Ai Chatbots & Agents A smarter way to learn Dialogflow



P2P Clouds

DialogFlow Agents

**	Chapte	er 1 : Ge	tting Started	3							
		Dialog	DialogFlow History								
	\succ	Dialog	Flow Introduction	4							
			Capabilities of DialogFlow	4							
	\checkmark	Differe	ence between Dialogflow ES and CX	4							
		-	Dialogflow ES (Essentials):	5							
		-	Dialogflow Cx (Customer Experience):	6							
	4	Chatbo	ot and Ai Agent:	7							
			How to start with DialogFlow tool	7							
	4	What a	are intents with example	8							
		-	Example: Pizza Order Intent	8							
			Steps to Create a Pizza Chatbot	9							
	\land	Contex	xts	13							
			Input Contexts	13							
		-	Output Contexts	13							
***	Chapte	er 2: Fulf	fillment and Webhooks with FastAPI andngrok	14							
		What i	s Fulfillment in DialogflowES?	14							

Chapter 1 : Getting Started

DialogFlow History: Dialogflow, originally known as API.AI, was founded in 2010 by Speaktoit. It was initially developed to provide conversational interfaces for virtual assistants and chatbots. Google acquired API.AI in 2016 and rebranded it as Dialogflow, integrating it with its cloud ecosystem to enhance its natural language processing (NLP) capabilities





Since then, Dialogflow has evolved into a powerful platform for building AI-powered chatbots and voice assistants across multiple channels. Over time, Google introduced new features, including the transition from Dialogflow ES (Essentials) to Dialogflow CX (Customer Experience), offering more advanced conversational flows and state management.

Reference: https://cloud.google.com/dialogflow/es/docs/history

Dialogflow Introduction

Dialogflow is a cloud-based platform that enables developers to design and implement conversational interfaces using natural language understanding (NLU) and machine learning (ML). It helps businesses create AI-driven chatbots and virtual agents that can interact with users via various communication channels, such as websites, mobile apps, messaging apps, and voice-enabled devices. Dialogflow supports multiple languages, making it a global solution for conversational AI.

Capabilities of Dialogflow

- Natural Language Understanding (NLU): Detects user intent and extracts relevant entities from conversations.
- **Omnichannel Integration:** Deploy chatbots on platforms like Google Assistant, Facebook Messenger, Slack, WhatsApp, and more.
- Context Management: Maintains conversation context to provide personalized interactions.
- **Fulfillment:** Connects with backend systems via webhooks to fetch real-time data and execute business logic.
- Multi-language Support: Offers support for multiple languages to cater to global users.
- **Rich Responses:** Enables multimedia-rich interactions such as images, cards, and quick replies.
- **Analytics and Insights:** Provides metrics and logs to track chatbot performance and user engagement.

Difference between Dialogflow ES and CX

Dialogflow Essentials Global +	CREATE INTENT :	Try it now
Fast_Stacks_Program - 🔆	Search interts Q T	D Please use test console above to sentence.
💬 Intents 🛛 🕂	Default Fallback Intent	
Entities +	Default Welcome Intent	
Knowledge [bets]		
🖗 Fulfillment		
Integrations		
Training		
Validation		
History		
ili Analytics		
Prebuilt Agents		
Small Talk		
> Docs 3		
Trial		
Free Upgrade		
Dialogflow CX [new]		
Support 2		

Dialogflow ES (Essentials):

- 1. Designed for simpler chatbot implementations.
- 2. Uses intents and contexts to manage conversations.
- 3. Best for small to medium-sized projects with linear workflows.
- 4. Limited state management capabilities.
- 5. Easier to set up and requires minimal training.

Dialogflow CX (Customer Experience):

≡ 🝦 Dialogflow CX	Project Agent CSR Agent en *		z q	0	P 🔞
Build Manage	Default Start Flow 🔍 🔍 🔄 🔊 Minimap	<> Publish	Agen	t settings	🖳 Test Agent
^ FLOWS + ₩					
🉏 Default Start Flow					
λ Al	Start Page				
🙏 Dev Courses	λ Al				
^ pages → 🖬					
0 0000					
O Start Page					
D AI :					
Designing Courses Outline					
Dev Courses Type					

https://dialogflow.cloud.google.com/cx/

- Provides advanced features with state machine-based flows.
- Designed for complex, large-scale conversational applications.
- Supports a visual flow builder for intuitive design.
- Allows for better scalability and more granular control over conversation paths.
- Ideal for enterprises needing sophisticated conversational flows.

