END TO END NLP BASED MINIPROJECT

INTEGRATING CHATBOT TO FOOD WEBSITE

The **primary cause** for developing the "Flavor Fusion" project is to streamline and enhance the food ordering process for a restaurant. The goal is to provide a seamless experience for customers to place and track their orders using a conversational interface.

The aim of the project is to:

Simplify the food ordering process for customers.
Reduce the workload on restaurant staff by automating order intake and tracking.
Improve customer satisfaction by providing real-time order status updates.
Ensure accurate and efficient order management through a robust backend system.

Flavor Fusion

Tools & Techniques

Frontend:

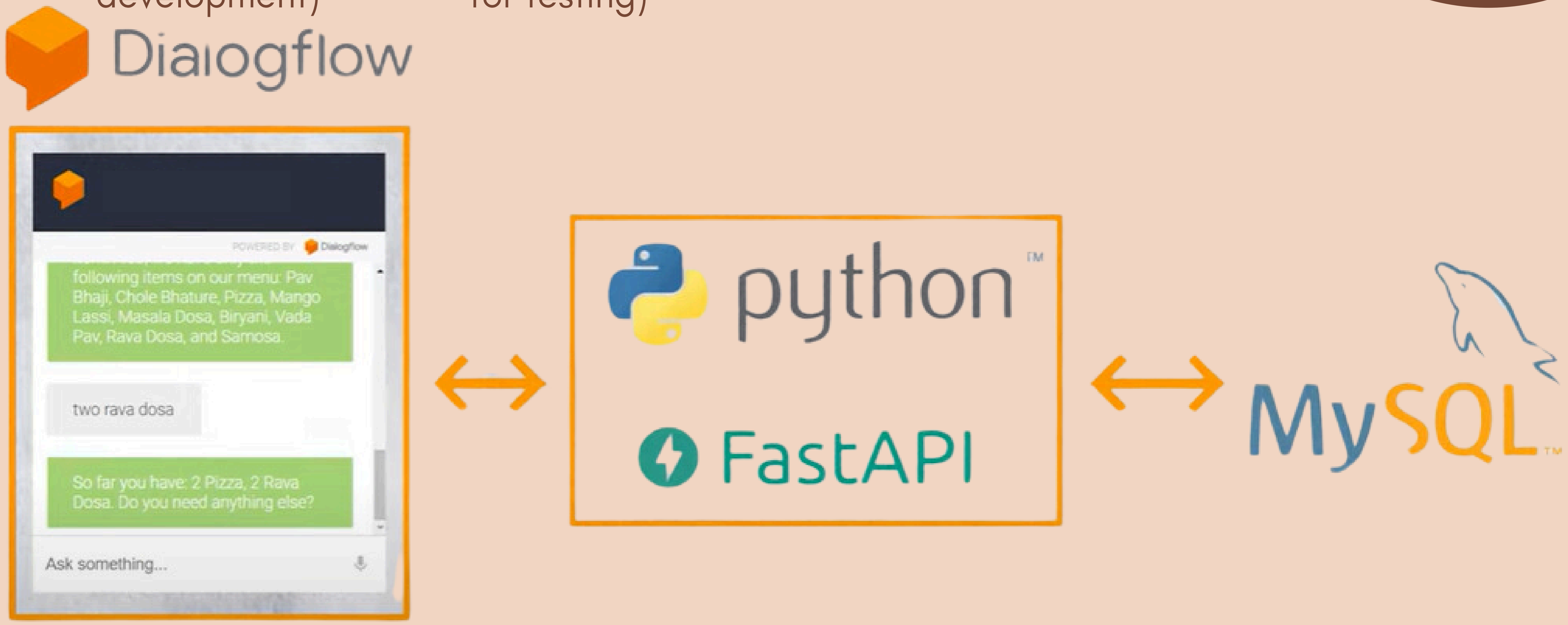
- HTML, CSS (for styling and layout)
- Dialogflow (for chatbot integration)

Backend:

- FastAPI (to handle webhook requests)
- Python (main.py, db_helper.py, generic_helper.py
- MySQL (for database storage)

Miscellaneous:

- PyCharm (IDE for development)
- Ngrok (to expose local server for testing)



following items on our menu: Pav Bhaji, Chole Bhature, Pizza, Mango Lassi, Masala Dosa, Biryani, Vada Pav, Rava Dosa, and Samosa.

two rava dosa

So far you have: 2 Pizza, 2 Rava Dosa. Do you need anything else?

