A

Semester Project-III Report

On

“Food Ordering Chatbot Using Dialogflow”

In partial fulfillment of requirements for the degree of

Bachelor of Technology

In

Artificial Intelligence and Machine Learning

Submitted By

1. Rajput Milind Rajendra	(TY-AIML-27)
2. Dhangar Gaurav Dnyaneshwar	(TY-AIML-37)
3. Pardeshi Mayuresh Ranjitsing	(TY-AIML-39)
4. More Parth Rakesh	(TY-AIML-40)
5. Vispute Tanmay Chandrashekhar	(TY-AIML-48)

Under the Guidance of

Prof. A. D. Mairale

R. C. PATEL
INSTITUTE OF TECHNOLOGY
An Autonomous Institute

The Shirpur Education Society’s

R. C. Patel Institute of Technology, Shirpur - 425405.

Computer Science & Engineering (Data Science) and AIML

[2023-24]

R. C. PATEL
INSTITUTE OF TECHNOLOGY
An Autonomous Institute

The Shirpur Education Society's

R. C. Patel Institute of Technology Shirpur, Dist. Dhule (M.S.)

Artificial Intelligence And Machine Learning

CERTIFICATE

This is to certify that the Semester Project-II entitled "**Chatbot Using DialogFlow"** has been carried out by team:

1. **Rajput Milind Rajendra**	**(TY-AIML-27)**
2. **Dhangar Gaurav Dnyaneshwar**	**(TY-AIML-37)**
3. **Pardeshi Mayuresh Ranjitsing**	**(TY-AIML-39)**
4. **More Parth Rakesh**	**(TY-AIML-40)**
5. **Vispute Tanmay Chandrashekhar**	**(TY-AIML-48)**

under the guidance of **Prof. A. D Mairale** in partial fulfillment of the requirement for the degree of Bachelor of Technology in Department of Artificial Intelligence and Machine Learning (Semester-V) of Dr. Babasaheb Ambedkar Technological University, Lonere during the academic year 2023-24.

Date:

Place: Shirpur

Guide
Prof. A. D. Mairale

Semester Project-II Coordinator
Prof. P. D. Lanjewar

H.O.D.
Prof. Dr. R. B. Wagh

Director
Prof. Dr. J. B. Patil

ACKNOWLEDGEMENT

Thank you very much **Prof. A. D. Mairale** for offering your assistance on the project. I really appreciate your willingness to help out outside your current position. It is helpful to have someone who has had experience with similar issues on previous projects to offer guidance and direction.

Prof. Dr . R. B. Wagh have truly helped us . We appreciate you taking the time and effort to mentor us and ensure that we're all doing our best for the team.

Director. Dr . J. B. Patil have truly helped us so we have expressed our gratitude towards them Thank you for all the years of guidance and advice you have bestowed upon us.

Project Team:

1. Rajput Milind Rajendra	**(TY-AIML-27)**
2. Dhangar Gaurav Dnyaneshwar	**(TY-AIML-37)**
3. Pardeshi Mayuresh Ranjitsing	**(TY-AIML-39)**
4. More Parth Rakesh	**(TY-AIML-40)**
5. Vispute Tanmay Chandrashekhar	**(TY-AIML-48)**

ABSTRACT

The FoodBot project represents a comprehensive and ambitious endeavor aimed at designing and implementing an end-to-end food delivery chatbot system. At its core lies Dialogflow, a robust natural language processing platform, serving as the project's initial building block. In the initial phases, we dive deep into Dialogflow's foundational components, including intents, entities, contexts, and more, ensuring a thorough grasp of these critical concepts.As the project progresses, we extend our efforts to create a resilient backend using Python and FastAPI. This backend acts as the backbone of the chatbot, facilitating seamless interactions between users and the system. It handles user requests, processes information, and responds intelligently, ensuring a smooth and efficient user experience. Furthermore, we integrate a MySQL database into the ecosystem, enabling effective data management. This database stores essential information such as user profiles, order histories, and menu items, allowing the chatbot to retrieve and update data seamlessly, thereby personalizing user interactions and streamlining food delivery processes.Throughout the project, we emphasize key aspects of chatbot development, including natural language understanding, intent recognition, context management, and real-time communication. The FoodBot project serves as a comprehensive guide for creating a sophisticated food delivery chatbot system, catering to developers of all levels of expertise in conversational AI and backend development, and promising to enhance the food delivery experience for users in a technologically advanced and user-friendly manner.

