A MINOR PROJECT REPORT ON
JARVIS – AI BASED WAITER

submitted in partial fulfilment of the requirements for the degree of

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

P. Vinod Kumar	O180905
S. Shahid Hussain	O180917
R. Rishi Pandya	O180921
M. Yaswanth	O180932

Under the supervision of

Mr.BEERALA MURALI M.Tech

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES,

ONGOLE CAMPUS

2022-23

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES,

ONGOLE CAMPUS

2022-23

CERTIFICATE

This is to certify that the project report entitled "**Jarvis – AI based waiter" s**ubmitted by **P. Vinod Kumar O180905, S. Shahid Hussain, O180917, R. Rishi Pandya O180921, M. Yaswanth O180932** to Department of Computer Science and Engineering, Rajiv Gandhi University of Knowledge Technologies, Ongole campus, during the academic year 2022-2023 is a partial fulfilment for the award of Under graduate degree of Bachelor of Technology in Computer Science and Engineering is a bonafide record carried out by him under my supervision. The project has fulfilled all the requirements as per as regulations of this institute and in my opinion reached the standard for submission.

Mr. BEERALA MURALI M. Tech,
Assistant Professor,
Department of CSE,
RGUKT, ONGOLE.

Mr. B. SAMPATH BABU M. Tech(Ph.D),
Head of Department,
Department of CSE,
RGUKT, ONGOLE.

Date :

Place :

APPROVAL SHEET

This report entitled **JARVIS – AI based waiter by** P. Vinod Kumar O180905, S. Shahid Hussain O180917, R. Rishi Pandya O180921, M. Yaswanth O180932 is **Assistant Professor Mr**. **BEERALA MURALI M.Tech** approved for the degree of Bachelor of Technology COMPUTER SCIENCE AND ENGINEERING.

Examiner

_____

_____

_____

Supervisor

_____

_____

_____

Date : _____

DECLARATION

We declare that this written submission represents my ideas in my own words and where others ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/ data/ fact/ source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

SIGNATURE

P. Vinod Kumar _____

S. Shahid Hussain _____

R. Rishi Pandya _____

M. Yaswanth _____

Date: _____

Place: _____

ACKNOWLEDGEMENT

We highly indebted to **Assistant Professor BEERALA MURALI M.Tech** for his guidance and constant supervision as well as for providing necessary information regarding the project and also for their kind cooperation encouragement and their support in completing the project.

We would like to express my special gratitude and thanks to our **COMPUTER SCIENCE AND ENGINEERING branch H.O.D Mr. B.SAMPATH BABU M.Tech (Ph.D)** and **Director of Ongole. RGUKT Prof. B. JAYARAMI REDDY** for giving me such attention and time.

We have taken efforts in this project. However, it could not have been possible without the kind support and help of many individuals and RGUKT. We would like to extend my sincere thanks to all of them.

My thanks and appreciation also go to my colleagues in developing the project and people who have willingly helped me out with their abilities.

With Sincere Regards,

P. Vinod Kumar O180905

S. Shahid Hussain O180917

R. Rishi Pandya O180921

M. Yaswanth O180932

ABSTRACT

With the improvement in technologies. AI based systems have appeared in homes as Voice assistants and as transporters of goods and equipment in the industries, including hospitals and mortuaries, The robotic technology and AI technology is replacing manual work at a fast pace throughout the world.

In restaurants and hotels the customers face a lot of problems due to congestion at peak hours, unavailability of waiters and due to manual order processing. The major problem is it is difficult to identify which customer has the first priority to be served. These shortcomings can be handled and overcome by using a restaurant automation system where “AI based waiter” are used for ordering the food.

We create AI named JARVIS using python that interacts with the customer based on voice commands and the JARVIS provides the menu through LCD screen and reads those items available on the menu and takes the orders from the customer. The desired order is transmitted on wireless network to the kitchen via menu bar by “JARVIS ”. JARVIS also receives the updates regarding the order and updates it to the customers. When the order is ready JARVIS alerts the customers to get the food. A service within the AI is suggested for the payment purposes.

The basic purpose of AI JARVIS is to assist human resource, reduce labour cost and provide quick and accurate services to enhance the performances of working environment in restaurants

INDEX

1.CHAPTER

INTRODUCTION

The main objective of the project is to create to an AI based waiter called JARVIS to assist human service in restaurants. That allows to receive orders from the customers and send the order details to the chef. When the customer arrives to restaurant the AI greets them and interacts with the customer and completes all the order queries of the customer like showing menu, describing the items in the menu, tracking the food order status, etc.

The main use of AI JARVIS is to assist human resource, reduce labour cost and provide quick and accurate services to enhance the performances of working environment in restaurants.

1.1.MOTIVATION:-

The motivation to create this project has many sources

- Interest to develop an AI – based application to provide a services in restaurants
- To increase my knowledge horizon in technologies like Artificial Intelligence, NLP Python development
- Efficiency and productivity - AI- based waiter can perform repetitive tasks with high accuracy and speed, leading to increased efficiency in restaurant operations
- Cost savings: Hiring and training human waitstaff can be expensive for restaurants, especially in with high labour cost. AI-based waiter can help reduce the labour expenses by automating certain tasks.

1.2.PROBLEM DEFINITION:-

The problem to be addressed in developing an AI-based waiter system is to create a technology that can effectively and efficiently perform various tasks traditionally carried out by human waitstaff in a restaurant setting. This includes tasks such as taking order, and providing customer service. The AI-based waiter system should aim to overcome the limitations and challenges associated with manual operations while enhancing the overall dining experience.

1.3.OBJECTIVE

The primary objective of creating an AI-based waiter is to automate repetitive tasks in restaurant operations, such as taking orders, delivering food and beverages, and clearing tables. By leveraging AI technology, the objective is to enhance efficiency, reduce errors, and improve overall productivity in the restaurant workflow. The AI-based waiter system aims to enhance the customer experience by providing personalized recommendations, answering queries, and offering efficient service. The objective is to create a seamless and engaging interaction between the AI waiter and customers, leading to higher satisfaction and loyalty. Another objective is to achieve cost savings for restaurant owners by reducing labor expenses associated with hiring and training human waitstaff. By utilizing AI-based waiters, the objective is to optimize resource allocation and minimize operational costs while maintaining service quality.The objective is to provide consistent and reliable service throughout the day by eliminating human errors, fatigue, and mood variations. AI-based waiters can follow predefined protocols and guidelines accurately, ensuring a consistent dining experience for customers. The objective is to design the AI-based waiter system with seamless integration capabilities into existing restaurant infrastructure, including POS systems, order management systems, and kitchen operations. Scalability is another objective, ensuring that the system can adapt to different restaurant sizes and accommodate increasing customer demands.

2.CHAPTER

REQUIREMENT ANALYSIS

2.1 REQUIREMENT SPECIFICATION

The project involved analysing the design of application so as to make the application more users friendly. To do so, it was really important to keep the simple and easy to understand and create UI to reduce time amount of typing the user needs to do. This also

2.2 Hardware Requirements

Hardware Requirements:

System will be using Processor: Core2Duo
Main Memory: 2 GB RAM (Minimum)
Hard Disk: 256 GB (Minimum)
Internet (256 kb/s Minimum)

2.3 Software Requirements

Python

Packages Used:

- SR
- OS
- PYTTSX3
- TKINTER
- PYQT5
- Flask

2.4 TECHNOLOGIES

Python:

Python is a high-level programming language known for its simplicity, readability, and versatility. It was created by Guido van Rossum and first released in 1991. Python emphasizes code readability, making it easier to write, understand, and maintain.Python's design philosophy focuses on code readability and the use of plain English-like syntax, which helps reduce the learning curve for beginners and promotes clean and elegant code. It utilizes whitespace indentation to indicate code blocks, eliminating the need for explicit braces or brackets.Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming. It provides a vast standard library that offers ready-to-use modules for various tasks, ranging from web development and data analysis to machine learning and artificial intelligence.

Advantages of Python

The merits of using Python are −

Readability and Simplicity: Python's clean and straightforward syntax makes it easy to read and write code. Its use of indentation for code blocks promotes clean and readable code, enhancing code maintainability and reducing the chances of introducing errors.

Large Standard Library: Python comes with an extensive standard library that provides a wide range of pre-built modules and functions for various tasks. This eliminates the need to reinvent the wheel and allows developers to leverage existing code, saving time and effort in development.

Wide Range of Libraries and Frameworks: Python has a vast ecosystem of third-party libraries and frameworks, such as NumPy, Pandas, Django, Flask, TensorFlow, and PyTorch. These libraries extend Python's capabilities for tasks like scientific computing, data analysis, web development, machine learning, and more. The availability of such rich resources enables developers to build complex applications efficiently.

Cross-platform Compatibility: Python is available on major operating systems, including Windows, macOS, and Linux. This cross-platform compatibility allows developers to write code once and run it on different platforms without major modifications, saving time and effort.

Scalability and Flexibility: Python is scalable and flexible, capable of handling projects of various sizes. It supports different programming paradigms, including procedural, object-oriented, and functional programming, allowing developers to choose the approach that best suits their project requirements

Integration Capabilities: Python offers seamless integration with other languages, allowing developers to incorporate modules or libraries written in languages like C, C++, and Java. This flexibility enables developers to leverage existing codebases and utilize Python's simplicity for glue code or high-level logic.

3.CHAPTER

ANALYSIS

3.1.EXISTED SYSTEM

There exist a system named ServeU to help restauranteurs to maintain the service quality by relieving the staffs' workload. ServeU provides a QR Code accessible online menu with a virtual waiter for restaurants that answer customers' questions and take orders. In addition, ServeU also offers personalization to customers and reduce the cost of human error during standard operation. ServeU provides a QR code accessible online menu with virtual waiter for restaurants that answer customers' questions and take orders. After a customer enters the restaurant, find a seat, and scans the QR code on the table, they will meet our AI-powered virtual waiter who will serve them, takes the order, and notifies the merchant.

3.2. PROPOSED SYSTEM

In this project you can interact with the AI – JARVIS just like how you interact with human in restaurant. It interacts with human and receives order from the users. It has the functionalities like interacting with customer and like receiving orders, tracking the order. Existed systems shows only the menu and QR code to get the menu there is no UI in it. AI JARVIS has the UI to interact with the user.

