**SUMMARY REPORT**

(AI-CHATBOT)

**Introduction:**

* The AI Chatbot is an intelligent application designed to simulate human-like conversations to assist users in getting information, answering queries, and providing services. This project leverages a React frontend for the chatbot interface and an Express.js backend for handling user interactions.
* Traditional methods like static FAQs and customer services calls are often slow and inefficient, this project aims to solve this problem by providing an AI-Powered chatbot that delivers fast, interactive and accurate responses, enhancing the user experience across different application.

**Implementation:**

* **Frontend:** (Vite + React) Step-By-Step implementation,
* First step is to initialize the project using **npm** and create **vite@latest** with react template.
* Then built a chatbot UI with a text input and send button.
* After that we will use **useState** to manage user input and message history.
* Installed **Axios** with npm install axios.
* On form submit, use Axios to make a POST request to [**http://localhost:5000/api/chatbot**](http://localhost:5000/api/chatbot)with user’s message.
* Updated the **UI** by displaying the bot’s response received from backend.
* **Backend:** (Node.js + Express) Step-By-Step implementation
* First setup Node.js project **npm init-y** and install express and axios.
* Then create index.js and setup an express server on port 5000.
* Define a POST route **/api/chatbot** to receive the message from the frontend.
* Then use axios to forward the message to an external chatbot API.
* Then after that it will receive the API response and send it back to the react frontend.
* Use **nodemon** for auto restart during development.

**Connected Components:**

The steps mentioned below will explain how we connected the frontend (React) and backend (Node.js/Express) using axios to handle HTTP requests.

* The frontend captures user input through a chat interface.
* When a message is sent, we will use axios to make a POST request to backend at the **/api/chatbot** route.
* The backend receives the request and forward the message to an external AI chatbot API (which we implemented earlier), than waits for a response.
* Once the response is received from the API, the backend sends it back to the frontend.
* Finally, the frontend displays the response in the chat UI

The setup allows the frontend and backend to stay clean and modular, with axios handling the sync communication smoothly.

**Docker Wrapping:**

The steps below explains how we wrapped the project in docker:

* Install the **docker desktop** from the official website and ste it up on the system.
* Then create docker files for both frontend and backend.
* Then backend runs a **Node.js server** on port 5000.
* Frontend (built with vite + react) is served using the **serve package** on port **5174**.
* Then create a docker-compose.yml file to run both containers together and manage networking.
* Tag the image properly (for e.g.)
  + - Docker tag backend-image yourusername/backend-chatbotai
    - Docker tag frontend-image yourusername/frontend-chatbotai
* Push both images to docker hub:
  + - Docker push yourusername/backend-chatbotai
    - Docker push yourusername/frontend-chatbotai.

**Conclusion:**

The AI Chatbot provides a foundation for an intelligent virtual assistant capable of enhancing user interactions across domains. With planned enhancements, it aims to deliver a smarter, more scalable, and personalized user experience.