AMBEDKAR INSTITUTE OF TECHNOLOGY

ISO 9001:2015 CERTIFIED INSTITUTE
Shakarpur, Near Nirman Vihar Metro Station, Delhi-110092.
Website- www.ambp.in

BCA DEPARTMENT

A Report on Major project

WEBSITE

SESSION: 2020-23

SUBMITTED TO:	**SUBMITTED BY:**
Mr Shivam Gupta	GAURAV BHANDARI **(020)**
(BCA DEPARTMENT)	UNNIKRISHNAN. V **(073)**

ACKNOWLEDGEMENTS

On this great occasion of accomplishment of our project on Website Designing (Tools AI), we would like to sincerely express our gratitude to **Mr.SHIVAM GUPTA** and **Mr.ABHISHEK GAUR** sir, who have been supported through the completion of this project.

We would also be thankful to our principal **Ms. NEERU MEHTA** for providing all the required facilities in completion of this project.

Finally, as one of the team members, I would like to appreciate all my group members for their support and coordination. I hope we will achieve more in our future endeavours.

Gaurav Bhandari

(02024802020)

Unnikrishnan. V

(07324802020)

Ambedkar DSEU Shakarpur Campus-1

Department of Bachelor of Computer Application

<u>CERTIFICATE</u>

This is to Certify that the minor project has been successfully completed by **Gaurav Bhandari (02024802020) and Unnikrishnan V (07240802020)** who have carried out the project work under my supervision.I approve this Major project

for submission

Prof.Shivam Gupta

Proctor BCA Department

TOOLS AI

-- A collection of some useful tools facilitated by AI technologies under a single website --

Table of content

INTRODUCTION

Welcome to the future of work! With the introduction of Power AI tools, we're taking a giant leap forward in how we approach our daily tasks. These powerful tools are designed to make our lives easier, more efficient, and more productive than ever before.

Imagine being able to generate stunning visuals with just a few clicks, or streamlining your coding process to save hours of time. With Power AI tools, these possibilities are now a reality. So let's dive in and explore the exciting world of Power AI tools!

This project is an attempt to make a utility website which contains some necessary daily life tools that will make our daily digital life easier and convenient.

It has four functions, the **Image Generator**, the **AI TRANSLATOR**, the **SQL generator** and **Code GPT**.

- ***Image Generator*** : This tool is used to create AI generated images based on user provided prompts/description. The images are fetched using OpenAI's Dall-E API.

- ***AI Translator*** : This tool is used to translate a given text to another language using ChatGPT API. It has an input and an output field. We enter our text in the input field and then select the language and hit the translate button. The app will generate the output on the right side.

- ***SQL Generator*** : This tool is used to convert english statements into SQL queries using AI Models by ChatGPT. It has an input section where the user is supposed to give the statement for which the app will generate the SQL query. OpenAI's ChatGPT API is used to generate the SQL query.

- ***Code GPT*** : This tool is used to facilitate many programmers, developers and software engineers. This tool helps in understanding a piece a code. When we give it a chunk of code and submit it. The ChatGPT model will give an explanation to the code.

WHY IT WAS NEEDED?

In today's interconnected world, translator, image generator, and SQL generator tools have emerged as vital components of various domains. Translator tools, like machine translation systems, break down language barriers and enable effective cross-cultural communication, facilitating collaboration and fostering cultural understanding. Image generator tools leverage advanced technologies to automate image creation, transforming creative industries and enhancing visual content creation in fields such as design, advertising, and virtual reality. SQL generator tools simplify complex database operations, improving productivity and optimizing performance for software developers, data analysts, and database administrators. These tools have become indispensable in our increasingly digital society, providing convenience, efficiency, and innovation. From facilitating international business transactions to empowering artists and designers to realize their creative visions, and from streamlining database management to extracting valuable insights, translator, image generator, and SQL generator tools play a vital role in enhancing communication, creativity, and data management. As technology continues to advance, these tools are poised to have an even greater impact, revolutionizing how we interact with language, visuals, and data.

Features OF TOOLS AI

- ***Image Generator:*** *Empowers users to automatically create or manipulate visually appealing images, graphics, and visual effects with advanced AI and deep learning techniques.*

- ***Chat AI:*** *Provides conversational capabilities and natural language understanding, enabling interactive and dynamic interactions with users, answering questions, and engaging in dialogue.*

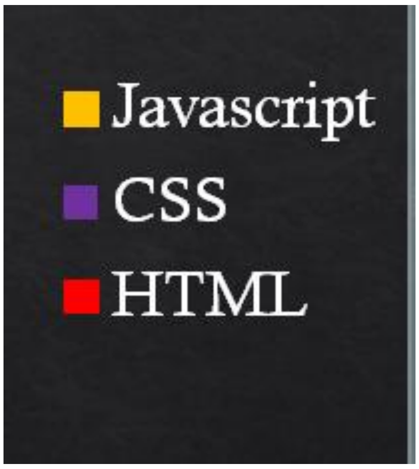
- ***SQL Generator:*** *Automates the generation of SQL queries, simplifying complex database operations and enhancing productivity and performance in managing and interacting with relational databases.*

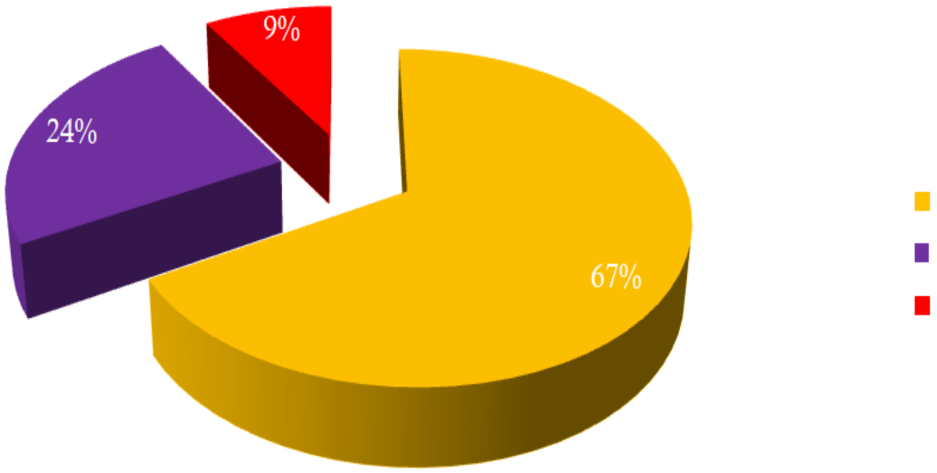
- ***Code GPT:*** *Assists developers in generating code snippets, providing coding suggestions, and offering programming assistance, streamlining the software development process and promoting efficiency and accuracy in coding tasks.*

SOFTWARE REQUIREMENTS

OPERATING SYSTEM :-	**RUN ON EVERY OS**
DESIGN :-	**FIGMA**
FRONT END :-	**REACT.JS**
BACK END :-	**NODE.JS**
IDE USED :-	**VISUAL STUDIO CODE**
PACKAGE MANAGER :-	**NODE PACKAGE MANAGER (NPM)**
PACKAGES USED :-	**CORS, EXPRESS, DOTENV, JSON**

LANGUAGES USED





<u>Technologies Used</u>

React.js

The React. js framework is an open-source JavaScript framework and library developed by Facebook.

It's used for building interactive user interfaces and web applications quickly and efficiently with significantly less code than you would with vanilla JavaScript

The website's frontend interface is built using React library. React makes it simple to use component based architecture which makes it easier to design independent components in the frontend

Node.js

Node. js (Node) is an open source, cross-platform runtime environment for executing JavaScript code.

Node is used extensively for server-side programming, making it possible for developers to use JavaScript for client-side and server-side code without needing to learn an additional language.

The website's backend is built using Node.js. Node has many in built functions for doing many backend operations in a simple way. Node.js helps us to understand and code many complex functions in the backend.

Express.js

Express is a node js web application framework that provides broad features for building web and mobile applications.

It is used **to build a single page, multipage, and hybrid web application**. It's a layer built on the top of the Node js that helps manage servers and routes.

Bootstrap

What is Bootstrap used for?

Bootstrap is **a free, open source front-end development framework for the creation of websites and web apps**.

Designed to enable responsive development of mobile-first websites, Bootstrap provides a collection of syntax for template designs.

Most of the frontend components are made using bootstrap templates which makes it easier to design components even faster and customise according to your choice

APIs Used

API (Application Programming Interface)

Application Programming Interface (API) is a software interface that allows two applications to interact with each other without any user intervention. API is a collection of software functions and procedures.

API is defined as a code that helps two different software's to communicate and exchange data with each other.

We have used these APIs in our project some of them are-----

OpenAI API

OpenAI is the same company who created the infamous ChatGPT. OpenAI has also created various APIs for developers based on their ChatGPT, Dall-E and Whisper models which generated AI responses for texts, images, audio etc. In this project we have used OpenAI's ChatGPT and Dall-E API's for image generation and text responses.

Packages Used

DotEnv

Dotenv is a zero-dependency module that loads environment variables from a .env file into **process.env**. Storing configuration in the environment separate from code is based on **The Twelve-Factor App** methodology.