3.3 PURPOSE

The main purpose of AI JARVIS is to assist human resource, reduce labour cost and provide quick and accurate services to enhance the performances of working environment in restaurants. It allows no compromise, even demand for higher service quality, with cleanliness and safety on top of mind.

3.4 SCOPE

The scope for AI-based waiter systems is quite broad and offers several opportunities for innovation and improvement in the restaurant industry. Here are some key areas where AI-based waiters can make a significant impact:

Order Taking and Delivery: AI-based waiters can automate the process of taking customer orders, reducing errors and improving order accuracy. They can also efficiently deliver orders to the correct tables, ensuring prompt service and minimizing wait times.

Personalized Customer Service: AI-based waiters can provide personalized recommendations based on customer preferences, dietary restrictions, and previous orders. By analyzing customer data and employing machine learning algorithms, they can offer tailored suggestions, enhancing the overall dining experience

3.5 OVERALL DESCRIPTION

An AI-based waiter, also known as a robotic waiter or smart waiter, is a system that utilizes artificial intelligence technology to automate and enhance various tasks typically performed by human waitstaff in a restaurant setting. It combines AI algorithms, natural language processing, computer vision, and robotics to provide efficient and personalized service to customers.

The AI-based waiter system can perform a range of functions, including taking orders, and interacting with customers. It leverages advanced natural language processing capabilities to understand and respond to customer inquiries, requests, and feedback. Through machine learning algorithms, it can analyze customer preferences and provide personalized recommendations for menu items.

The system is designed to navigate the restaurant space autonomously, avoiding obstacles and locating tables accurately. It can identify and interact with customers using touchscreens or voice commands, providing a seamless and interactive dining experience.

4.CHAPTER

SYSTEM DESIGN

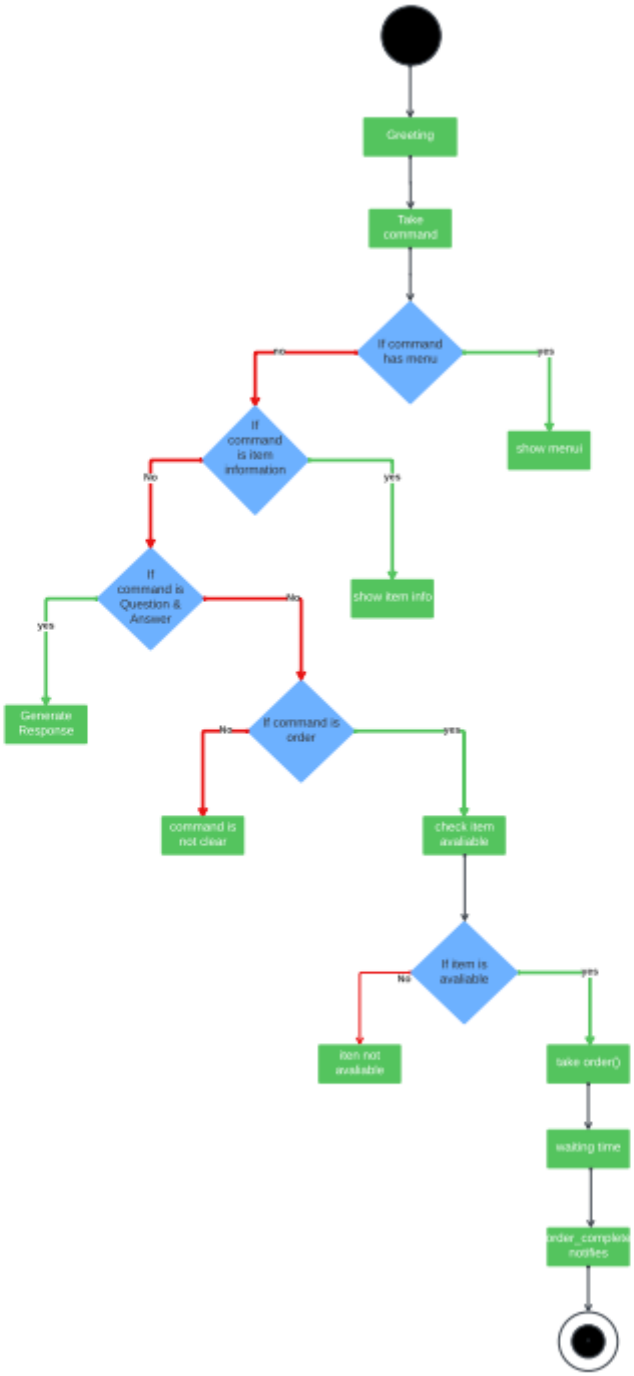
System Design is the process of designing the architecture, components, and interfaces for a system so that it meets the end-user requirements. System Design for tech interviews is something that can't be ignored!

Design concepts such as scalability, load-balancing, caching, etc. in the interview. This specifically designed System Design tutorial will help you to learn and master System Design concepts in the most efficient way from basics to advanced level.
Systems development is systematic process which includes phases such as planning, analysis, design, deployment, and maintenance. Here, in this tutorial, we will primarily focus on − System Analysis System Design.

Systems Analysis :

It is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components.System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose. Analysis specifies **what the system should do**.

Data Flow Diagram



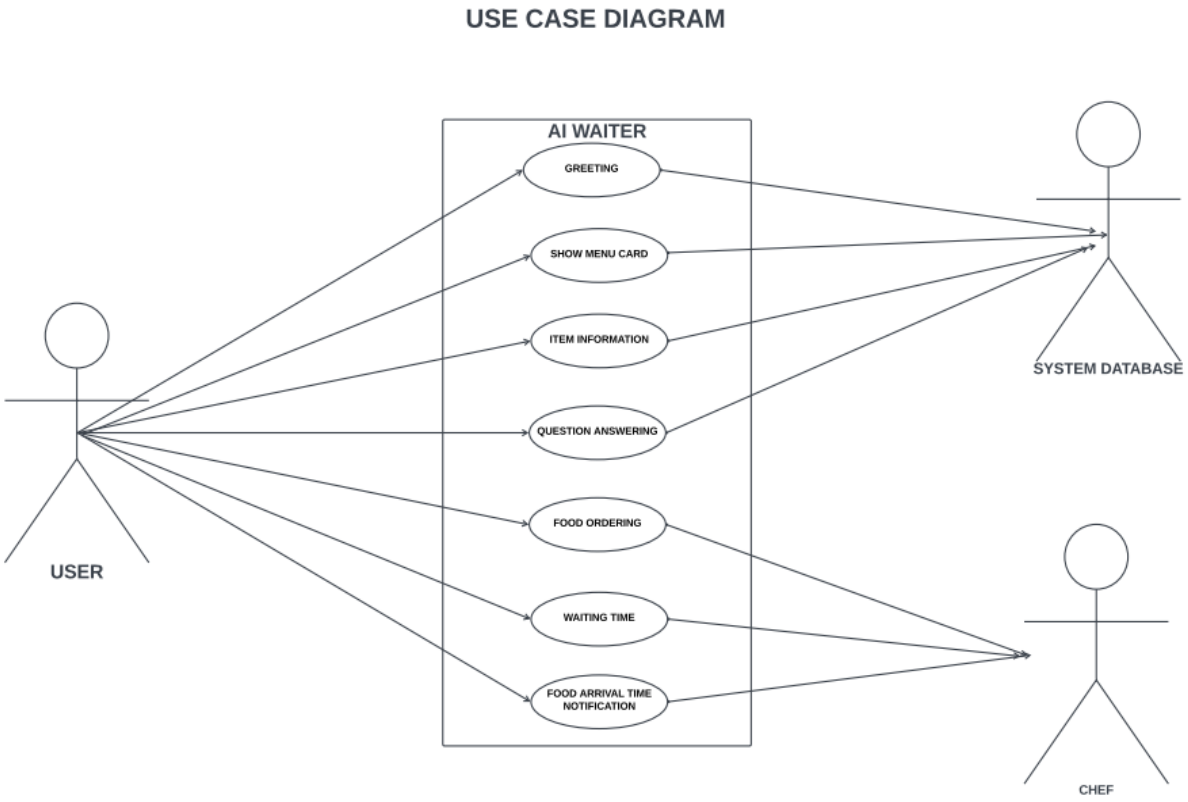
4.1 UML DIAGRAMS

A use case diagram is a visual representation of how a user might interact with a program. A use case diagram depicts the system's numerous use cases and different sorts of users. The circles or ellipses are used to depict the use cases.

Usecase Diagram

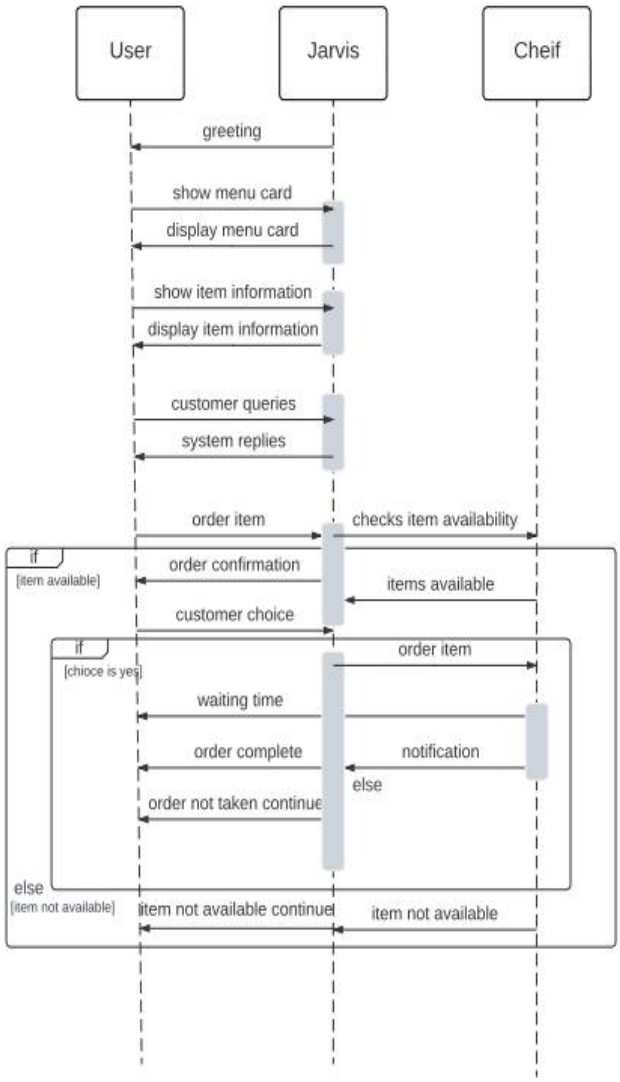
To model a system,the most important aspect is to capture the dynamic behaviour.

