AWS Bedrock (Creating Chatbot)

A **chatbot** is an artificial intelligence (AI) program designed to simulate conversation with users, typically through text or voice interactions. It can be integrated into websites, messaging platforms, apps, or devices to automate communication and provide real-time assistance or information. Chatbots use natural language processing (NLP) to understand user input and generate appropriate responses, and they can range from simple, rule-based systems (following pre-defined scripts) to advanced AI-powered bots that learn from interactions and adapt over time.

Types of Chatbots:

1. **Rule-based Chatbots**: Operate based on predefined rules or decision trees. They respond to specific commands or keywords and are limited in scope.
2. **AI-powered Chatbots**: Use machine learning (ML) and NLP to understand complex queries, engage in natural conversations, and improve responses over time.

Common Uses:

- **Customer Support**: Handling FAQs, troubleshooting, and 24/7 support.
- **E-commerce**: Assisting users with product recommendations, order tracking, and purchases.
- **Healthcare**: Providing health advice, scheduling appointments, and offering mental health support.
- **Finance**: Managing account information, conducting transactions, and financial planning assistance.

Key Technologies:

- **Natural Language Processing (NLP)**: Allows chatbots to understand and process human language.
- **Machine Learning (ML)**: Helps chatbots learn from interactions and enhance their performance.
- **APIs**: Enable chatbots to interact with various systems and services, such as databases, payment gateways, or third-party applications.

Chatbots improve efficiency, reduce response time, and offer scalable solutions for businesses to interact with users across various industries.

To begin with the Lab:

1. In this lab we are going to create a chatbot using Bedrock. So, here we will use Amazon Bedrock Foundational Models, Streamlit, and Langchain.

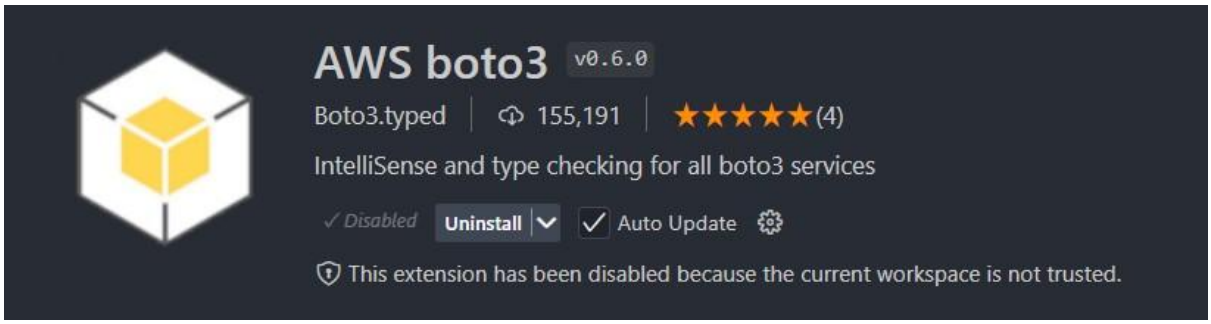
2. Create a new folder
3. Open VSCode and create a virtual environment
   Use command – python -m venv . (this dot represents your current raw directory)

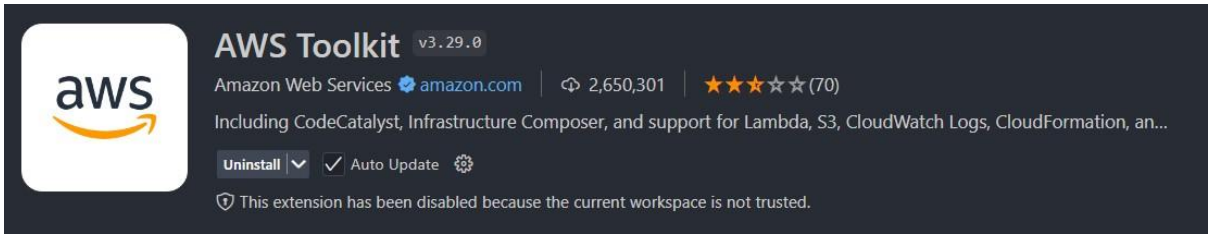
```
PS C:\Users\amanr\Desktop\HR_QA_RAG> python -m venv .
```

