**TEAM NOVA**

## Aurora, an Ai chatbot works by guiding users to live a sustainable lifestyle

## Aurora Chatbot Documentation

**Introduction**

Aurora is an AI chatbot application built using **Streamlit** and various AI tools and APIs, designed to assist users in understanding and adopting a sustainable lifestyle. It can answer questions on sustainability and provide general information based on user queries.

**Installation and Setup**

To run Aurora, make sure you have the following prerequisites installed:

**Requirements:**

* Python 3.x
* Libraries:
  + Streamlit
  + Transformers (T5Tokenizer, T5ForConditionalGeneration)
  + Pinecone
  + Groq
  + Torch
  + Base64
  + Time
  + OS

**Steps to Install:**

1. Clone or download the project files.
2. Install required libraries using:

bash

Copy code

pip install streamlit transformers pinecone-client groq torch

1. Run the Streamlit app:

bash

Copy code

streamlit run Aurora.py

**Components:**

**1. T5 Language Model**

The application uses **T5 (Text-to-Text Transfer Transformer)** model to process and generate text responses based on the user’s input.

* **Tokenizer**: Converts text input into tokens for model processing.
* **Model**: Generates conditional text responses.

**Functions:**

* generate\_response(input\_text): Generates a response from the user's input using the T5 model.

**2. Pinecone and Groq Integration**

Aurora uses **Pinecone** to store and query embeddings and **Groq** for chat completions:

* **Pinecone Indexing**: Queries relevant information from Pinecone based on user inputs.
* **Groq**: Provides an API for managing chat completions using the groq.ChatCompletion feature.

**3. UI Customization:**

* **Background Image**: A custom background image is set using base64 encoding.
* **CSS Styling**: The UI features a glowing title effect with animations. The font used is Roboto.
  + **Main Title**: A glowing, animated title "Aurora".
  + **Subheading**: A subheading introducing the app’s purpose.

**4. Chat Interface**

* A chat interface is set up to handle conversation flow:
  + **User Input**: Prompts entered by the user are displayed in the chat box.
  + **Typing Effect**: Aurora simulates typing for better user experience.
  + **Message History**: User messages and responses are stored and displayed in chat bubbles.
  + **API Key Management**: Users can input and store their API keys for using Groq and Pinecone.

**Functions:**

* display\_typing\_effect(text, role): Simulates a typing effect for the chatbot’s response.
* generate\_response\_with\_context(user\_input): Generates a context-aware response by accumulating conversation history.

**How It Works:**

1. **User Interaction**: Users can type in queries related to sustainable lifestyles.
2. **Contextual Response Generation**:
   * The input is processed by the T5 model, generating a response.
   * If relevant, the Pinecone index is queried to provide contextual data.
3. **Response Display**: Aurora responds with a dynamic typing effect for a more engaging experience.

**Customization Options:**

* **Background Image**: Change the image by modifying the background\_image path in the code.
* **Font and Animation**: Customize the CSS animations and fonts by modifying the styles in the st.markdown section.

**Example Usage:**

When the user interacts with Aurora, they may ask questions like:

* "How can I reduce my carbon footprint?"
* "What are some sustainable alternatives to plastic?"