RAG Chatbot

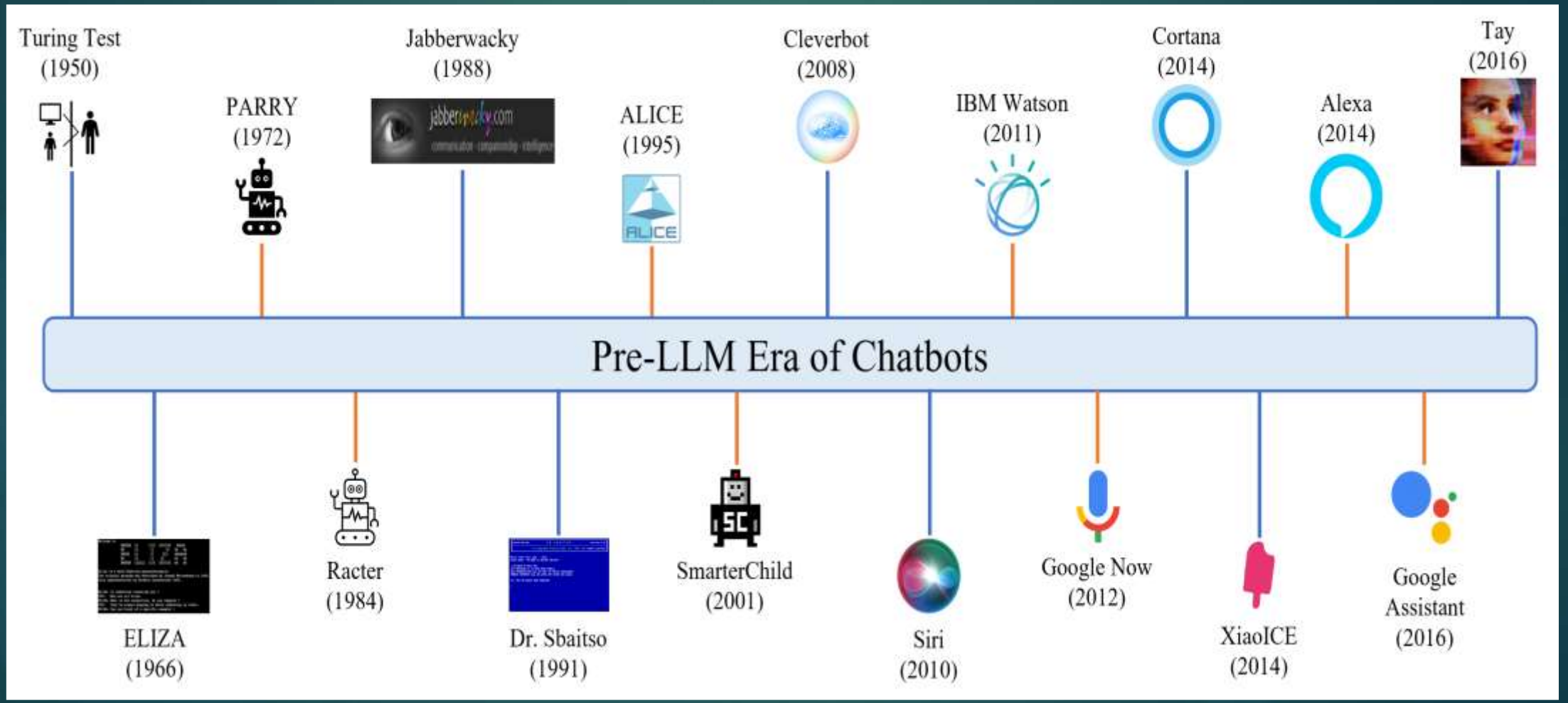
SUBMITTED BY: CHATBOT FOR SABUDH TEAM-A

PROJECT MENTOR: MR. PARTHA SARATHI MUKHERJEE

What is a Chatbot?