Dynamic behaviour means the behaviour of the system when it is running or operating.



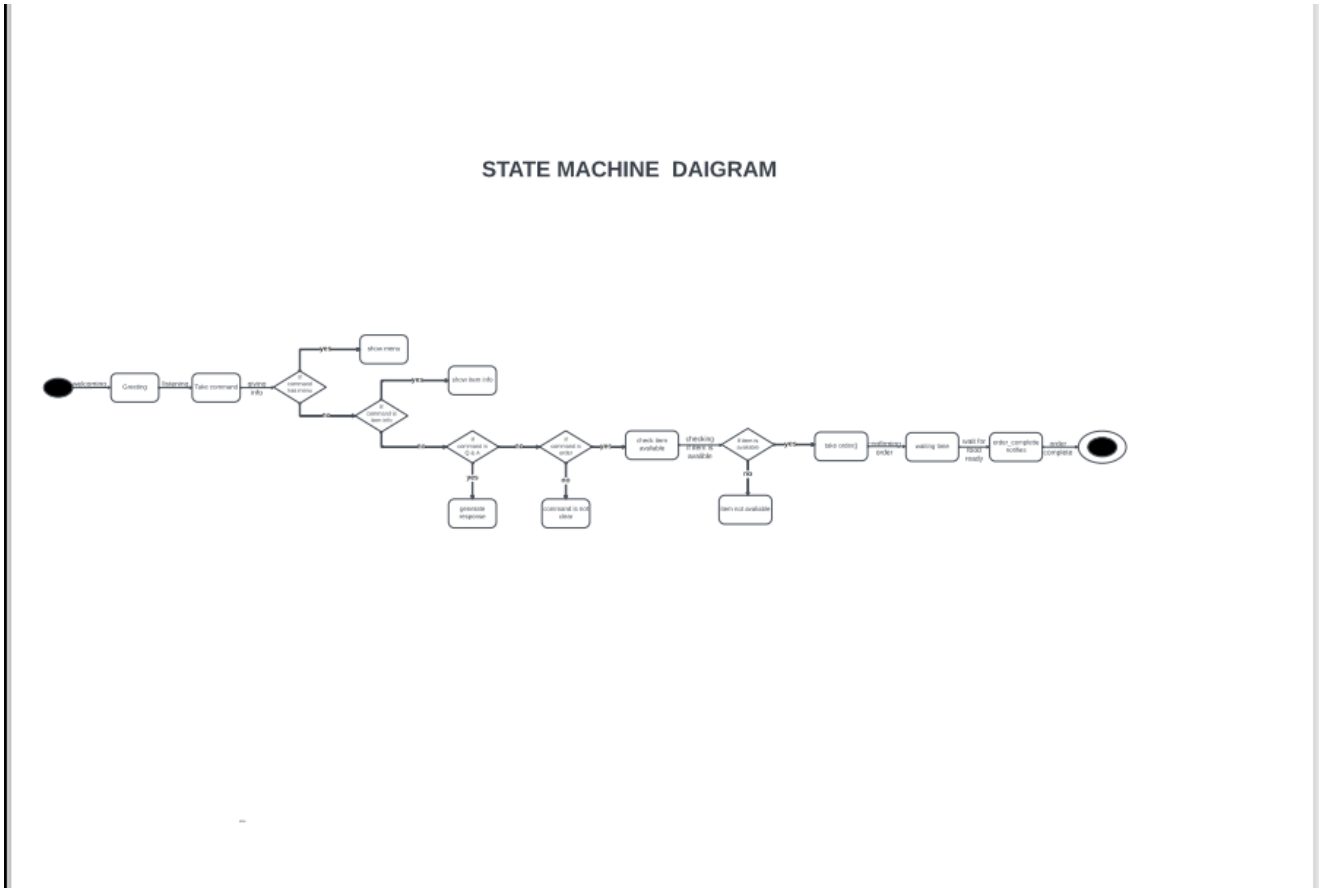
Sequence Diagram

A sequence diagram is simply depicts the interaction between objects in sequential order such that the order in which these interactions takes place.



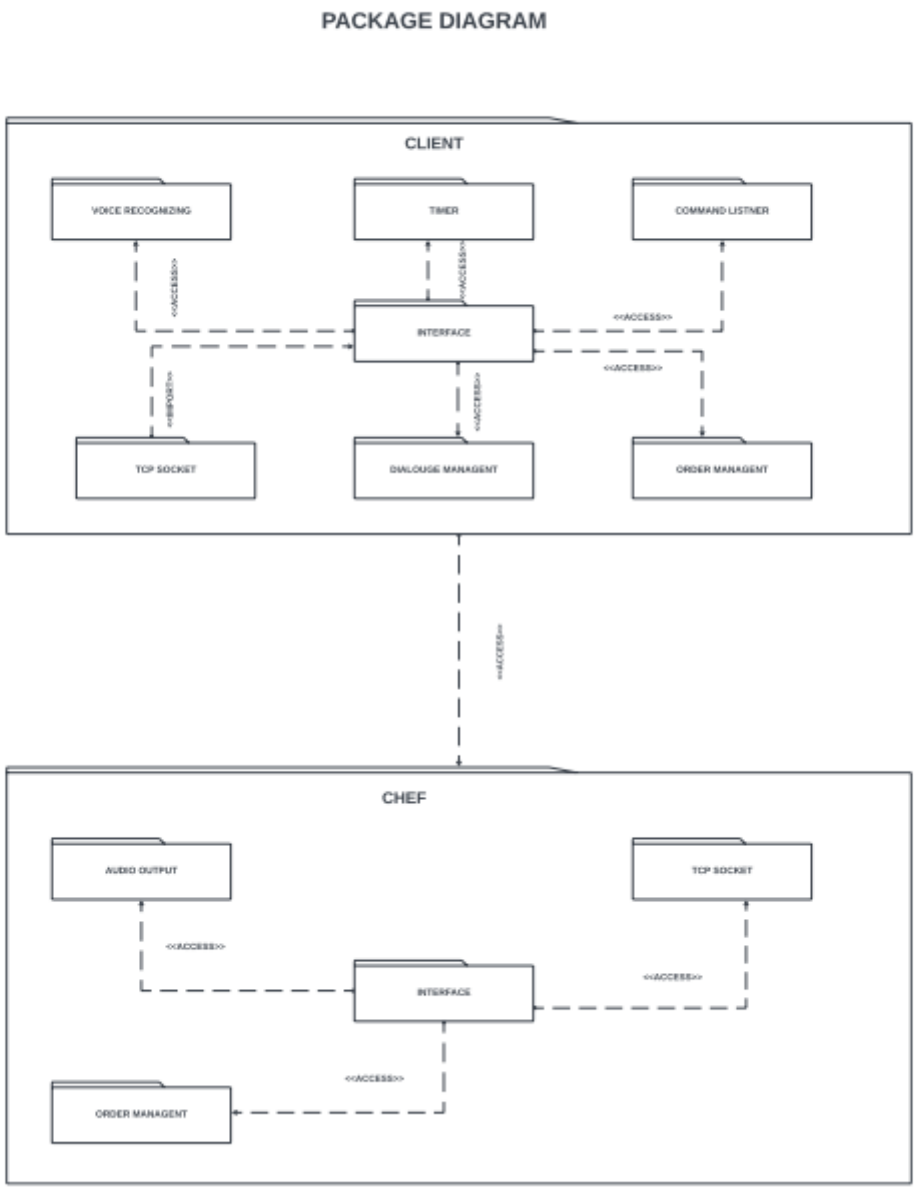
State Diagram

For representing dynamic view, we use state chart diagram. State diagram will be dealing with different states of components or objects. States are changed based on some external conditions.



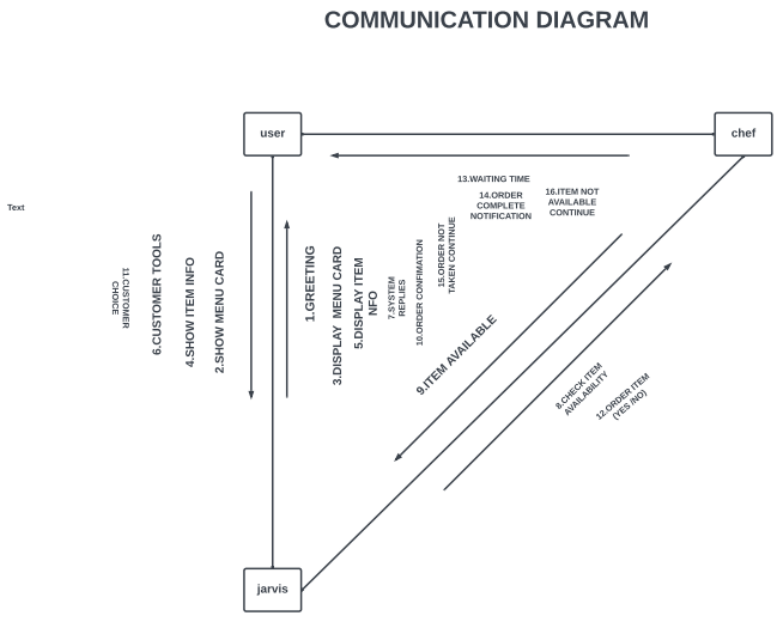
Package Diagram

Package diagrams are structural diagrams used to show the organization and argument of various model elements in the form of packages. A package is grouping of related uml elements, such as diagrams, documets, classes are even other packages.



Communication diagram

A communication diagram is an extension of object diagram that shows the objects along with the messages that travel from one to another. In the condition to association among objects, communication diagram shows the messages the objects send each other.



5.CHAPTER

IMPLEMENTATION

5.1 GRAPHICAL USER INTERFACE

The user interface is kept simple and understandable. The user need not take any additional effort to understand the functionality and navigation in the application. The colours are chosen in such a way that the user can easily understand where the input has to be given. A mic icon is present in the interface for taking voice command as input from the user. The output will be read out loud and output text will be presented on the screen. Interface contains different labels to display menu card and order tracking feature.