4. Now activate the virtual environment
   Use command - Scripts\activate

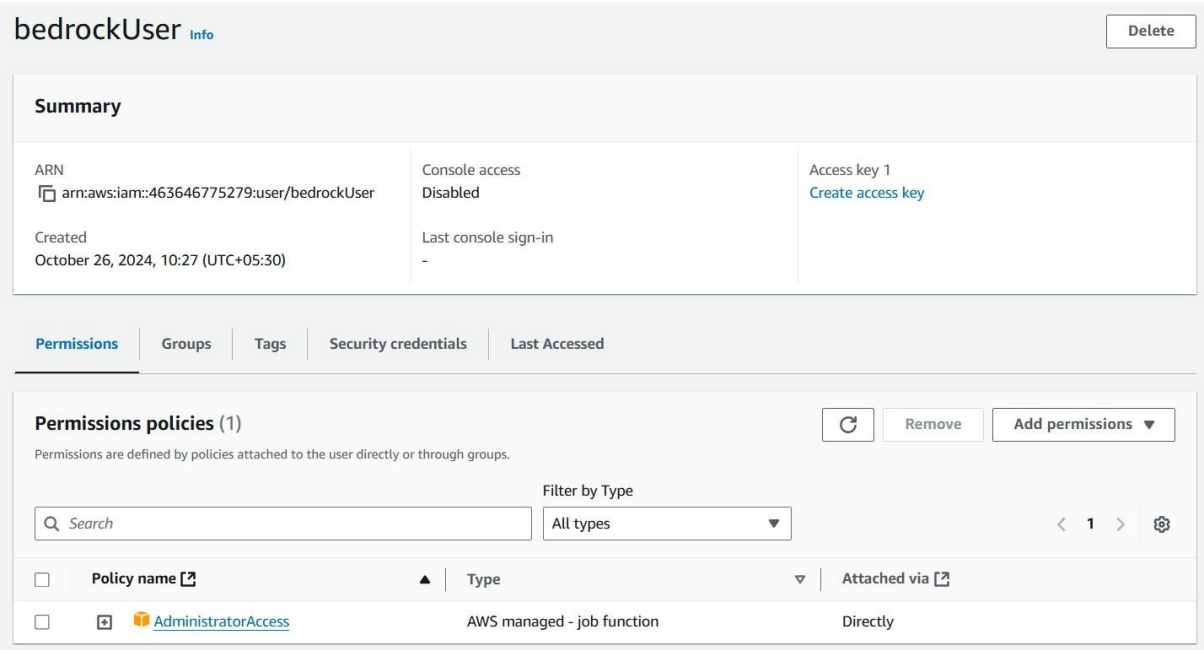
```
PS C:\Users\amanr\Desktop\HR_QA_RAG> Scripts\activate
```

5. After activating the virtual environment, configure the aws account.
   Use command – aws configure
6. Now copy and paste the access Id, key and region.
7. Now for this lab there are some prerequisites that you need to do. First, you should have VS Code on your laptop and in this, you should have these two extensions installed.





8. Next you should have Python on your laptop with that you should also have the AWS CLI installed.
9. Then you need to come to your console, create a User from IAM, and attach Administrator access to that user.
10. After that you need to go to the security credentials of this user and create access and secret access key for this user. Then configure this user in AWS CLI on your laptop.




```
PS D:\AWS Bedrock\3 Bedrock Creating Chatbot\Chatbot> aws configure
AWS Access Key ID [****************4JMH]: AKIAWX44AK7XSJHLZLPA
AWS Secret Access Key [****************UYdX]: BvL+AcBc5Z7kFLEzCG6ANQBaa0xHgISSOC3SHNAJ
Default region name [ap-south-1]: us-east-1
Default output format [json]: json
```

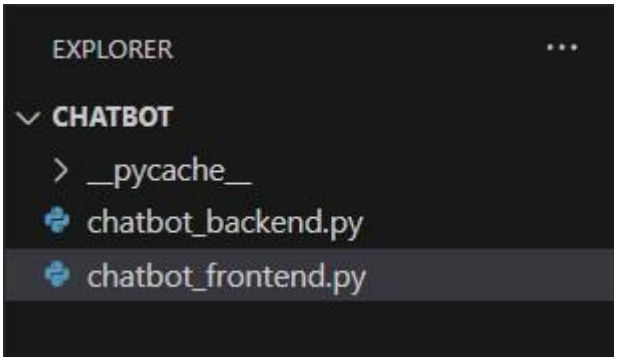
11. In your VS Code you need to open the terminal and install boto3, langchain, and streamlit using the below commands.