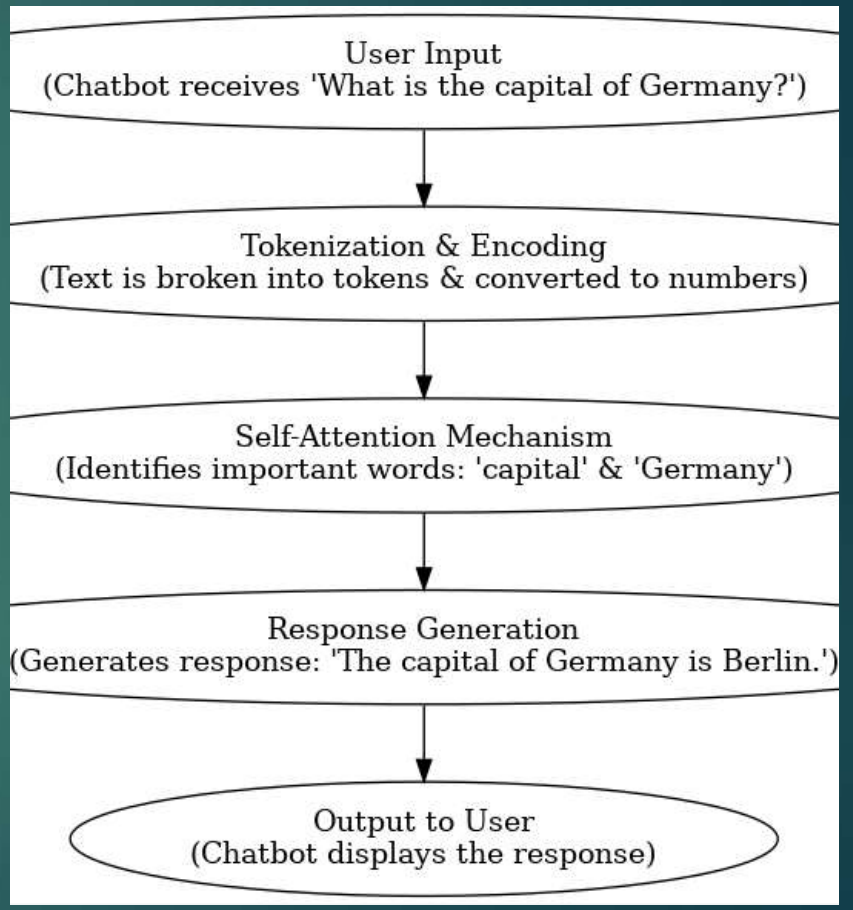
- A chatbot is basically a software application or interface that users can converse with through text or voice.
- It simulates human conversations.
- A business organization typically uses chatbots to scale, personalize, and improve communications for a better user experience.
- Today they are used in several sectors such as finance, defense, healthcare, etc.
- Examples: ChatGPT, DeepSeek, Domino's Chatbot, etc.