5.2.MODULES:

- Voice recognizing
- TCP socket
- Order Management
- Audio Output
- Timer

5.3.MODULES DESCRIPTION:

Voice Recognizing

Voice recognition, also known as speech recognition, is a technology that enables computers or systems to interpret and understand spoken language.

It involves the conversion of spoken words into text or commands that can be processed by machines.In our project it is used to recognize the user audio commands and covert in to text for order management

TCP Socket

TCP (Transmission Control Protocol) socket is a communication endpoint that allows two devices to establish a reliable, bidirectional, and stream-oriented connection over an IP network. It provides a means for applications to exchange data in a structured and ordered manner.

We used TCP socket for the communication between the AI and Chef regarding the details about the preparation of order.

Order Management

Order management in an AI-based waiter system involves efficiently handling the process of taking customer orders, managing order details, and coordinating with kitchen staff for order preparation and delivery.

Audio Output

The audio output module in an AI-based waiter system is responsible for delivering auditory information and notifications to customers or restaurant staff. It enables the system to communicate effectively. For the implementation of audio output we use the python module known as PYTTSX3

Timer

The timer module in an AI-based waiter system provides functionality to track and manage time-related operations, such as timing food preparation, monitoring cooking durations, and estimating delivery or pickup times. We implement this function by using a python module known as time.

5.4 SAMPLE CODE

```
# importing libraries
import tkinter as tk
import speech_recognition as sr
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from PIL import Image, ImageTk
import pyttsx3
import threading
import time
from translate import Translator
import pandas as pd
import socket
import re

# Set up NLP
# nltk.download('punkt')
# nltk.download('stopwords')
stop_words = set(stopwords.words('english'))

#declaing global variables
global order_status
global new_time
```

```
new_time = []
order_status = 0
total_amount = 0
global responses
responses = []
global response_index
global response_index2
response_index = -1
response_index2 = response_index
bill_details = []
bill_details2 = []
global menu_status
menu_status = 0
global time_place


# Loading the Excel sheets into a DataFrame
dialogue_data = pd.read_excel("C:/Users/Rishi/Desktop/jarvis/dialogue_data.xlsx")
menu_price_data = pd.read_excel("C:/Users/Rishi/Desktop/jarvis/menu_price.xlsx")

# Convert the DataFrame to a dictionary
dialogue_dict = dialogue_data.set_index('command')['response'].to_dict()

menu = menu_price_data.set_index('item')['price'].to_dict()
```

```
description = menu_price_data.set_index('item')['description'].to_dict()


# Dialogue management


# defining greetings
def handle_greeting():
    return dialogue_dict['greeting']


# menu displaying
def handle_menu_request():
    global menu_status
    menu_status = 1
    global menu_carousel
    # Create the menu carousel
    menu_carousel = MenuCarousel(window)


    return dialogue_dict['menu_request']


#defining menucarousal for sliding menu cards
class MenuCarousel:
    def _init_(self, master):
        self.master = master
        self.menu_images = [
            "C:/Users/Rishi/Desktop/jarvis/1.jpg",
            "C:/Users/Rishi/Desktop/jarvis/2.jpg",
```

```
            "C:/Users/Rishi/Desktop/jarvis/3.jpg",
            "C:/Users/Rishi/Desktop/jarvis/4.jpg",
            "C:/Users/Rishi/Desktop/jarvis/5.jpg",
            "C:/Users/Rishi/Desktop/jarvis/6.jpg",
            "C:/Users/Rishi/Desktop/jarvis/7.jpg"
        ]
        self.current_image = 0

        self.carousel_frame = tk.Frame(self.master)
        self.carousel_frame.pack()
        self.carousel_frame.place(x=95, y=0)

        self.menu_image_label = tk.Label(self.carousel_frame)
        self.menu_image_label.pack()

        # Calculate desired button size based on frame size
        frame_width = self.carousel_frame.winfo_width()
        button_width = frame_width // 10  # Adjust the denominator to control the button
size ratio
        button_height = button_width

        # Load previous and next button images and resize them
        prev_image = Image.open("C:/Users/Rishi/Desktop/jarvis/left_arrow2.jpg")
        prev_image = prev_image.resize((50,50))
        prev_image = ImageTk.PhotoImage(prev_image)
```

```
        next_image = Image.open("C:/Users/Rishi/Desktop/jarvis/right_arrow2.jpg")
        next_image = next_image.resize((50,50))
        next_image = ImageTk.PhotoImage(next_image)

        cancel_image = Image.open("C:/Users/Rishi/Desktop/jarvis/cancel_image3.jpg")
        cancel_image = cancel_image.resize((100,50))
        cancel_image = ImageTk.PhotoImage(cancel_image)

        self.prev_button = tk.Button(self.master, image=prev_image,
command=self.show_prev_image, bd=0)#,bg="#10104E", fg="white")
        self.prev_button.image = prev_image
        self.prev_button.pack()
        self.prev_button.place(x=40, y=390)

        self.next_button = tk.Button(self.master, image=next_image,
command=self.show_next_image, bd=0)# bg="#10104E", fg="white",
bd=0,font=("Helvetica", 12))
        self.next_button.image = next_image
        self.next_button.pack()
        self.next_button.place(x=645, y=390)

        self.cancel_button = tk.Button(self.master, image = cancel_image, command =
self.cancel_display,bd=0)
        self.cancel_button.image = cancel_image
        self.cancel_button.pack()
```

```
        self.cancel_button.place(x=340, y=700)
        # Store the PhotoImage object
        self.menu_image = None

        self.show_current_image()

    def show_current_image(self):
        image_path = self.menu_images[self.current_image]
        image = Image.open(image_path)
        image = image.resize((540, 800))
        self.menu_image = ImageTk.PhotoImage(image)
        self.menu_image_label.configure(image=self.menu_image)

    def show_prev_image(self):
        self.current_image = (self.current_image - 1) % len(self.menu_images)
        self.show_current_image()

    def show_next_image(self):
        self.current_image = (self.current_image + 1) % len(self.menu_images)
        self.show_current_image()
    def cancel_display(self):
        global menu_status
        self.carousel_frame.destroy()
        self.prev_button.destroy()
        self.next_button.destroy()
```

```
        self.cancel_button.destroy()
        menu_status -= 1

#defining function to display item information
def handle_item_info(command):

    response = description[command]
    output_text.insert(tk.END, response)
    speak_response(response)

#defining function to take order from user
global confirmation_buttons
def handle_order(items):
    global order_status
    global send_items
    global confirmation_buttons
    total_amount2 = 0
    for item in items:
        item = item.lower()
        if item in menu:
            price = menu[item]
            bill_details2.append((item, price))  # Append the item and price to the bill details
list
            total_amount2 += price
        else:
```

```
            response = "I'm sorry, {} is not available on the menu.".format(item)
            items.remove(item)
            output_text.insert(tk.END, response)
            speak_response(response)
    if total_amount2 > 0:
        order_status = 5
        send_items = items
        confirmation_buttons = ConfirmationButtons(window, confirm_order,
cancel_order)
        confirmation_buttons.create_buttons()
        items = ",".join(items)
        response = "Great choice! Your order is {}. Would you like to confirm your
order?".format(items)
        output_text.insert(tk.END, response)
        speak_response(response)


class ConfirmationButtons:
    def _init_(self, window, confirm_func, cancel_func):
        self.window = window
        self.confirm_func = confirm_func
        self.cancel_func = cancel_func
        self.yes_button = None
        self.no_button = None


    def create_buttons(self):
```

```
        self.yes_button = tk.Button(self.window, text="Yes", width=10,
command=self.confirm_func)
        self.yes_button.pack(side="left", padx=10)
        self.yes_button.place(relx=0.65, rely=0.55)

        self.no_button = tk.Button(self.window, text="No", width=10,
command=self.cancel_func)
        self.no_button.pack(side="left", padx=10)
        self.no_button.place(relx=0.8, rely=0.55)

    def destroy_buttons(self):
        if self.yes_button:
            self.yes_button.destroy()
        if self.no_button:
            self.no_button.destroy()




def confirm_order():
    confirmation("yes")

def cancel_order():
    confirmation("no")

#function for processing the confirmation command from user
```

```
def confirmation(command):
    global confirmation_buttons
    # After some time or action, call the following method to destroy the buttons:
    confirmation_buttons.destroy_buttons()
    if "yes" in command or "yeah" in command or "confirm" in command or "haa" in command:
        handle_order_confirmation('yes',send_items)
    else:
        handle_order_confirmation('no',send_items)


# function for order confirmation from user and sending it to chef
def handle_order_confirmation(choice, items):
    global order_status
    total_amount = 0
    if choice.lower() == 'yes':
        for item in items:
            price = menu[item]
            # Append the item and price to the bill details list
            bill_details.append((item, price))
            total_amount += price
        # Convert the order details to a string
        order_details = items
        resp = "Waiting for chef confirmation "
        input_text.delete(0, tk.END)
        output_text.insert(tk.END,resp)
```

```
        # Send the order details to the server and receive the response
        new_order = ','.join(order_details)
        my_thread = threading.Thread(target=send_order_details,
args=(new_order,)).start()
        while(len(responses)==0):
            pass
        response = responses[0]
        responses.remove(responses[0])
        count = find_time(response)

        response2 = "Order confirmed! Your order will be served in {}. Enjoy your
meal!".format(response)

        order_status = 0

        output_text.insert(tk.END, response2)
        speak_response(response2)
        new_time.append(timer(count,window))
        new_time[-1].start_countdown()
    else:
        response = "Okay, please continue with your order."
        output_text.insert(tk.END, response)
        speak_response(response)
        order_status = 0
```

```
#sending order to the chef using tcp protocol
def send_order_details(order_details):
    try:

        # Create a socket object
        client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        # Define the server's IP address and port number
        server_address = ('192.168.39.54', 10000)

        # Connect to the server
        client_socket.connect(server_address)

        client_socket.send(order_details.encode())

        # Receive the response from the server
        response = client_socket.recv(1024).decode()
        # Close the socket connection
        client_socket.close()

        responses.append(response)
    except:
        response = "The chef is not available . please wait for some time."
        output_text.insert(tk.END, response)
```

```
        speak_response(response)


# defining class for creating a timer for order waiting time
class timer():


    def _init_(self,count,window):
        self.times = count
        self.window = window


    def start_countdown(self):
        self.remaining_time = int(self.times)
        # Create a Label widget to display the countdown
        self.countdown_label = tk.Label(self.window, text="Order will be ready in: 00:00",
font=("Arial", 16))
        self.countdown_label.pack(pady=10)
        timer_thread = threading.Thread(target=self.update_countdown).start()


    def update_countdown(self):
        if self.remaining_time > 0:
            minutes = self.remaining_time // 60
            seconds = self.remaining_time % 60
            self.countdown_label.config(text=f"Order will be ready in:
{minutes:02d}:{seconds:02d}")
            self.remaining_time -= 1
            self.countdown_label.after(1000, self.update_countdown)
```

```
        else:
            self.countdown_label.config(text="Order is ready sir.\n your order will be served shortly")  # Update label text
            self.countdown_label.after(3000, self.remove_label)

    def remove_label(self):
        # Remove the label from the window
        self.countdown_label.destroy()

# defining function for finding waiting time from the string given
def find_time(new_word):
    numbers = re.findall(r'\d+',new_word)
    return (int(numbers[0])*60)

def order_cancelation(command):
    if 'yes' in command or 'yeah' in command or 'confirm' in command:
        for i in new_time:
            i.remove_label()
        response = "order cancelled ."
        output_text.insert(tk.END, response)
        speak_response(response)
    else:
        response = "ok sir. order not cancelled."
        output_text.insert(tk.END, response)
        speak_response(response)
```

```
#function for handling unknown request
def handle_unknown():
    return "I'm sorry, I didn't understand that. Can you please rephrase?"



def handle_bill_request():
    total_amount = 0
    bill_description = "Bill Details:\n"
    for item, price in bill_details:
        bill_description += "{} - Rs.{}\n".format(item, price)
        total_amount += price
    bill_description += "Total Amount: Rs.{}".format(total_amount)
    return bill_description



def handle_conversation(command):
    global order_status
    k=0
    info = 0
    command = command.lower()
    is_order = []
    for item in menu :
        if item in command and 'order' not in command :
            handle_item_info(item)
            info = 5
```

```
    elif item in command and 'order' in command:
        is_order.append(item)
if is_order :
    handle_order(is_order)


if 'menu' in command and 'remove' not in command :
    response = handle_menu_request()
    output_text.insert(tk.END, response)
    speak_response(response)
elif 'information' in command or 'info' in command or 'about' in command:
    for item in menu:
        if item in menu:
            temp1 = 5
    if temp1 == 0:
        response = "Please specify an item from the menu to place an order."
        output_text.insert(tk.END, response)
        speak_response(response)
elif 'order' in command:
    if is_order :
        pass
    else:
        response = "Please specify an item from the menu to place an order."
        output_text.insert(tk.END, response)
        speak_response(response)
elif 'no' in command:
```

```
    tokens = word_tokenize(command)
    tokens = [token for token in tokens if token not in stop_words]
    item = ' '.join(tokens)  # Combine tokens into a sequence of words

    if item in menu:
        handle_order_confirmation('no', item)
    else:
        response = "Please specify an item from the menu to continue with your order."
        output_text.insert(tk.END, response)
        speak_response(response)
elif 'bill' in command or 'total bill' in command:
    bill_description = handle_bill_request()
    output_text.insert(tk.END, bill_description)
    speak_response(bill_description)

elif 'good bye' in command or 'goodbye' in command:
    response = "Goodbye! Have a great day!"
    output_text.insert(tk.END, response)
    speak_response(response)
    window.after(3000, window.destroy)  # Close the window after 3 seconds
elif "cancel" in command :
    response = "do you want to cancel the order sir?"
    output_text.insert(tk.END, response)
    speak_response(response)
    order_status = 10
```

```
elif 'remove' in command or 'close' in command :
    global menu_carousel
    if menu_status == 1 :
        response = "Got it sir"
        output_text.insert(tk.END, response)
        speak_response(response)
        menu_carousel.cancel_display()
    else:
        response = "menu is already removed sir"
        output_text.insert(tk.END, response)
        speak_response(response)
elif info == 5 :
    pass
else:
    chat1 = 0
    for i in dialogue_dict :
        if i in command:
            response = dialogue_dict[i]
            output_text.insert(tk.END, response)
            speak_response(response)
            chat1 = 5
    for j in command :
        if j in menu :
            handle_item_info(j)
            chat1 = 5
```

```
        if(chat1 == 0):
            response = handle_unknown()
            output_text.insert(tk.END, response)
            speak_response(response)


def process_command(command):
    output_text.delete(1.0, tk.END)  # Clear previous output
    handle_conversation(command)

def process_command2(command):
    confirmation(command)

def process_command3(command):
    order_cancelation(command)

# Speech recognition
def listen_command():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        listening_label.config(text="Listening...", fg="blue")  # Update label text and color
        window.update_idletasks()  # Update the GUI immediately
        audio = r.listen(source)
```

```
    try:
        listening_label.config(text="Recognizing...", fg="blue")  # Update label text and color
        window.update_idletasks()  # Update the GUI immediately
        command = r.recognize_google(audio)
        input_text.delete(0, tk.END)  # Clear previous input
        input_text.insert(tk.END, command)

        listening_thread = threading.Thread(target=process_command, args=(command,)).start()
    except sr.UnknownValueError:
        print("Could not understand audio")
    except sr.RequestError as e:
        print("Could not request results; {0}".format(e))
    finally:
        listening_label.config(text="")  # Clear the listening label


def speak_response(response):
    output_text.pack()
    output_text.delete(1.0, tk.END)  # Clear previous output

    def perform_speech():
        for i, char in enumerate(response):
            output_text.insert(tk.END, char)
```

```
        output_text.update_idletasks()  # Update the GUI immediately

    engine = pyttsx3.init()
    engine.say(response)
    engine.runAndWait()

speech_thread = threading.Thread(target=perform_speech).start()



def submit_command(event=None):
    global order_status
    if(order_status == 0):
        print("order status 1")
        command = input_text.get()
        process_command(command)
    elif(order_status == 10):
        print("order satus 3 ")
        command = input_text.get()
        order_status = 0
        process_command3(command)
    else:
        print("order status 2")
        command = input_text.get()
        process_command2(command)
```

```
# Create GUI
window = tk.Tk()
window.title("Rishi Restaurant")
window.attributes('-fullscreen', True)  # Set the window to fullscreen

# Load and display background image
background_image = Image.open("jarvis.jpg")
background_photo = ImageTk.PhotoImage(background_image)
background_label = tk.Label(window, image=background_photo)
background_label.place(x=0, y=0, relwidth=1, relheight=1)

# Load the background images for input and output frames
input_bg_image = Image.open("C:/Users/Rishi/Desktop/jarvis/input_pic2.png")
output_bg_image = Image.open("C:/Users/Rishi/Desktop/jarvis/output_pic.png")


# Convert the resized images to PhotoImage objects
input_bg_photo = ImageTk.PhotoImage(input_bg_image)
output_bg_photo = ImageTk.PhotoImage(output_bg_image)

# Fonts
title_font = ("Helvetica", 28, "bold")
label_font = ("arial", 14)
input_font = ("arial", 14)
```

```
output_font = ("arial", 12)

# Input Frame
input_frame = tk.Frame(window, bd=0)
input_frame.pack(pady=50)
input_frame.place(relx=0.75, rely=0.2, anchor="n", relwidth=0.25, relheight=0.1)

# Create labels for the frames and set the background images
input_bg_label = tk.Label(input_frame, image=input_bg_photo)
input_bg_label.place(x=0, y=0, relwidth=1, relheight=1)

# Resize the background images to match the frame dimensions.0.5
input_bg_image = input_bg_image.resize((50,50))

# Create a transparent Entry widget
input_text = tk.Entry(input_frame, width=30, font=input_font, bg="#10104E",
fg="white",highlightthickness=0)
input_text.pack(side=tk.LEFT)
input_text.bind("<Return>", submit_command)  # Bind the Return key event

listen_frame = tk.Frame(window, bd=5)
listen_frame.pack(pady=10)
listen_frame.place(relx=0.5, rely=0.1, relwidth=0.03, relheight=0.06, anchor="n")

# Load and resize the image
```