Chapter 1
Introduction

1.1 Introduction

We are developing a food order receiving and order tracking chatbot for a restaurant website, utilizing Google Dialogflow and Python FastAPI. This comprehensive system is designed to optimize both user experience and restaurant operations. Dialogflow serves as the user-facing component, enabling customers to interact naturally with the chatbot, placing orders, inquiring about menu items, and checking the status of their orders through conversational language. Within Dialogflow, we set up a dedicated agent, create various intents, and configure training phrases to capture a wide range of user inputs. These intents represent different user actions, such as placing orders, menu inquiries, and order tracking. We also utilize entities to extract important details from user queries, like specific food items for ordering and order-related information. Dialogflow's natural language processing capabilities ensure that customers can interact with the chatbot in a manner that feels intuitive and human-like, significantly enhancing the user experience. On the backend, Python FastAPI plays a crucial role in processing user requests and connecting them to the restaurant's internal systems. We define endpoints to handle actions like order placement and status checks, and FastAPI interacts with a database to store and retrieve order information, customer data, and menu items. This architecture enables real-time order confirmation, order tracking, and access to up-to-date menu details. FastAPI can also integrate seamlessly with external systems, such as the kitchen order processing system, to ensure efficient and timely order fulfillment. We prioritize security, implementing measures to protect sensitive customer data, and rigorous testing and deployment procedures guarantee that the chatbot runs smoothly on the restaurant's website. Finally, our user documentation provides clear guidance on how customers can interact with the chatbot and access assistance when needed, contributing to a user-friendly and efficient ordering process.

1.2 Problem statement

The restaurant industry faces the challenge of streamlining the food ordering process while providing a convenient and personalized experience for customers. To address this issue, we aim to develop a food order receiving and order tracking chatbot for our restaurant's website. The chatbot needs to be capable of understanding natural language inputs from customers, allowing them to easily place food orders, inquire about menu items, and track the status of their orders. The solution must integrate seamlessly with our existing restaurant systems, such as the kitchen order processing system, while ensuring data security and user authentication. Moreover, the chatbot should offer a user-friendly and intuitive interface, guiding customers through the order placement process. By implementing this chatbot, we aim to enhance the customer experience, reduce order processing times, and ultimately boost customer satisfaction and operational efficiency in our restaurant.

1.3 Objective

The objective of developing a Dialogflow-powered chatbot for a hotel's online food ordering system is to create an engaging and efficient user experience. The chatbot aims to facilitate seamless interactions, allowing customers to browse menus, customize orders, and place requests effortlessly through natural language conversations. The 24/7 availability enhances convenience, while personalized recommendations and realtime order tracking contribute to a personalized and informed ordering process. Integration with secure payment systems ensures smooth transactions, and multi-platform accessibility enables users to place orders through various channels. The chatbot also serves as a feedback and support mechanism, encouraging customer engagement and satisfaction. Additionally, the system incorporates dynamic updates for menus and specials, leveraging data analytics to gain insights into customer preferences and optimize overall service delivery. The overarching goal is to enhance customer satisfaction, streamline operations, and contribute to the success of the hotel's online food ordering service.