    pip install boto3 langchain langchain-aws streamlit transformers

    this command includes libraries like - boto3, langchain, streamlit, langchain-aws, anthropic, transformers

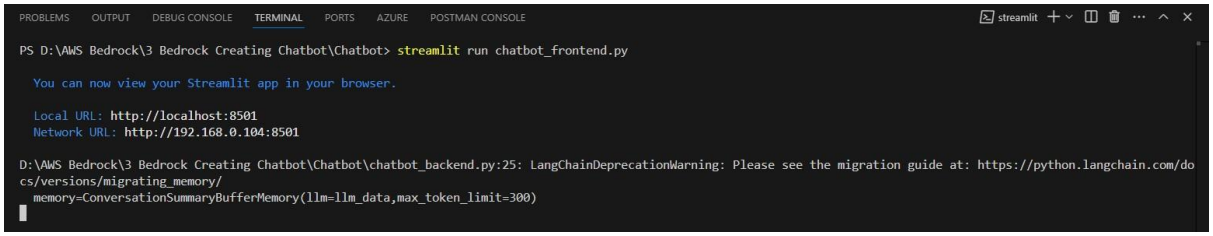
```
PROBLEMS 1  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  AZURE  POSTMAN CONSOLE                    pwsh
PS D:\AWS Bedrock\3 Bedrock Creating Chatbot\Chatbot> pip show boto3
Name: boto3
Version: 1.35.49
Summary: The AWS SDK for Python
Home-page: https://github.com/boto/boto3
Author: Amazon Web Services
Author-email:
License: Apache License 2.0
Location: C:\Users\PULKIT\anaconda3\Lib\site-packages
Requires: botocore, jmespath, s3transfer
Required-by:
PS D:\AWS Bedrock\3 Bedrock Creating Chatbot\Chatbot> pip show langchain
Name: langchain
Version: 0.3.4
Summary: Building applications with LLMs through composability
Home-page: https://github.com/langchain-ai/langchain
Author:
Author-email:
License: MIT
Location: C:\Users\PULKIT\anaconda3\Lib\site-packages
Requires: aiohttp, langchain-core, langchain-text-splitters, langsmith, numpy, pydantic, PyYAML, requests, SQLAlchemy, tenacity
Required-by:
PS D:\AWS Bedrock\3 Bedrock Creating Chatbot\Chatbot> pip show streamlit
Name: streamlit
Version: 1.39.0
Summary: A faster way to build and share data apps
Home-page: https://streamlit.io
Author: Snowflake Inc
Author-email: hello@streamlit.io
License: Apache License 2.0
Location: C:\Users\PULKIT\anaconda3\Lib\site-packages
Requires: altair, blinker, cachetools, click, gitpython, numpy, packaging, pandas, pillow, protobuf, pyarrow, pydeck, requests, rich, tenacity, toml, tornado, typing-ex
tensions, watchdog
Required-by:
PS D:\AWS Bedrock\3 Bedrock Creating Chatbot\Chatbot>
```

12. Once you have set up all the required things then you need to open the code folder that you get with lab in your VS Code. Here you will see two files first is for backend and other one is for frontend.