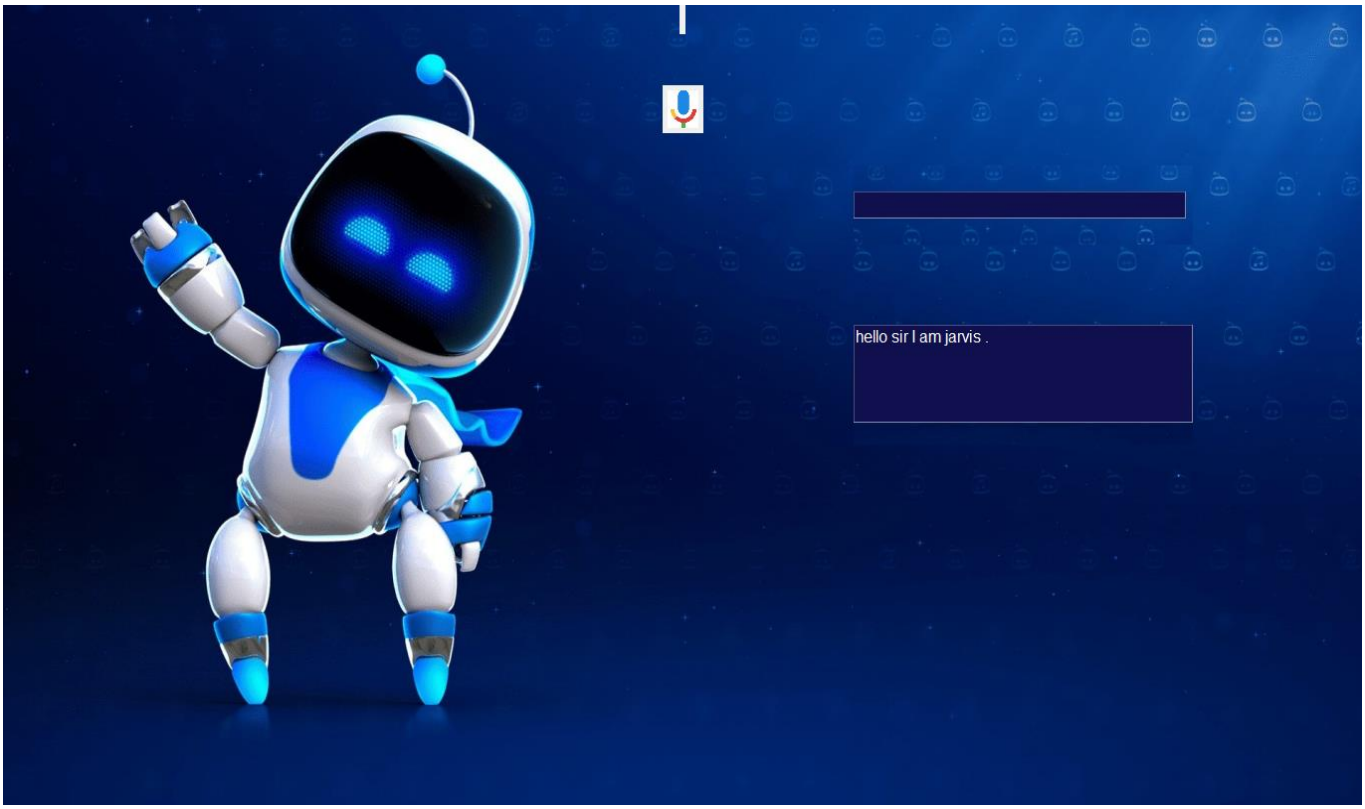
```
submit_image = Image.open("googlemic.png")

submit_image = submit_image.resize((58, 58))  # Resize the image

submit_photo = ImageTk.PhotoImage(submit_image)


# Create button with custom image

submit_button = tk.Button(listen_frame, image=submit_photo,
command=listen_command, bd=0, relief="flat", bg="blue")

submit_button.image = submit_photo  # Save reference to prevent garbage collection

submit_button.pack(side=tk.LEFT)


# Output Frame

output_frame = tk.Frame(window, bd=0)

output_frame.pack(pady=80)

output_frame.place(relx=0.75, rely=0.4, anchor="n", relwidth=0.25, relheight=0.15)


output_bg_image = output_bg_image.resize((output_frame.winfo_width(),
output_frame.winfo_height()))


output_bg_label = tk.Label(output_frame, image=output_bg_photo)

output_bg_label.place(x=0, y=0, relwidth=1, relheight=1)


output_text = tk.Text(output_frame, height=5, width=60,
font=output_font,bg="#10104E", fg="white", wrap=tk.WORD)



# Listening Label
```

```
listening_label = tk.Label(window, text="", font=label_font)
listening_label.pack()



# Hide the menu card initially
# Call the handle_greeting function
def display_greeting():
    greeting_response = handle_greeting()
    output_text.insert(tk.END, greeting_response)
    speak_response(greeting_response)



# Start the greeting thread
greeting_thread = threading.Thread(target=display_greeting)
greeting_thread.start()

window.mainloop()
```