1.4 Need of Project

The need for developing a chatbot for online food orders on a hotel website lies in addressing several key objectives. Firstly, it enhances user engagement by providing a conversational interface, making the ordering process more user-friendly and enjoyable. This, in turn, contributes to efficient order processing and allows for 24/7 availability, accommodating users in different time zones. The chatbot facilitates personalized interactions, offering recommendations based on user preferences, and integrates seamlessly with secure payment systems for convenient transactions. Multi-platform accessibility ensures a broader reach, and real-time order tracking provides transparency for customers. Additionally, the chatbot serves as a valuable tool for customer feedback and support, fostering engagement and satisfaction. The project also aims to leverage data analytics to gain insights into customer behavior and preferences, enabling continuous improvement and optimization of the online food ordering service. Ultimately, the project addresses the evolving demands of the digital marketplace, offering a streamlined and modern approach to enhance the overall customer experience for online food orders.

1.5 Purpose and Scope

The scope of the project encompasses various dimensions to create a comprehensive and effective online food ordering system. The chatbot's primary scope includes facilitating order placements through natural language conversations, supporting 24/7 availability, and integrating with secure payment systems for seamless transactions. It extends to personalized user experiences with recommendations based on preferences and real-time order tracking. The project also involves multiplatform accessibility, ensuring users can place orders through diverse channels. Additionally, the chatbot serves as a platform for customer feedback and support, promoting engagement and satisfaction. The incorporation of data analytics widens the scope by providing insights into customer behavior, preferences, and system performance, enabling continuous improvement and optimization over time.

Chapter 2
Related Concept

2.1 Related concept

The development of a chatbot for online food orders on a hotel website entails a multifaceted integration of various related concepts to optimize user engagement and operational efficiency. Central to this is Natural Language Processing (NLP), a fundamental aspect enabling the chatbot to comprehend and respond to user inputs in a human-like manner, fostering a more interactive and intuitive conversational experience. The design of the Conversational User Interface (CUI) plays a pivotal role, focusing on creating an interface that facilitates seamless communication and ensures a positive overall user experience. User Experience (UX) design principles are crucial for refining the chatbot's interface, considering user feedback, and streamlining user flows to enhance satisfaction throughout the ordering process.

The concept of Menu Management involves structuring and presenting the diverse array of food items in a logical and user-friendly manner, encompassing categories, pricing, detailed descriptions, and customization options. Secure Payment Integration is paramount, ensuring the implementation of robust systems to safeguard user payment information during transactions, thereby establishing trust and security in the online ordering process.

Multi-Platform Development is another critical consideration, requiring the adaptation of the chatbot's functionality and design to various platforms, including the hotel website, mobile applications, and diverse messaging channels, thereby expanding its reach and accessibility. Real-Time Tracking functionality enhances user transparency, providing live updates on the status of orders from preparation to delivery.

Moreover, the incorporation of Personalization and Recommendation Systems adds a layer of sophistication, leveraging data and algorithms to tailor the user experience based on individual preferences. By analyzing user interactions and behaviors, Data Analytics becomes a powerful tool for extracting valuable insights, contributing to informed decision-making, continuous system improvement, and a deeper understanding of evolving customer preferences over time. Finally, the inclusion of Feedback Mechanisms allows users to actively participate in the improvement process, providing ratings, reviews, and suggestions, thereby fostering a dynamic and customer-centric approach to the development of the online food ordering system.

2.2 What is DialogFlow ?

Dialogflow, a comprehensive natural language processing (NLP) platform developed by Google, serves as a robust tool for constructing intelligent and dynamic conversational applications, including chatbots and voice interfaces. At its core, Dialogflow employs sophisticated machine learning algorithms to discern user intent, allowing developers to define various intents, each associated with specific actions or responses. Additionally, the platform facilitates entity recognition, capturing crucial parameters within user input. Agents, the virtual conversational entities created on Dialogflow, comprise a collection of intents, entities, and contextual settings, forming the backbone of a chatbot or voice application.