Timeline of Pre-LLM Era Chatbots



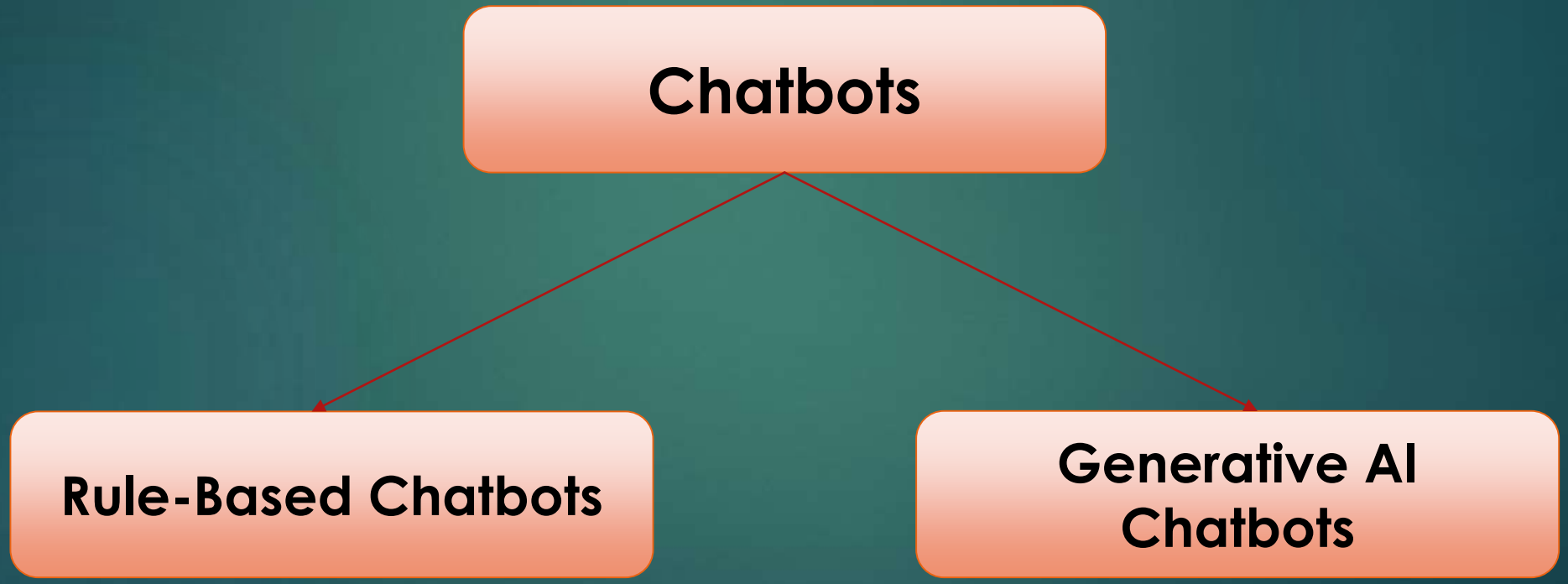
How do Chatbots work?

- **Natural Language Processing (NLP):** It explains how chatbots process human language. It helps the chatbot break down sentences, extract meaning, and generate appropriate responses.
- **Intent Recognition**: It is used to understand what a user wants and entities to extract key information.
- **Response Generation**: Responses can be pre-defined (in rule-based systems) or generated dynamically (in generative AI chatbots). These are the final outputs by a chatbot.