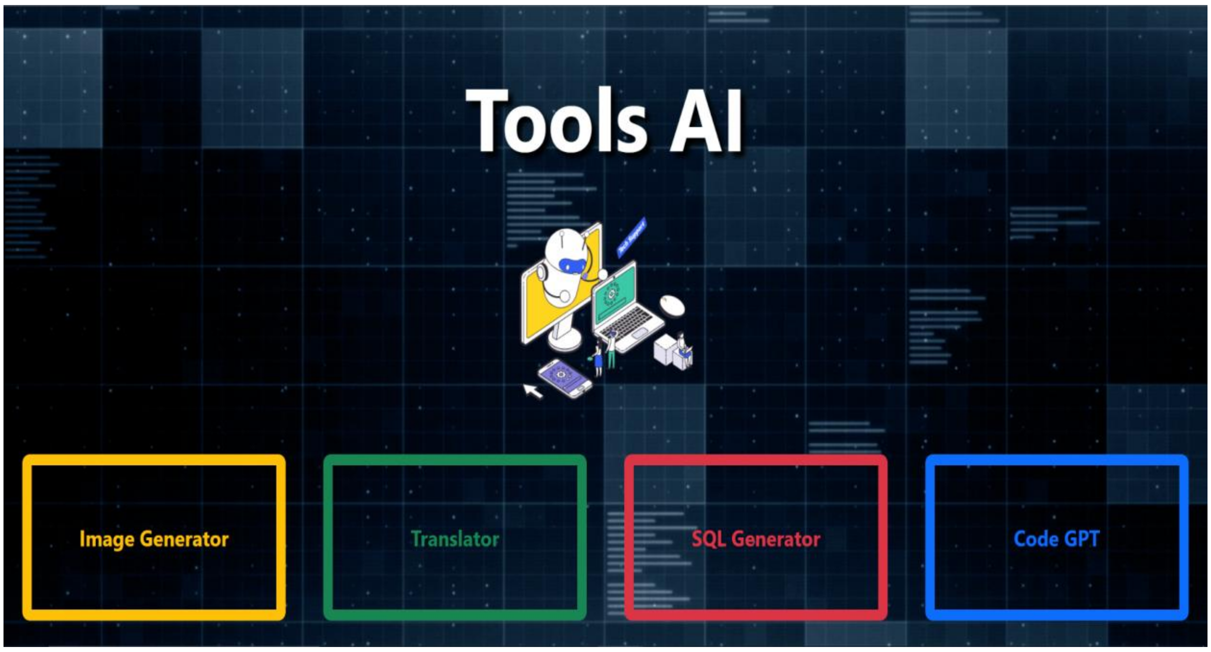
Cors

Today, there are many applications that depend on APIs to access different resources. Some of the popular APIs include weather, time, and fonts. There are also servers that host these APIs and ensure that information is delivered to websites and other end points. Therefore, making cross-origin calls, is a popular use case for the modern web application.

Let’s say accessing images, videos, iframes, or scripts from another server. This means that the website is accessing resources from a different origin or domain. When building an application to serve up these resources with Express, a request to such external origins may fail. This is where CORS comes in to handle cross-origin requests.

HOME PAGE

This is how the home page of our website looks like



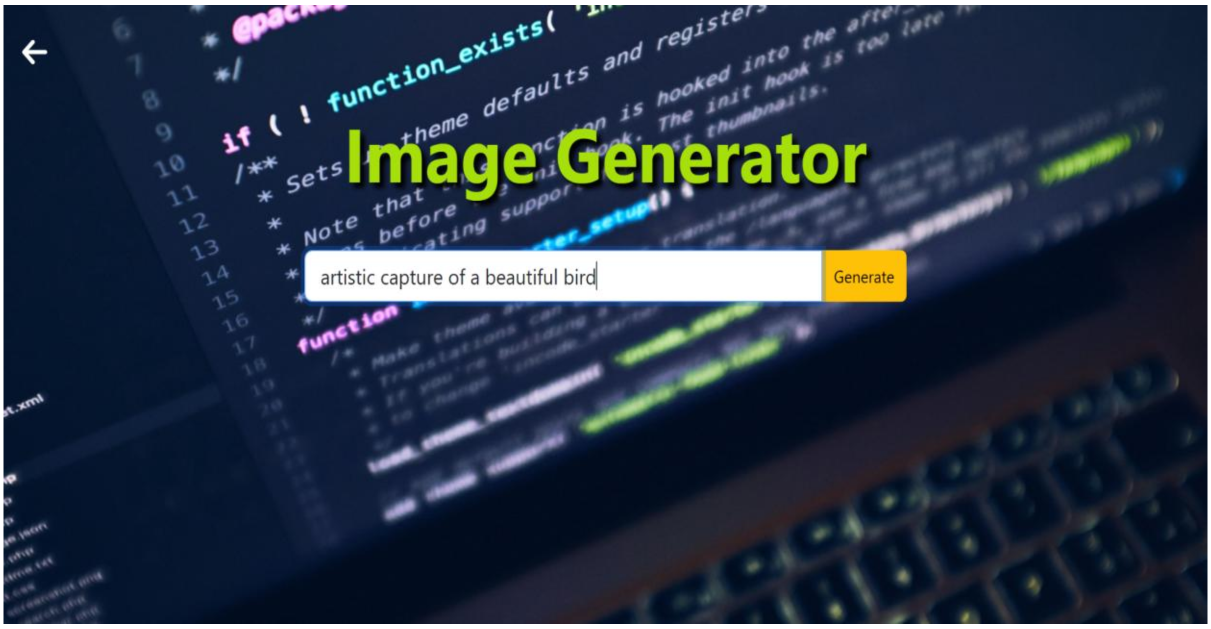
HOW IT WORKS

This Website comprises of 4 convenient tools :

IMAGE GENERATOR

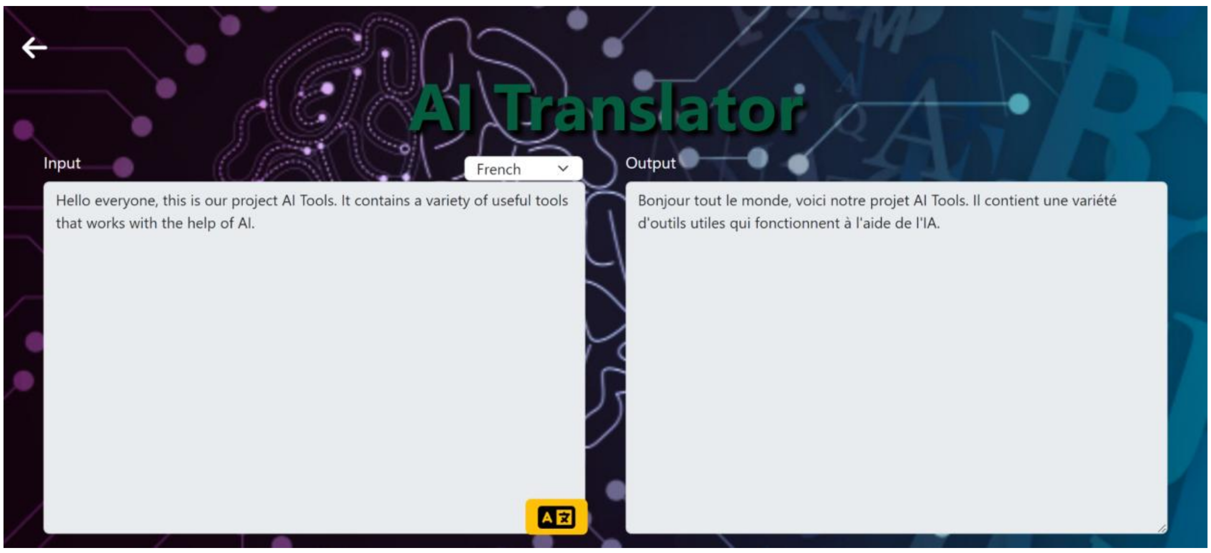
As the name suggests this tools helps in fetching AI generated images based on the prompts given.

We have to give the prompt in the input field and press the Generate button. In a few seconds the tool will generate various images based on the prompt given (currently only four images will be generated because we are using the APIs free tier).



AI TRANSLATOR

This tool is an AI translator. When we give it an input and then select the language and hit submit, it will generate the translation in the provided language.



Hello everyone, this is our project AI Tools. It contains a variety of useful tools that works with the help of AI.

Bonjour tout le monde, voici notre projet AI Tools. Il contient une variété d'outils utiles qui fonctionnent à l'aide de l'IA.

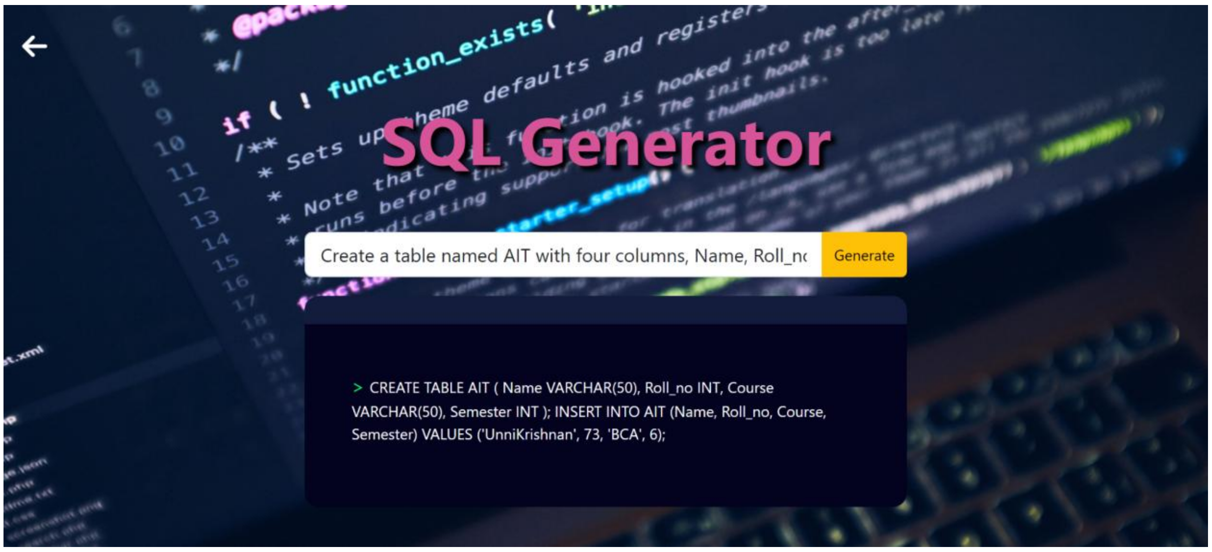
SQL GENERATOR

This is a tool which can be very handy and useful for students and Software developers. This tool can translate your english statements to SQL queries which will be accurate and useful for people who face difficulty in remembering SQL queries.