Chatbots and Ai Agents



A **chatbot** is a software application that simulates human conversation through text or voice interactions. It typically follows predefined scripts or simple decision trees to respond to user inputs.

An **AI agent**, on the other hand, is a more advanced version of a chatbot that leverages machine learning, natural language processing, and artificial intelligence to understand user intent, maintain context, and provide more intelligent, context-aware responses.

How to Start with Dialogflow Tool

- Sign up for Google Cloud Platform (GCP):
 - Visit the Google Cloud Console and create an account.
- Enable Dialogflow API:
 - Navigate to the API & Services section and enable the Dialogflow API.

• Access Dialogflow Console:

• Go to Dialogflow Console and create a new agent.

• Create an Agent:

- Define a project, set the default language, and choose a time zone.
- Define Intents:
 - Create intents to handle different user queries.

• Train the Agent:

- Add training phrases and responses to fine-tune the chatbot's understanding.
- Integrate with Platforms:
 - Use built-in integrations to deploy the agent on platforms like Google Assistant,
 Facebook Messenger, and more.

• Testing and Optimization:

 Use the Dialogflow simulator to test responses and refine the agent based on user interactions.

What are Intents with Example

Intents in Dialogflow represent the mapping between what a user says and how the system should respond. They are used to categorize user requests and provide appropriate responses. Each intent contains training phrases, responses, and optional actions.

Example: Pizza Order Intent Intent Name: OrderPizza



Training Phrases:

- I want to order a pizza
- Can I get a large pepperoni pizza?
- I'd like to order two medium-sized pizzas

Response:

- Sure! You choosen large one pepporoni pizza?
- Great choice! Would you like to add any drinks?

By setting up an intent like this, Dialogflow can handle different ways users request a pizza order and respond accordingly.

Essentials Global - Entities	CREATE ENTITY	Try it now
Fast_Stacks_Program - ** Custom System		Agent
	-	USER SAYS COPY CL
C Intents + Search entities	Q	ds sdflkdsj f
Kninkes		DEFAULT RESPONSE I missed that, say that again?
Kinderger Prizza, Size		
C2 integrations		CONTEXTS RESET CONTED
🕞 Training		INTENT Default Fallback Intent
Validation		
C History		input.unknown
a Anaytos		DIAGNOSTIC INFO
Prebuilt Agents		
💭 Small Talk		
> Docs (3*		
Trial Upgrade		
Dialogflow CX ^[see]		
Dialogflow alcoal + Entity name	SAVE	Try it now
Dialogflow Global * Entity name Fast_Stacks_Program * * * Ø Define synonyms @ Regerp entity @ Allow automated expansion Fuzzy matching @	SAVE	Try it now 🦆
Dialogflow Global + Essentials Entity name Fast_Stacks_Program * Image: Stacks_Program * <td>SAVE</td> <td>Try it now 4</td>	SAVE	Try it now 4
Dialogflow Global + Entity name Fast_Stacks_Program * Control + Contro	SAVE [‡]	Try It now Agent User SAVS ds sdfikdsj f
Dialog/flow alcoal Entity name East_Stacks_Program ** imtents trates	swe i	Try it now Agent USER SMS ds softikisj f COPY Cut
Dialog/fow alcoal + Entity name East_Stacks_Program + C Interes + C Interes + C Knowledge limit Knowledge limit	I	Try It now Agent USER SAYS COPY CM dis adfikidij f corxuct response Imissed that, say that again?
	*	Try it now Agent USER SAYS COPY CU USER SAYS copy cu ds adfikdaj f copy cu cu copy cu c
	X	Try It now Agent USER SAYS ds adfikdaj f convocut response response response convocut response convocut response respon
	x	Try It now Agent URER SAVS ds softikdij f COPY CUI CUTER SAVS COPY CUI CUTER SAVS COPY CUI CUTER SAVS CONTROL CUTER SAVS CONTR
Dialogffor aloal * Entity name Fast_Stacks_Program		Try It now Agent USER SAVS ds adfikdsj f COPY Cut ds adfikdsj f corruct response rmissed that, say that again? Contrextrs Contrextrs Contrextrs Perfault Fallback Intent Action
Dialogffor global * Entity name Fast_Stacks_Program *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ******	X	Try It now Agent USER SAVS ds softwids f COPY Cut USER SAVS ds softwids f COPY Cut System_counters_ RESET Control NTENT Default Failback Intent ACTION Input Inknown
Dialogflow actal * Entity name Fast_Stacks_Program	X	Try It now Agent USER SAVS ds softwals ds softwals f COMPOSE I missed that, say that again? COMPOSTS COMPOSTS RESET COMPOSESystem_counters NTENT Default Tablock Intent ACTON Input unknown DIAGNOSTIC INFO
Pialogffor cloal * Entity name Furt.Stacks_Program	_ <u>\$</u>	Try It now Agent USER BANS COPY OU Software Sof
Calcogffor accel* Entity name Fat.Stacks_Program	x	Try It now Agent USER BANS ds softwals ds softwals f COMPECTS CONTENTS CON