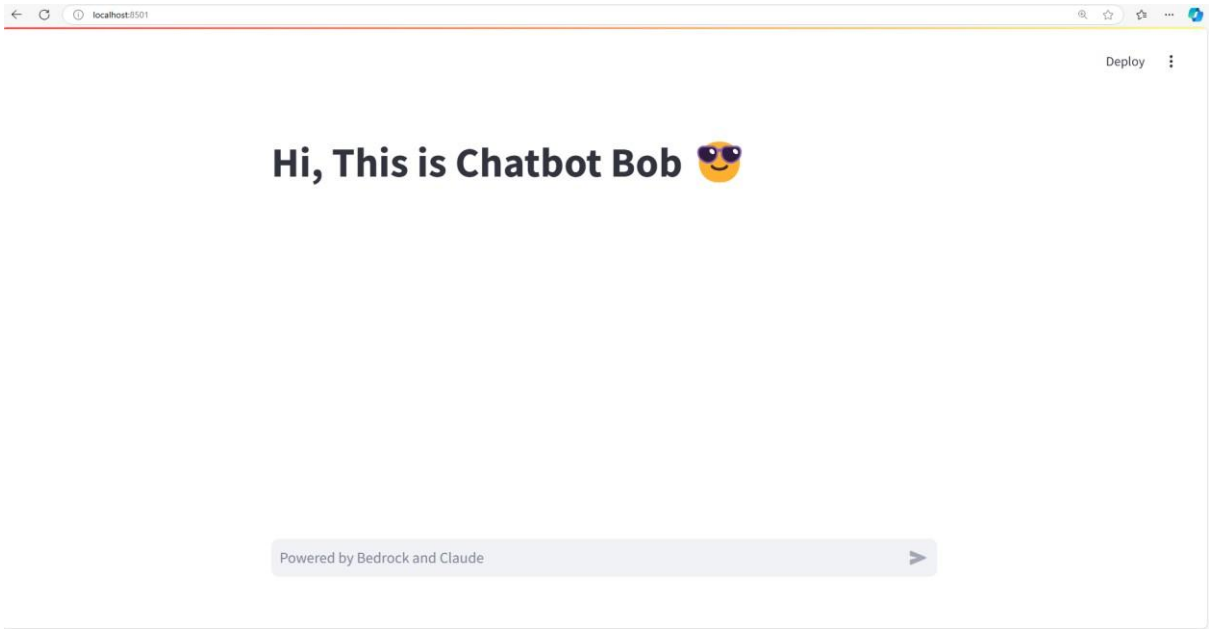


13. In the backend code we integrate **Langchain** with Amazon Bedrock's Anthropic model to manage conversational memory and create dynamic interactions. It defines functions for setting up an LLM client, using **ConversationSummaryBufferMemory** to retain conversation context, and building a **ConversationChain** to handle chat responses. The invoke method is used to generate model responses, with parameters like **max_tokens,** temperature, and **top_p** controlling the output behaviour.
14. In the frontend code we created a **Streamlit-based** chatbot using AWS Bedrock and **LangChain**. It initializes the chatbot with a title, manages memory and chat history using Streamlit's session state, and renders previous conversation history. User input is captured in a chatbox, and passed to the chatbot backend for response via the **demo_conversation** method, and both the user message and chatbot reply are displayed and stored for continuous conversation flow.
15. Now you can read the code and understand it for your knowledge. Now we need to run this chatbot.
16. For that we need to open the terminal and run the command given below. Here you can see that our chatbot is running.