We have to give the statement in the input section and it will generate an SQL query based on that.

Statement : "Create a table named AIT with four columns, Name, Roll_no, Course and Semester and add a value UnniKrishnan, 73, BCA, 6 to it"

Output :

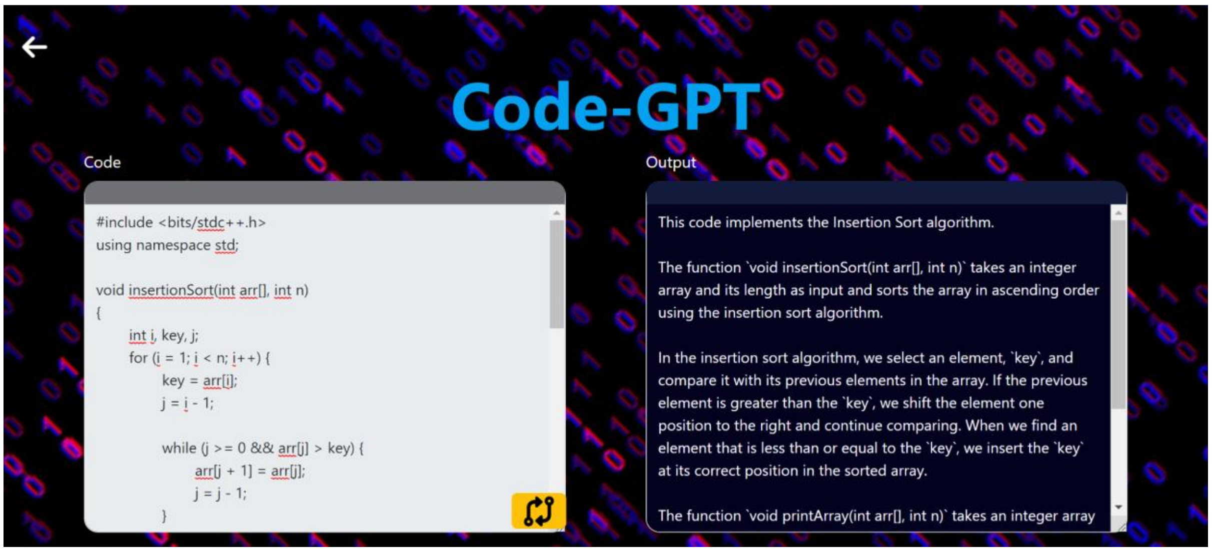


```
> CREATE TABLE AIT ( Name VARCHAR(50), Roll_no INT, Course
VARCHAR(50), Semester INT ); INSERT INTO AIT (Name, Roll_no, Course,
Semester) VALUES ('UnniKrishnan', 73, 'BCA', 6);
```

CODE GPT

This is a tool that will facilitate many developers in understanding a piece of code. This will generate an AI explanation of the given piece of code.

We have to just copy paste the code in the input section and hit submit. In a few seconds it will generate the explanation of the code.



```
#include <bits/stdc++.h>
using namespace std;

void insertionSort(int arr[], int n)
{
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;

        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
```

This code implements the Insertion Sort algorithm.

The function `void insertionSort(int arr[], int n)` takes an integer array and its length as input and sorts the array in ascending order using the insertion sort algorithm.

In the insertion sort algorithm, we select an element, `key`, and compare it with its previous elements in the array. If the previous element is greater than the `key`, we shift the element one position to the right and continue comparing. When we find an element that is less than or equal to the `key`, we insert the `key` at its correct position in the sorted array.

The function `void printArray(int arr[], int n)` takes an integer array

SOURCE CODE

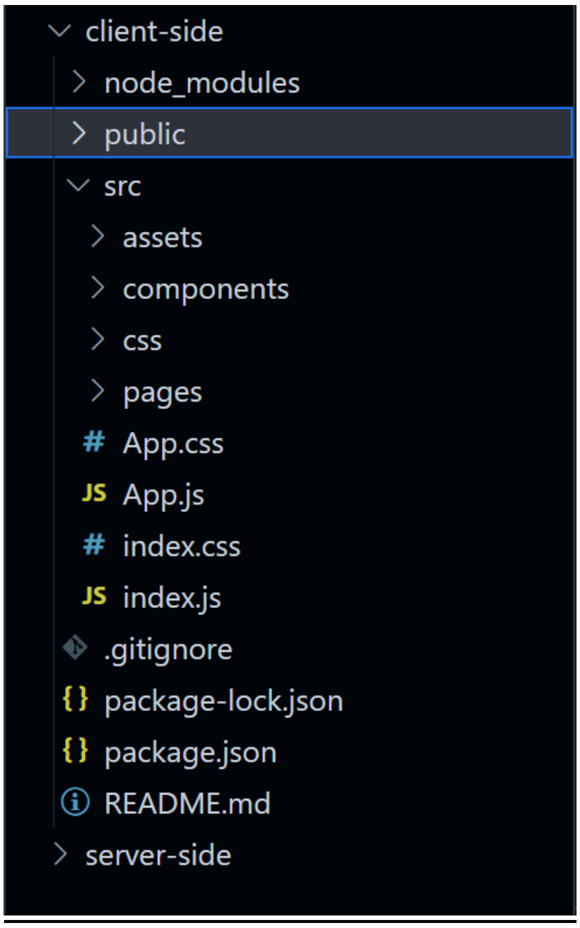
Our project is mainly divided into two sections - Client side and Server side.

Client side

In the client side we have done all the frontend design. All the frontend code elements are in this section and this section communicates with the backend data and populates the data in the frontend. We have used a Javascript framework called React.js to write all the design code. React.js uses components based architecture for the application. React helps in writing the code quickly and efficiently. It has a predefined template and a clear folder structure to be worked upon.

Component based architecture is a process where we create various components in the website and store it in separate files. This method becomes very useful when we have to style every component differently and independently. Every component is stored in separate files for easy access and easy error detection. We can easily find in which files are the error and fix them quickly. All the components are imported to the main page file of the website. Using this approach is also useful when you don’t have to write all your code in a single file

FOLDER STRUCTURE



```
client-side
  node_modules
  public
  src
    assets
    components
    css
    pages
    App.css
    App.js
    index.css
    index.js
  .gitignore
  package-lock.json
  package.json
  README.md
server-side
```

The above image shows the folder structure of the client folder. The client side folder is mainly divided into two subfolders which are public and src. In the public folder, the main file is index.html. index.html is the main entry point of the website where our javascript files are mounted. Now next comes the source folder which is the main section of the client side where all our source code of the frontend is written. In source folder we have 4 subfolders - assets, components, css and pages. The assets folder contains all the custom fonts, images, icons etc. The component folder has all the components used in the website. Components are independent elements which are stored in separate files for the ease of writing code and accessibility. In the Css folder we have written all the custom css codes of the components and pages. In the last folder which is the pages folder we have stored all the pages in our website in separate files and the components are imported into these pages and shown in the output. Apart from the subfolders you can see some extra files like index.js, app.js etc. These files are nothing but the inbuilt files by React.

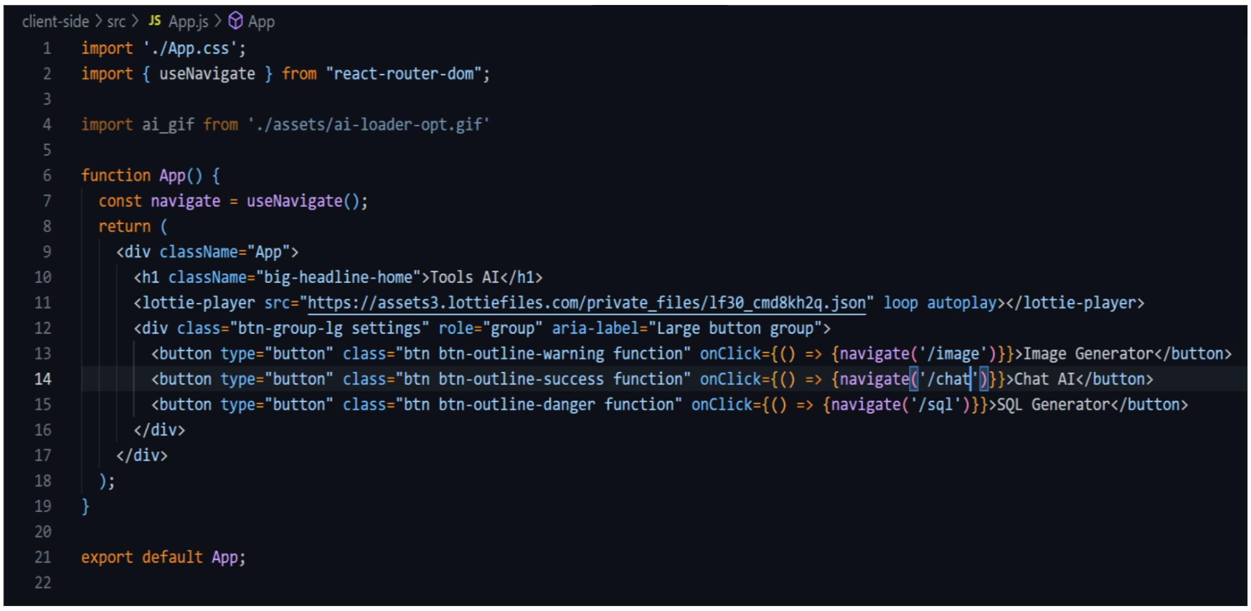
CODE

Let's understand the different pages and components of the website. The website has fiver main pages, Home, SQL, Image, Translate and Code-GPT. All these pages carry out different functions.

