

Topic: Plant Analysis Tool using Gemini AI and Express.js Part 1

Speaker: *Masynctech* / Notebook: *Node.js (JavaScript) Projects*



We used NODE.JS (Javascript) and NVM.

MAIN VIDEO RESOURCE:

1. We created a new folder, PLANTANALYSIS TOOL.
2. We open a Gitbash Terminal here and use CODE . (code dot) to open our VS CODE editor.
3. We [Install NODE.JS](#), CODEIUM EXTENSION IN VS CODE EXTENSIONS and get Google API key from Google API dashboard.



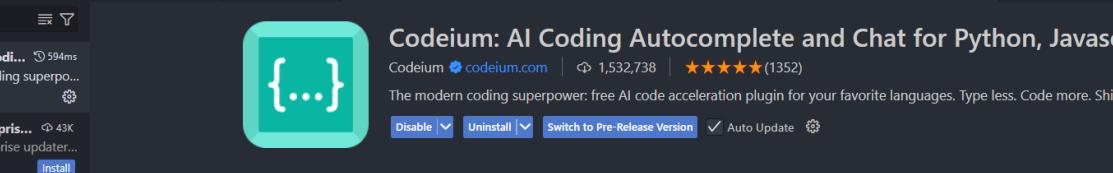
Run JavaScript Everywhere

Node.js® is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools and scripts.

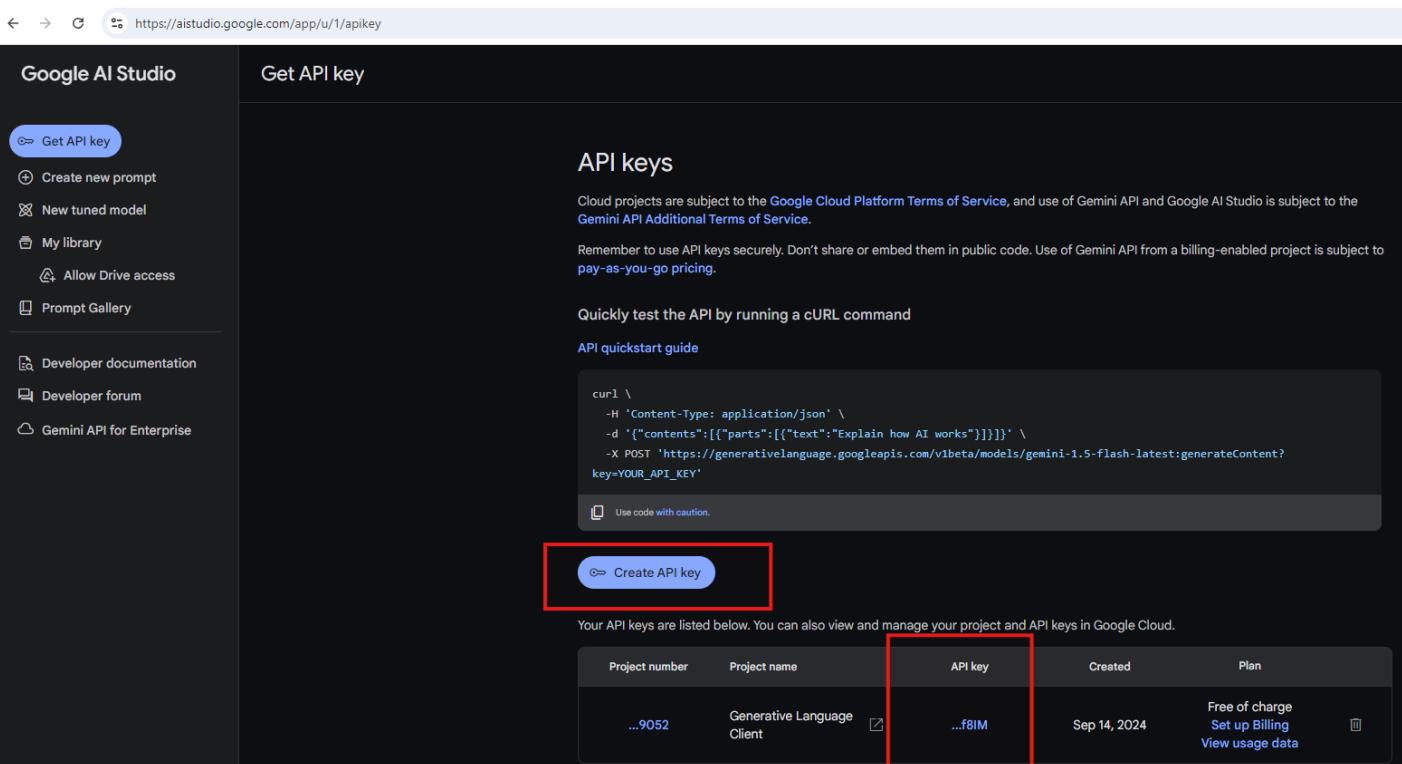
[Download Node.js \(LTS\) !\[\]\(de95854c7ee024cfadc48187bbb781b2_img.jpg\)](#)

Downloads Node.js **v20.17.0¹** with long-term support.
Node.js can also be installed via [package managers](#).

