PROJECT REPORT

ON

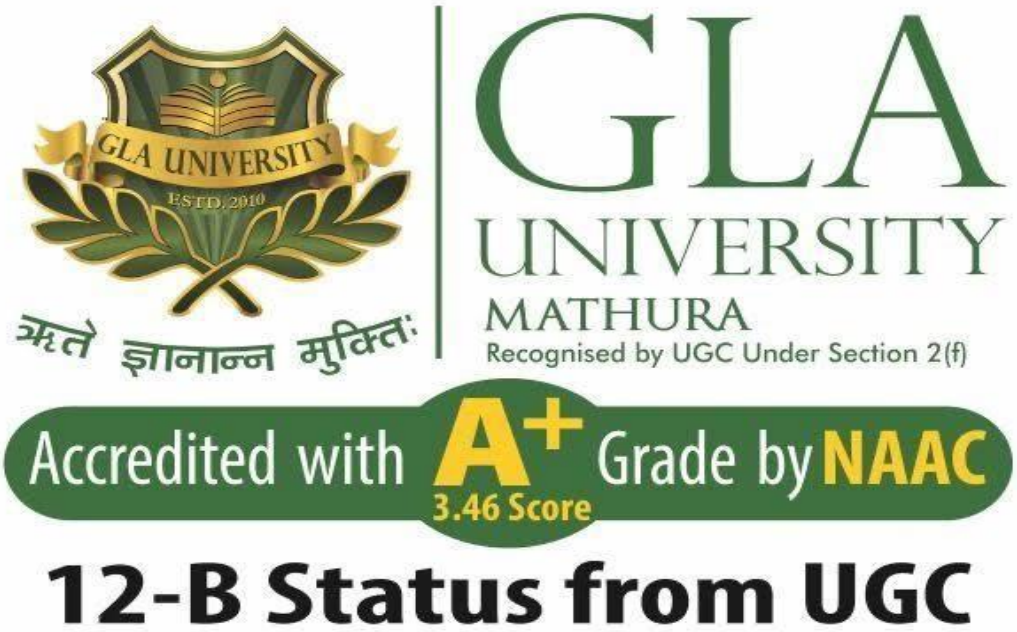
OPEN AI API VERSE

SUBMITTED BY:

Abhishek Shukla (201500024)

Divik Singh (201500227)

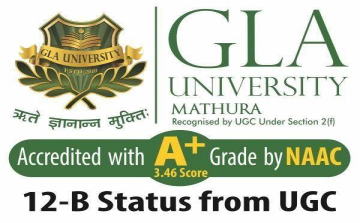
Tushar Tomar (201500748)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSITUTE OF ENGINNERING & TECHNOLOGY

GLA UNIVERSITY, MATHURA



DECLARATION

I would like to express my special thanks of gratitude to my project guide Ms. Gurpreet Kaur ma'am who gave me the golden opportunity to do this wonderful project on the topic OPEN AI API VERSE which connect usto a friends and I came to know about so many new things I am reallythankful to them.

Secondly, I would also like to thank my friends who helped me a lot in finalizing this project within the limited time frame.

Candidate Names:

Abhishek Shukla (201500024)

Divik Singh (201500227)

Tushar Tomar (201500748)

CERTIFICATE

This is to certify that the project entitled “OPEN AI API VERSE”, carried out in Mini Project – II Lab, is a bonafide work by our respective team and is submitted in partial fulfillment of the requirements for the award of the degree Bachelor of Technology (Computer Science & Engineering).

Project Supervisor

Ms. Gurpreeet Kaur

Assistant Professor

Date: 27-April-2023

Table of Content

INTRODUCTION

The OpenAI API provides a format for describing the structure of APIs and their operations, including information such as the available endpoints, input and output parameters, authentication requirements, and response formats. This information can be used to automatically generate documentation, code libraries, and even client SDKs in a variety of programming languages.

For ChatGPT, the motivation is to develop a natural language processing (NLP) system that can interact with users in a more human-like way, enabling more intuitive and efficient communication with machines. This technology has the potential to be used in a wide range of applications, such as customer service, virtual assistants, and educational tools. Similarly, the motivation behind AI image generators is to develop computer algorithms that can create images with high levels of visual fidelity and creative expression. This technology has the potential to be used in a variety of fields, including graphic design, video game development, and filmmaking. AI image generators can also be used to augment and enhance human creativity, providing new tools and inspiration for artists and designers.