Home page

The Home page consists of all the tools at one place making it easier for navigation along the website.

The code of the Home page is very simple and easy to understand which has the job of linking all the other pages to the website :



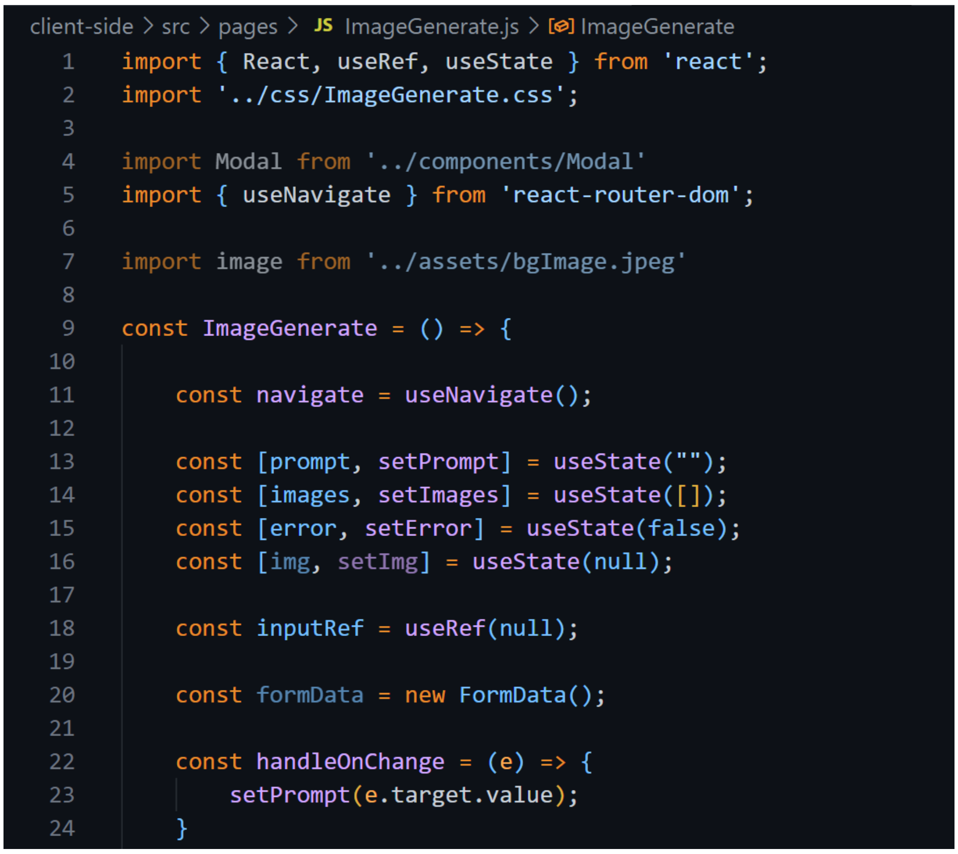
```
import './App.css';
import { useNavigate } from "react-router-dom";

import ai_gif from './assets/ai-loader-opt.gif'

function App() {
  const navigate = useNavigate();
  return (
    <div className="App">
      <h1 className="big-headline-home">Tools AI</h1>
      <lottie-player src="https://assets3.lottiefiles.com/private_files/lf30_cmd8kh2q.json" loop autoplay></lottie-player>
      <div class="btn-group-lg settings" role="group" aria-label="Large button group">
        <button type="button" class="btn btn-outline-warning function" onClick={() => {navigate('/image')}}>Image Generator</button>
        <button type="button" class="btn btn-outline-success function" onClick={() => {navigate('/chat')}}>Chat AI</button>
        <button type="button" class="btn btn-outline-danger function" onClick={() => {navigate('/sql')}}>SQL Generator</button>
      </div>
    </div>
  );
}

export default App;
```

We have used Bootstrap Button component and then we have added custom css to make it attractive. The Lotties animation library is used to add the animation at the home page

Image Generation Page

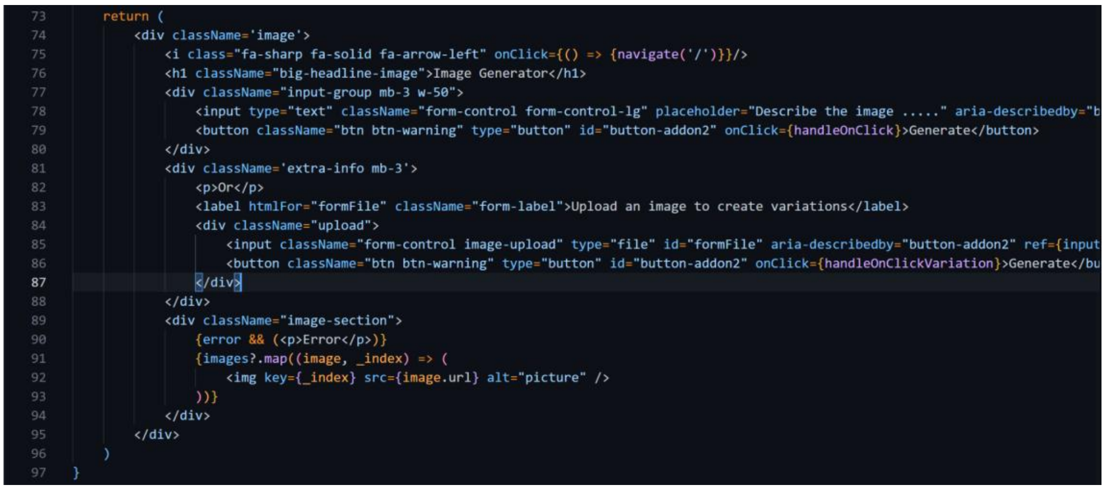
This page as mentioned before helps us to generate AI generated images based on prompts.

The code of the Image generation page is given below :

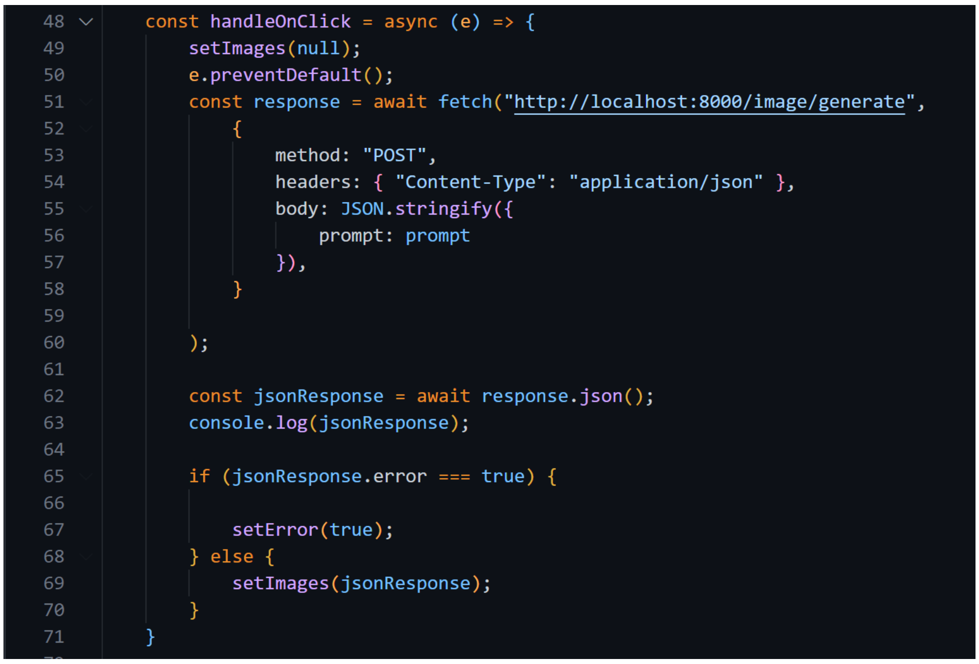


client-side > src > pages > JS ImageGenerate.js > ImageGenerate

```
import { React, useRef, useState } from 'react';
import '../css/ImageGenerate.css';

import Modal from '../components/Modal'
import { useNavigate } from 'react-router-dom';

import image from '../assets/bgImage.jpeg'

const ImageGenerate = () => {

    const navigate = useNavigate();

    const [prompt, setPrompt] = useState("");
    const [images, setImages] = useState([]);
    const [error, setError] = useState(false);
    const [img, setImg] = useState(null);

    const inputRef = useRef(null);

    const formData = new FormData();

    const handleOnChange = (e) => {
        setPrompt(e.target.value);
    }
```



```
    return (
        <div className='image'>
            <i class="fa-sharp fa-solid fa-arrow-left" onClick={() => {navigate('/')}}/>
            <h1 className="big-headline-image">Image Generator</h1>
            <div className="input-group mb-3 w-50">
                <input type="text" className="form-control form-control-lg" placeholder="Describe the image ....." aria-describedby="b
                <button className="btn btn-warning" type="button" id="button-addon2" onClick={handleOnClick}>Generate</button>
            </div>
            <div className='extra-info mb-3'>
                <p>Or</p>
                <label htmlFor="formFile" className="form-label">Upload an image to create variations</label>
                <div className="upload">
                    <input className="form-control image-upload" type="file" id="formFile" aria-describedby="button-addon2" ref={input
                    <button className="btn btn-warning" type="button" id="button-addon2" onClick={handleOnClickVariation}>Generate</bu
                </div>
            </div>
            <div className="image-section">
                {error && (<p>Error</p>)}
                {images?.map((image, _index) => (
                    <img key={_index} src={image.url} alt="picture" />
                ))}
            </div>
        </div>
    )
}
```