★ Create Entities:

- Go to the Dialogflow console and navigate to the "Entities" section.
- Create an entity named pizza_name with values like "pepperoni," "margherita,"
 "veggie," etc.
- Create another entity named pizza_size with values like "small," "medium,"
 "large."

Dialogflow Essentials Global -	CREATE INTENT :	Try it now
Fast_Stacks_Program - 🌾	Search intents Q T	Agent
💬 intents 🛛 🕂	Complain_Reistration	USER SAYS COPY CUR ds sdfikdsj f
L Fotities	Default Fallback Intent	
	Default Welcome Intent	DEFAULT RESPONSE I missed that say that again?
Knowledge (vera)	Order Pizza	
4 Fulfillment		CONTEXTS RESET CONTEXT
C2 Integrations		system_counters
Training		DITENT
Validation		Default Failback Intent
		ACTION
History		input.unknown
In Analytics		DIACNOSTIC INFO
Prebuilt Agents		Divento she har o
🗇 Small Talk		
> Docs I		
Trial Upgrade		
Dialogflow CX [new]		
Dialogflow Essentials Global -	Pizza_Name	SAVE
Fast_Stacks_Program -	*	
en	Define synonyms 🛛 📋 Regexp entity 🖓 📋 Allow automated expansion 🔄 Fuzzy matching 🚱	
	Faiita Faiita, fi, honey	
💬 Intents	+ Halian Halian	

	- 19	
	Fajita	Fajita, fj. honey
intents +	Italian	Italian
🖁 Entities 🕂 🕂	Chinease	Chinease
Knowledge [beta]		Click here to edit entry
4 Fulfillment	+ Add a row	
Integrations		
🗇 Training		
Validation		

Dialogflow Global -	Pizza_Size			
Fast_Stacks_Program - 🔆	Cefine synonyms 🛛 🗌 Regexp entity 🖉	Z Define synonyms 🛛 🗌 Regexp entity 🛛 🗌 Allow automated expansion 📄 Fuzzy matching 🚱		
	Medium	Medium, md		
📮 Intents 🗕	Large	Large, Ig		
🔒 Entities 🛛 🕂	Small	Small, sm		
Knowledge [beta]		Click here to edit entry		
4 Fulfillment	+ Add a row			
Integrations				

★ Define an Order Pizza Intent:

- Create an intent named OrderPizza.
- Add training phrases such as:

- "I want a medium pepperoni pizza."
- "Can I get a large margherita?"
- Use the pizza_name and pizza_size entities in the training phrases.