The functionality of ChatGPT is to engage in natural language conversations with users and generate responses that are appropriate and informative. ChatGPT achieves this through a combination of machine learning and natural language processing techniques.

TECHNOLOGY USED

HTML -

The **HyperText Markup Language** or **HTML** is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes, and other items. HTML elements are delineated by *tags*, written using angle brackets. HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. The inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.[2] A form of HTML, known as HTML5, is used to display video and audio, primarily using the <**canvas**> element, in collaboration with javascript.

CSS -

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colours, layout, and fonts, thus making our web pages presentable to the users.

CSS is designed to make style sheets for the web. It is independent of HTML and can be used with any XML-based markup language. Now let’s try to break the acronym:

- Cascading: Falling of Styles
- Style: Adding designs/Styling our HTML tags
- Sheets: Writing our style in different documents

Java Script -

JavaScript was initially created to “make web pages alive”.
The programs in this language are called *scripts*. They can be written right in a web page’s HTML and run automatically as the page loads.
Scripts are provided and executed as plain text. They don’t need special preparation or compilation to run.
In this aspect, JavaScript is very different from another language called Java.
Today, JavaScript can execute not only in the browser, but also on the server, or actually on any device that has a special program called the java script.
The browser has an embedded engine sometimes called a “JavaScript virtual machine”.
Different engines have different “code names”. For example:

- V8 – in Chrome, Opera and Edge.
- Spider Monkey – in Firefox.
- There are other code names like “Chakra” for IE, “JavaScript”, “Nitro” and “Squirrel Fish” for Safari, etc.

The terms above are good to remember because they are used in developer articles on the internet. We’ll use them too. For instance, if “a feature X is supported by V8”, then it probably works in Chrome, Opera and Edge.

SYSTEM REQUIREMENTS

Software Requirement-

To build web application –

- 64-bit Windows 8/10/11
- VS code (version 1.73)
- XAMPP (version 8.1.10)

Hardware Requirement –

- x86_64 CPU architecture; 2nd generation Intel Core or newer
- 2 GB RAM or more
- 4 GB of available disk space minimum

IMPLEMENTATION

Explanation of Source Code :-

Data Preparation: Gather a large dataset of text conversations for your model to learn from. You can use pre-existing datasets like Cornell Movie Dialogues Corpus or scrape chat logs from online forums.

OpenAI API Setup: Sign up for the OpenAI API and get your API key. You can use the OpenAI GPT-3 API to generate text responses for your chatbot.

Model Training: Use the OpenAI API to fine-tune the GPT-3 model on your conversational dataset. You will need to use the OpenAI API to send requests to the GPT-3 model and receive responses. You can use Node.js's axios library to make HTTP requests to the OpenAI API.

Deployment: Once you have trained your model, you can deploy it using a web framework like Express.js or Hapi.js. You can create a simple UI that allows users to input text and get a response from the chatbot. Alternatively, you can create a command-line interface for users to use the model locally.

Data Preparation: Gather a large dataset of images for your model to learn from. You can use pre-existing datasets like ImageNet or COCO or scrape images from online sources.

OpenAI API Setup: Sign up for the OpenAI API and get your API key. You can use the OpenAI DALL-E API to generate new images based on your input.

Model Training: Use the OpenAI API to generate new images based on your input. You will need to use Node.js's axios library to make HTTP requests to the OpenAI API. You can create a function that takes an input image and returns a new generated image.

Deployment: Once you have created your function, you can deploy it using a web framework like Express.js or Hapi.js. You can create a simple UI that allows users to upload an input image and get a generated image in return. Alternatively, you can create a command-line interface for users to use the model locally.

Final Code :-

Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="./style.css">
</head>
<body>
    <a href="https://chat-gpt-clone-liard.vercel.app/">
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        CHAT GPT
    </a>
    <a href="https://rich-red-elephant-wig.cyclic.app/">
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        AI IMAGE GENERATOR
    </a>
</body>
</html>
```

Style.css:

```
@import
url('https://fonts.googleapis.com/css2?family=Raleway:wght@400;700&display=swap');
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
body{
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background: #050801;
    font-family: 'Raleway', sans-serif;
    font-weight: bold;
}
a{
    position: relative;
    display: inline-block;
    padding: 25px 30px;
    margin: 40px 0;
```

```
    color: #03e9f4;
    text-decoration: none;
    text-transform: uppercase;
    transition: 0.5s;
    letter-spacing: 4px;
    overflow: hidden;
    margin-right: 50px;

