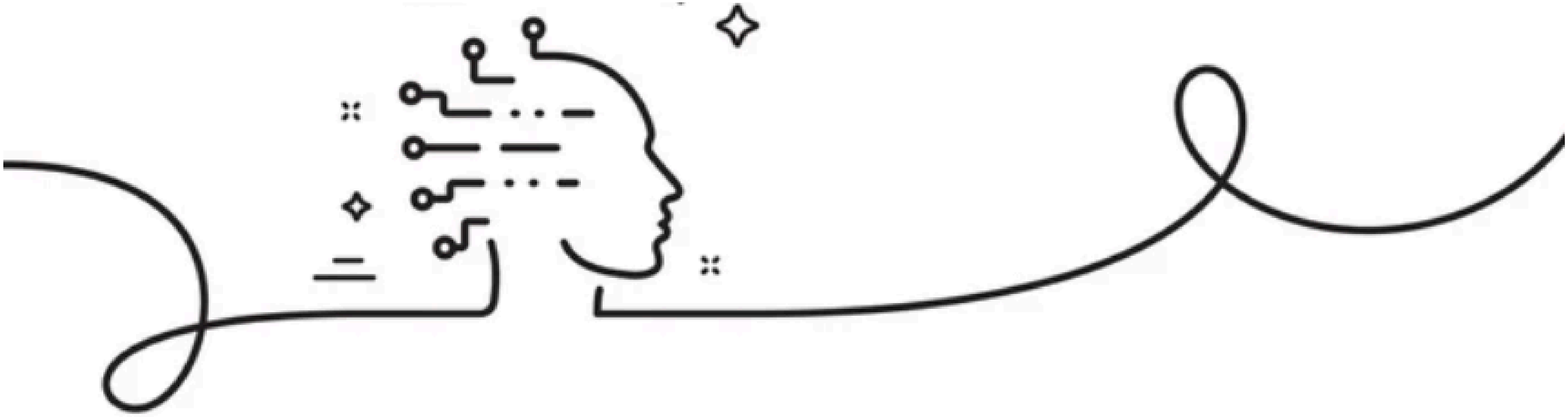
ARTIFICIAL INTELLIGENCE MACHINE LEARNING

LEARNFLU - FINAL PROJECT



Team Name : THE NBHD

Problem Statement : CHAT BOT

BIO PROFILE :

NAME: **PRAKASH RAJ A**

TEAM NAME : THE NBHD

PROJECT : CHAT BOT - Individual Project

DEPARTMENT : B.TECH-IT (2rd year)

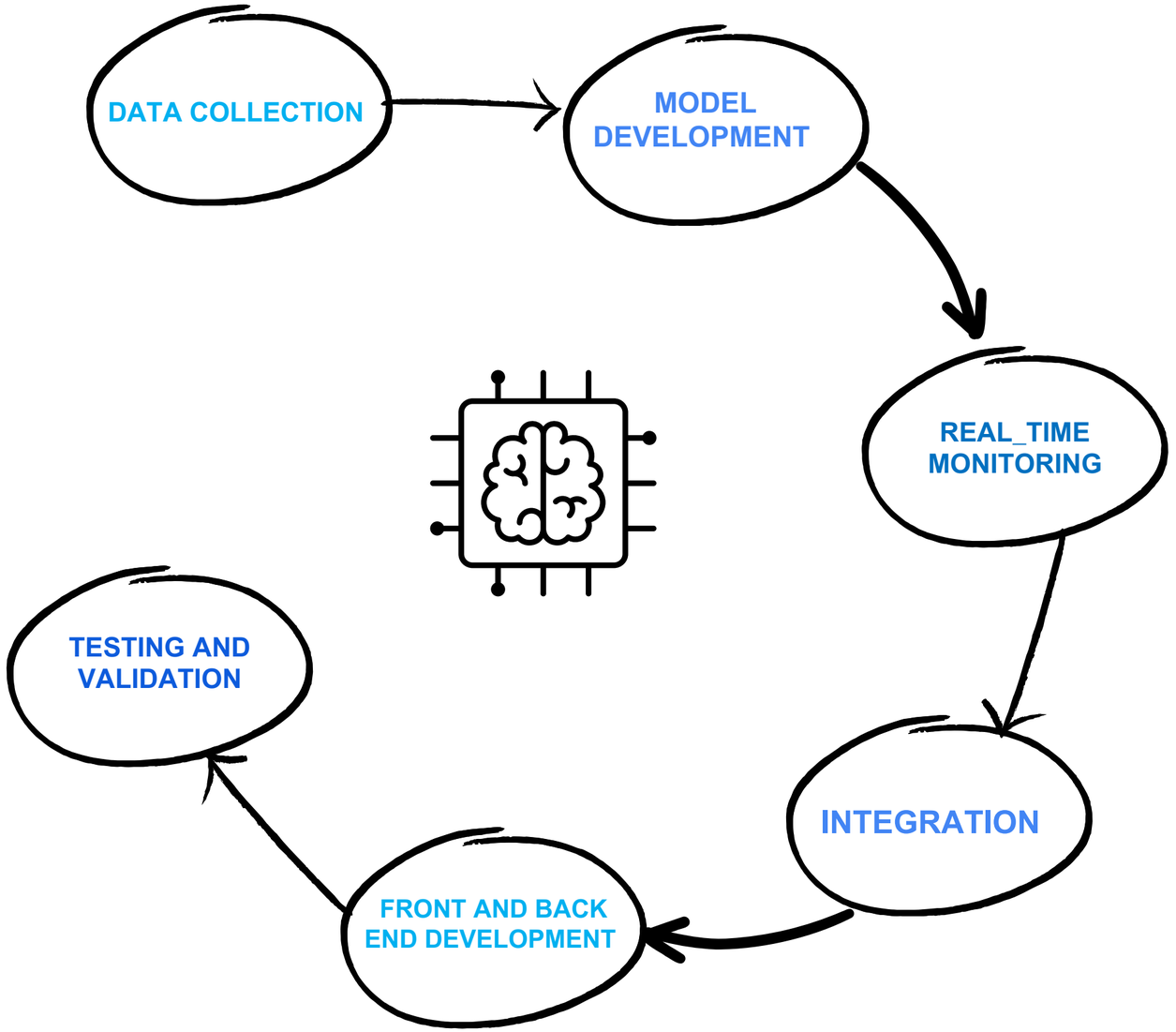
INSTITUTE : St. JOSEPH'S COLLEGE OF ENGINEERING