Dialogflow Essentials Global -	Order Pizza				SAVE	Try it now	٩
Fast_Stacks_Program - 🔆	Contexts 🕢				^	Agent	
C Intents +	Add input context					USER SAYS ds sdfikdsj f	COPY CURL
Entities +	2 OrderPizza-followup 🛞 Add output o	context			×	🟓 DEFAULT RESPONSE	•
Knowledge [Deta]						I missed that, say that again?	
🖌 Fulfillment	Events 🔞				~	CONTEXTS	RESET CONTEXTS
C2 Integrations					· · · · ·	system_counters	
😚 Training	Training phrases 🔞			Search training phra:	ses Q A	INTENT	
Validation	Template phrases are deprecated and	will be ignored in training time. M	ore details here.			Default Fallback Intent	
History	When a user says something similar to a trainin	g phrase, Dialogflow matches it to	the intent. You don't have to create	an exhaustive list. Dialogflow will fill o	out the list with similar	ACTION input.unknown	
in Analytics	expressions. To extract parameter values, use a	nnotations with available system (or custom entity types.				
Prebuilt Agents	99 Add user expression					DIAGNOSTIC I	NFO
🗊 Small Talk	99 order 3 italian pizza						
> Docs 12*	75 Please order one fajita large						
Trial Upgrade	99 I want to order a pizza						
Dialogflow Global -	Order Pizza				SAVE	Try It now	Ŷ
						Agent	
🕞 intents 🛛 🕂	Action and parameters				^	USER SAVS ds sdfikdsj f	COPY CURL
🛱 Entities +	Enter action name					DEFAULT RESPONSE	
Knowledge [beta]	REQUIRED O PARAMETER NAME O	ENTITY O	VALUE	IS UST PROMPTS O		I missed that, say that again?	
4 Fulfillment	pizza_name	@Pizza_Name	Spizza_name	Please tell the.		CONTEXTS	RESET CONTEXTS
C2 Integrations	pizza_size	@Pizza_Size	Spizza_size	Please tell the.		system_counters	
💮 Training	quantity	@sys.number	Squantity	please tell the.		INTENT	
Validation	Enter name	Enter entity	Enter value			Default Fallback Intent	
History	+ New parameter					ACTION	
Analytics						mpacanentown	
Prebuilt Agents	Responses 🔞				^	DIAGNOSTIC I	
Comail Talk	DETRUCT 1						NFO
Lar ornali taik	DEFAULT +						NFO
> Docs @	Text Response				٥		NFO
Svraat talk Does@ Trial Upgrade	Text Response 1 You selected Squantity Spizza_size S	pizza_name			0		NFO

★ Create the order_pizza Entity:

- Define a composite entity called order_pizza.
- Include pizza_name and pizza_size as attributes.

★ Configure Fulfillment:

 Enable webhook fulfillment to connect the chatbot with a backend system for processing orders.

★ Test the Agent:

• Use the simulator in Dialogflow to test user inputs and verify correct responses.

Contexts

Contexts are an essential feature in Dialogflow that help manage the flow of conversation by maintaining relevant information across multiple interactions. They allow the chatbot to understand and remember what the user has said earlier and use it to provide meaningful responses.

There are two types of contexts in Dialogflow:

Input Contexts

Input contexts are used to control which intents should be triggered based on the current conversation context. They help the agent identify relevant intents when multiple intents could match a user's query.

Example: If a user says "I want to order a pizza," an input context named order-pizza can be set. Subsequent intents, such as choosing a pizza type or size, will only be triggered if the input context is active.

Output Contexts

Output contexts are used to maintain state and pass information from one intent to another. They help in carrying forward user-provided data through the conversation.

Example: After the user specifies the pizza type, an output context can be set to retain that information for the next intent, such as selecting the pizza size.

Adding Input and Output Contexts in Dialogflow

To add contexts in Dialogflow:

1. Go to the Dialogflow console and open the desired intent.

- 2. In the "Contexts" section, add an **input context** to restrict the intent to a specific conversational state.
- 3. Add an **output context** to maintain state across interactions.
- 4. Test the agent to ensure the context is being set and carried correctly.

Using contexts effectively helps create smooth and personalized conversations that feel natural and efficient to the user.

Chapter 2: Fulfillment and Webhooks with FastAPI and ngrok

What is Fulfillment in Dialogflow ES?