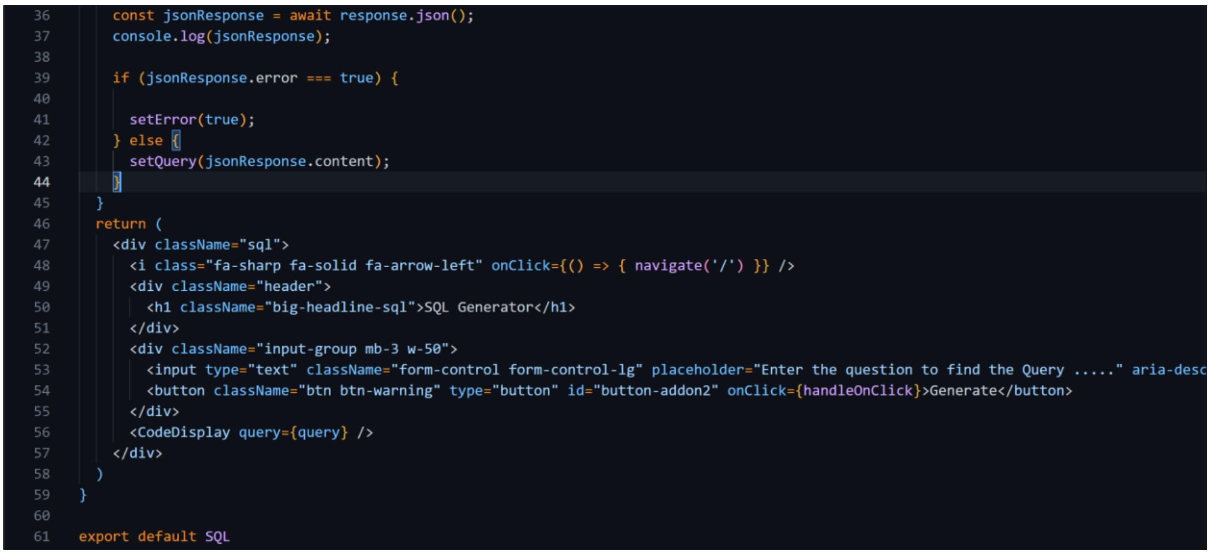


```
const handleOnClick = async (e) => {
    setImages(null);
    e.preventDefault();
    const response = await fetch("http://localhost:8000/image/generate",
        {
            method: "POST",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify({
                prompt: prompt
            }),
        }

    );

    const jsonResponse = await response.json();
    console.log(jsonResponse);

    if (jsonResponse.error === true) {

        setError(true);
    } else {
        setImages(jsonResponse);
    }
}
```

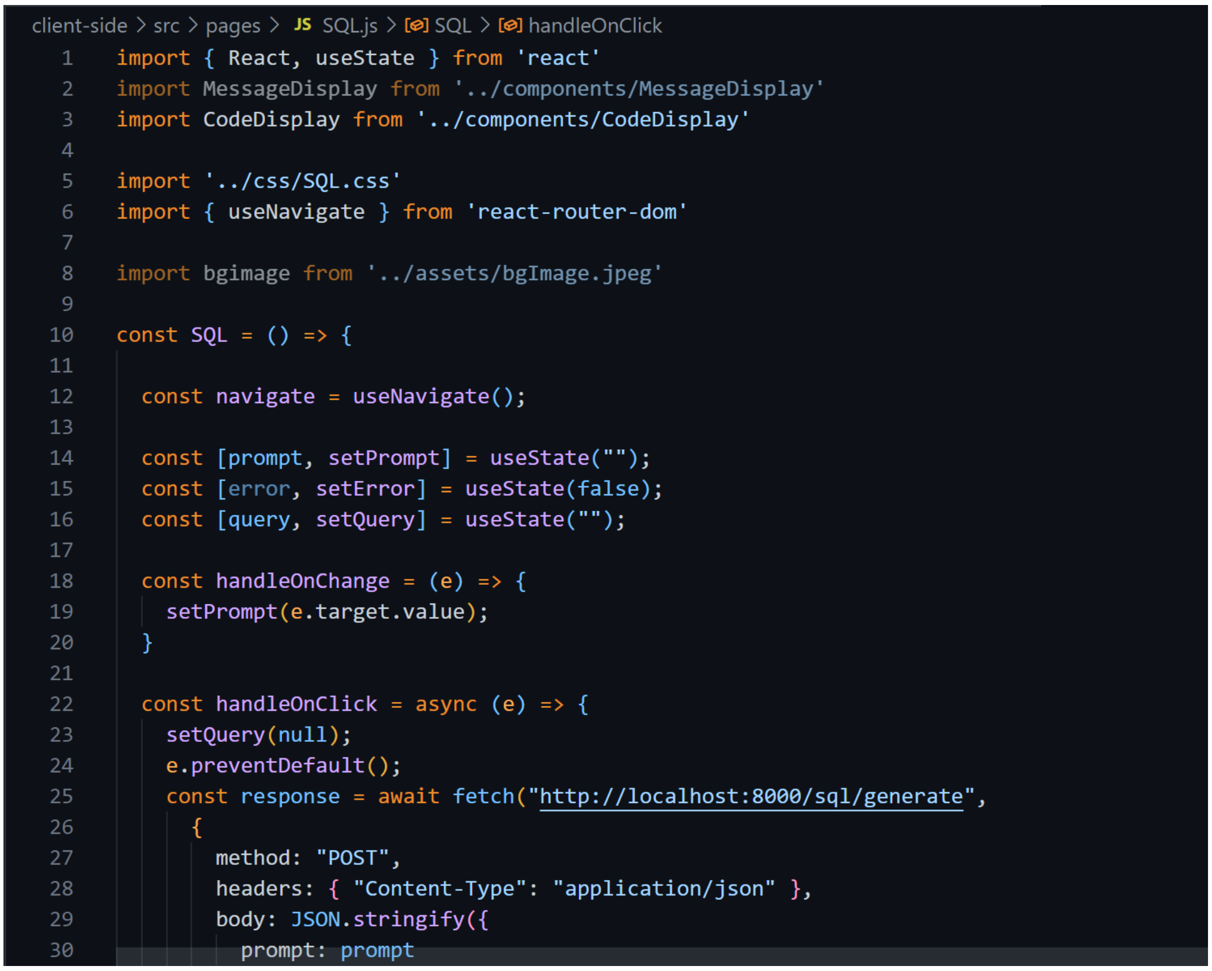
SQL Genrator Page

This page as mentioned before helps us to generate SQL Queries based on the statements.

The code of the Image generation page is given below :



```
    const jsonResponse = await response.json();
    console.log(jsonResponse);

    if (jsonResponse.error === true) {

      setError(true);
    } else {
      setQuery(jsonResponse.content);
    }
  }
  return (
    <div className="sql">
      <i class="fa-sharp fa-solid fa-arrow-left" onClick={() => { navigate('/') }} />
      <div className="header">
        <h1 className="big-headline-sql">SQL Generator</h1>
      </div>
      <div className="input-group mb-3 w-50">
        <input type="text" className="form-control form-control-lg" placeholder="Enter the question to find the Query ....." aria-desc
        <button className="btn btn-warning" type="button" id="button-addon2" onClick={handleOnClick}>Generate</button>
      </div>
      <CodeDisplay query={query} />
    </div>
  )
}

export default SQL
```