Ask something...

Flavor Fusion Project Overview

Features

Chatbot Integration:
A Dialogflow chatbot embedded in the front-end to facilitate user interaction for ordering and tracking food items.

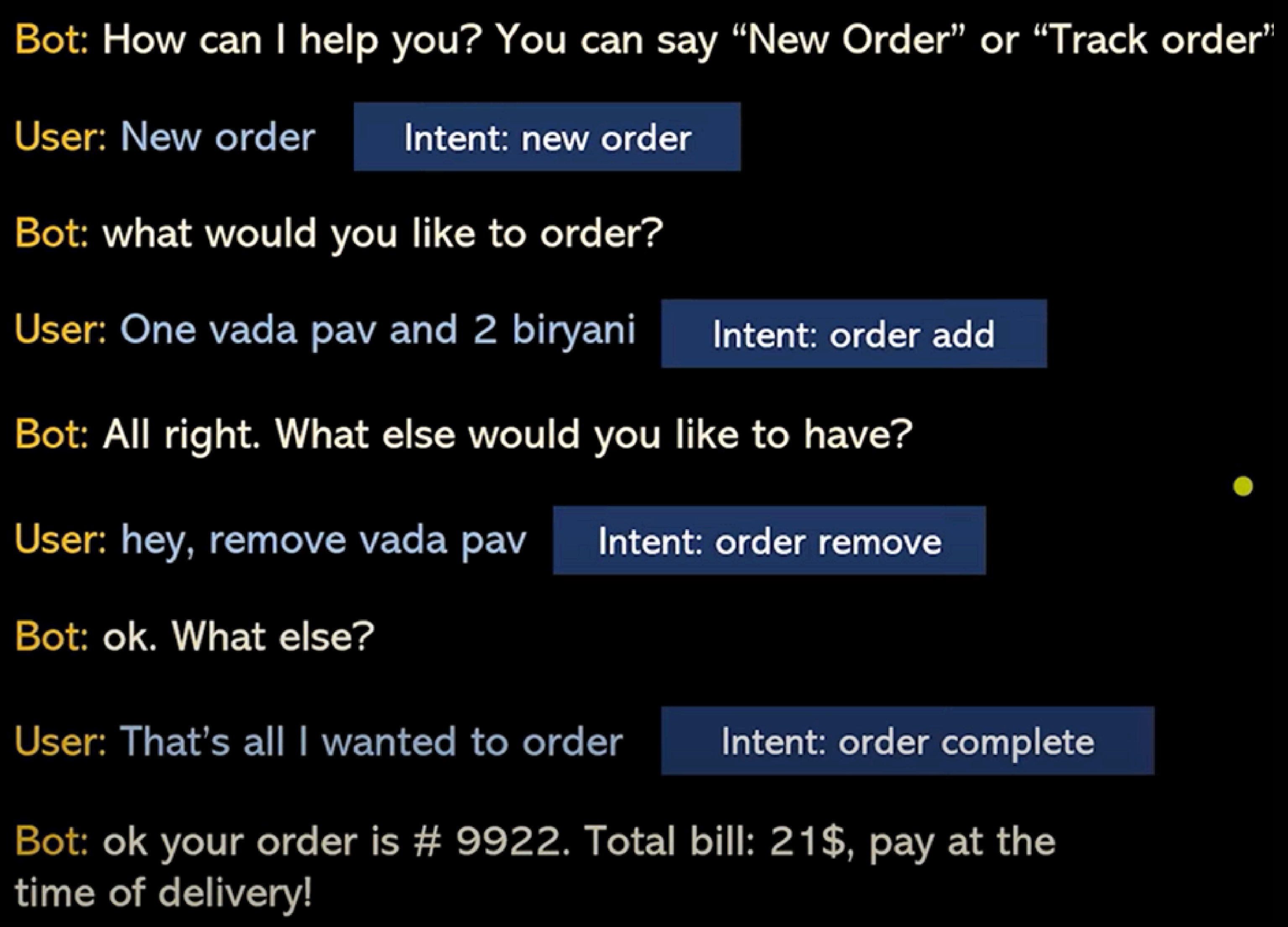
Order Management:
Users can place new orders, add or remove items from an ongoing order, and complete orders.