Dialogflow Global -	4 Fulfillment				Try it now
Fast_Stacks_Program 👻 🔆	Webhook			ENABLED	O Please use test o sentence.
Intents +	all the webhook requirem	eive a POST request from Dialogflow in the form of the re- nents specific to the API version enabled in this agent.	ponse to a user query matched by intents with webhook enabled. Be sure that y	our web service meets	
🖁 Entities +	URL*	https://e010-182-189-28-213.ngrok-free.app			
Knowledge [beta]	BASIC AUTH	Enter username	Enter password		
4 Fulfillment	LIEADEDO	Enter los	Entervalue		
	HEADERS		Einer Value		
😚 Training	SMALL TALK	Disable webhook for Smalltalk		*	
Validation	Inline Editor (
History		Powered by Google Cloud Functions)		DISABLED	
in Analytics	Newly created	cloud functions now use Node is 10 as runtime engine. C	heck migration guide for more details.		
Prebuilt Agents					
🗊 Small Talk	index.js package.j:				
> Docs 🗗	1 // See https: 2 // for Dialog	//github.com/dialogflow/dialogflow-fulfillment flow fulfillment library docs, samples, and to			
Trial Upgrade	3 'use strict'; 4 5 const functio	<pre>ns = require('firebase-functions');</pre>			
Dialogflow CX [new]	6 const {Webhoo 7 const {Card, 8	<pre>kClient} = require('dialogflow-fulfillment'); Suggestion} = require('dialogflow-fulfillment'); reput</pre>			

Fulfillment is the mechanism that allows your Dialogflow agent to interact with external services. It lets you perform dynamic actions such as querying a database, calling APIs, or responding with personalized content.

Key Components of Fulfillment:

- **Webhook**: An HTTP endpoint that processes requests sent by Dialogflow.
- **Request and Response Structure**: JSON format for communication between Dialogflow and your webhook.
- **Enablement**: Fulfillment must be enabled on the Dialogflow Console at the intent level.

Building a Webhook with FastAPI

FastAPI is a high-performance, easy-to-use Python framework ideal for building webhooks for Dialogflow.

Fast API Doc: https://fastapi.tiangolo.com/

Uvicorn: https://www.uvicorn.org/

Step 1: Setting Up FastAPI

Install FastAPI and Uvicorn:

pip install fastapi uvicorn

Create a Basic FastAPI Application:

```
from fastapi import FastAPI
app = FastAPI()
@app.get("/")
def read_root():
   return {"message": "Webhook is running!"}
```

Run the Application:

uvicorn main:app --reload

Step 2: Implementing Dialogflow Webhook Logic



1. Define the Webhook Endpoint:

Easy Starter Way

```
from fastapi import FastAPI, Request
app = FastAPI()
@app.post("/webhook")
async def webhook(request: Request):
    req_data = await request.json()
```

```
print("Request received:", req_data)
# Construct a response
fulfillment_response = {
    "fulfillmentText": "This is a response from FastAPI"
}
return fulfillment_response
```

Normal Way

```
from fastapi import FastAPI, Request
app = FastAPI()
@app.post("/webhook")
async def webhook(request: Request):
    req_data = await request.json()
    print("Request received:", req_data)

# Construct a response
fulfillment_response = {
    "fulfillmentMessages": [
        {"text": {"text": ["This is a response from FastAPI!"]}}
    ]
    }
    return fulfillment_response
```

2. **Test the Endpoint Locally**: Use tools like Postman or curl to send a test request and ensure the endpoint behaves as expected.



Exposing Localhost with ngrok

For Dialogflow to communicate with your webhook, it needs a publicly accessible URL. Ngrok creates secure tunnels to your localhost, making it accessible over the internet.

Step 1: Install ngrok

- 1. **Download and Install**: Visit <u>ngrok's website</u> to download and install the tool for your OS.
- 2. Authenticate:
- 3. ngrok authtoken YOUR_AUTH_TOKEN

PROBLEMS OUTPUT DEBUG CO	ONSOLE	TERMINAL	PORTS	сомм	ENTS				+~ ^ ×
o ngrok ∳ Route traffic by anythin							(נהאל	to quit)	と uvicorn と ngrok []) 自
Session Status Account Version Region Latency Web Interface Forwarding	online zeesha 3.19.6 Europe 874ms http:/ https:	e anali9023 e (eu) //127.0.0 ://5ce3-1	3@gmail.).1:4040 .82-189-2	com (Pla 8-213.ng	in: Free) trok-free	.app ->	http://localhost:8000		
Connections	ttl 0	opn Ø	rt1 0.00	rt5 0.00	p50 0.00	p90 0.00			

Step 2: Run ngrok

- 1. Expose Localhost:
- 2. ngrok http 8000
- 3. **Copy the Public URL**: Ngrok will display a URL like https://<randomstring>.ngrok.io. Use this in the Dialogflow Console.