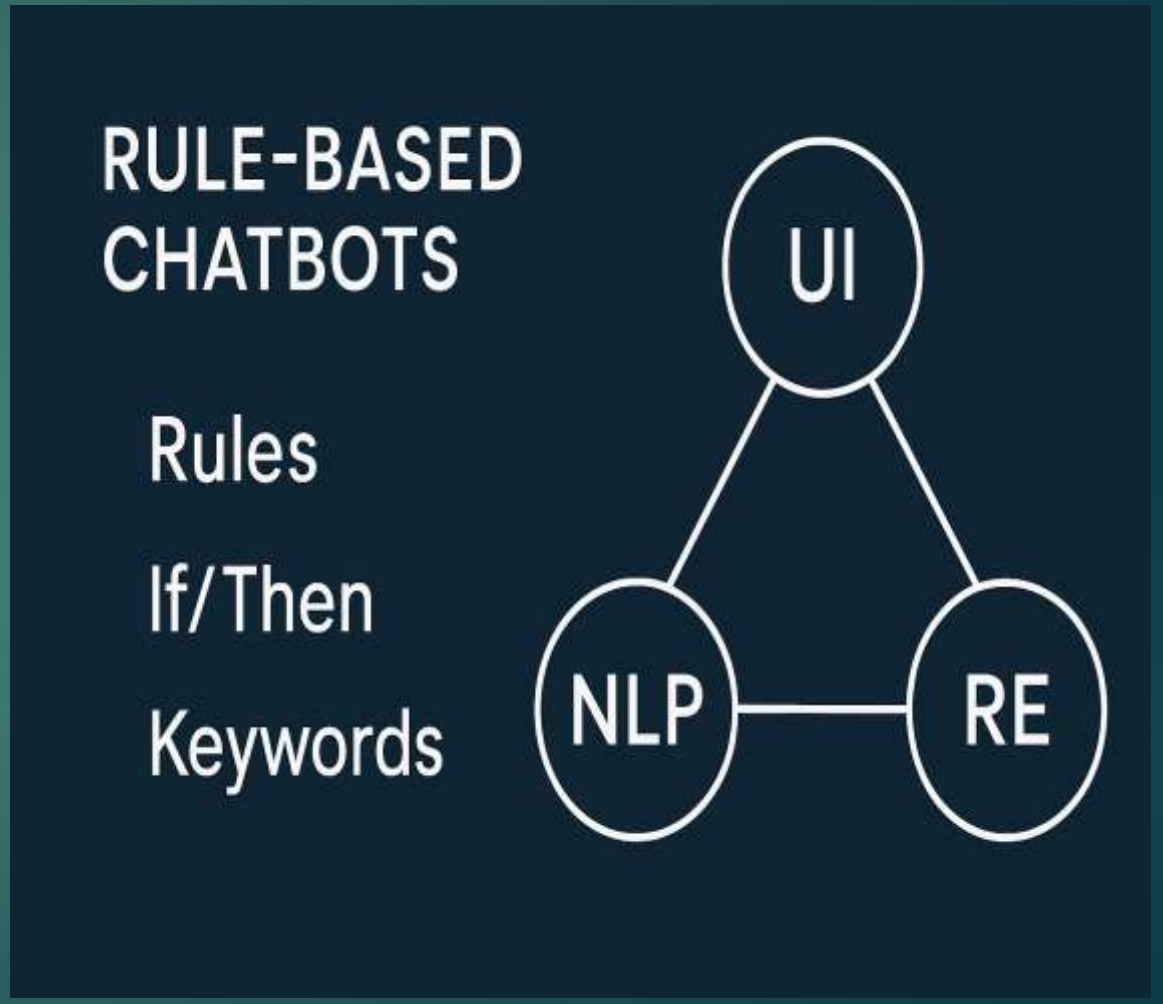
Types of Chatbots

- There are broadly two types of chatbots:



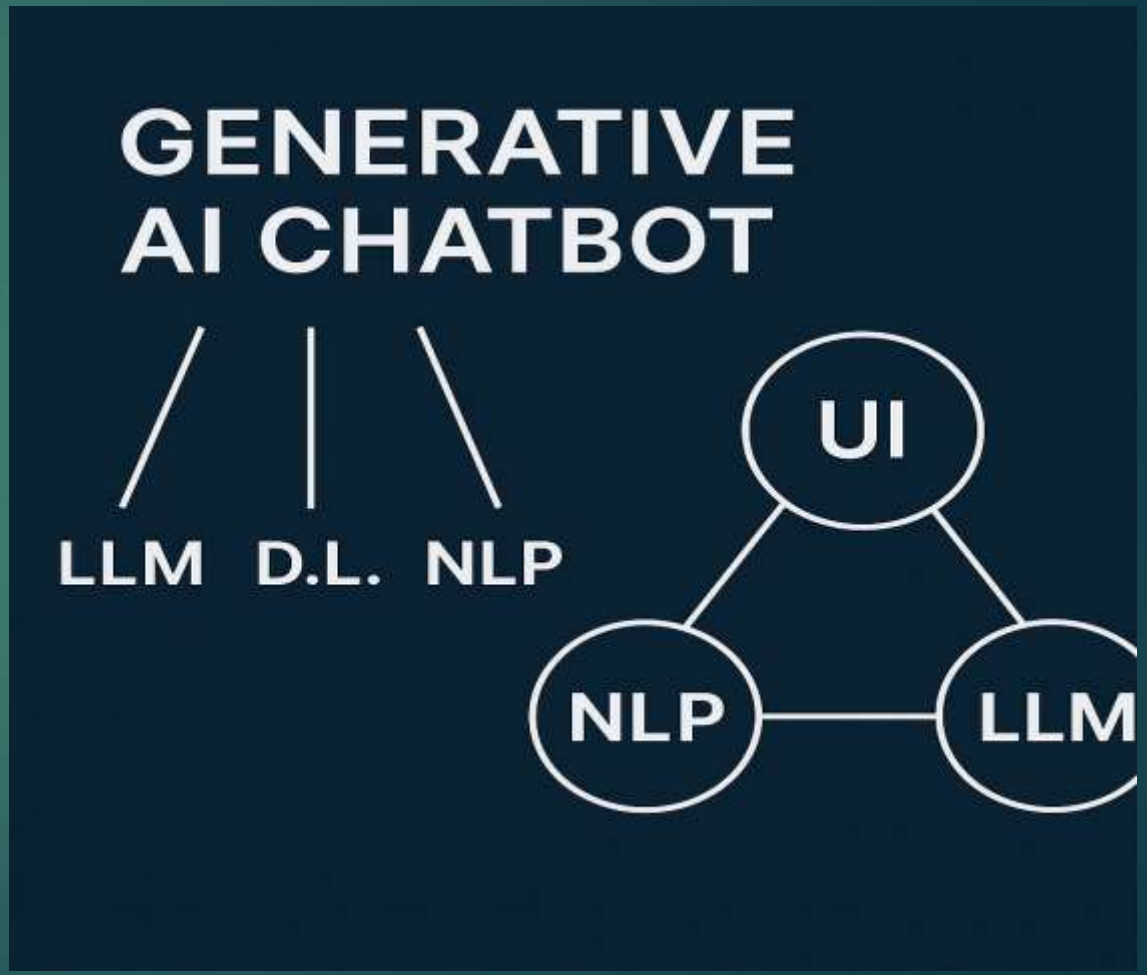
Rule-based Chatbots

- Rule-Based Chatbots utilize pre-determined rules, keywords, and a bunch of if/then statements that help the chatbot produce results based on these.
- They are limited to the knowledge programmed into them and give results conforming to a set of pre-determined rules.
- They do not have any "creativity" in the sense that they only produce responses based on the rules programmed into them and can not create anything distinctive on their own.
- Use cases: answering FAQs, booking tickets, customer support, etc.



