**Documentation for Chat Bot + System Assistant in C++**

**Author:**

**Muhmmad Ali Zafar**

**241813**

**BSCS-2B**

**Overview:**

This project implements a voice-activated chatbot and system assistant in C++ using:

* **Vosk** for Speech-to-Text (STT)
* **SAPI** for Text-to-Speech (TTS)
* **CURL & JSON** for API calls (Gemini & Weather)
* **System Commands** for launching apps, shutdown, etc.
* **Multithreading & Audio Handling** using PortAudio

**Modules and Classes:**

**1. Class: Time**

* **Responsibility:** Handles time-related queries.
* **Key Methods:**
  + bool timeInitializer(string): Detects if command is for time.
  + string currentTime(): Returns formatted current time using chrono and put\_time.

**2. Class: sttSystem**

* **Responsibility:** Captures voice input and converts it to text.
* **Key Components:**
  + Initializes Vosk recognizer and PortAudio.
  + Listens until user presses Enter.
* **Key Method:**
  + string listen\_until\_enter(): Captures real-time voice input and converts to string.

**3. Class: ttsSystem**

* **Responsibility:** Converts text responses to speech.
* **Uses:** Microsoft's Speech API (SAPI)
* **Key Methods:**
  + void speak(string): Speaks the text input.
  + void printTypingEffect(string, int delay): Prints response with a typing animation.

**4. Class: SystemCmd**

* **Responsibility:** Detects and executes system-level commands.
* **Key Methods:**
  + bool initializeSystemBot(string): Detects if system command is activated.
  + void listen(string): Receives command input.
  + void systemResponse(): Executes mapped system commands (e.g., open Chrome, shutdown PC).

**5. Class: API (Base Class)**

* **Responsibility:** Provides core CURL and API key setup functionality.
* **Key Members:**
  + apiKey, url, buffer
  + WriteCallback() - Static function to write API response to string buffer.

**6. Class: WeatherAPI (Derived from API)**

* **Responsibility:** Fetches weather information for a given city.
* **Key Methods:**
  + bool initializeWeatherAPI(string&): Detects weather query.
  + string fetchWeather(const string&): Sends HTTP request to weatherapi.com and parses JSON result.

**7. Class: GeminiAPI (Derived from API)**

* **Responsibility:** Sends prompt to Gemini and receives response.
* **Key Features:**
  + Prompt engineering to ensure sarcastic, witty responses.
  + Markdown cleanup for TTS compatibility.
* **Key Methods:**
  + bool initializeGemini(string&)
  + string askGemini(string&): Sends POST request and parses the Gemini response.
  + string modifiedPrompt(string&): Adds sarcasm and behavior instruction to prompt.
  + string cleanMarkdown(string): Strips characters unsuitable for TTS.

**External Libraries and Tools:**

* **Vosk API**: Open-source STT engine.
* **Microsoft SAPI**: Windows-only TTS engine.
* **PortAudio**: Low-latency audio input stream.
* **libcurl**: Used for making REST API requests.
* **nlohmann/json**: Header-only C++ JSON parser.

**Functionality Flow:**

1. **Voice Capture**: User speaks; sttSystem records until Enter is pressed.
2. **Command Parsing**: Input is passed through different command initializers:
   * Time → If matched, returns current time.
   * SystemCmd → Executes app/system commands.
   * WeatherAPI → Makes API call to fetch weather data.
   * GeminiAPI → Sends prompt and returns AI-generated response.
3. **Response Output**:
   * Terminal output with typing effect.
   * Voice feedback using TTS.

**Potential Improvements:**

* Add fallback response for unrecognized commands.
* Store chat history.
* Add voice activation (wake-word detection).
* Add support for additional APIs or custom commands.

**Notes:**

* The program is Windows-specific due to SAPI and system() commands.
* API keys must be secured; current implementation stores them as plain strings.
* All voice and API features require working internet and microphone.