Developers leverage training phrases and responses within intents to enable the system to comprehend a diverse range of user inputs. Integration capabilities extend Dialogflow's functionality across different messaging platforms, like Facebook Messenger or Slack, as well as voice platforms such as Google Assistant. Furthermore, Dialogflow offers multilingual support, ensuring adaptability to diverse linguistic preferences and global user bases. The platform emphasizes continuous learning, utilizing machine learning to improve intent recognition and overall performance based on real-world interactions. Security features, including data encryption and compliance with industry standards, enhance the platform's suitability for applications demanding robust privacy measures. In essence, Dialogflow stands as a versatile and powerful solution, empowering developers to craft sophisticated conversational interfaces that seamlessly interpret and respond to user input in a natural, context-aware, and globally accessible manner.

2.3 What is MySQL WorkBench ?

MySQL Workbench stands out as a comprehensive and versatile database management tool developed by Oracle, designed to cater to the diverse needs of developers, database administrators, and data architects engaged with MySQL databases. This robust application combines a visually intuitive design interface with powerful administrative capabilities, providing users with a multifaceted solution for tasks ranging from conceptualizing and designing database schemas to executing complex SQL queries and managing user access and permissions.At its core, MySQL Workbench facilitates the creation and modification of database structures through a graphical interface, allowing users to effortlessly model and visualize intricate relationships between different tables and entities. This visual design aspect not only enhances the efficiency of database development but also provides a clear and insightful representation of the interconnectedness within the data model.In addition to its design features, MySQL Workbench serves as a potent administrative tool. It empowers users with tools for performance monitoring, enabling the identification and optimization of query execution and resource usage. The tool's backup and restore functionalities contribute to data security and recovery strategies, ensuring the integrity of critical information.Furthermore, MySQL Workbench offers support for various MySQL server versions, making it adaptable to diverse database environments. Its versatility extends to encompassing tasks such as database migration, making it a valuable asset for projects involving the transition between different database systems or versions.The tool's user-friendly interface, coupled with its ability to handle both design and administrative aspects seamlessly, positions MySQL Workbench as an indispensable asset in the realm of MySQL database management. Whether used for small-scale development projects or large-scale enterprise applications, its feature-rich capabilities contribute to enhanced productivity and streamlined workflows in the dynamic landscape of database administration and development.

2.4 What is NGROK?

Ngrok is a powerful and versatile tool that serves as a secure tunneling service, facilitating the creation of secure connections between local servers and the internet. It was designed to address the challenges associated with exposing locally hosted applications, particularly during development, testing, or collaboration scenarios. Ngrok simplifies the process of making local servers publicly accessible by creating a secure tunnel and assigning a temporary public URL to the local environment. This eliminates the complexities often associated with network configurations, firewalls, or NAT traversal issues. The way Ngrok operates is by establishing a connection to its servers in the cloud, which act as intermediaries between the local server and the internet. This connection is secured using TLS/SSL, ensuring the confidentiality and integrity of data transferred through the tunnel. Ngrok supports various protocols, including HTTP, HTTPS, and TCP, making it compatible with a wide range of applications and services. One of Ngrok's notable features is its ability to dynamically assign a public URL to the local server, eliminating the need for a static IP address or domain. This dynamic nature proves particularly beneficial during development and testing phases when frequent changes are made to the application. Developers often leverage Ngrok to share their work with clients, collaborators, or stakeholders without deploying it to a public server, enabling seamless remote access for demonstration and feedback purposes. The tool also offers insights into the traffic being tunneled, making it valuable for monitoring and debugging purposes. Ngrok's ease of use, security features, and flexibility have made it a popular choice in the development community, streamlining the process of exposing local services to the internet while maintaining a robust level of security and facilitating collaboration across different environments.

Chapter 3
Hardware and Software requirement

3.1 Hardware requirement:

PROCESSOR	INTEL i3 OR HIGHER
RAM	512 MB & ABOVE
HARD DISK DRIVE	50 GB FREE SPACE OR ABOVE

3.2 Software requirement

- Operating System : Windows 10 or Windows 11
- PyCharm
- VS Code
- MySQL WorkBench
- NGROK

Chapter 4

Implementation

4.1 Implementation

Set Up Dialogflow:

Create a Dialogflow agent and define various intents for ordering, menu inquiries, and order
tracking.
Configure entities to extract key information like menu items, order details, and customer data.