client-side > src > pages > JS SQL.js > [ø] SQL > [ø] handleOnClick

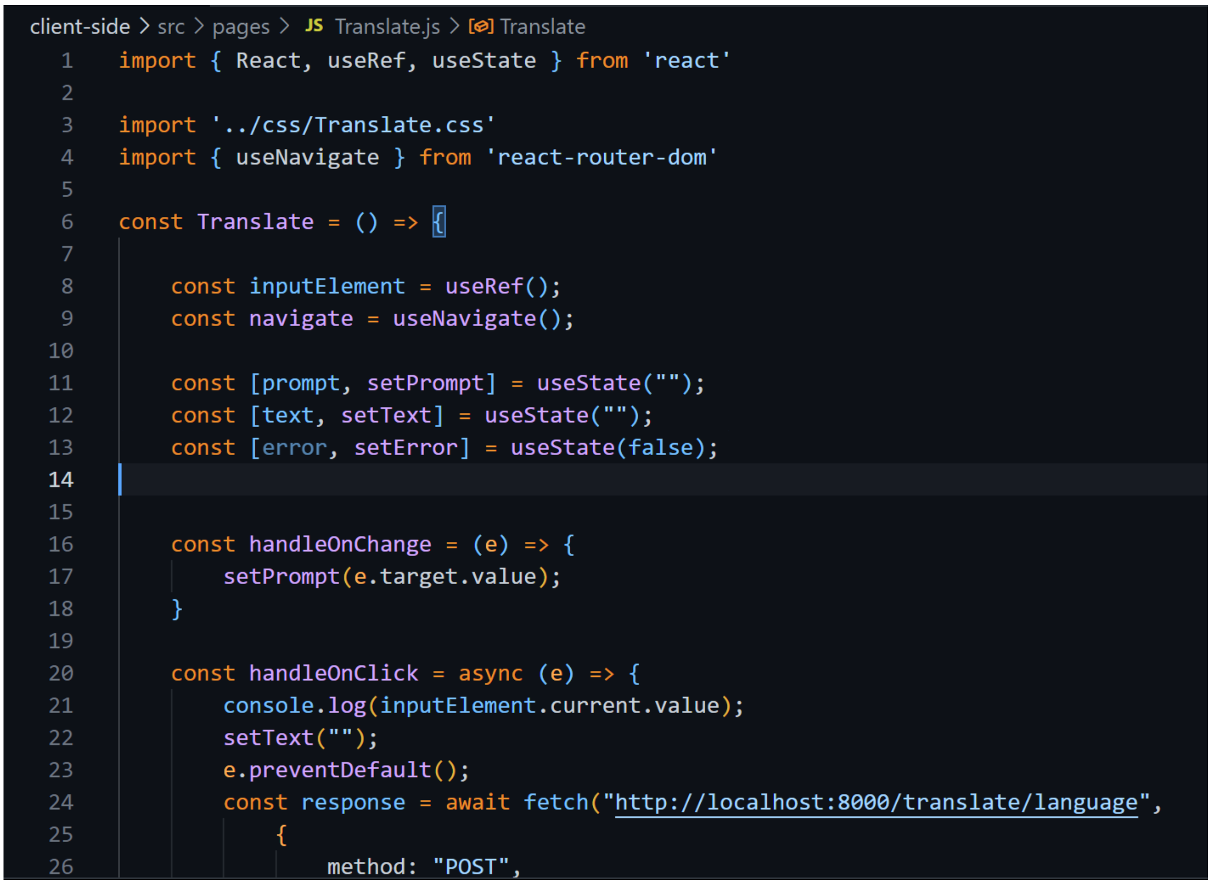
```javascript
import { React, useState } from 'react'
import MessageDisplay from '../components/MessageDisplay'
import CodeDisplay from '../components/CodeDisplay'

import '../css/SQL.css'
import { useNavigate } from 'react-router-dom'

import bgimage from '../assets/bgImage.jpeg'

const SQL = () => {

  const navigate = useNavigate();

  const [prompt, setPrompt] = useState("");
  const [error, setError] = useState(false);
  const [query, setQuery] = useState("");

  const handleOnChange = (e) => {
    setPrompt(e.target.value);
  }

  const handleOnClick = async (e) => {
    setQuery(null);
    e.preventDefault();
    const response = await fetch("http://localhost:8000/sql/generate",
      {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({
          prompt: prompt
```

This page has further components like CodeDisplay which helps to create the output

<u>Translator Page</u>

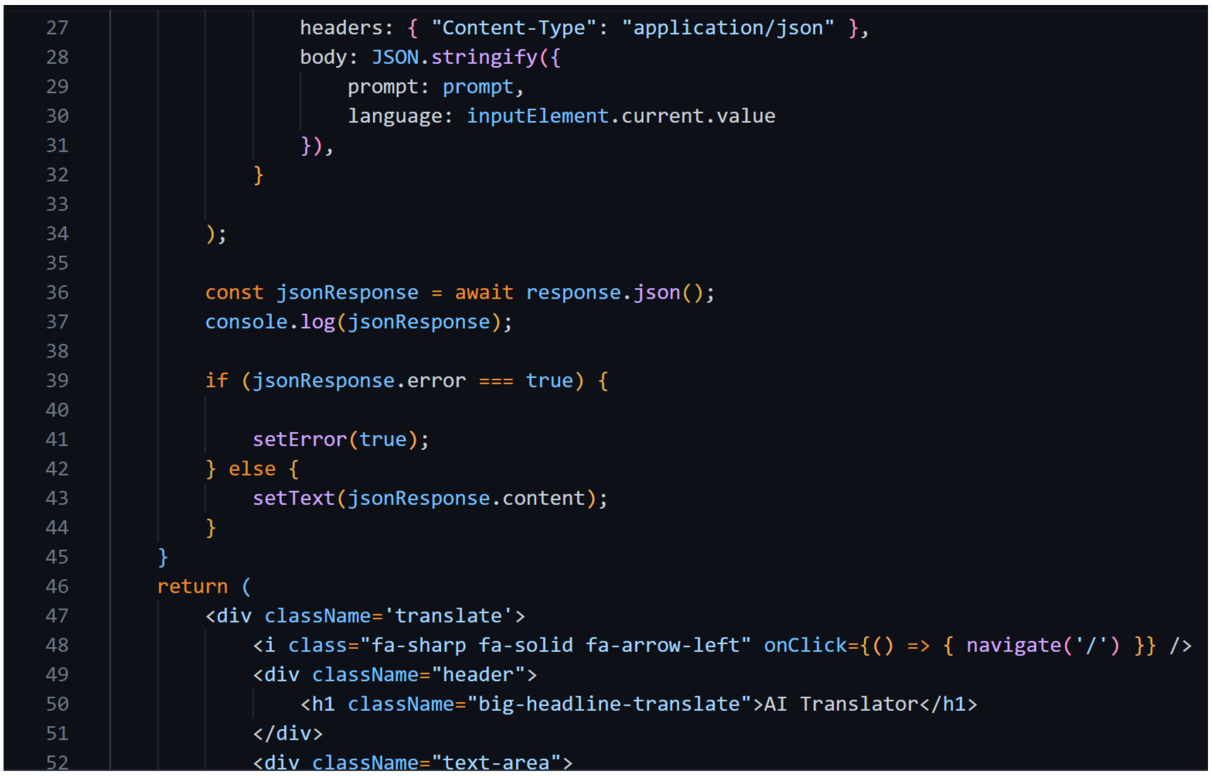
This page as mentioned above will translate any text to another language. The language will be provided by the user.

The code of the Image generation page is given below :



client-side > src > pages > JS Translate.js > Translate

```
import { React, useRef, useState } from 'react'

import '../css/Translate.css'
import { useNavigate } from 'react-router-dom'

const Translate = () => {

    const inputElement = useRef();
    const navigate = useNavigate();

    const [prompt, setPrompt] = useState("");
    const [text, setText] = useState("");
    const [error, setError] = useState(false);


    const handleOnChange = (e) => {
        setPrompt(e.target.value);
    }

    const handleOnClick = async (e) => {
        console.log(inputElement.current.value);
        setText("");
        e.preventDefault();
        const response = await fetch("http://localhost:8000/translate/language",
            {
                method: "POST",
```

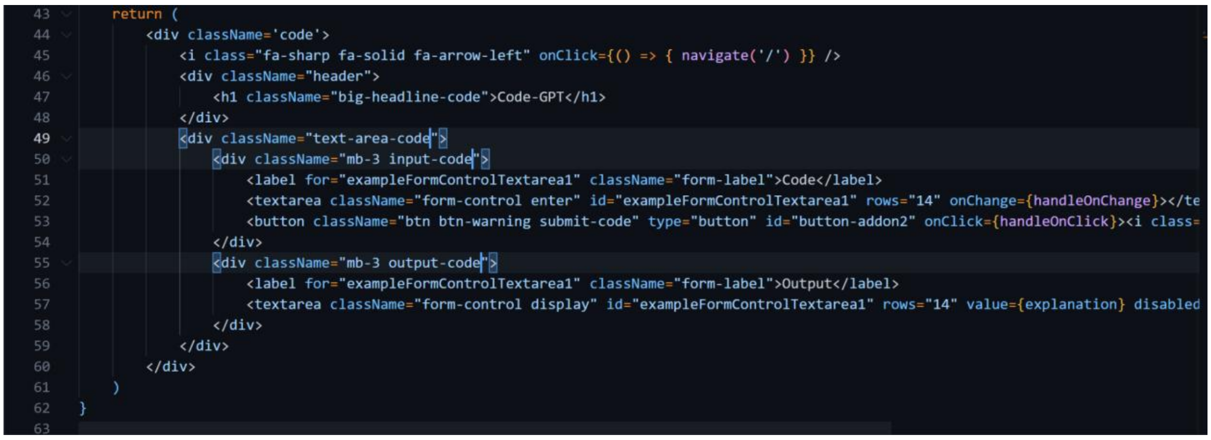


```
                headers: { "Content-Type": "application/json" },
                body: JSON.stringify({
                    prompt: prompt,
                    language: inputElement.current.value
                }),
            }

        );

        const jsonResponse = await response.json();
        console.log(jsonResponse);

        if (jsonResponse.error === true) {

            setError(true);
        } else {
            setText(jsonResponse.content);
        }
    }
    return (
        <div className='translate'>
            <i class="fa-sharp fa-solid fa-arrow-left" onClick={() => { navigate('/') }} />
            <div className="header">
                <h1 className="big-headline-translate">AI Translator</h1>
            </div>
            <div className="text-area">
```