Order Tracking:
Users can track the status of their orders using the chatbot.

Database Management:
Orders are stored in a MySQL database, which keeps track of order items, quantities, prices, and statuses.

Responsive Design:
The front-end is designed to be user-friendly and responsive, providing a good user experience across different devices.

Scenarious



Bot: How can I help you? You can say “New Order” or “Track order”

User: New order

Intent: new order

Bot: what would you like to order?

User: One vada pav and 2 biryani

Intent: order add

Bot: All right. What else would you like to have?

User: hey, remove vada pav

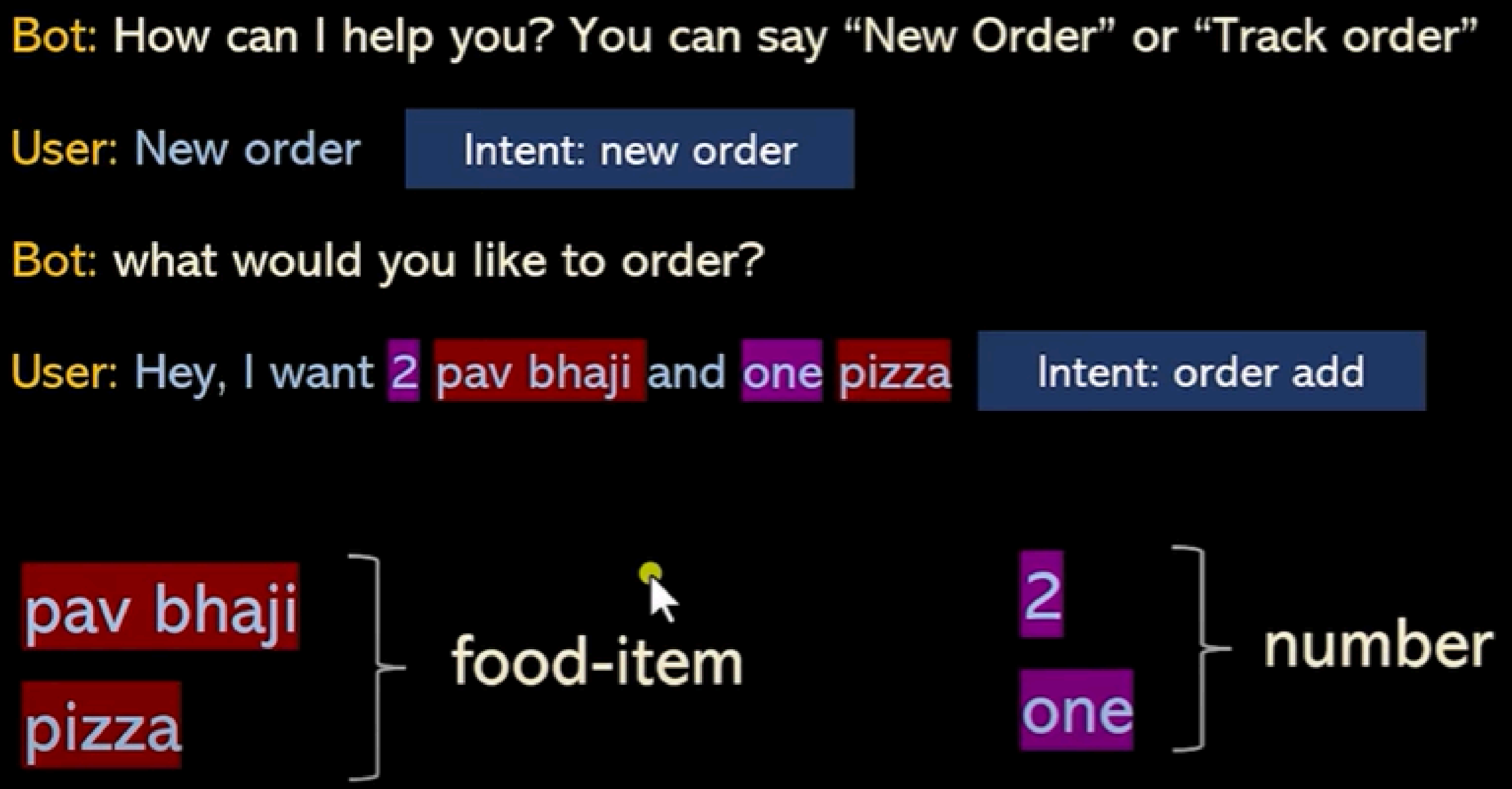
Intent: order remove

Bot: ok. What else?

User: That’s all I wanted to order

Intent: order complete

Bot: ok your order is # 9922. Total bill: 21$, pay at the time of delivery!



Bot: How can I help you? You can say “New Order” or “Track order”

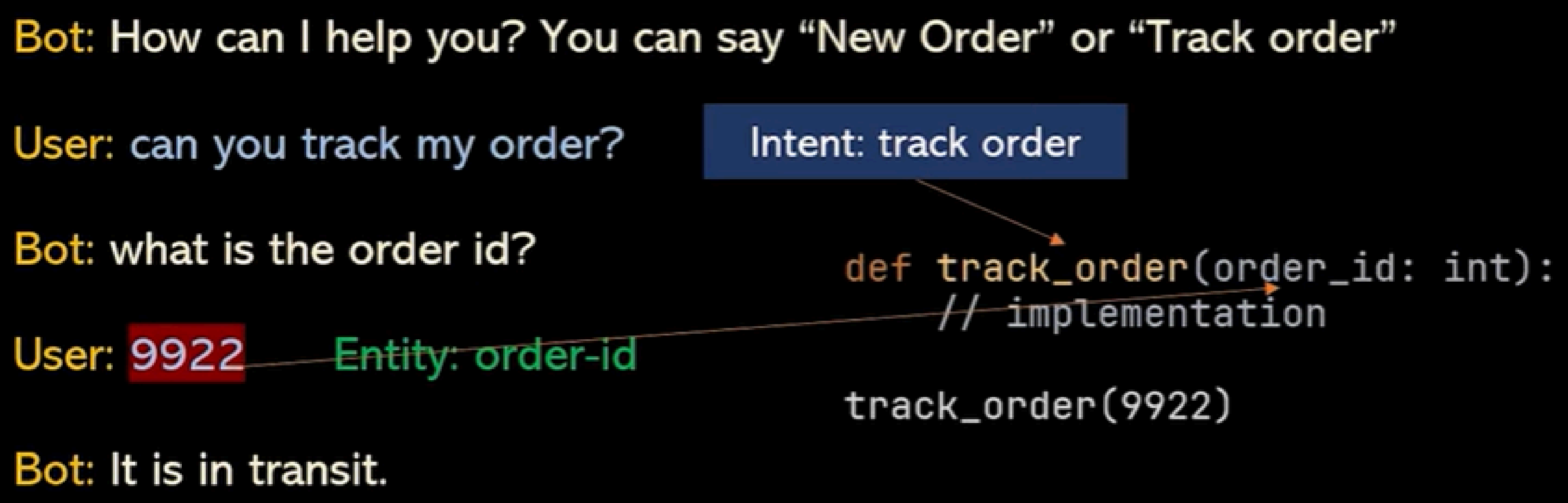
User: New order

Intent: new order

Bot: what would you like to order?

