**Module: OpenAI Avenger Pet Name Generator**

**Introduction**

The OpenAI Animal Name Generator is a module designed to generate creative and imaginative superhero names for animals using the OpenAI API. It utilizes the GPT-3.5-turbo-instruct model for text generation. This documentation provides an overview of the code structure, import statements, configuration setup, and the main route handler exported as an asynchronous function.

**Import Statements**

import { Configuration, OpenAIApi } from "openai";

The code begins by importing the necessary modules from the "openai" library. It specifically imports the Configuration and OpenAIApi classes.

**Configuration Setup**

const configuration = new Configuration({

  apiKey: process.env.OPENAI\_API\_KEY,

});

A configuration object is created using the Configuration class. The OpenAI API key is obtained from the environment variables (stored in process.env.OPENAI\_API\_KEY) and passed to the configuration object.

**OpenAIApi Initialization**

const openai = new OpenAIApi(configuration);

An instance of the OpenAIApi class is created by passing the configuration object to it. This instance, named openai, will be used to make API requests.

**Exported Async Function**

export default async function (req, res) {...}

The main functionality is encapsulated in an asynchronous function exported as the default export. This function serves as the route handler for incoming requests.

**API Key Check**

if (!configuration.apiKey) {

    res.status(500).json({

      error: {

        message: "OpenAI API key not configured, please follow instructions in README.md",

      }

    });

    return;

  }

This block checks if the OpenAI API key is configured. If not, it returns a 500 Internal Server Error with an error message indicating that the API key needs to be configured.

**Request Body Validation**

const animal = req.body.animal || '';

  if (animal.trim().length === 0) {

    res.status(400).json({

      error: {

        message: "Please enter a valid animal",

      }

    });

    return;

  }

Validates the request body to ensure that a valid animal name is provided. If not, it returns a 400 Bad Request with an error message.

**OpenAI API Request**

try {

    const completion = await openai.createCompletion({

      model: "gpt-3.5-turbo-instruct",

      prompt: generatePrompt(req.body.animal),

      temperature: 0.6, // \* creative text generation

    });

    res.status(200).json({ result: completion.data.choices[0].text });

  }

**Response Handling**

catch(error) {

    // Consider adjusting the error handling logic for your use case

    if (error.response) {

      console.error(error.response.status, error.response.data);

      res.status(error.response.status).json(error.response.data);

    } else {

      console.error(`Error with OpenAI API request: ${error.message}`);

      res.status(500).json({

        error: {

          message: 'An error occurred during your request.',

        }

      });

    }

  }

Handles errors that may occur during the API request. It checks for both network-related errors and errors returned by the OpenAI API. The appropriate status code and error message are sent as a response.

**Prompt Generation Function**

function generatePrompt(animal) {

  const capitalizedAnimal =

    animal[0].toUpperCase() + animal.slice(1).toLowerCase();

  return `Suggest three names for an animal that is a superhero.

Animal: Cat

Names: Captain Sharpclaw, Agent Fluffball, The Incredible Feline

Animal: Dog

Names: Ruff the Protector, Wonder Canine, Sir Barks-a-Lot

Animal: ${capitalizedAnimal}

Names:`;

}

Defines a function generatePrompt that takes an animal name as input and returns a formatted prompt. The prompt includes the animal’s name and examples of superhero names for that animal.