Connecting the Webhook to Dialogflow

1. Enable Fulfillment:

- o Go to the intent where you want to use fulfillment.
- Enable the "Use webhook" checkbox.

2. Set the Webhook URL:

- Navigate to the Dialogflow Console -> Fulfillment section.
- Add the ngrok public URL (https://<random-string>.ngrok.io/webhook).

3. Test Your Integration:

• Use the Dialogflow Simulator to trigger the intent and verify that your webhook is responding correctly.

Advanced Use Cases with FastAPI Webhooks

1. Custom Payloads for Rich Responses:

2. Accessing User Parameters:

Extract and use parameters provided by the user:

```
user_params = req_data["queryResult"]["parameters"]
print("User Parameters:", user_params)
```

3. Error Handling:

Add robust error handling to ensure reliable responses:

try: # Processing logic here return fulfillment_response except Exception as e: return {"fulfillmentText": "Something went wrong!"}

Chapter 3: Dialogflow CX Setup and Pizza Chatbot

≡ 🛊 Dialogflow CX Test Project	roject 👻					0 8	1
Agents							
	Location			Use prebuilt agents	Create agent		
	Q Search Search agents by ID or disp	lay name					
	Display Name	Default language	Region 个	Using CMEK			
	Weather_Agent	en	asia-south1 (Mumbai, India)	No	:		
	test	en	us-central1 (Iowa, USA)	No	:		
Quick access to useful resources							
Documentation D	Tutorial Video	•					
Empower human agents with continuous support during live calls.	Create conversational experiences acros devices and platforms.	55					
		Term	ns of Service Privacy				

Introduction to Dialogflow CX

Dialogflow CX (Customer Experience) is designed for **complex, multi-turn conversations** with advanced state management. Unlike ES, CX uses a **visual flow builder** to map conversational journeys, making it ideal for enterprise-grade chatbots. Key components include:

- **Agents**: The core chatbot instance.
- Flows: Modular sections of a conversation (e.g., "Order Pizza" flow).
- **Pages**: Steps within a flow (e.g., "Choose Size," "Select Toppings").
- **Transitions**: Rules to navigate between pages.
- **Parameters**: Variables to store user inputs (e.g., pizza_size).



Setting Up a Dialogflow CX Agent

- 1. Create a CX Agent:
 - o Go to Dialogflow CX Console.
 - Click **Create Agent** \rightarrow Name it "PizzaBotCX" \rightarrow Set language/time zone.
 - Enable **Advanced Settings** for Google Cloud logging if needed.
- 2. **Design the Flow:**
 - Click **Build a Flow** \rightarrow Name it "OrderPizzaFlow".
 - Start with the **Default Start Page** to handle initial user requests.

Building the Pizza Order Flow

= 😝 Dialogflow CX	Test Project - We	ather_Agent • en •		(z) q 🛛 📌 🚳
Build Manage	Intents + Create 1 Import			<> Publish 🛞 Ag	ent settings 🛛 🖳 Test Agent
RESOURCES	Intents Suggestions Over	aps			
Intents					
♣ Entity types ♣ Webhooks	Q Search Search intents by ID, display	name, or label			
۲ ^{,4} Route groups	Display name	Labels	# of Training phrases	Last modified	
E Data stores 🖉 [new]	Default Welcome Intent		17	Jan 24, 2025 08:44 PM	Ū
♣ Generators	Default Negative Intent		0	Jan 24, 2025 08:44 PM	Ū
TEST & FEEDBACK	weather.cityname		13	Jan 24, 2025 09:49 PM	Ū
Test cases					
Validations					
Change history					
Conversation history					
Analytics [new]					
FESTING & DEPLOYMENT					
≓ Git					
Versions					
Environments					

Step 1: Define Intents and Entities

- 1. Create Intents:
 - StartOrderIntent:
 - Training Phrases: "I want to order a pizza," "Start pizza order."
 - SizeSelectionIntent:
 - Training Phrases: "Large," "Medium size," "I'll take a small."

• ToppingsIntent:

Training Phrases: "Add pepperoni," "Mushrooms and olives."

2. Create Entities:

- **pizza_size**: Values: small, medium, large.
- **toppings**: Values: pepperoni, mushrooms, olives.