User: Hey, I want 2 pav bhaji and one pizza

Intent: order add



Bot: How can I help you? You can say “New Order” or “Track order”

User: can you track my order?

Intent: track order

Bot: what is the order id?

User: 9922

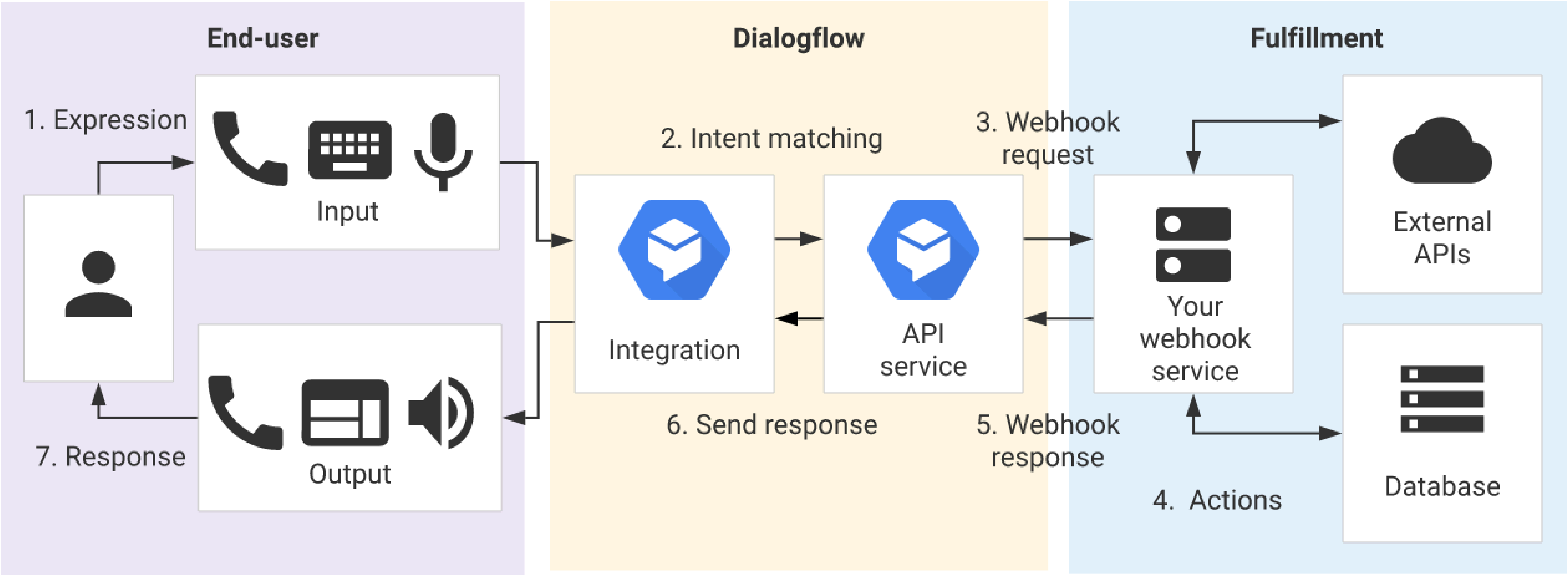
Entity: order-id

Bot: It is in transit.

```
def track_order(order_id: int):
    // implementation

track_order(9922)
```

WORKING


End-user
Dialogflow
Fulfillment
1. Expression
Input
2. Intent matching
3. Webhook request
External APIs
Integration
API service
Your webhook service
Database
7. Response
Output
6. Send response
5. Webhook response
4. Actions

What is Dialogflow?

Dialogflow is a natural language understanding platform that enables developers to design and integrate a conversational user interface into mobile apps, web applications, devices, and bots. It can understand and process user input, and respond with appropriate information.

Role of Dialogflow in Our Project

Dialogflow acts as the conversational interface that allows users to place and manage orders through a chatbot. Users interact with the chatbot embedded in our main.html file, and Dialogflow processes these interactions, determines the user's intent, and triggers appropriate responses or actions.