5.5 SAMPLE SCREENSHOTS

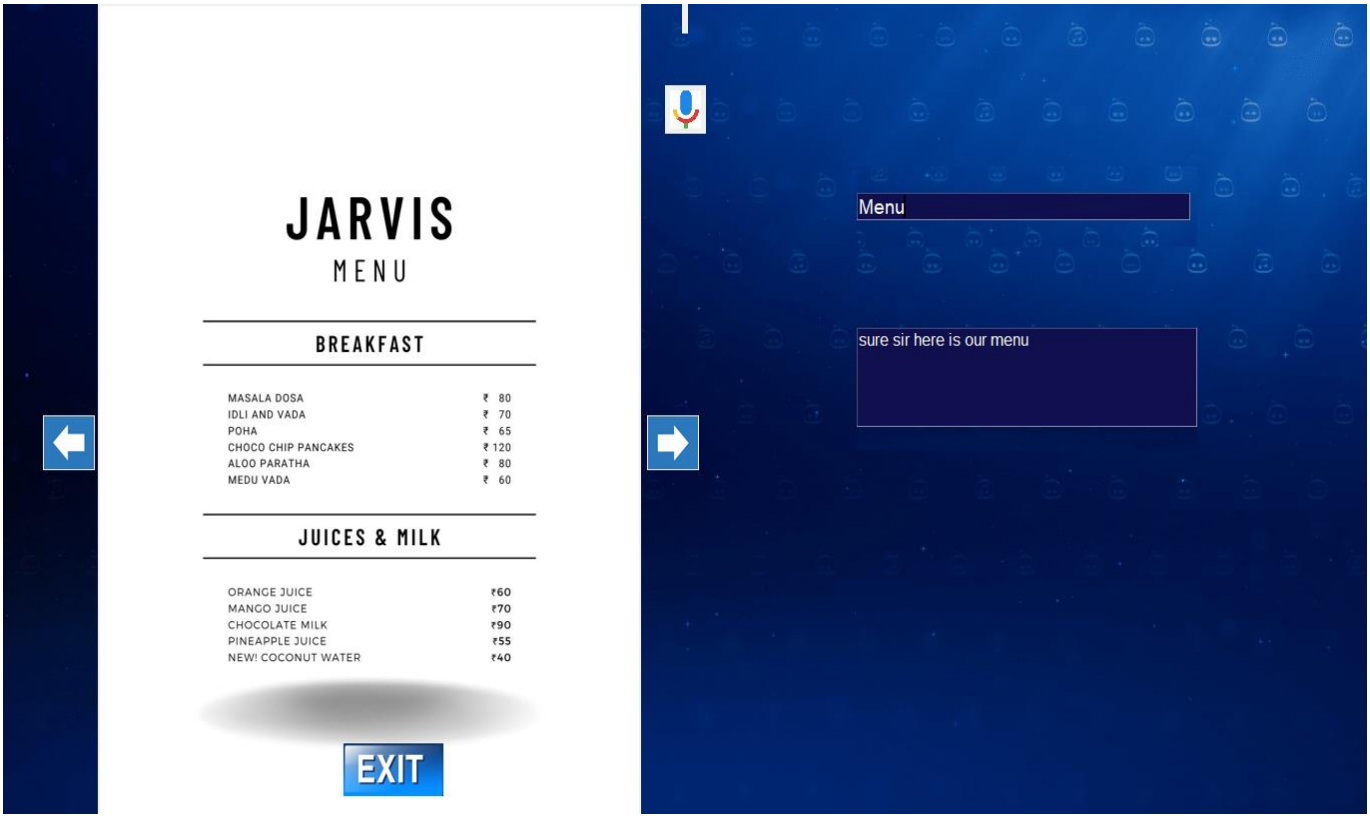
User Interface

It refers to the visual and interactive elements of a software application or system that enable users to interact with and control the functionality of the software. Our UI contains the elements mic for taking voice input, input text for the manual input, and output textbox to display the output



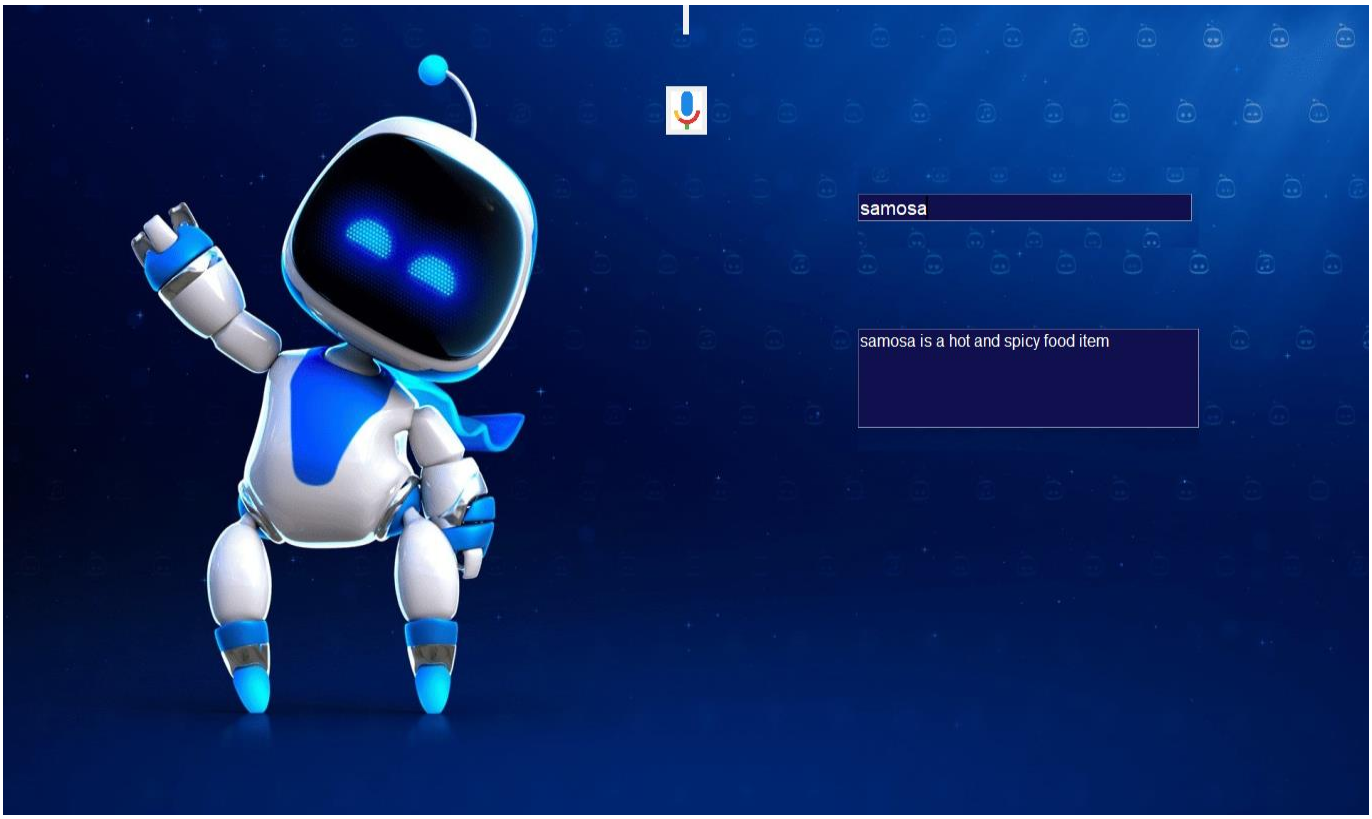
Menu page

Its display the menu of items available in the restaurant and it contains 2 button to navigate to other menu cards