Want new features sooner? Get **Node.js v22.8.0¹** instead.



The screenshot shows the Microsoft Store interface with the search bar set to 'PlantAnalysisTool'. The results list the 'Codeium' extension, which is highlighted. The extension page for 'Codeium: AI Coding Autocomplete and Chat for Python, Javascript, Typescript, Java, Go, and more' is displayed. The page features a large green icon with three dots and brackets, the extension's name, a brief description, and a rating of 4.5 stars. It includes buttons for 'Disable', 'Uninstall', 'Switch to Pre-Release Version', and 'Auto Update'. Below the main title, there are tabs for 'DETAILS', 'FEATURES', and 'CHANGELOG'. A large section titled 'Codeium: Free AI-powered code acceleration toolkit' is present, followed by a question 'What is Codeium?' and a detailed description of its features. To the right, there are sections for 'Categories' (AI, Chat, Programming Languages, Machine Learning, Snippets, Education) and 'Resources' (Marketplace, Issues, License, Codeium). The left sidebar shows other popular extensions like 'VSCode Essentials' and 'Front-End Extension'.



Google AI Studio

Get API key

Get API key

Create new prompt

New tuned model

My library

Allow Drive access

Prompt Gallery

Developer documentation

Developer forum

Gemini API for Enterprise

API keys

Cloud projects are subject to the [Google Cloud Platform Terms of Service](#), and use of Gemini API and Google AI Studio is subject to the [Gemini API Additional Terms of Service](#).

Remember to use API keys securely. Don't share or embed them in public code. Use of Gemini API from a billing-enabled project is subject to [pay-as-you-go pricing](#).

Quickly test the API by running a cURL command

API quickstart guide

```
curl \
  -H 'Content-Type: application/json' \
  -d '{"contents": [{"parts": [{"text": "Explain how AI works"}]}]}' \
  -X POST 'https://generativelanguage.googleapis.com/vbeta/models/gemini-1.5-flash-latest:generateContent?key=YOUR_API_KEY'
```

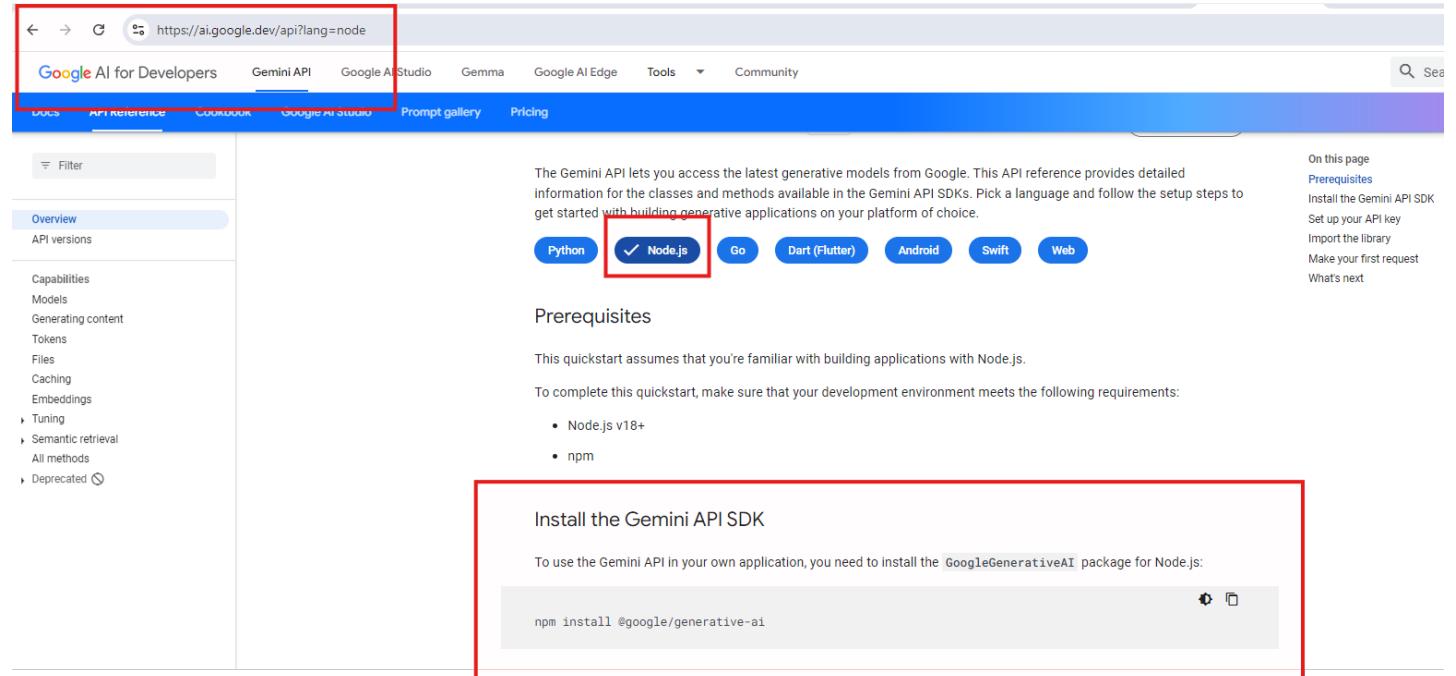
Use code with caution.