Intents in Dialogflow

Intents are used to categorize a user's intentions for a conversation turn. Each intent represents a mapping between what a user says and how the system should respond.

Our Defined Intents:

Default Welcome Intent: Greets the user when they initiate a conversation.
new.order: Initiates a new order.
order.add - context: ongoing-order: Adds items to an ongoing order.
order.complete - context: ongoing-order: Completes and places the order.
order.remove - context: ongoing-order: Removes items from an ongoing order.
track.order: Tracks the status of an order.
track.order - context: ongoing-tracking: Specific context for tracking an order.

Webhook Fulfillment

Webhook fulfillment is a mechanism that allows Dialogflow to call an external service (your FastAPI server) to execute logic or fetch data. When an intent is matched, Dialogflow can send a request to your webhook, which processes the request and sends a response back to Dialogflow, which in turn replies to the user.

How Webhook Fulfillment Works:

User Input: The user interacts with the chatbot.
Intent Matching: Dialogflow matches the user input to an intent.
Webhook Call: If the intent has webhook fulfillment enabled, Dialogflow sends a request to the webhook URL.
Webhook Processing: FastAPI server (acting as the webhook) processes the request, executes necessary actions (e.g., adding an item to an order), and prepares a response.
Response: The FastAPI server sends a response back to Dialogflow.
User Response: Dialogflow sends the response back to the user via the chatbot.

What is ngrok?

ngrok is a tool that allows you to expose a local server to the internet securely. It creates a secure tunnel to your localhost and provides a public URL that can be accessed from anywhere.

Role of ngrok in Our Project

ngrok plays a crucial role by allowing Dialogflow to interact with locally hosted FastAPI server. Since Dialogflow needs to send webhook requests to a publicly accessible URL, ngrok provides this by tunneling requests from a public URL to our local machine.

How ngrok Works in our Project

Run ngrok: start ngrok and point it to your local FastAPI server (e.g., localhost:8000).

Generate Public URL: ngrok provides a public URL that forwards requests to our local FastAPI server.

Configure Dialogflow: Configure Dialogflow to use the public URL provided by ngrok as the webhook URL.

Development and Testing: develop and test FastAPI server locally, and Dialogflow can send webhook requests to this public URL, which ngrok will forward to our local server.

What is PyCharm?

PyCharm is a popular integrated development environment (IDE) developed by JetBrains, designed specifically for Python development. It provides a range of tools and features to enhance productivity and efficiency, including code analysis, debugging, testing, and version control integration.

Role of PyCharm in our Project

PyCharm serves as the primary development environment for writing, testing, and debugging our FastAPI server code, as well as managing other aspects of our project such as virtual environments and dependencies.

Backend Components in Our Project

main.py

This is the main FastAPI server file. It defines the API endpoints and the logic for handling different intents from Dialogflow.

Key Components:

FastAPI Initialization: Initializes the FastAPI app.

In-progress Orders: A dictionary to keep track of orders that are currently being processed.

API Endpoint: The handle_request endpoint to receive and handle requests from Dialogflow.

Intent Handlers: Functions to handle various intents such as adding to order, removing from order, completing an order, and tracking an order.

Database Operations: Functions that interact with the database to save orders and track order statuses.

MySQL Database

MySQL is a relational database management system used to store and manage the data related to food orders, order tracking, and other related information.

db_helper.py

This file contains functions for interacting with the MySQL database. These functions perform operations such as inserting order items, tracking orders, and fetching order details.

Key Functions:

Insert Order Item: Calls a stored procedure to insert an item into the order.
Insert Order Tracking: Inserts the order status into the order_tracking table.
Get Total Order Price: Retrieves the total price of an order.
Get Next Order ID: Fetches the next available order ID from the database.
Get Order Status: Retrieves the status of an order based on the order ID.

generic_helper.py

This file includes utility functions that are used across the project. These functions handle common tasks like string manipulation and session ID extraction.

Key Functions:

Get String from Food Dictionary: Converts a dictionary of food items and their quantities to a readable string format.
Extract Session ID: Extracts the session ID from a given session string.