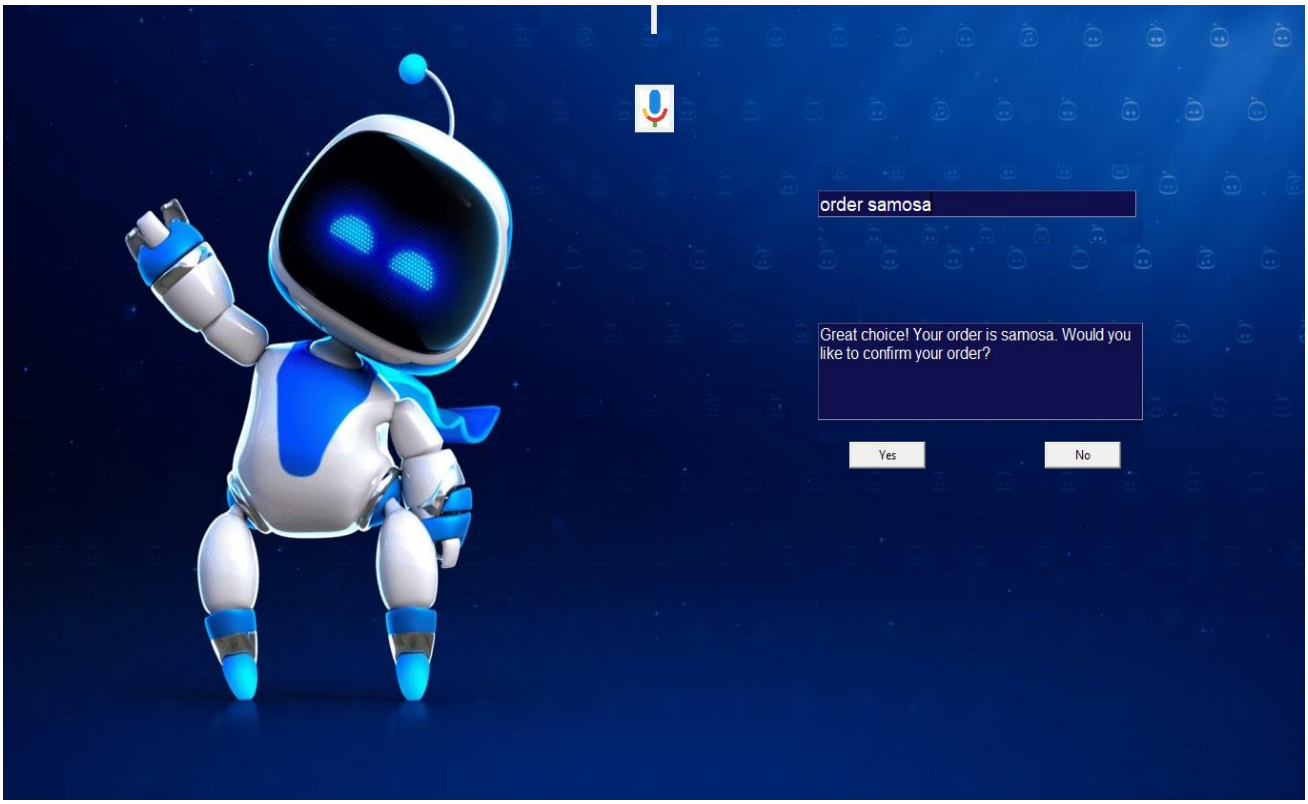
Item Description

It shows the description of each item available in the menu.



