# CHATBOT DEPLOYMENT WITH IBM CLOUD WATSON ASSISTANT

DEVELOPMENT PART 2

## SET UP DEVELOPMENT ENVIRONMENTS:

* + **CREATE DEVELOPER ACCOUNTS:** IF YOU DON'T ALREADY HAVE DEVELOPER ACCOUNTS, SIGN UP FOR FACEBOOK FOR DEVELOPERS AND SLACK DEVELOPER ACCOUNTS.
  + **CREATE APPS/BOTS:** CREATE NEW APPS OR BOTS ON THE RESPECTIVE PLATFORMS. FOR FACEBOOK, CREATE A FACEBOOK APP, AND FOR SLACK, CREATE A SLACK APP. THIS STEP WILL PROVIDE YOU WITH THE NECESSARY API KEYS AND TOKENS.
  + **INSTALL DEVELOPMENT TOOLS:** DEPENDING ON YOUR PROGRAMMING LANGUAGE (E.G., NODE.JS, PYTHON), YOU MAY NEED TO INSTALL RELEVANT SDKS AND DEVELOPMENT TOOLS. FOR EXAMPLE, YOU MIGHT USE THE FACEBOOK GRAPH API OR SLACK API LIBRARIES TO INTERACT WITH THESE PLATFORMS.

## Build the Chatbot Integration:

* + Use the documentation and guides provided by Facebook and Slack to set up the integration. This will involve configuring webhooks and endpoints for your chatbot to listen for incoming messages and events

## Handling Incoming Messages:

* + Your chatbot should be able to receive messages from both Facebook Messenger and Slack. This involves setting up webhooks to receive POST requests from these platforms when users send messages.
  + Parse the incoming messages to extract user IDs, message content, and any relevant metadata such as message timestamps or attachments.

**Refine Responses:**

* + Enhance your chatbot's NLP capabilities. You can use NLP libraries like spaCy or NLTK (Python) to understand and respond to user messages accurately.
  + Implement context handling to make conversations flow more naturally. Remembering past interactions and maintaining context is crucial for creating engaging conversations.

## Informative and Accurate Responses:

* + To ensure informative responses, you can connect your chatbot to a knowledge base or a database to fetch real-time data. For instance, if your chatbot provides weather updates, it should connect to a weather API to provide accurate information.
  + Implement error handling and fallback responses for questions the chatbot can't answer to manage user expectations.

## User Authentication (Optional):

* + Depending on your use case, you may need to implement user authentication. This can be useful for providing personalized responses or granting access to specific features. For instance, in a workplace Slack bot, you might want to authenticate users with their corporate credentials.

## Testing:

* + Thoroughly test your chatbot on both Facebook Messenger and Slack. Test various user scenarios to ensure it functions correctly.
  + Create test cases to simulate different user interactions, including user queries, commands, and possible edge cases.

## Deployment:

* + Deploy your chatbot to a server or a cloud platform. Ensure it is accessible from both Facebook Messenger and Slack. You might use platforms like AWS, Heroku, or a custom server.

## Monitoring and Analytics:

* + Implement monitoring and analytics to track user interactions and gather insights into how users are using your chatbot. Tools like Google Analytics or custom tracking solutions can help with this.
  + Analyze the data to understand user behavior and improve your chatbot's performance.

## Compliance and Privacy:

* + Ensure that your chatbot complies with the terms and conditions of both Facebook and Slack. Pay attention to user data privacy and security.

# #PROGRAM

IMPORT OS

FROM FLASK IMPORT FLASK, REQUEST FROM SLACK IMPORT WEBCLIENT

FROM SLACKEVENTSAPI IMPORT SLACKEVENTADAPTER FROM PYMESSENGER IMPORT BOT

APP = FLASK( NAME )

MESSENGER = BOT("YOUR\_FACEBOOK\_TOKEN")

SLACK\_TOKEN = "ABC123DEFABC123DEFABC123DEFABC123DEFXYZ"

SLACK\_CLIENT = WEBCLIENT(SLACK\_TOKEN) SLACK\_EVENTS\_ADAPTER =

SLACKEVENTADAPTER("YOUR\_SLACK\_SIGNING\_SECRET", "/SLACK/EVENTS", APP)

@APP.ROUTE("/WEBHOOK", METHODS=["POST"]) DEF WEBHOOK():

DATA = REQUEST.GET\_JSON() IF DATA["OBJECT"] == "PAGE":

FOR ENTRY IN DATA["ENTRY"]:

FOR MESSAGING\_EVENT IN ENTRY["MESSAGING"]: IF MESSAGING\_EVENT.GET("MESSAGE"):

SENDER\_ID = MESSAGING\_EVENT["SENDER"]["ID"] MESSAGE\_TEXT = MESSAGING\_EVENT["MESSAGE"]["TEXT"]

MESSENGER.SEND\_TEXT\_MESSAGE(SENDER\_ID, "HELLO, YOU SAID: " + MESSAGE\_TEXT)

RETURN "OK", 200

@SLACK\_EVENTS\_ADAPTER.ON("MESSAGE") DEF HANDLE\_MESSAGE(EVENT\_DATA):

MESSAGE = EVENT\_DATA["EVENT"] USER = MESSAGE["USER"] MESSAGE\_TEXT = MESSAGE["TEXT"]

.

SLACK\_CLIENT.CHAT\_POSTMESSAGE(CHANNEL=MESSAGE["CHANNEL"], TEXT="HELLO, YOU SAID: " + MESSAGE\_TEXT)

IF NAME == " MAIN ": APP.RUN(DEBUG=TRUE)