= 🗳 Di	ialogflow CX	Project Veather_Agent Weather_Agent	Canguage en	2 < 0 P
Build	Manage	Default Start Flow @ @ 6 🔊 🖨 Minin	map	🗘 Publish 🔞 Agent settings 🖳 Test Age
FLOWS	+ 🖽			
🙏 Default Star	rt Flow			
		Start Page		
		CityName		
		Спуталь		
	+ 🖽			
O Start Page				
CityName				

Step 2: Configure Pages and Transitions

- 1. Start Page:
 - Entry Route: Triggered by StartOrderIntent.
 - Set Parameter: pizza_type (if needed).
 - **Transition**: Route to **SizePage**.
- 2. SizePage:
 - **Parameter**: pizza_size (required).
 - Entry Route: Use SizeSelectionIntent to capture size.
 - **Condition**: \$session.params.pizza_size != null.
 - **Transition**: Route to **ToppingsPage**.
- 3. ToppingsPage:
 - **Parameter**: toppings (list).
 - Entry Route: Use ToppingsIntent.
 - **Transition**: Route to **ConfirmationPage**.

Step 3: Add Conditional Responses

In **ConfirmationPage**, use a **conditional response**:

Fulfillment in CX with FastAPI

Dialogflow CX uses a **different JSON structure** for webhook requests compared to ES. Adapt the FastAPI webhook from Chapter 2:

Sample CX Webhook Request:

```
{
    "session":
    "projects/PROJECT_ID/locations/LOCATION/agents/AGENT_ID/sessions/SES
SION_ID",
    "pageInfo": {
        "currentPage":
        "projects/PROJECT_ID/locations/LOCATION/agents/AGENT_ID/flows/FLOW_
ID/pages/PAGE_ID"
        ,
        "sessionInfo": {
            "parameters": {
            "pizza_size": "large",
            "toppings": ["pepperoni"]
        }
        }
    }
}
```

FastAPI Endpoint:

```
from fastapi import FastAPI, Request
app = FastAPI()
@ app post("/cx-webhook")
async def cx_webhook(request: Request):
    data = await request json()
    params = data get("sessionInfo", {}).get("parameters", {})
    size = params get("pizza_size", "medium")
    toppings = params get("toppings", [])
    response_text = f"Order confirmed: {size} pizza with {toppings if toppings
    else 'no toppings'}."
    return {
```

```
"sessionInfo": {
    "parameters": params # Preserve parameters for future steps
    "fulfillmentResponse": {
```



Configure Fulfillment in CX:

- 1. In the **ConfirmationPage**, enable **Webhook** under *Entry Fulfillment*.
- 2. Set the endpoint to your ngrok URL (e.g., https://abcd1234.ngrok.io/cx-webhook).

Testing the Flow

- 1. Use the **CX Simulator** to test:
 - o User: "I want to order a pizza."
 - o Bot: "What size would you like?"
 - o User: "Large with pepperoni."
 - o Bot: "Order confirmed: large pizza with pepperoni."

Advanced CX Features

- 1. Slot Filling:
 - Mark parameters as **required** in a page. The bot will auto-prompt users for missing data.
- 2. Event Handlers:
 - Trigger intents on specific events (e.g., timeout, retry).

3. Parameter Persistence:

o Use \$session.params.param_name to retain data across flows.

Migrating from ES to CX

- Flows ≈ Contexts: CX flows replace ES's input/output contexts with visual workflows.
- **Parameters** ≈ **Entities**: Store user inputs in parameters instead of composite entities.

• **State Management**: CX automatically preserves session parameters, reducing manual context handling.

Conclusion

Dialogflow CX empowers developers to build **non-linear**, **scalable chatbots** with intuitive visual tools. By structuring conversations into flows and pages, you can handle complex scenarios like pizza orders effortlessly.

Chapter 4: Building a Weather Chatbot and FastAPI

Introduction to the Weather Chatbot

In this chapter, you'll build a **weather chatbot** using **Dialogflow CX** and **FastAPI**. The chatbot will:

- 1. Capture a **city name** and **country name** from the user.
- 2. Call an external weather API (e.g., <u>WeatherAPI</u>) to fetch real-time weather data.
- 3. Return a formatted response with temperature, humidity, wind speed, and more.

Step 1: Set Up the Dialogflow CX Agent

- 1. Create a New Agent:
 - o Go to Dialogflow CX Console.
 - Name the agent **WeatherBotCX** and set the default language.