Generative AI Chatbots

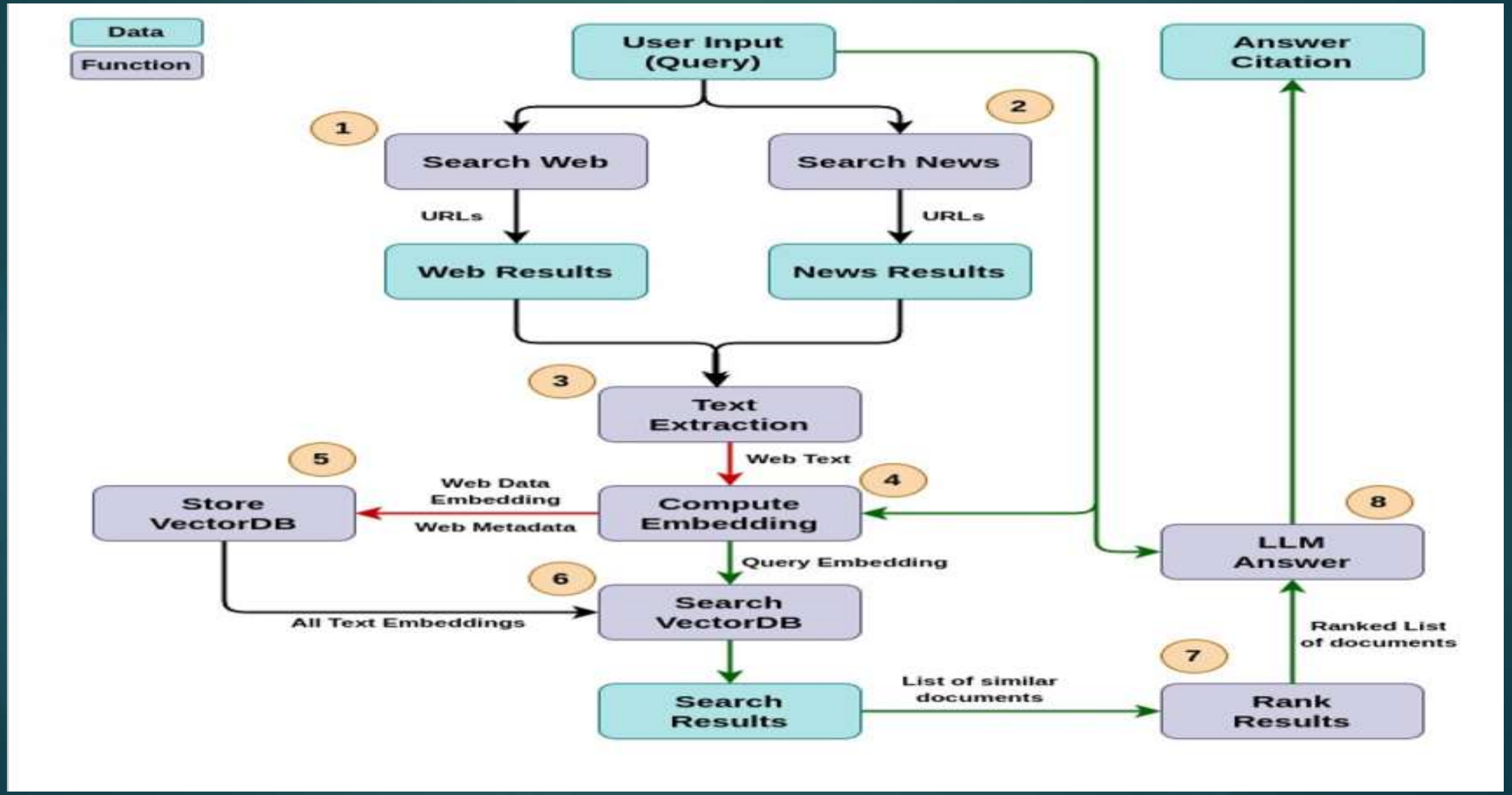
- Generative AI Chatbots utilize Deep Learning, Large Language Models (LLMs), and Natural Language Processing (NLP) in conjunction that help the chatbot understand and produce human-like responses.
- They understand the context and can create new responses instead of picking from a fixed set, unlike Rule-Based Chatbots.
- They continuously improve by learning from interactions.
- They are "creative" in the sense that they can create distinctive new responses that suit the context without relying solely on the programming logic.
- Use cases: virtual assistants, content generation, image generation, etc.



Retrieval-Augmented Generation

- Retrieval-Augmented Generation (RAG) is the process of optimizing the performance of a large language model, enabling it to reference external credible sources of information to generate an appropriate response. It enhances the performance of an LLM.
- It involves:
  - **Retrieve**: Retrieving relevant information from a credible source.
  - **Augment**: Feeding the retrieved information into the LLM.
  - **Generate**: Generating a response based on the retrieved information.
- By combining the information with LLM skills, it helps generate better responses and overcome hallucinations (providing misleading or false information that is not grounded in reality).

Workflow of the Project



Workflow of the Project (contd.)

1. The user input query hits the Serper web and news API endpoints. It extracts all the associated links.

2. The links are scraped to get text content only. The scraped content is stored in MongoDB Atlas database in a specific schema.

3. The extracted text is then broken down into chunks. These chunks are also stored in MongoDB Atlas database in a specific schema.

4. The embeddings are computed corresponding to these chunks and stored in Milvus vector database.

5. The stored chunks are indexed accordingly to mark that they have been processed once.

6. The user query is then converted into an embedding.

7. The embedding corresponding to the user query is then compared against all embeddings in Milvus vector database.

8. The top k (here, k = 5) embeddings are then elicited and chunks corresponding to these embeddings are obtained.

9. These chunks are then reordered in the context of the user query from most to least relevant via an LLM call.

10. The reordered chunks are then passed to the LLM as context along with the user query to get the final answer.

Detailed Module Breakdown

We have various modules in our project as follows:

- **src module**: It stands for source code. It keeps the overall code clean from the root directory. Within this, we have 2 files:
  - etl.py: It is used for extract, transform, and load.
  - retrieval_engine.py: It is used to retrieve relevant information.
- **utils module**: It stands for utilities. It keeps all the various code utilities that can be used across other modules. Within this, we have 5 files:
  - search_utils.py: It is used for search-related functionality to get the URLs.
  - db_utils.py: It is used for database-related functionality.
  - text_utils.py: It is used for text-related functionality.
  - LLM_utils.py: It is used for LLM-related functionality.
  - user_interface.py: It is used for developing the user interface.