Brief about the Idea:

A chatbot project involves creating a conversational AI system designed to simulate human-like interactions with users through text or voice. It typically uses natural language processing (NLP) and machine learning techniques to understand and respond to user inputs. The goal is to build a bot that can handle various tasks such as answering questions, providing recommendations, automating customer support, or performing specific actions within a business or service.

Key components of a chatbot project include:

- Intent Recognition: Identifying the user's purpose or request.
- Entity Extraction: Understanding specific details in the conversation (e.g., dates, locations).
- Dialog Management: Managing the flow of the conversation and maintaining context.
- Integration: Connecting the chatbot with external APIs or databases to provide dynamic responses.



WORK FLOW

Diagram Overview of Workflow:

User Interaction
- User accesses the chat interface (HTML).
- User types a message and submits it (POST request to /get).

Backend Processing
- Flask receives the POST request.
- Extract user input (msg) from the request.
- Pass input to get_Chat_response function.

Model Interaction
- Model processes the input, keeps track of chat history, generates response using DialoGPT.
- Return response to Flask app.

Frontend Response
- Flask sends the chatbot's response to the frontend.
- The response is displayed in the chat interface

Improvements/Considerations for Production:

State Management:

The current code doesn't maintain user-specific chat histories across sessions. In a production application, you'd need to manage unique sessions or users using session management tools (e.g., Flask session or database).

Error Handling:

The application lacks error handling (e.g., what happens if the user input is empty or the model fails to generate a response). Adding proper error messages or fallback mechanisms could improve user experience.

Scalability and Optimization:

Loading the model on each request could be inefficient for production. You might want to load the model once when the app starts and keep it in memory for the lifetime of the app.

Frontend Features:

The HTML page should be designed to accept input dynamically (via forms or WebSockets) and update the UI without requiring page reloads.

FEATURES

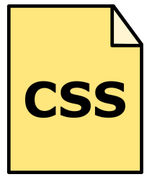
A HUGGING FACE MODEL

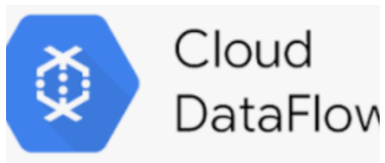
Microsoft DialoGPT, a pre-trained language model that can generate human-like responses to given prompts. We will be integrating DialoGPT with Flask, a popular Python web framework, to create a web application that can communicate with users via a chat interface.

For the frontend of our application, we will be using HTML, CSS, and JavaScript to create a visually appealing and interactive chat interface. Additionally, we will be using jQuery to handle the HTTP requests that are made to the backend server

TECH STACK

Google's AutoML





SUMMARY

Project Summary: Flask-based Chatbot using DialoGPT
This project is a simple chatbot application built with Flask, utilizing Microsoft's DialoGPT model to generate conversational responses. The goal is to create an interactive web-based interface where users can input messages and receive responses generated by the AI.

Workflow:

1. Flask Web Server:
   - The Flask app runs a local web server, serving the frontend and handling user requests.
2. Frontend (Chat Interface):
   - The user interacts with a chat interface rendered by the chat.html template. Users input their messages and submit them via a form.
3. Handling User Input:
   - The /get route processes the user's message using a POST request. The input is extracted and passed to the model for response generation.
4. Model Interaction (DialoGPT):
   - The input is tokenized and passed through the DialoGPT model, which maintains a history of the conversation and generates an appropriate response.
   - The generated response is decoded and returned to the frontend.
5. Frontend Displays Response:
   - The chatbot's response is displayed in the chat interface, creating a seamless conversation flow between the user and the AI.

Key Components:

- Flask: Handles routing and communication between the frontend and backend.
- DialoGPT: A pre-trained conversational model used to generate responses based on the user's input.
- Tokenization: The input is tokenized and processed by the model to understand and generate contextually relevant responses.
- Chat History: Maintains conversation context within a session, though improvements can be made for long-term state management.

Considerations for Improvement:

- Session Management: Store chat histories per user to maintain context across different sessions.
- Error Handling: Improve handling of empty inputs, failed responses, or system errors.
- Efficiency: Load the model once during the app's startup to avoid redundant reloading on every request.
- UI Enhancements: Create a dynamic, real-time chat experience with JavaScript for better user interaction.