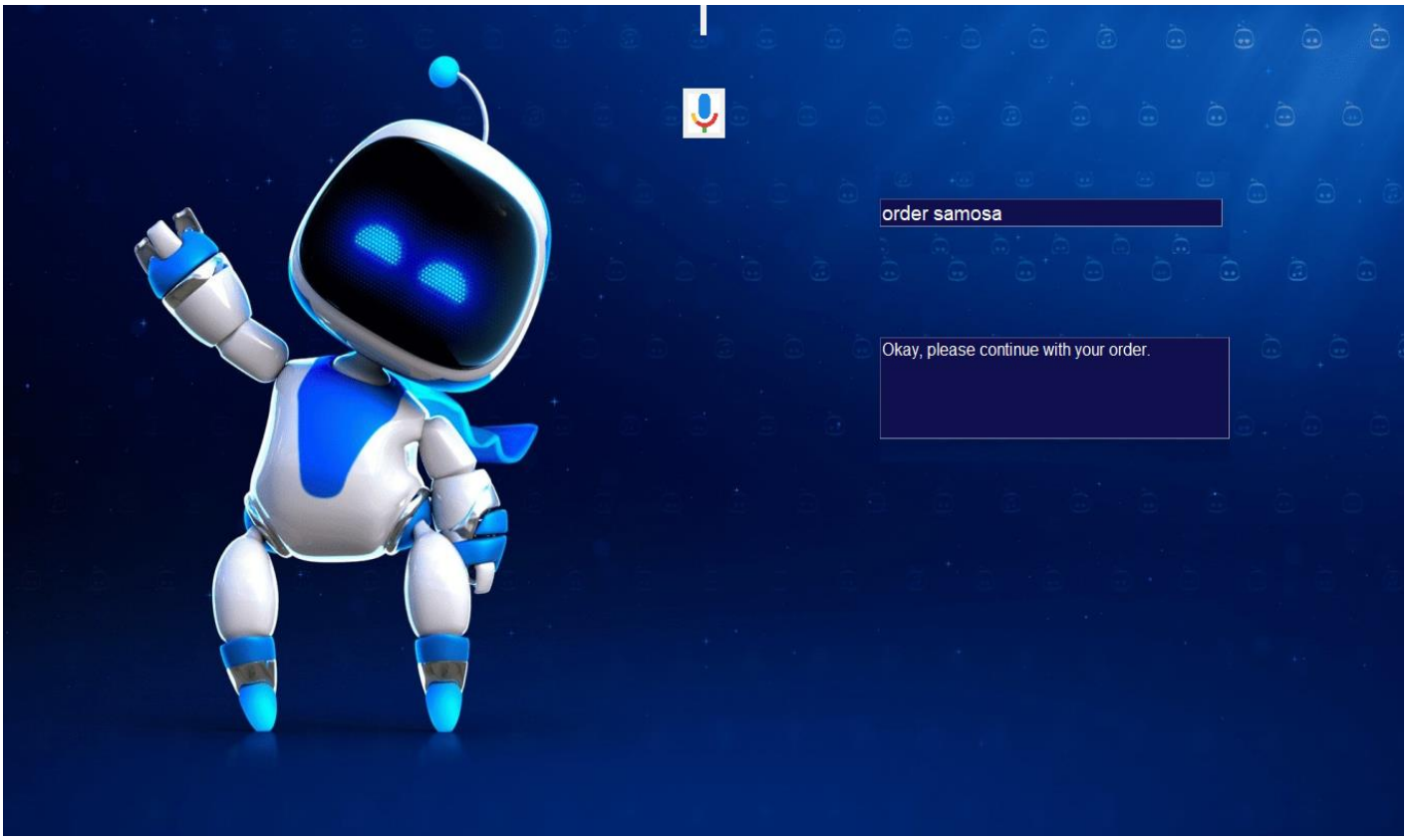
Item Ordering

Ordering an food item



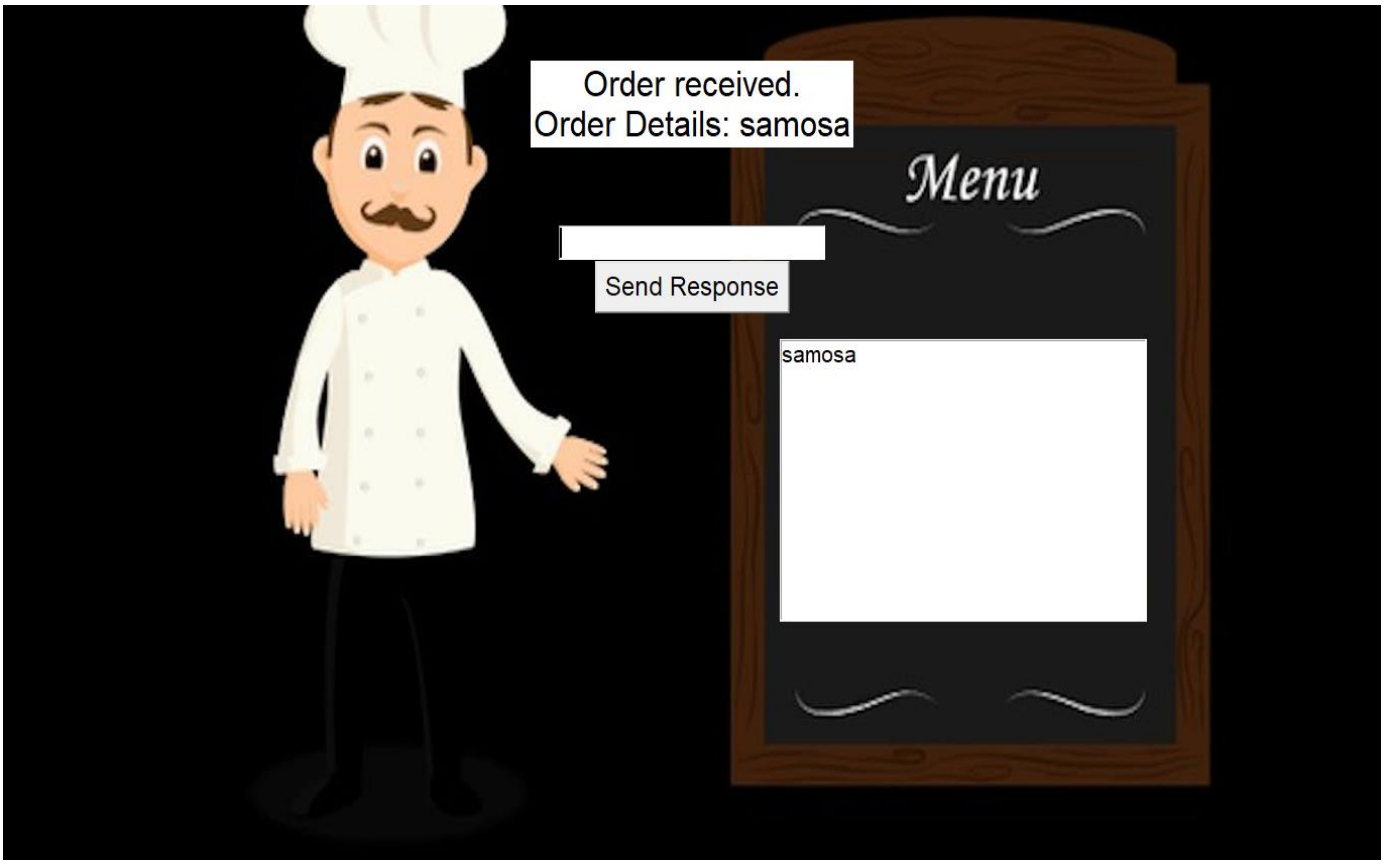
Order cancelling

Canceling the current order and selecting other items



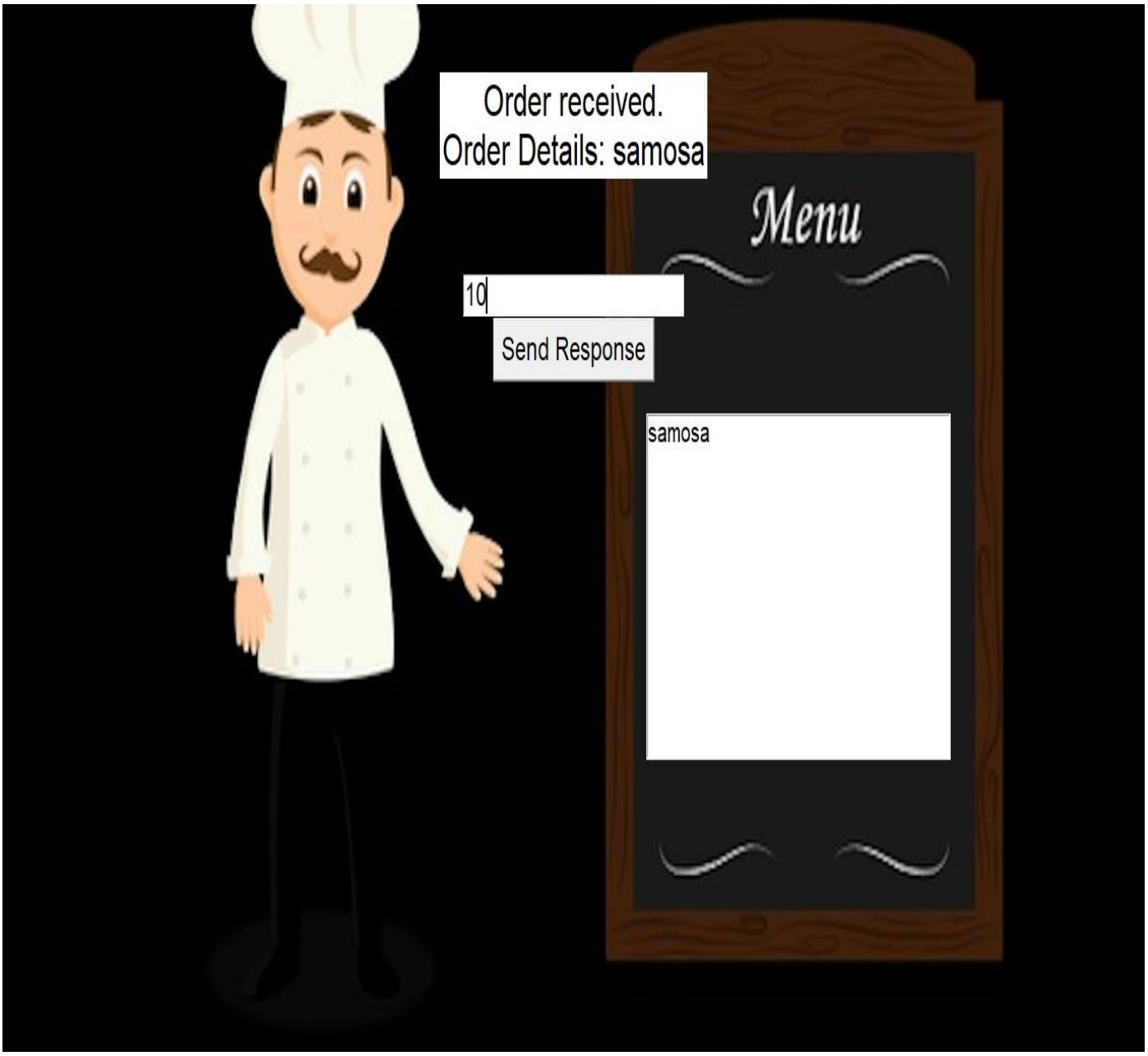
Order Received

Order details sent to chef and chef will start making of the dish and list of items will appear on the right label box which need to be prepared



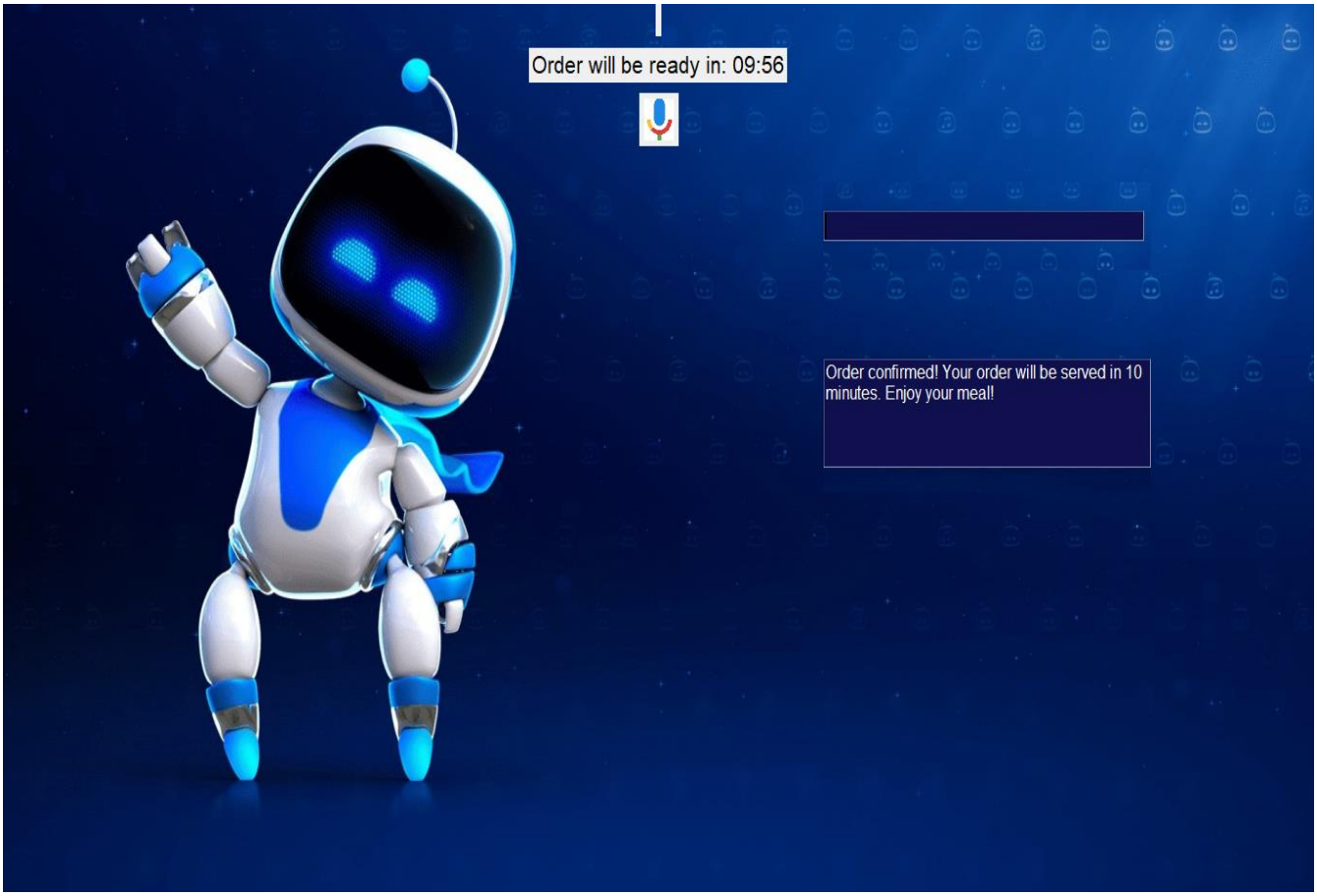
Send Response

Chef will send the response to AI regarding the food tracking



Order Tracking

Tracking of the at user end using UI



6.CHAPTER

TEST CASES

Voice Recognizing:

Without clicking mic button giving input, it doesn't take any input.
By clicking the mic button, it take user input and processes it.
Giving input in other languages, system don't recognize the language

TCP Socket:

By giving wrong ip address to both systems, connection is failed

By giving correct ip address credentials, connection is established between the systems

Multiple orders

By giving multiple orders at a time it prioritize the orders items by following FIFO.

Order Cancellation

order cancellation when user requests for a order cancel, the AI cancels the orders.

7.CHAPTER

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

In conclusion, AI-based waiters have the potential to revolutionize the food service industry by automating and enhancing various aspects of the dining experience. AI technologies, such as natural language processing, computer vision, and machine learning, can be utilized to create intelligent systems that can assist with order taking, food delivery, and customer service.

One of the key advantages of AI-based waiters is their ability to improve efficiency and reduce human errors. They can accurately process and record customer orders, ensuring that the right items are delivered to the right tables. AI-based systems can also integrate with kitchen systems, enabling faster and more streamlined food preparation and delivery.

Moreover, AI-based waiters can enhance the customer experience by providing personalized recommendations based on customer preferences and past orders. They can analyze data and make suggestions for food and beverage pairings, special offers, and promotions, leading to increased customer satisfaction and loyalty.

7.2FUTURE SCOPE

This project further can be developed by some enhancements.

- Facial recognition and saving user details like preferences in food items
- Multilingual
- Food Suggestions
- Advance NLP
- NER
- Better Human interaction

8.CHAPTER

REFERENCES

https://www.alibabacloud.com/blog/serveu-ai-powered-virtual-waiter-for-restaurants_597185

https://www.w3schools.in/python/gui-programming

https://www.javatpoint.com/pyqt-library-in-python

https://www.javatpoint.com/create-the-first-gui-application-using-pyqt5-in-pyth