}
a:hover{
    background: #03e9f4;
    color: #050801;
    box-shadow: 0 0 5px #03e9f4,
                0 0 25px #03e9f4,
                0 0 50px #03e9f4,
                0 0 200px #03e9f4;
     -webkit-box-reflect:below 1px linear-gradient(transparent, #0005);
}
a:nth-child(1){
    filter: hue-rotate(270deg);
}
a:nth-child(2){
    filter: hue-rotate(110deg);
}
a span{
    position: absolute;
    display: block;
}
a span:nth-child(1){
    top: 0;
    left: 0;
    width: 100%;
    height: 2px;
    background: linear-gradient(90deg,transparent,#03e9f4);
    animation: animate1 1s linear infinite;
}
@keyframes animate1{
    0%{
        left: -100%;
    }
    50%,100%{
        left: 100%;
    }
}
a span:nth-child(2){
    top: -100%;
    right: 0;
    width: 2px;
    height: 100%;
    background: linear-gradient(180deg,transparent,#03e9f4);
    animation: animate2 1s linear infinite;
    animation-delay: 0.25s;
}
```

```css
@keyframes animate2{
    0%{
        top: -100%;
    }
    50%,100%{
        top: 100%;
    }
}
a span:nth-child(3){
    bottom: 0;
    right: 0;
    width: 100%;
    height: 2px;
    background: linear-gradient(270deg,transparent,#03e9f4);
    animation: animate3 1s linear infinite;
    animation-delay: 0.50s;
}
@keyframes animate3{
    0%{
        right: -100%;
    }
    50%,100%{
        right: 100%;
    }
}


a span:nth-child(4){
    bottom: -100%;
    left: 0;
    width: 2px;
    height: 100%;
    background: linear-gradient(360deg,transparent,#03e9f4);
    animation: animate4 1s linear infinite;
    animation-delay: 0.75s;
}
@keyframes animate4{
    0%{
        bottom: -100%;
    }
    50%,100%{
        bottom: 100%;
    }
}
```

CHAT GPT CODE:

Client side

```
import bot from './assets/bot.svg'
import user from './assets/user.svg'

const form = document.querySelector('form')
const chatContainer = document.querySelector('#chat_container')

let loadInterval

function loader(element) {
    element.textContent = ''

    loadInterval = setInterval(() => {
        // Update the text content of the loading indicator
        element.textContent += '.';

        // If the loading indicator has reached three dots, reset it
        if (element.textContent === '....') {
            element.textContent = '';
        }
    }, 300);
}

function typeText(element, text) {
    let index = 0

    let interval = setInterval(() => {
        if (index < text.length) {
            element.innerHTML += text.charAt(index)
            index++
        } else {
            clearInterval(interval)
        }
    }, 20)
}

// generate unique ID for each message div of bot
// necessary for typing text effect for that specific reply
// without unique ID, typing text will work on every element
function generateUniqueId() {
    const timestamp = Date.now();
    const randomNumber = Math.random();
    const hexadecimalString = randomNumber.toString(16);

    return `id-${timestamp}-${hexadecimalString}`;
}

function chatStripe(isAi, value, uniqueId) {
    return (
```

```
        `
        <div class="wrapper ${isAi && 'ai'}">
            <div class="chat">
                <div class="profile">
                    <img
                      src=${isAi ? bot : user}
                      alt="${isAi ? 'bot' : 'user'}"
                    />
                </div>
                <div class="message" id=${uniqueId}>${value}</div>
            </div>
        </div>
    `
    )
}

const handleSubmit = async (e) => {
    e.preventDefault()

    const data = new FormData(form);

    // user's chatstripe
    chatContainer.innerHTML += chatStripe(false, data.get('prompt'));

    // to clear the textarea input
    form.reset();

    // bot's chatstripe
    const uniqueId = generateUniqueId();
    chatContainer.innerHTML += chatStripe(true, " ", uniqueId);

    // to focus scroll to the bottom
    chatContainer.scrollTop = chatContainer.scrollHeight;

    // specific message div
    const messageDiv = document.getElementById(uniqueId);

    // messageDiv.innerHTML = "..."
    loader(messageDiv);

    //fetch data from server -> bot's response

    const response = await fetch('https://chatgpt-clone-4f1d.onrender.com', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({
            prompt: data.get('prompt')
        })
    })
```