Detailed Module Breakdown (contd.)

- **app.py**: It is the main file which serves as the entry point for the entire program. This is where the code runs.
- **README.md**: It contains the overall code organization.
- **Docker-compose.yml**: It contains the docker specifications that is useful for setting up the Milvus Vector Database.
- **Reports**: It contains all the project reports done after completion of the project.

Software and Services used

The various platforms used are:

- **Milvus Vector Database**: To store vector embeddings of chunks.
- **MongoDB Atlas**: To store documents as well as chunks.
- **Docker Compose**: To support Milvus connection setup.
- **Serper API**: To extract data from the web.

The various frameworks used are:

- **Langchain**: Aids in building a Large Language Model.
- **Gradio**: To build the user interface.
- **Sentence Transformers**: Used for embedding strings.

Software and Services used (contd.)

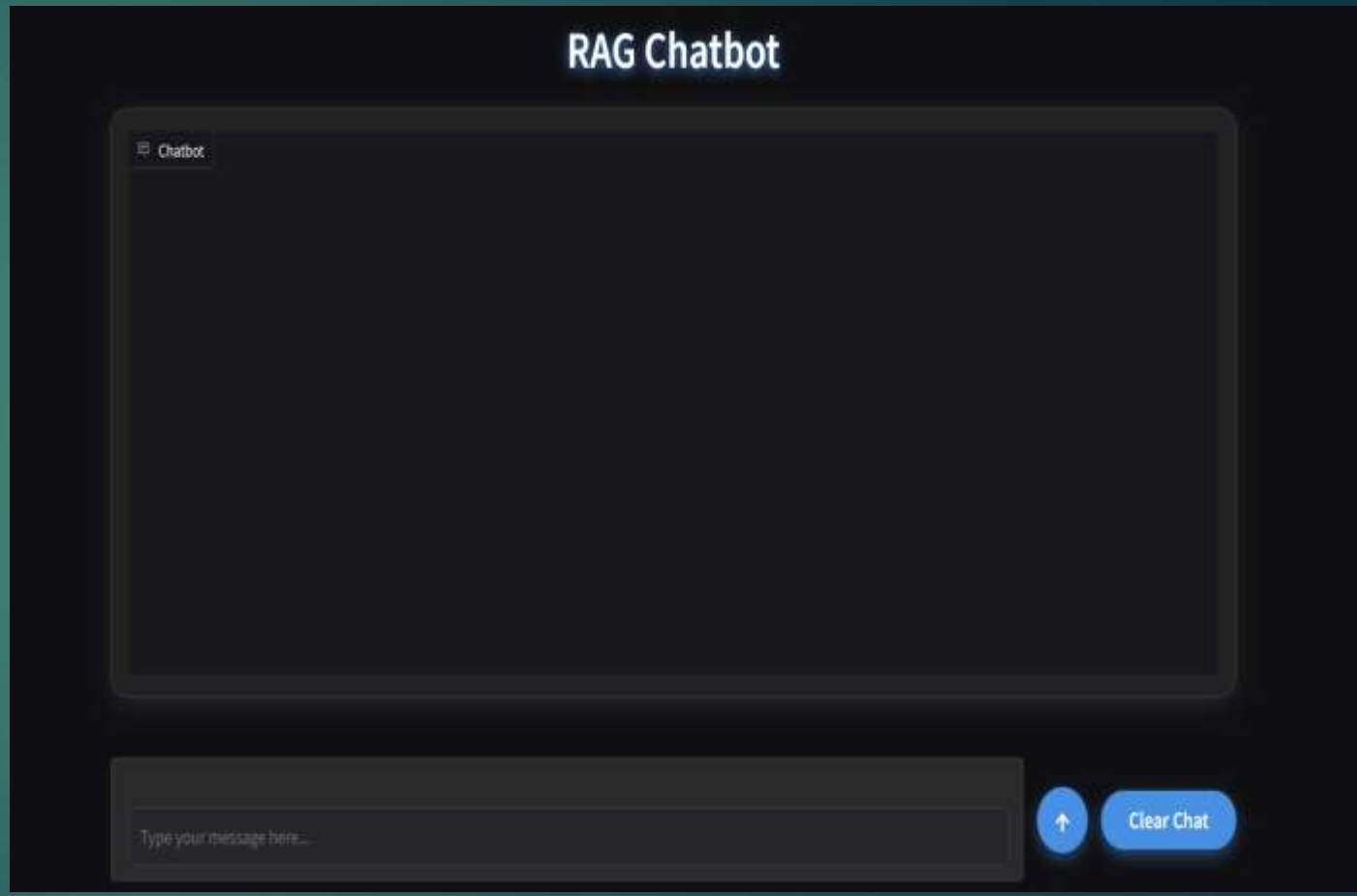
Various functionalities used are:

- API requests
- Web scraping
- HTML boilerplate cleaning
- Text cleaning
- Database storage and handling
- Chunking
- Vector embedding
- Semantic search and retrieval
- Large Language Models
- Prompt Engineering
- Conversation Buffer Memory
- User Interface

Results

Components of User Interface

- **Chat window**: This is the workspace where the chatbot's responses to the user's questions will be displayed.
- **Typing area**: This is the field where the user types the message.
- **Send button**: This is used to send the message to the chatbot after typing it.
- **Clear chat button**: This is used to clear chat with the chatbot.



User Interface

Results (contd.)

Example conversation

- User: "Where is India loacted?"
- Assistant: "India is located in South Asia."
- User: "What is its capital?"
- Assistant: "The capital of India is New Delhi."
- User: "What is its largest city?"
- Assistant: "Mumbai is the largest city in India."

Results

Sample usage

Upcoming Technologies and Future Scope: RIG

- Retrieval-Interleaved Generation (RIG) is the process of dynamically combining data retrieval and generation of appropriate responses.
- Unlike RAG where information is first fetched and then a response is generated, RIG integrates the retrieval process into the generation process itself.
- So here in RIG, we do not follow a sequential pattern of one process completing and then the next starting. Instead, the process are interleaved (multiple processes can alternate or run simultaneously without any one of them having to sequentially complete before the next can begin).

RAG vs RIG

Feature	Retrieval-Augmented Generation	Retrieval-Interleaved Generation
Procedure	It consists of two steps: retrieve first and then generate.	It consists of just one step: retrieve while generating.
Static/Dynamic	Retrieval is fixed before response generation.	Retrieval is carried out dynamically alongside response generation.
Example Tools and Frameworks	LangChain, LlamaIndex, Haystack, etc.	DRAGON, FiD-RAG, ReAct-style, etc. (emerging)
Speed	Faster	Slower (due to dynamic retrievals)
Use cases	Customer support chatbots, Enterprise search, live document analysis, etc.	Live financial reporting, scientific literature summarization, report writing, etc.

References

- https://datasciencedojo.com/blog/what-is-langchain/
- https://sebastian-petrus.medium.com/top-10-rag-frameworks-github-repos-2024-12b2a81f4a49
- https://www.tonic.ai/guides/enterprise-rag
- https://blog.google/technology/ai/google-datagemma-ai-llm/
- https://pub.towardsai.net/retrieval-interleaved-generation-rig-when-real-time-data-retrieval-meets-response-generation-a33e44ddbd74
- https://blog.google/technology/ai/google-datagemma-ai-llm/
- https://prajnaaiwisdom.medium.com/llm-trends-2025-a-deep-dive-into-the-future-of-large-language-models-bff23aa7cdbc

Thank you!