FastAPI Backend:

Develop FastAPI endpoints to handle incoming requests from Dialogflow.
Implement routes for actions such as order placement, order status checks, and external system
integration.

Database Integration:

Choose and set up a database system (e.g., SQL or NoSQL) to store order information, menu
items, and customer data.
Develop database schemas and implement data storage and retrieval logic in FastAPI.

User Interaction Flow:

Design the conversation flow within Dialogflow, including how the chatbot responds to user queries and guides them through the order process.

Ensure a user-friendly experience with informative responses and clear instructions.

Authentication and Security:

Implement user authentication to secure sensitive customer data. Apply encryption and access control measures to protect the chatbot and database.

Integration with External Systems:

Connect the chatbot and FastAPI with the restaurant's kitchen order processing system or any other relevant external systems. Ensure realtime order processing and tracking.

Testing and Quality Assurance:

Thoroughly test the chatbot to ensure it handles various user scenarios effectively. Test the integration with external systems for reliability.

Verify security measures.

Deployment:

Deploy the chatbot and FastAPI backend on the restaurant's website.

Ensure that it's accessible and responsive to user requests.

User Documentation:

Create clear and concise user documentation to guide customers on how to interact with the chatbot. Provide instructions for order placement, menu inquiries, and order tracking.

Monitoring and Maintenance:

Set up monitoring and alerting to track the chatbot's performance and user interactions. Regularly maintain and update the chatbot to improve its functionality and address any issues orfeedback from customers.