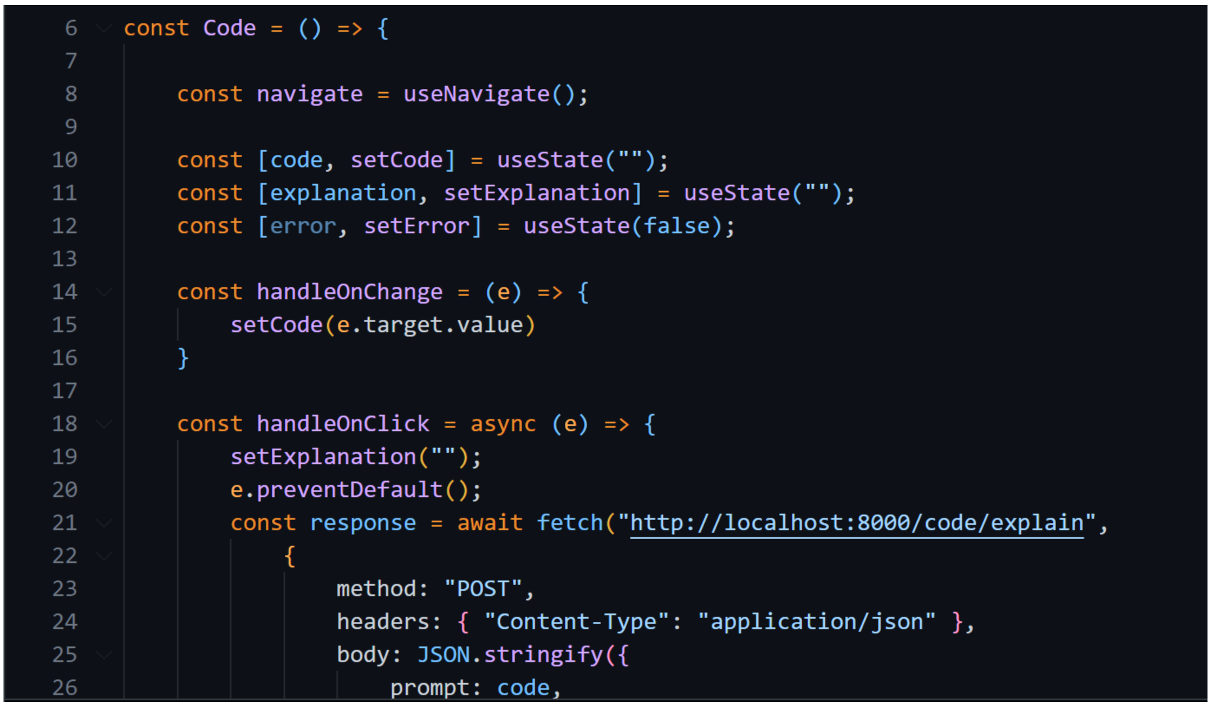
Code GPT Page

This page as mentioned above will generate an AI explanation describing the provided code sample.

The code of the Image generation page is given below :



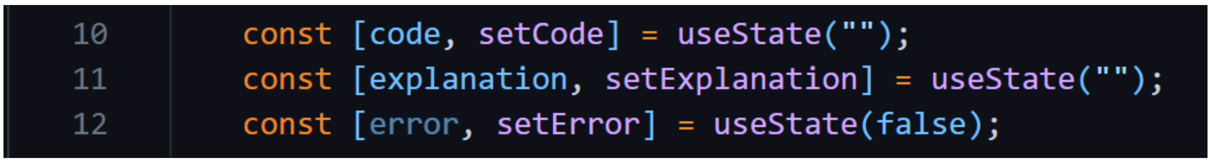
```
    return (
        <div className='code'>
            <i class="fa-sharp fa-solid fa-arrow-left" onClick={() => { navigate('/') }} />
            <div className="header">
                <h1 className="big-headline-code">Code-GPT</h1>
            </div>
            <div className="text-area-code">
                <div className="mb-3 input-code">
                    <label for="exampleFormControlTextarea1" className="form-label">Code</label>
                    <textarea className="form-control enter" id="exampleFormControlTextarea1" rows="14" onChange={handleOnChange}></te
                    <button className="btn btn-warning submit-code" type="button" id="button-addon2" onClick={handleOnClick}><i class=
                </div>
                <div className="mb-3 output-code">
                    <label for="exampleFormControlTextarea1" className="form-label">Output</label>
                    <textarea className="form-control display" id="exampleFormControlTextarea1" rows="14" value={explanation} disabled
                </div>
            </div>
        </div>
    )
}
```



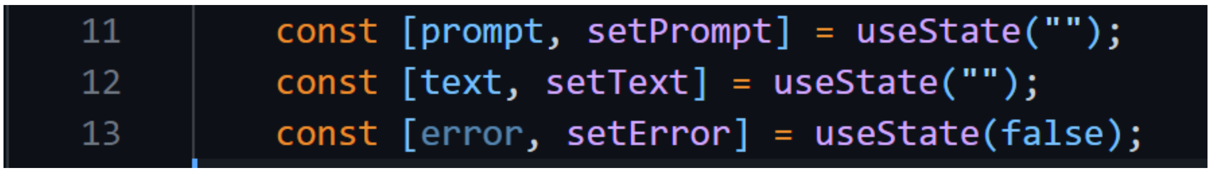
```
const Code = () => {

    const navigate = useNavigate();

    const [code, setCode] = useState("");
    const [explanation, setExplanation] = useState("");
    const [error, setError] = useState(false);

    const handleOnChange = (e) => {
        setCode(e.target.value)
    }

    const handleOnClick = async (e) => {
        setExplanation("");
        e.preventDefault();
        const response = await fetch("http://localhost:8000/code/explain",
            {
                method: "POST",
                headers: { "Content-Type": "application/json" },
                body: JSON.stringify({
                    prompt: code,
```

Main functions in the application

For the main operations we have used useStates to store variable data's . useState are a concept of React, this function is used to update a variable data during the runtime.



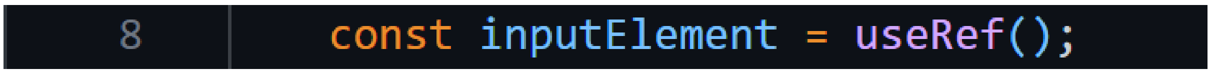
```
    const [code, setCode] = useState("");
    const [explanation, setExplanation] = useState("");
    const [error, setError] = useState(false);
```



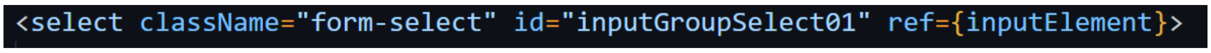
```
    const [prompt, setPrompt] = useState("");
    const [text, setText] = useState("");
    const [error, setError] = useState(false);
```

These are the useStates we have used in the application.

Further we have used React's useRef hook to get the value of a different component after on clicking.



```
    const inputElement = useRef();
```



```
<select className="form-select" id="inputGroupSelect01" ref={inputElement}>
```

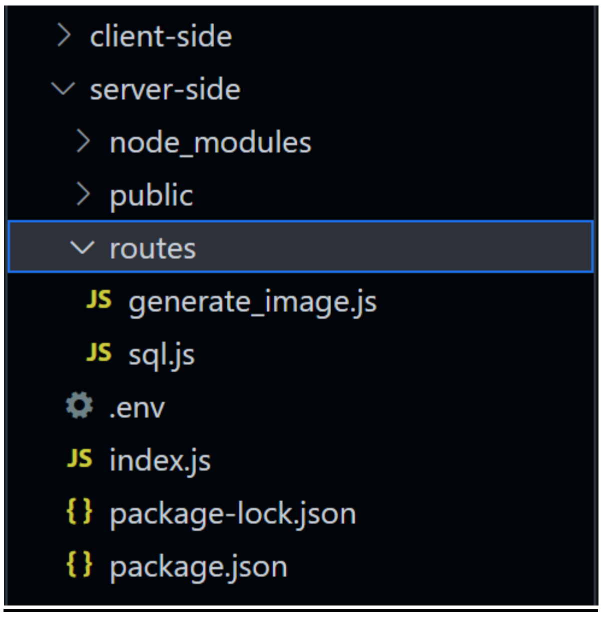
As you can see we have referenced another element while clicking submit to get its value. In this case we are getting the language from the user.

One more important package we have used is react router dom. This package facilitates in defining website routes, navigating to specific routes, etc.

Server side

In the backend we have written all the logic for shortening the URL, sending the responses, etc. Another javascript framework Node.js is used for the backend. Node.js helps in simplifying many backend concepts using simple functions. In Node.js there is a concept of file based routing which we have used here. We have defined various routes for different functionalities like image generation, sql generation etc.

FOLDER STRUCTURE



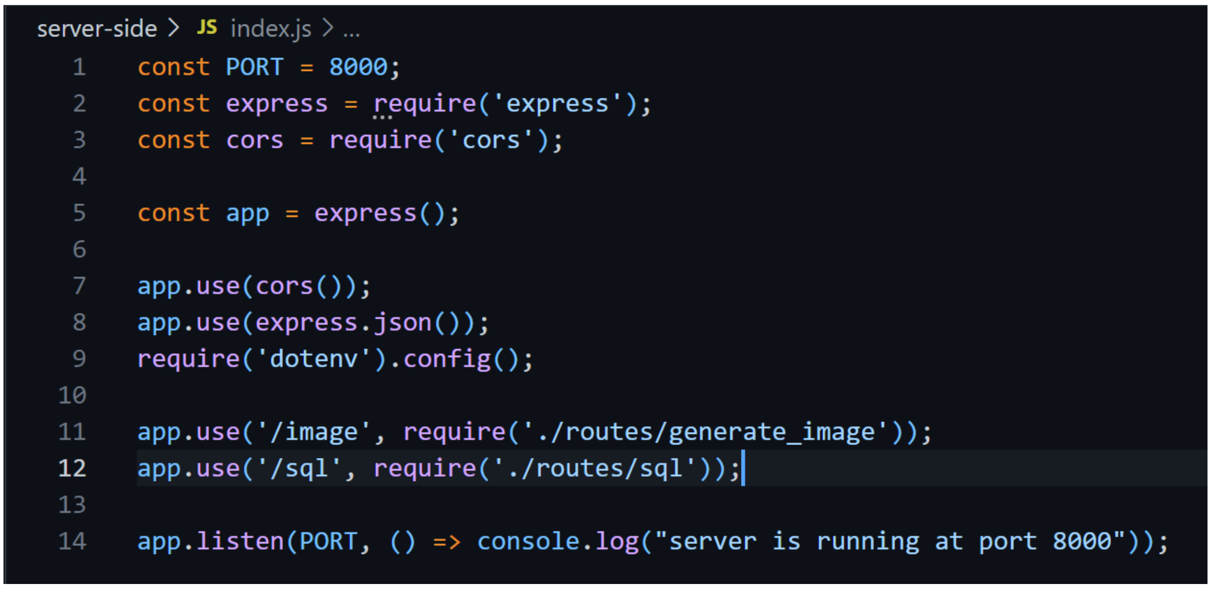
```
> client-side
v server-side
  > node_modules
  > public
  v routes
    JS generate_image.js
    JS sql.js
  .env
  JS index.js
  {} package-lock.json
  {} package.json
```

The above image shows the folder structure of the server folder. The server side has the router folder where we define the routes of the server.Routes as we discussed earlier are for defining various functionalities in various endpoints. After that we have 1 more file, which is index.js. index.js is the main file where the execution starts.