```
    messageDiv.innerHTML = '';

    if(response.ok){
        const data = await response.json();
        const parseData = data.bot.trim();

        typeText(messageDiv, parseData);
    } else {
        const err = await response.text();

        messageDiv.innerHTML = "SOMETHING WENT WRONG";

        alert(err);
    }
}

form.addEventListener('submit', handleSubmit)
form.addEventListener('keyup', (e) => {
    if (e.keyCode === 13) {
        handleSubmit(e)
    }
})
```

Server side

```
import express from 'express';
import * as dotenv from 'dotenv';
import cors from 'cors'; //allows us to make cross origin requests
import { Configuration , OpenAIApi } from 'openai';

dotenv.config();

// console.log(process.env.OPENAI_API_KEY)

const configuration = new Configuration({
    apiKey: process.env.OPENAI_API_KEY,
});

const openai = new OpenAIApi(configuration);

const app = express();
app.use(cors());
app.use(express.json());  //allows us to send json from froentend to backend

app.get('/', async(req, res) => {
    res.status(200).send({
        message: 'Hello from this side',
    })
});

app.post('/', async(req, res) => {
```

```
        const prompt = req.body.prompt;

        const response = await openai.createCompletion({
            model: "text-davinci-003",
            prompt: `${prompt}`,
            temperature: 0,
            max_tokens: 3000,
            top_p: 1,
            frequency_penalty: 0.5,
            presence_penalty: 0,
        });;

        res.status(200).send({
            bot: response.data.choices[0].text
        })
    } catch (error){
        console.log(error);
        res.status(500).send({ error })
    }
})

app.listen(5000, () => console.log('Server is running on port
http://localhost:5000'));
```

AI IMAGE GENERATOR CODE:

index.js

```
const path = require('path');
const express = require('express');
const dotenv = require('dotenv').config();
const port = process.env.PORT || 5000;

const app = express();
// ENABLE BODY PARSER TO ACCESS BODY DATA
app.use(express.json());
app.use(express.urlencoded({extended: false}));

// SET STATIC FOLDER
app.use(express.static(path.join(__dirname, 'public')));


app.use('/openai', require('./routes/openAIroutes'))