4.2 Pseudo code for Webhook

```
from fastapi import FastAPI from
fastapi import Request
from fastapi.responses import JSONResponse
import db_helper import gener

app = FastAPI()

inprogress_orders = {}

@app.post("/") async def
handle_request(request: Request):     #
Retrieve the JSON data from the request
payload = await request.json()

    # Extract the necessary information from the payload
    # based on the structure of the WebhookRequest from Dialogflow
intent = payload['queryResult']['intent']['displayName']     parameters
```

```
= payload['queryResult']['parameters']     output_contexts =
payload['queryResult']['outputContexts']     session_id =
generic_helper.extract_session_id(output_contexts[0]["name"])

    intent_handler_dict = {
        'order.add - context: ongoing-order': add_to_order,
        'order.remove - context : ongoing-order': remove_from_order,
        'order.complete - context: ongoing-order': complete_order,
        'track.order-context: ongoing-tracking': track_order
    }

    return intent_handler_dict[intent](parameters, session_id)

def save_to_db(order: dict):
    next_order_id = db_helper.get_next_order_id()

    # Insert individual items along with quantity in orders table
for food_item, quantity in order.items():        rcode =
db_helper.insert_order_item(          food_item,          quantity,
next_order_id
        )

        if rcode == -1:
            return -1

    # Now insert order tracking status
    db_helper.insert_order_tracking(next_order_id, "in progress")

    return next_order_id
```

```
def complete_order(parameters: dict, session_id: str):
if session_id not in inprogress_orders:
    fulfillment_text = "I'm having a trouble finding your order. Sorry! Can
you place a new order please?"    else:
    order = inprogress_orders[session_id]      order_id =
save_to_db(order)      if order_id == -1:         fulfillment_text =
"Sorry, I couldn't process your order due to a backend error. " \
                  "Please place a new order again"
else:
      order_total = db_helper.get_total_order_price(order_id)

      fulfillment_text = f"Awesome. We have placed your order. " \
f"Here is your order id # {order_id}. " \
                f"Your order total is {order_total} which you can pay at the
time of delivery!"

    del inprogress_orders[session_id]

  return JSONResponse(content={
    "fulfillmentText": fulfillment_text
  })



def add_to_order(parameters: dict, session_id: str):
  food_items      =      parameters["food-item"]
quantities = parameters["number"]

  if len(food_items) != len(quantities):
        fulfillment_text = "Sorry I didn't understand. Can you please specify
food items and quantities clearly?"     else:
        new_food_dict = dict(zip(food_items, quantities))

        if session_id in inprogress_orders:
            current_food_dict = inprogress_orders[session_id]
current_food_dict.update(new_food_dict)
inprogress_orders[session_id] = current_food_dict
else:
            inprogress_orders[session_id] = new_food_dict

        order_str =
generic_helper.get_str_from_food_dict(inprogress_orders[session_id])
fulfillment_text = f"So far you have: {order_str}. Do you need anything
else?"

    return JSONResponse(content={
        "fulfillmentText": fulfillment_text
    })


def remove_from_order(parameters: dict, session_id: str):
if session_id not in inprogress_orders:         return
JSONResponse(content={
            "fulfillmentText": "I'm having a trouble finding your order. Sorry!
Can you place a new order please?"
        })
```

```
   food_items                =               parameters["food-item"]
current_order = inprogress_orders[session_id]

   removed_items               =               []
no_such_items = []

   for item in food_items:         if
item not in current_order:
no_such_items.append(item)
else:
         removed_items.append(item)
del current_order[item]

   if len(removed_items) > 0:
      fulfillment_text   =   f'Removed   {",".join(removed_items)}   from   your
order!'

   if len(no_such_items) > 0:         fulfillment_text = f' Your
current order does not have
{",".join(no_such_items)}'

   if len(current_order.keys()) == 0:
      fulfillment_text   +=   "   Your   order   is   empty!"
else:
      order_str      =      generic_helper.get_str_from_food_dict(current_order)
fulfillment_text += f" Here is what is left in your order: {order_str}"

   return JSONResponse(content={
      "fulfillmentText": fulfillment_text
   })
```

```
def track_order(parameters: dict, session_id: str):
   order_id = int(parameters['order_id'])     order_status =
db_helper.get_order_status(order_id)     if order_status:
fulfillment_text = f"The order status for order id: {order_id} is:
{order_status}"
else:
      fulfillment_text = f"No order found with order id: {order_id}"

   return JSONResponse(content={
      "fulfillmentText": fulfillment_text
   })
```