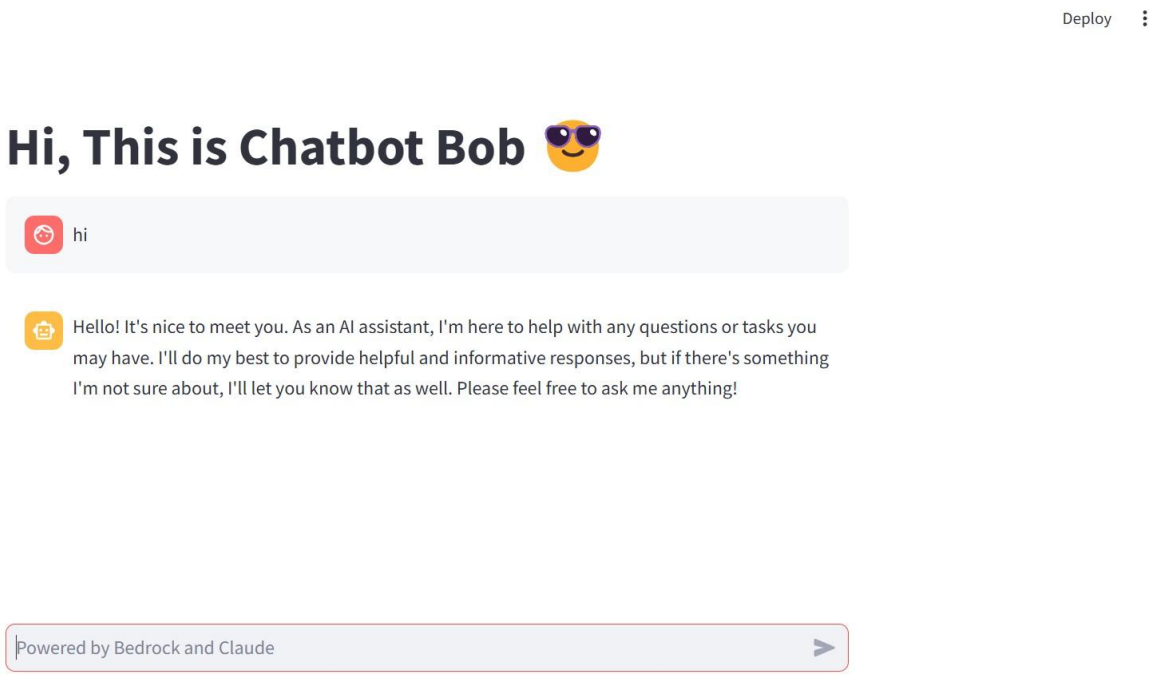
    streamlit run chatbot_frontend.py



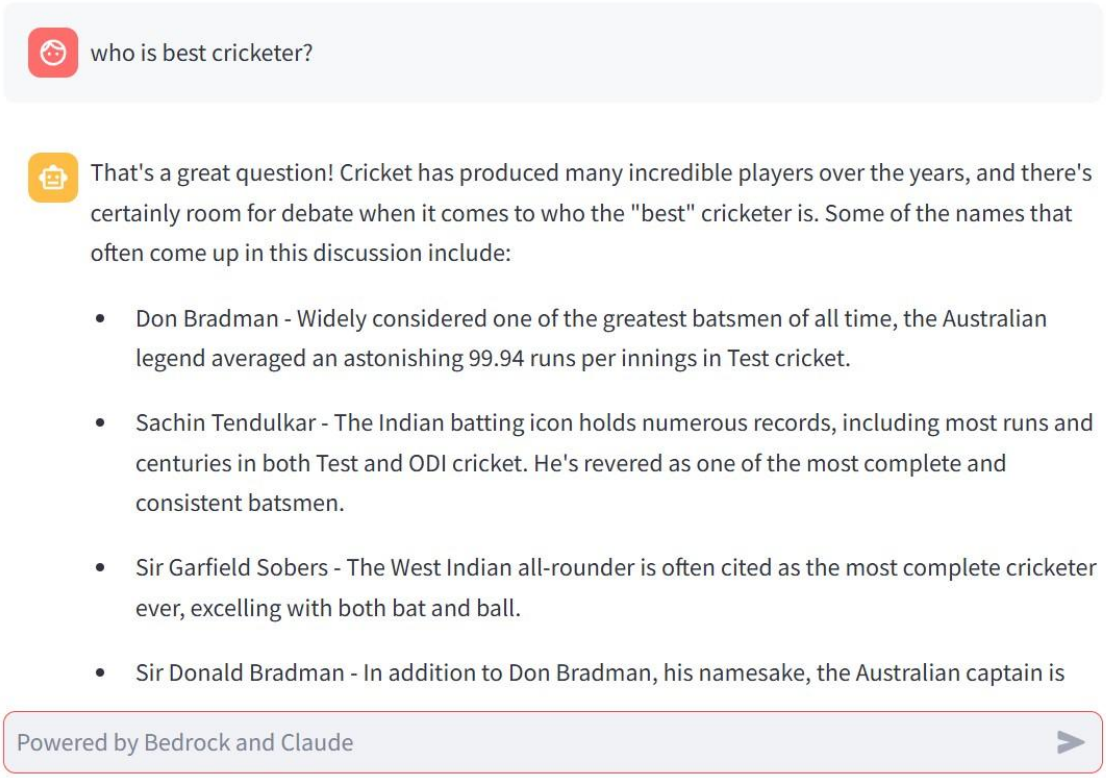
17. You can see that it is running in your browser. Now you need to ask this bot some questions.



18. We just wrote ‘hi’ and we got the response as expected.



19. Below you can see that we asked it a question and it responded with a proper answer. Now following this question, we are going to ask more questions.



20. Below you can see that it followed the previous question and based on that it answered me.
21. So, this is how we can create our own chatbot.

so, who are the best players after them?

That's a great follow-up question! After the all-time greats like Don Bradman, Sachin Tendulkar, Garfield Sobers, and Viv Richards, there are several other cricketers who are widely considered among the best players of more recent generations:

- Ricky Ponting - The former Australian captain is one of the most successful batsmen and leaders in the modern era, with over 13,000 Test runs.
- Jacques Kallis - The South African all-rounder is one of the most complete cricketers ever, excelling with both bat and ball.
- Kumar Sangakkara - The Sri Lankan wicket-keeper batsman is one of the highest run-scorers in ODI and Test cricket history.
- Muttiah Muralitharan - The Sri Lankan spin wizard holds the record for most wickets taken in Test cricket with over 800 scalps.