app.listen(port, () => console.log(`server started on port ${port}`));
```

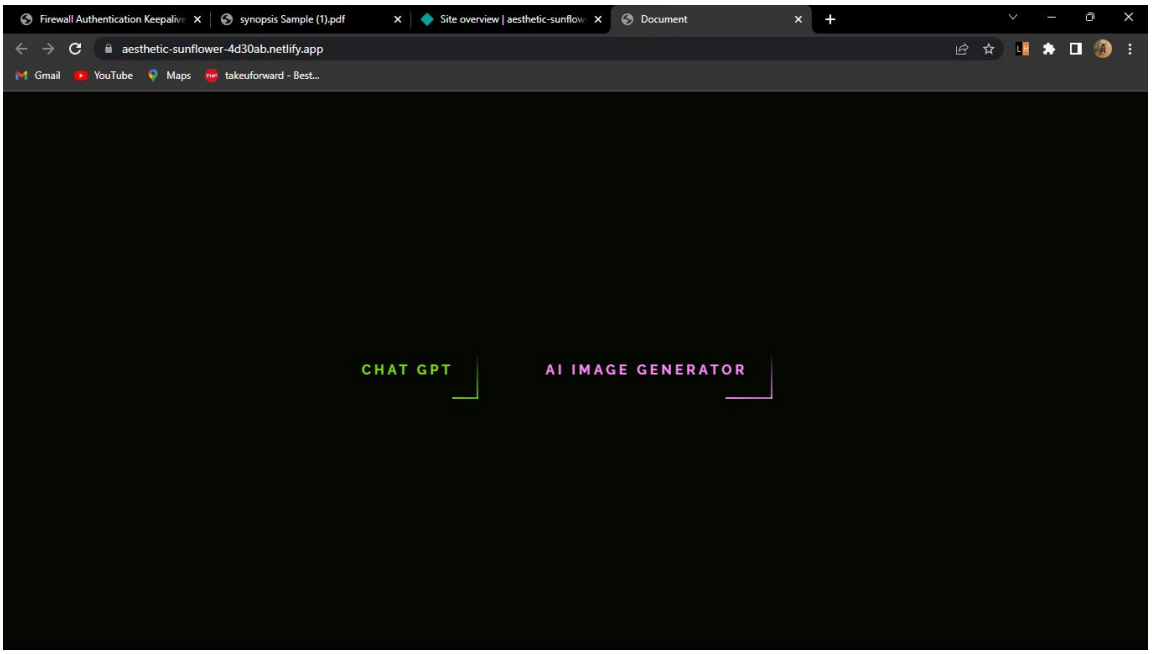
controller.js

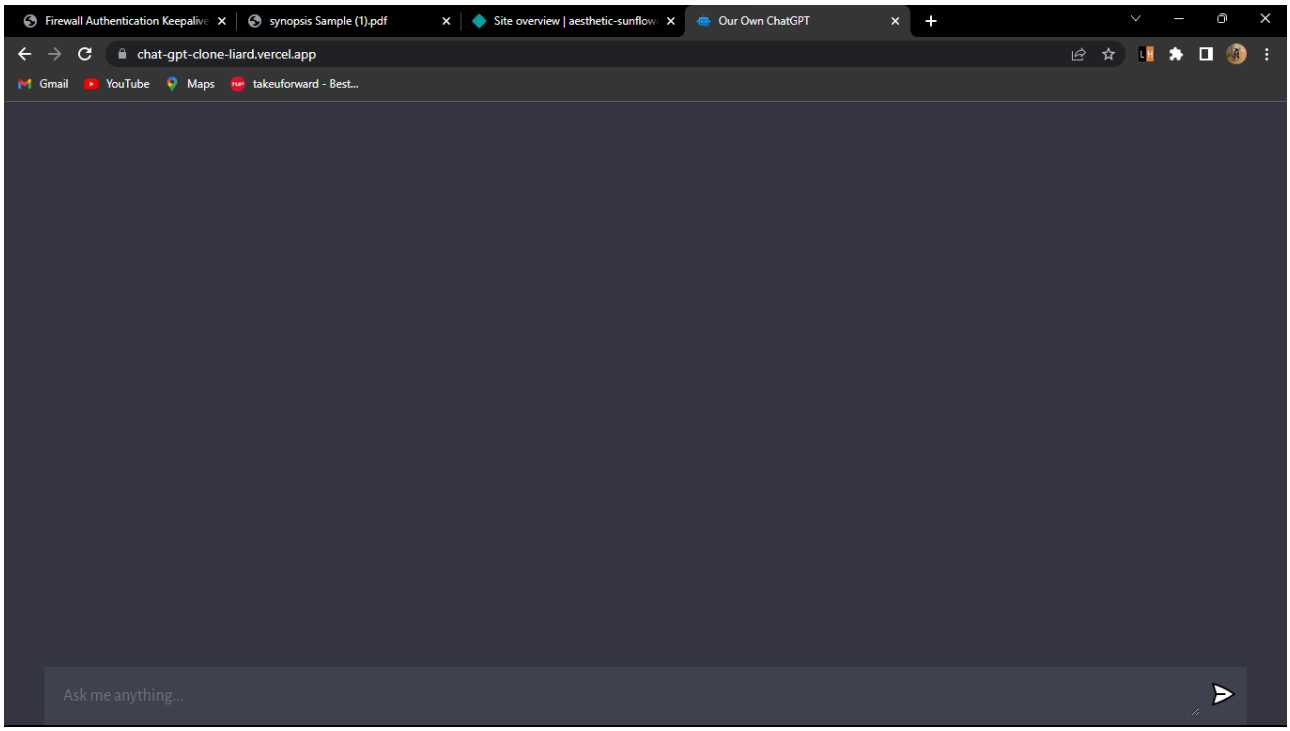
```
// HERE WE WILL USE THE LIBRARIES RELATED TO OPENAI
const { Configuration, OpenAIApi } = require('openai');

const configuration = new Configuration({
  apiKey: process.env.OPENAI_API_KEY,
});
const openai = new OpenAIApi(configuration);
// copied from openAI docs

const generateimage = async (req, res) => { //async because the open ai library is
going to give us promise
  const{prompt, size} = req.body;

  const imageSize = size === 'small' ? '256x256' : size === 'medium' ? '512x512' :
'1024x1024'


  try {
    const response = await openai.createImage({
      prompt,
      n: 1,  //number of images
      size: imageSize
    });

    const imageUrl = response.data.data[0].url;

    res.status(200).json({
      success: true,
      data: imageUrl
    });
  }

  catch (error) {
    if (error.response) {
      console.log(error.response.status);
      console.log(error.response.data);
    } else {
      console.log(error.message);
    }
    res.status(400).json({
      success: true,
      error: 'The image could not be genrated'
    });
  }
}

