GitHub repository link with API calling and output: https://github.com/amanrock005/GenAIApps

Case 1: ChatGPT

Overview

- ChatGPT is an AI model developed by OpenAI, built upon the GPT (Generative Pre-trained transformer) architecture.
- It is designed to understand and generate human-like text based on a given prompt.

Key Features

1. Transformer-based Architecture
2. Pre-training and Fine-tuning
3. API Integration
4. Reinforcement Learning from Human Feedback (RLHF)

Transformer-based Architecture

- ChatGPT is based on the Transformer architecture, which uses self-attention mechanism to process sequences of data in parallel.
- It enables the model to handle long-range dependencies and context better than previous models like RNNs or LSTMs.

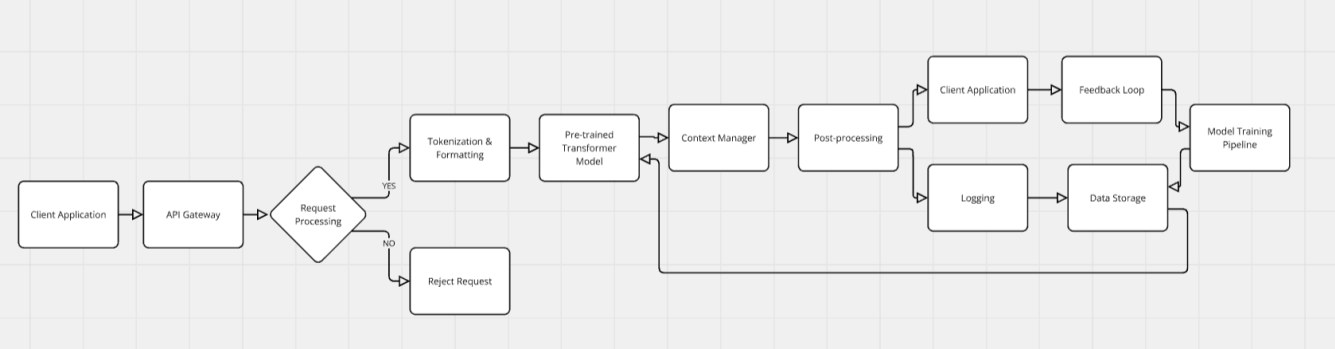
Pre-training and Fine-tuning

- The model is pre-trained on massive datasets from different sources, such as books, websites etc.
- The model is fine-tuned using supervised learning technique and reinforcement learning RLHF.

API and Integration

- OpenAI provides an API for integrating ChatGPT into various applications, allowing software engineers to integrate chatGPT functionality into their own apps, website or internal system.
- The API response can be tweaked by altering the parameters such as temperature, max_token etc.

High-Level Architecture Design



- **Client Application**: The frontend where users interact with ChatGPT (web app, mobile app or chatbot interface)
- **API Gateway**: The entry point that routes user requests to the backend. The input passes through the API gate for authentication and basic validation.
- **Request Processing**: Handles authentication, API Key verification, request validation and preprocessing of input.
    - Text is tokenized and formatted into a structure understandable by the chatGPT model.
- **Pre-trained transformer model**: The core of chatGPT, built using a large scale transformer architecture.
    - The context manager retrieves prior conversation history for multi-turn response.
- **Post-processing**: Filters output to ensure they are safe, relevant, and compliant with ethical guidelines.
- **Logging**: Logs req/res for debugging, monitoring performance, and improving the model.
- **Feedback Loop**: capture user feedback for reinforcement learning and continuous improvement.
- **Data Storage**: Stores model parameters, training data, user preferences, and interaction logs.
- **Model Training Pipeline**: Uses RLFH for continuous improvement.

Reference: Attention is All you Need

API Endpoints

1. **Generate Response (*POST /v1/chat/completions)***
   Sends a chat prompt to the API and receives a conversational response based on the specified model and parameters.

2. **Retrieve Model List (*GET /v1/models)***
   Fetches a list of available models, including their ID's and capabilities.

3. **Retrieve Usage Metrics (*GET /v1/usage)***
   Provides data on API usage, such as token counts and limits, for monitoring and billing purposes.

4. **File Upload (*POST /v1/files)***
   Uploads files to be used in fine-tuning or other API functionalities.

5. **Delete File (*DELETE /v1/files/{file_id}*)**
   Delete a previously uploaded file from your OpenAI account.

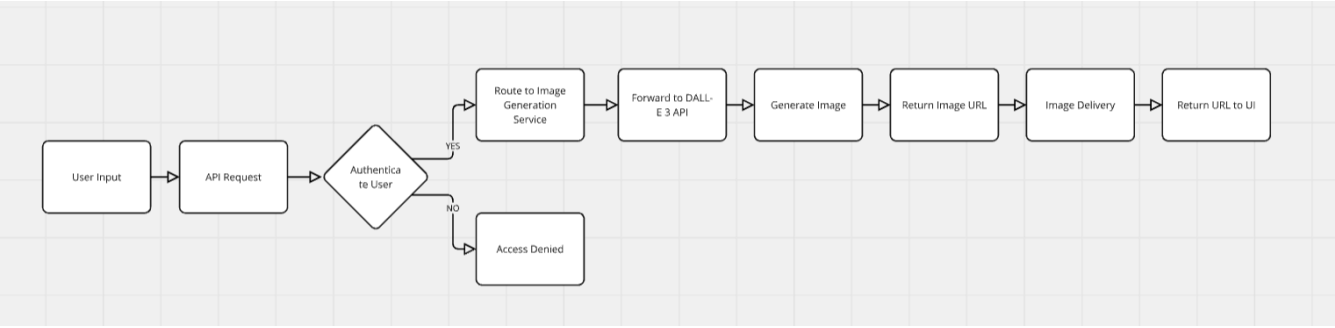
6. **Fine-Tuning Jobs(*POST /v1/fine-tunes)***
   Starts a fine-tuning process to customize a model using uploaded training data.

Case 2: Dalle 3

Overview

- DALLE 3, is OpenAI's image-generation model. It can create high-quality, detailed and visually compelling images from natural language descriptions.
- DALLE 3 allows users to generate and refine images through conversational prompts.

High-Level Architecture Design



- **User Input:** The user provides a text prompt via the UI
- **API Request:** The UI sends the prompt as a request to the API Gateway, which authenticates the user and routes the request to the image generation service.
- **Image Generation**: The image generation services forwards the prompt to the DALLE3 API, which processes the text and generates the image.
- **Image URL:** The OpenAI API returns an image URL to the image generation service.
- **Image Delivery**: The image generation service sends the image URL back to the API gateway and then returns the URL to the UI.

API Endpoints

1. **POST /v1/generate-image**
   It allows the user to request the generation of an image based on a text prompt. The user enters the prompt and the API returns the generated image url.