2. **Define Entities**:

- **city**: System entity type @sys.geo-city (prebuilt).
- **country**: System entity type @sys.geo-country (prebuilt).

Step 2: Create the Weather Intent

- 1. Intent Name: GetWeather
- 2. Training Phrases:
 - o "What's the weather in Lahore, Pakistan?"
 - o "Tell me the weather forecast for Tokyo, Japan."
 - o "How's the weather in Paris?"

- 3. Parameters:
 - **cityname**: Map to @sys.geo-city.
 - countryname: Map to @sys.geo-country (optional but recommended for accuracy).

Step 3: Design the Flow

- 1. Start Page:
 - Add an **entry route** triggered by GetWeather intent.

```
Set parameters:
```

```
"parameters": {
"cityname": "$session.params.cityname",
"countryname": "$session.params.countryname"
```

Transition to **WeatherPage**.

WeatherPage:

• Enable **webhook fulfillment** to call your FastAPI endpoint.

Step 4: Implement the FastAPI Webhook

Use the provided code to create a webhook that processes city/country inputs and fetches weather data.

Code Explanation

```
from fastapi import FastAPI, Request
import requests
app = FastAPI()
@app post("/webhook/weather")
async def webhook(request: Request):
    try:
        body = await request json()
```

```
# Extract city and country from Dialogflow parameters
    city_name = body.get("intentInfo", {}).get("parameters",
{}).get("cityname", {}).get("resolvedValue")
    country name = body.get("intentInfo", {}).get("parameters",
{}).get("countryname", {}).get("resolvedValue")
    response text =
    try:
       # Call the weather API (replace with your API endpoint)
       url
f"https://p2pclouds.up.railway.app/v1/learn/weather?city={city_name}"
       response = requests.get(url)
       data = response.json()
       # Extract weather details
       temp c = data.get("current", {}).get("temp c")
      feelslike_c = data.get("current", {}).get("feelslike_c")
wind_kph = data.get("current", {}).get("wind_kph")
       humidity = data.get("current", {}).get("humidity")
api_city = data.get("location", {}).get("name")
       api_country = data.get("location", {}).get("country")
       # Format the response
       response_text = f"""
       Weather in {api_city}, {api_country}**:
       - Temperature: {temp c}°C (Feels like {feelslike c}°C)
       - Wind Speed: {wind_kph} km/h
       - Humidity: {humidity}%
       .....
    except Exception as e:
       response_text = f"  Failed to fetch weather for {city_name}. Please
trv again later."
    # Return the response to Dialogflow
    return {
       "fulfillmentResponse": {
         "messages": [{"text": [response text]}]
  except Exception as e:
    return {
       "fulfillmentResponse": {
         "messages": [{"text": {"text": ["X An error occurred. Please try
again!"]}}]
```

Step 5: Configure Fulfillment in Dialogflow CX

1. Enable Webhook:

- In the **WeatherPage**, go to *Entry Fulfillment* \rightarrow Enable **Webhook**.
- Set the URL to your FastAPI endpoint (e.g., https://your-ngrokurl.ngrok.io/webhook/weather).

2. Deploy with ngrok:

- Run ngrok http 8000 to expose your local FastAPI server.
- Update the webhook URL in Dialogflow CX with the ngrok HTTPS URL.

Step 6: Test the Chatbot

- 1. Use the **Dialogflow CX Simulator**:
 - **User**: "What's the weather in London, UK?"
 - Bot: " Weather in London, UK: Temperature: 18°C (Feels like 17°C), Wind Speed: 15 km/h, Humidity: 65%"

2. Edge Cases:

o Test invalid cities: "What's the weather in XYZ?"→ Bot returns an error message.

Common Issues and Fixes

1. API Rate Limits:

o Use a paid API key (e.g., WeatherAPI Premium) for higher request limits.

2. Parameter Extraction Failures:

o Add validation logic to handle missing city/country names:

if not city_name:

response_text = " Please specify a city name!"

Conclusion

You've now built a fully functional weather chatbot using **Dialogflow CX** and **FastAPI**. This chatbot can be integrated with platforms like WhatsApp, Google Assistant, or your website to provide real-time weather updates. In the next chapter, you'll learn how to **deploy the chatbot at scale** and add analytics for performance monitoring.