4.3 Pseudo code for Database Connectivity

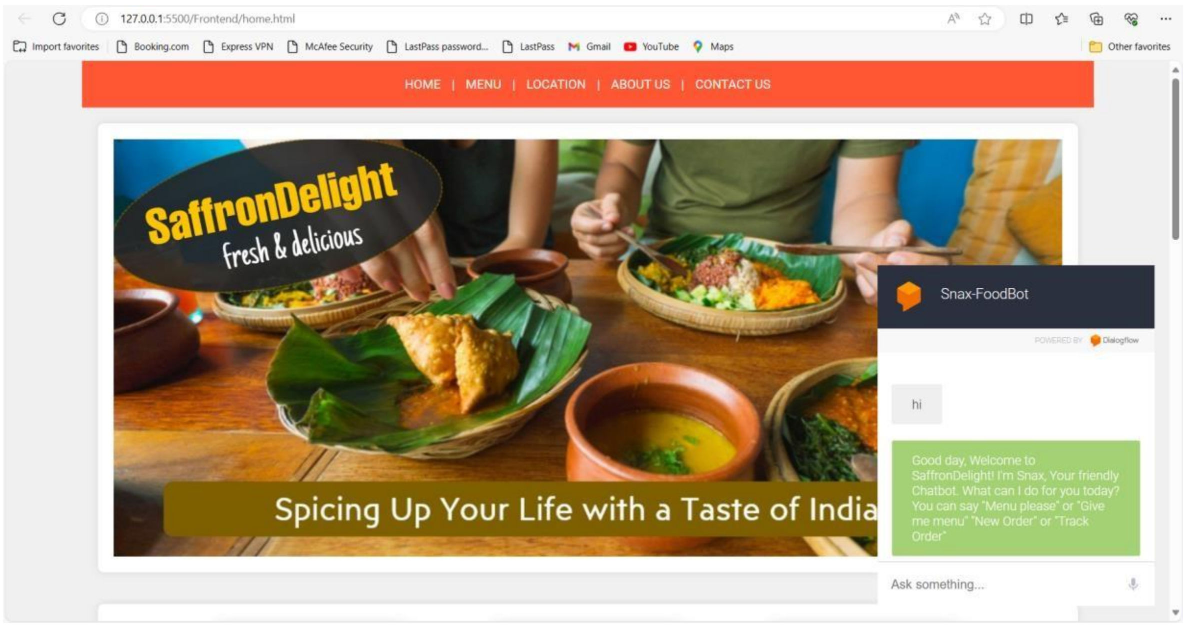
```
import mysql.connector global
cnx

cnx = mysql.connector.connect(
host="localhost",     user="root",
password="root",
database="pandeyji_eatery"
)

# Function to call the MySQL stored procedure and insert an order item
def insert_order_item(food_item, quantity, order_id):     try:
      cursor = cnx.cursor()
```

```
        # Calling the stored procedure
        cursor.callproc('insert_order_item', (food_item, quantity, order_id))

        #       Committing       the       changes
cnx.commit()

        # Closing the cursor
cursor.close()       print("Order item
inserted successfully!")


        return 1

    except mysql.connector.Error as err:
        print(f"Error inserting order item: {err}")

        #      Rollback      changes      if      necessary
cnx.rollback()

        return -1

    except Exception as e:
        print(f"An error occurred: {e}")
# Rollback changes if necessary
cnx.rollback()

        return -1

# Function to insert a record into the order_tracking table def
insert_order_tracking(order_id, status):
    cursor = cnx.cursor()

    # Inserting the record into the order_tracking table
    insert_query = "INSERT INTO order_tracking (order_id, status) VALUES
(%s, %s)" cursor.execute(insert_query, (order_id,
    status))
    #       Committing       the       changes
cnx.commit()

    #       Closing       the       cursor
cursor.close()

def get_total_order_price(order_id):
    cursor = cnx.cursor()

    # Executing the SQL query to get the total order price
query = f"SELECT get_total_order_price({order_id})"
cursor.execute(query)

    # Fetching the result     result
= cursor.fetchone()[0]

    #       Closing       the       cursor
cursor.close()

    return result
```

```
# Function to get the next available order_id
def get_next_order_id():     cursor =
cnx.cursor()

   # Executing the SQL query to get the next available order_id
query = "SELECT MAX(order_id) FROM orders"
cursor.execute(query)

   # Fetching the result     result
= cursor.fetchone()[0]

   #        Closing        the        cursor
cursor.close()

   #  Returning  the  next  available  order_id
if result is None:
      return          1
else:
      return result + 1

# Function to fetch the order status from the order_tracking table def
get_order_status(order_id):
   cursor = cnx.cursor()

   # Executing the SQL query to fetch the order status       query =
f"SELECT status FROM order_tracking WHERE order_id =
{order_id}"
cursor.execute(query)
```