module.exports = {generateimage};
```

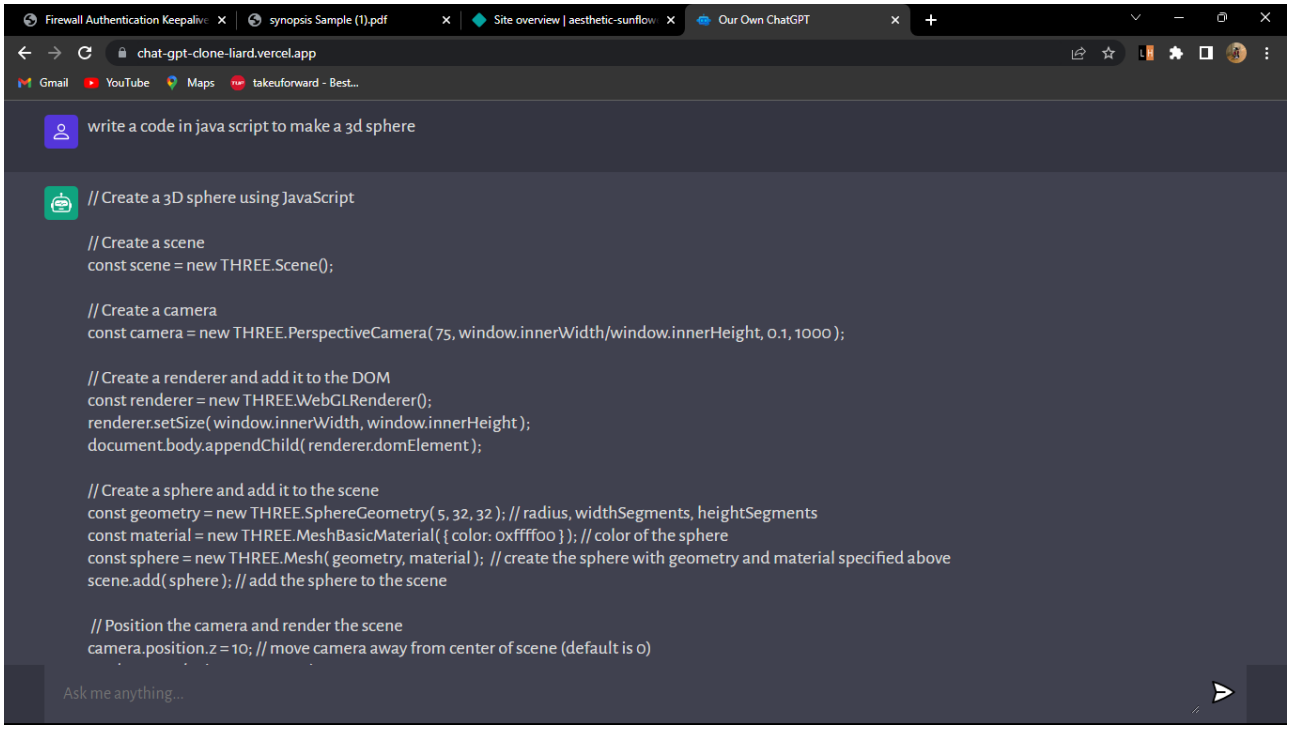
WORKING

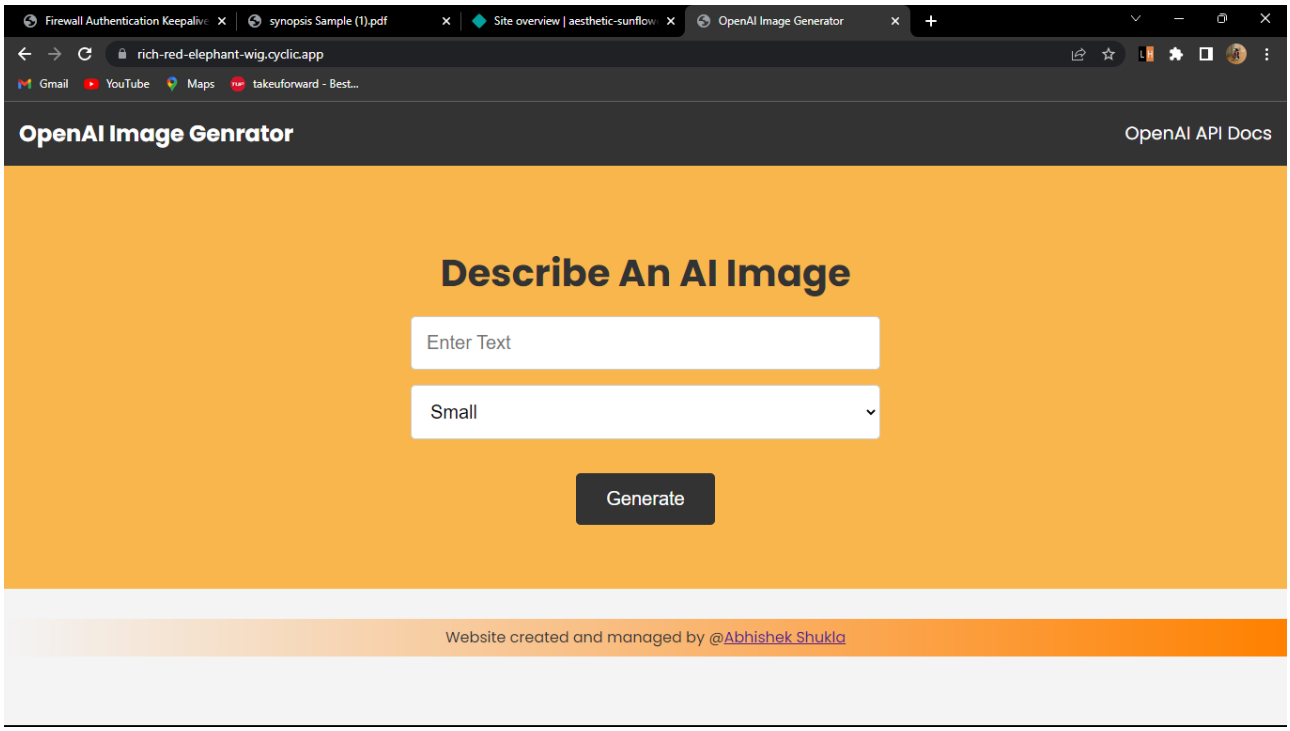
1. Website start with a index page where two buttons are given.
2. Then click on any button .
3. ChatGPt wil open ChatGPT clone .
4. AIMAGE will open the Ai image generator.
5. Search for any code on chatGPT.
6. Search for any image on AI image generator

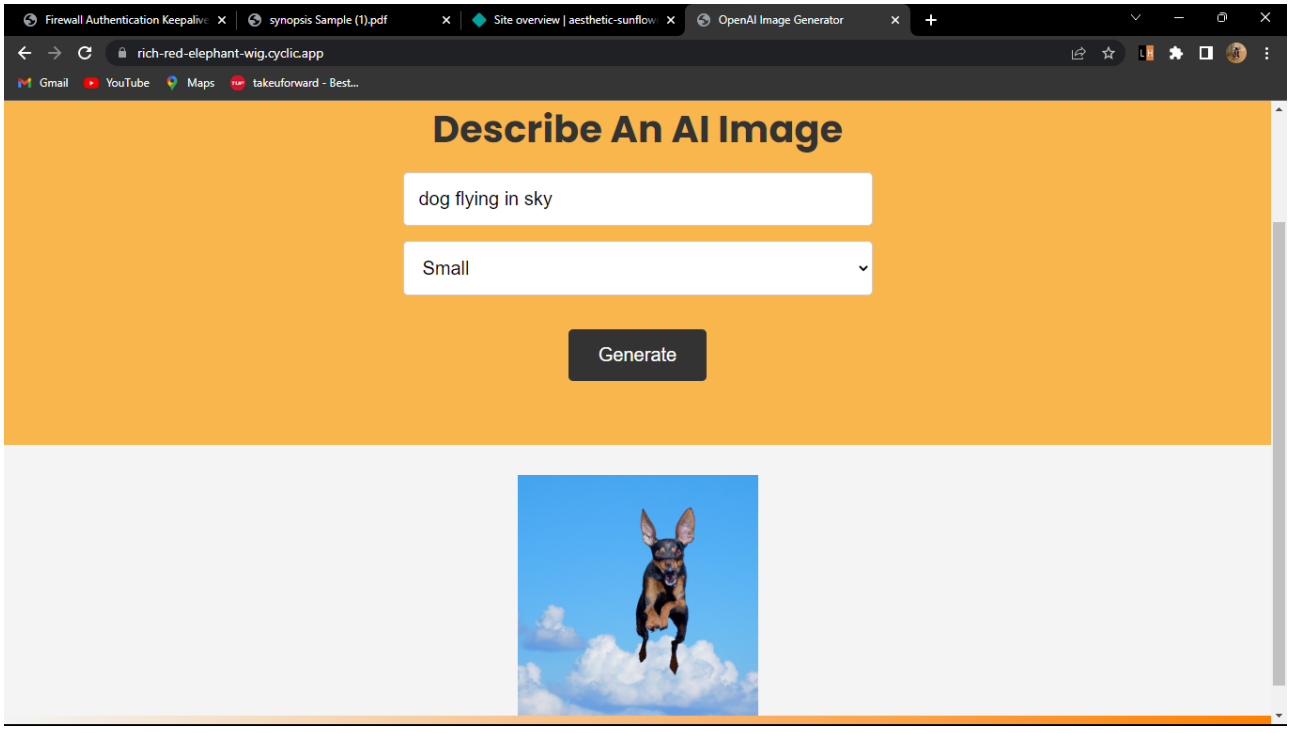
WEBSITE SCREENSHOOTS




Firewall Authentication Keepalive
synopsis Sample (1).pdf
Site overview | aesthetic-sunflow
Our Own ChatGPT
chat-gpt-clone-liard.vercel.app
Gmail
YouTube
Maps
takeuforward - Best...
Ask me anything...


Firewall Authentication Keepalive
synopsis Sample (1).pdf
Site overview | aesthetic-sunflow
Our Own ChatGPT
chat-gpt-clone-liard.vercel.app
Gmail
YouTube
Maps
takeuforward - Best...
write a code in java script to make a 3d sphere
// Create a 3D sphere using JavaScript
// Create a scene
const scene = new THREE.Scene();
// Create a camera
const camera = new THREE.PerspectiveCamera(75, window.innerWidth/window.innerHeight, 0.1, 1000);
// Create a renderer and add it to the DOM
const renderer = new THREE.WebGLRenderer();
renderer.setSize(window.innerWidth, window.innerHeight);
document.body.appendChild(renderer.domElement);
// Create a sphere and add it to the scene