CODE

Index.js file

This file is the main entry point of the backend application. In this file we will do all the function imports and route defining



server-side > JS index.js > ...

```
const PORT = 8000;
const express = require('express');
const cors = require('cors');

const app = express();

app.use(cors());
app.use(express.json());
require('dotenv').config();

app.use('/image', require('./routes/generate_image'));
app.use('/sql', require('./routes/sql'));

app.listen(PORT, () => console.log("server is running at port 8000"));
```

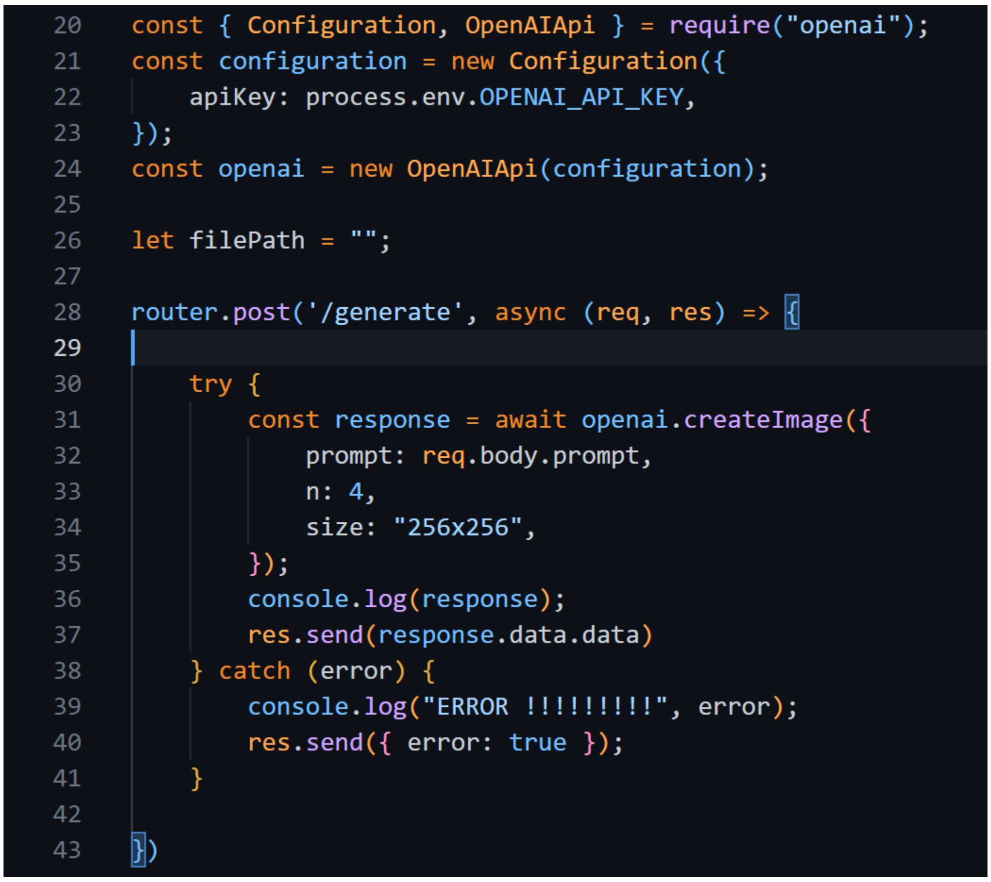
As you can see, we have imported all the necessary packages and we have defined two routes for image, sql and chat.

After that we have wrote the command to start the server

Route files

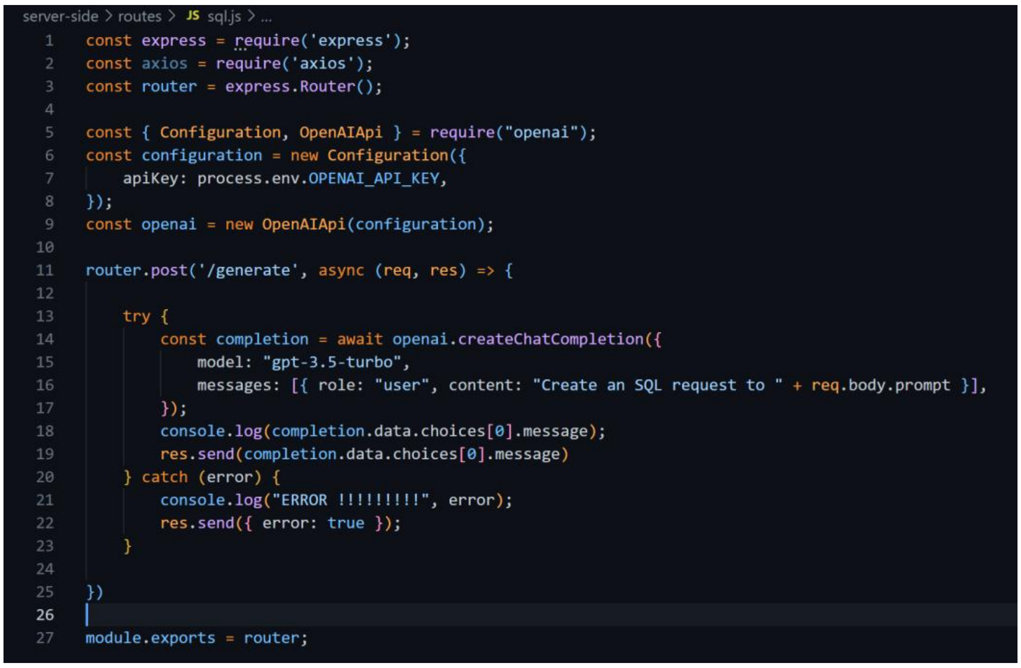
We have three route files which are image, sql and chat.

First one is the image route where we are interacting with the openAI API and getting the response. This file takes the user input given at the frontend and generates the image based on the prompt.



```javascript
const { Configuration, OpenAIApi } = require("openai");
const configuration = new Configuration({
    apiKey: process.env.OPENAI_API_KEY,
});
const openai = new OpenAIApi(configuration);

let filePath = "";

router.post('/generate', async (req, res) => {

    try {
        const response = await openai.createImage({
            prompt: req.body.prompt,
            n: 4,
            size: "256x256",
        });
        console.log(response);
        res.send(response.data.data)
    } catch (error) {
        console.log("ERROR !!!!!!!!!", error);
        res.send({ error: true });
    }

})
```

Next file is the sql.js file where we contact the api and generate sql queries based on the statements given by the user



server-side > routes > JS sql.js > ...

```javascript
const express = require('express');
const axios = require('axios');
const router = express.Router();

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

router.post('/generate', async (req, res) => {

    try {
        const completion = await openai.createChatCompletion({
            model: "gpt-3.5-turbo",
            messages: [{ role: "user", content: "Create an SQL request to " + req.body.prompt }],
        });
        console.log(completion.data.choices[0].message);
        res.send(completion.data.choices[0].message)
    } catch (error) {
        console.log("ERROR !!!!!!!!!", error);
        res.send({ error: true });
    }

})

module.exports = router;
```

Conclusion

In this project, a suite of AI tools comprising a translator, image generator, SQL generator, and Code GPT has been utilized to enhance various aspects of language processing, visual content creation, and database management.

The translator tool harnesses the power of machine translation, breaking down language barriers and enabling seamless communication and collaboration across different cultures and languages. It facilitates international exchanges, business interactions, and cultural understanding.

The image generator tool leverages advanced AI and deep learning techniques to automate the creation and manipulation of visually appealing images, graphics, and visual effects. This tool empowers professionals in creative industries, such as design, advertising, and virtual reality, to bring their ideas to life swiftly and produce high-quality visual content.

The SQL generator tool simplifies complex database operations by automating the generation of SQL queries. It streamlines data retrieval, insertion, modification, and deletion tasks, improving productivity and optimizing performance for developers and data analysts.

The Code GPT tool assists developers in coding tasks by generating code snippets, providing coding suggestions, and offering programming assistance. It accelerates the software development process, promoting efficiency and accuracy in coding tasks.

Collectively, these AI tools revolutionize language processing, visual content creation, and database management, offering convenience, efficiency, and innovation. By harnessing the power of AI, this project demonstrates how these tools can greatly enhance productivity, creativity, and effectiveness across a wide range of applications.

References

- https://www.geeksforgeeks.org/reactjs-basics-concepts-complete-reference/
- https://www.youtube.com/watch?v=z-Z5radvnOA&list=PLwGdqUZWnOp3aROg4wypcRhZqJG3ajZWJ&index=15
- https://www.techtarget.com/searchapparchitecture/application-program-interface-API
- Advanced web development by Mehul Mohan.
- https://www.youtube.com/watch?v=ExcRbA7fy_A&list=PL4cUxeGkcC9h77dJ-QJlwGlZlTd4ecZOA
- https://www.youtube.com/watch?v=zb3Qk8SG5Ms&list=PL4cUxeGkcC9jsz4LDYc6kv3ymONOKxwBU
- https://platform.openai.com/docs/introduction
- https://goqr.me/api/