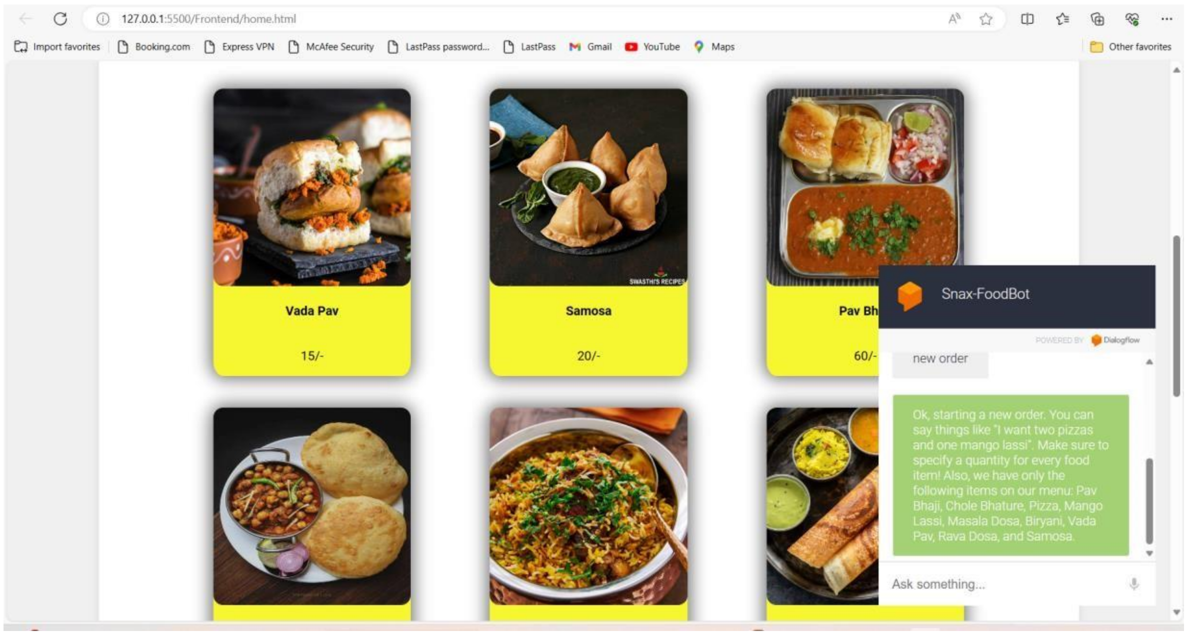
```
    #    Fetching    the    result
result = cursor.fetchone()


    # Closing the cursor cursor.close()

    #  Returning  the  order  status
if result:
        return         result[0]
else:
        return None



if __name__ == "__main__":
    # print(get_total_order_price(56))
    # insert_order_item('Samosa', 3, 99)
    # insert_order_item('Pav Bhaji', 1, 99)     #
insert_order_tracking(99, "in progress")
print(get_next_order_id())
```

4.4 Snapshot of Output


127.0.0.1:5500/Frontend/home.html
HOME | MENU | LOCATION | ABOUT US | CONTACT US
SaffronDelight
fresh & delicious
Spicing Up Your Life with a Taste of India
Snax-FoodBot
Ask something...

hi

Good day, Welcome to SaffronDelight! I'm Snax, Your friendly Chatbot. What can I do for you today? You can say "Menu please" or "Give me menu" "New Order" or "Track Order"



new order

Ok, starting a new order. You can say things like "I want two pizzas and one mango lassi". Make sure to specify a quantity for every food item! Also, we have only the following items on our menu: Pav Bhaji, Chole Bhature, Pizza, Mango Lassi, Masala Dosa, Biryani, Vada Pav, Rava Dosa, and Samosa.

Chapter 4
Conclusion

In conclusion, the development of a food order receiving and order tracking chatbot for a restaurant website, integrating Google Dialogflow and Python FastAPI, presents a compelling solution to improve both customer satisfaction and operational efficiency in the restaurant industry. By leveraging natural language understanding, secure data management, and real -time integration with restaurant systems, this chatbot offers a user -friendly experience for customers to place orders, inquire about menu items, and track their orders conveniently. The implementation process involves setting up Dialogflow, creating a FastAPI backend, integrating with a database, ensuring data security, and providing clear user documentation. Once deployed, the chatbot serves as a valuable tool to streamline restaurant operations, reduce order processing times, and enhance the overall customer experience

Chapter 6
BIBLIOGRAPHY

1. Youtube – CodeBasics

https://youtu.be/2e5pQqBvGco?si=pKi3m6VYtV_vmwv7