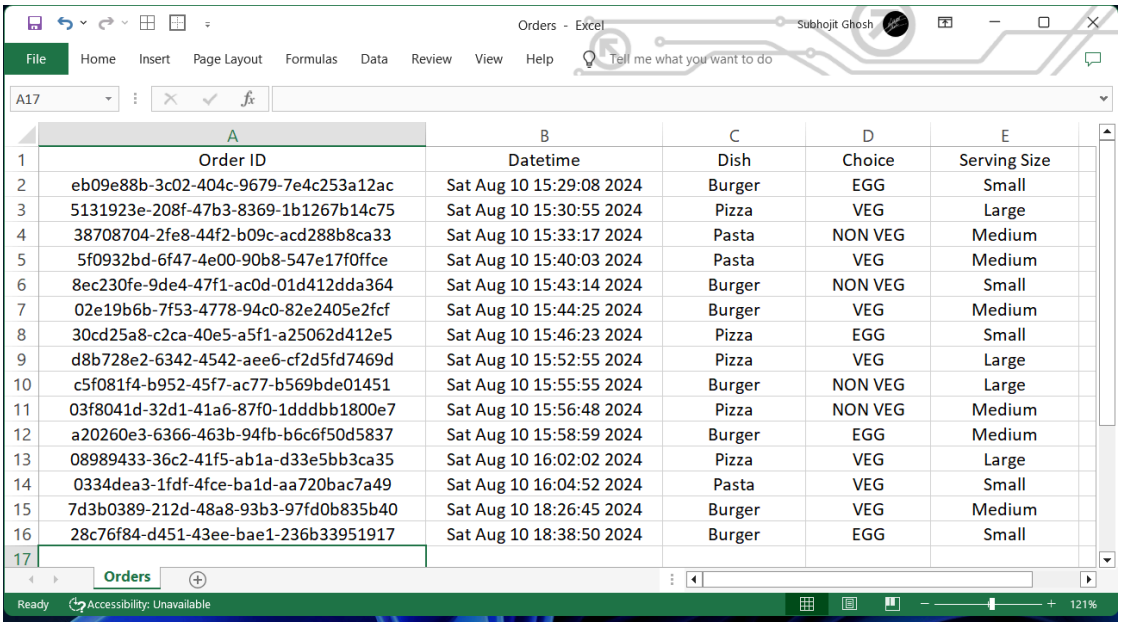





GESTURE BASED ORDERING SYSTEM BY SUBHOJIT GHOSH
GESTURE BASED ORDERING SYSTEM
Right
Select Choice
1
VEG
NON VEG
2
3
EGG


GESTURE BASED ORDERING SYSTEM BY SUBHOJIT GHOSH
GESTURE BASED ORDERING SYSTEM
Right
Select Serving Size
1
SMALL
MEDIUM
2
3
LARGE





Orders - Excel

Subhojit Ghosh

File | Home | Insert | Page Layout | Formulas | Data | Review | View | Help | Tell me what you want to do

A17

	A	B	C	D	E
1	Order ID	Datetime	Dish	Choice	Serving Size
2	eb09e88b-3c02-404c-9679-7e4c253a12ac	Sat Aug 10 15:29:08 2024	Burger	EGG	Small
3	5131923e-208f-47b3-8369-1b1267b14c75	Sat Aug 10 15:30:55 2024	Pizza	VEG	Large
4	38708704-2fe8-44f2-b09c-acd288b8ca33	Sat Aug 10 15:33:17 2024	Pasta	NON VEG	Medium
5	5f0932bd-6f47-4e00-90b8-547e17f0ffce	Sat Aug 10 15:40:03 2024	Pasta	VEG	Medium
6	8ec230fe-9de4-47f1-ac0d-01d412dda364	Sat Aug 10 15:43:14 2024	Burger	NON VEG	Small
7	02e19b6b-7f53-4778-94c0-82e2405e2fcf	Sat Aug 10 15:44:25 2024	Burger	VEG	Medium
8	30cd25a8-c2ca-40e5-a5f1-a25062d412e5	Sat Aug 10 15:46:23 2024	Pizza	EGG	Small
9	d8b728e2-6342-4542-aee6-cf2d5fd7469d	Sat Aug 10 15:52:55 2024	Pizza	VEG	Large
10	c5f081f4-b952-45f7-ac77-b569bde01451	Sat Aug 10 15:55:55 2024	Burger	NON VEG	Large
11	03f8041d-32d1-41a6-87f0-1dddbb1800e7	Sat Aug 10 15:56:48 2024	Pizza	NON VEG	Medium
12	a20260e3-6366-463b-94fb-b6c6f50d5837	Sat Aug 10 15:58:59 2024	Burger	EGG	Medium
13	08989433-36c2-41f5-ab1a-d33e5bb3ca35	Sat Aug 10 16:02:02 2024	Pizza	VEG	Large
14	0334dea3-1fdf-4fce-ba1d-aa720bac7a49	Sat Aug 10 16:04:52 2024	Pasta	VEG	Small
15	7d3b0389-212d-48a8-93b3-97fd0b835b40	Sat Aug 10 18:26:45 2024	Burger	VEG	Medium
16	28c76f84-d451-43ee-bae1-236b33951917	Sat Aug 10 18:38:50 2024	Burger	EGG	Small
17					

Orders

Ready | Accessibility: Unavailable | 121%

BIBLIOGRAPHY

- **Computer Science with Python Class XII by Preeti Arora**