const geometry = new THREE.SphereGeometry(5, 32, 32); // radius, widthSegments, heightSegments
const material = new THREE.MeshBasicMaterial({ color: 0xffff00 }); // color of the sphere
const sphere = new THREE.Mesh(geometry, material); // create the sphere with geometry and material specified above
scene.add(sphere); // add the sphere to the scene
// Position the camera and render the scene
camera.position.z = 10; // move camera away from center of scene (default is 0)
Ask me anything...


Firewall Authentication Keepalive
synopsis Sample (1).pdf
Site overview | aesthetic-sunflow
OpenAI Image Generator
rich-red-elephant-wig.cyclic.app
Gmail
YouTube
Maps
takeuforward - Best...
OpenAI Image Genrator
OpenAI API Docs
Describe An AI Image
Enter Text
Small
Generate
Website created and managed by @Abhishek Shukla


Firewall Authentication Keepalive
synopsis Sample (1).pdf
Site overview | aesthetic-sunflow
OpenAI Image Generator
rich-red-elephant-wig.cyclic.app
Gmail
YouTube
Maps
takeuforward - Best...
Describe An AI Image
dog flying in sky
Small
Generate

CONCLUSION AND FUTURE WORK

Conclusion:

For the ChatGPT clone project, the implementation of the GPT-3.5 architecture has enabled the creation of a highly intelligent and responsive language model. The model can understand natural language and provide relevant responses to various prompts. However, further improvements can be made by fine-tuning the model on specific domains or datasets to improve its accuracy and relevance.

As for the AI image generator project, the use of GANs has enabled the creation of highly realistic and convincing images that can be used for various applications such as product design, marketing, and gaming.

Future work:-

In terms of future work, there are several areas of improvement for both projects. For the ChatGPT clone, the model can be fine-tuned on specific domains such as finance, healthcare, or legal to provide more accurate and relevant responses in those areas. Additionally, incorporating multi-modal inputs such as audio and video can enhance the model's capabilities.

For the AI image generator project, the generator can be improved by incorporating more advanced techniques such as attention mechanisms, and exploring more advanced GAN architectures. Furthermore, developing methods for generating more diverse and varied images can also improve the overall quality of the images.

GITHUB:

https://github.com/Shuklaaa/MINIPROJECT

https://github.com/Shuklaaa/OpenAI-Image-Generator

https://github.com/Shuklaaa/ChatGPT-Clone

Website Is Live on:

https://aesthetic-sunflower-4d30ab.netlify.app/