Create API key

Your API keys are listed below. You can also view and manage your project and API keys in Google Cloud.

Project number	Project name	API key	Created	Plan
...9052	Generative Language Client	...f8IM	Sep 14, 2024	Free of charge Set up Billing View usage data

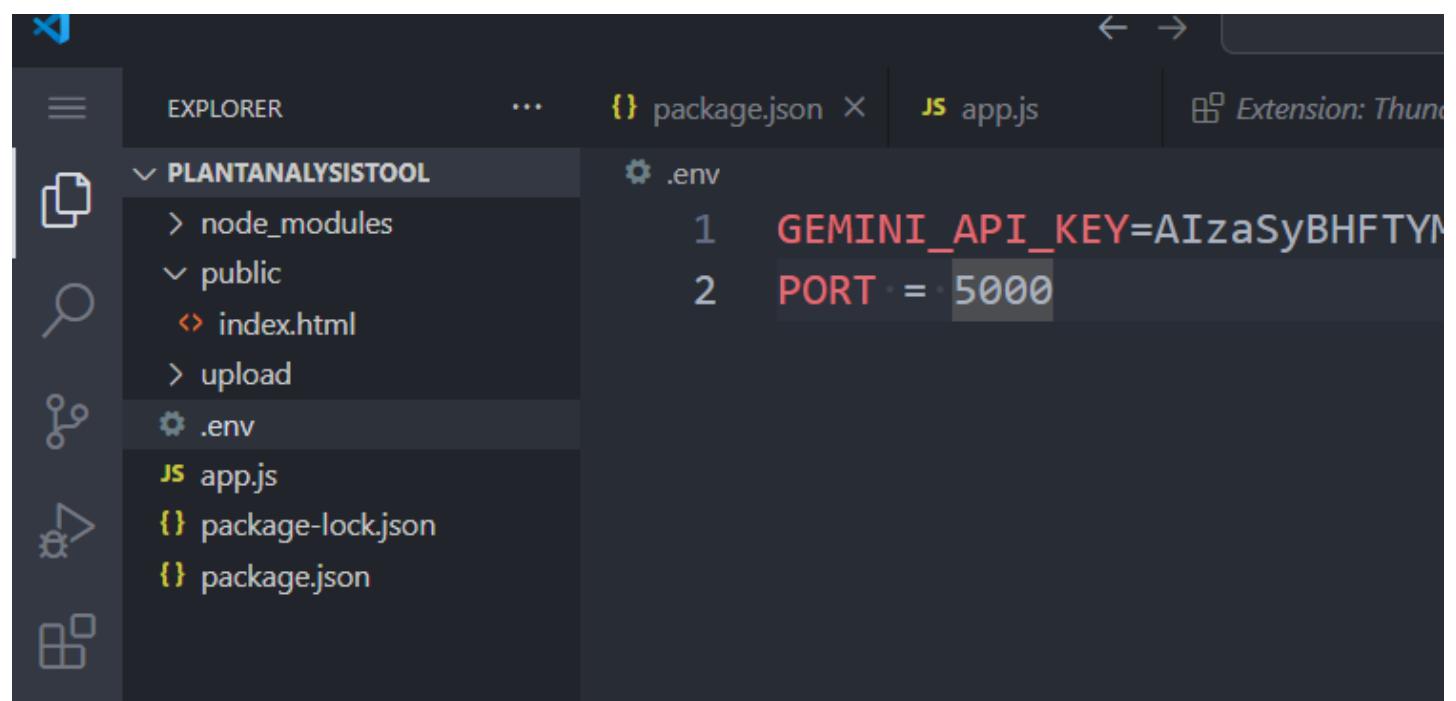
We use GOOGLE API REFERENCE to install our GENERATIVE AI in NODE.JS:



The screenshot shows the Google AI for Developers API Reference page for the Gemini API. The URL is <https://ai.google.dev/api?lang=node>. The page has a navigation bar with links for Google AI for Developers, Gemini API, Google AI Studio, Gemma, Google AI Edge, Tools, and Community. Below the navigation bar is a blue header with links for Docs, API Reference, Lookbook, Google AI Studio, Prompt gallery, and Pricing. On the left, there is a sidebar with sections for Overview, API versions, Capabilities, Models, Generating content, Tokens, Files, Caching, Embeddings, Tuning, Semantic retrieval, All methods, and Deprecated. The main content area has a heading 'Prerequisites' and a list of requirements: Node.js v18+ and npm. A red box highlights the 'Node.js' button in the language selection bar. Another red box highlights the 'Install the Gemini API SDK' section, which contains the command 'npm install @google/generative-ai'.

4. We created new folders like UPLOAD, PUBLIC and created APPS.JS and .ENV files

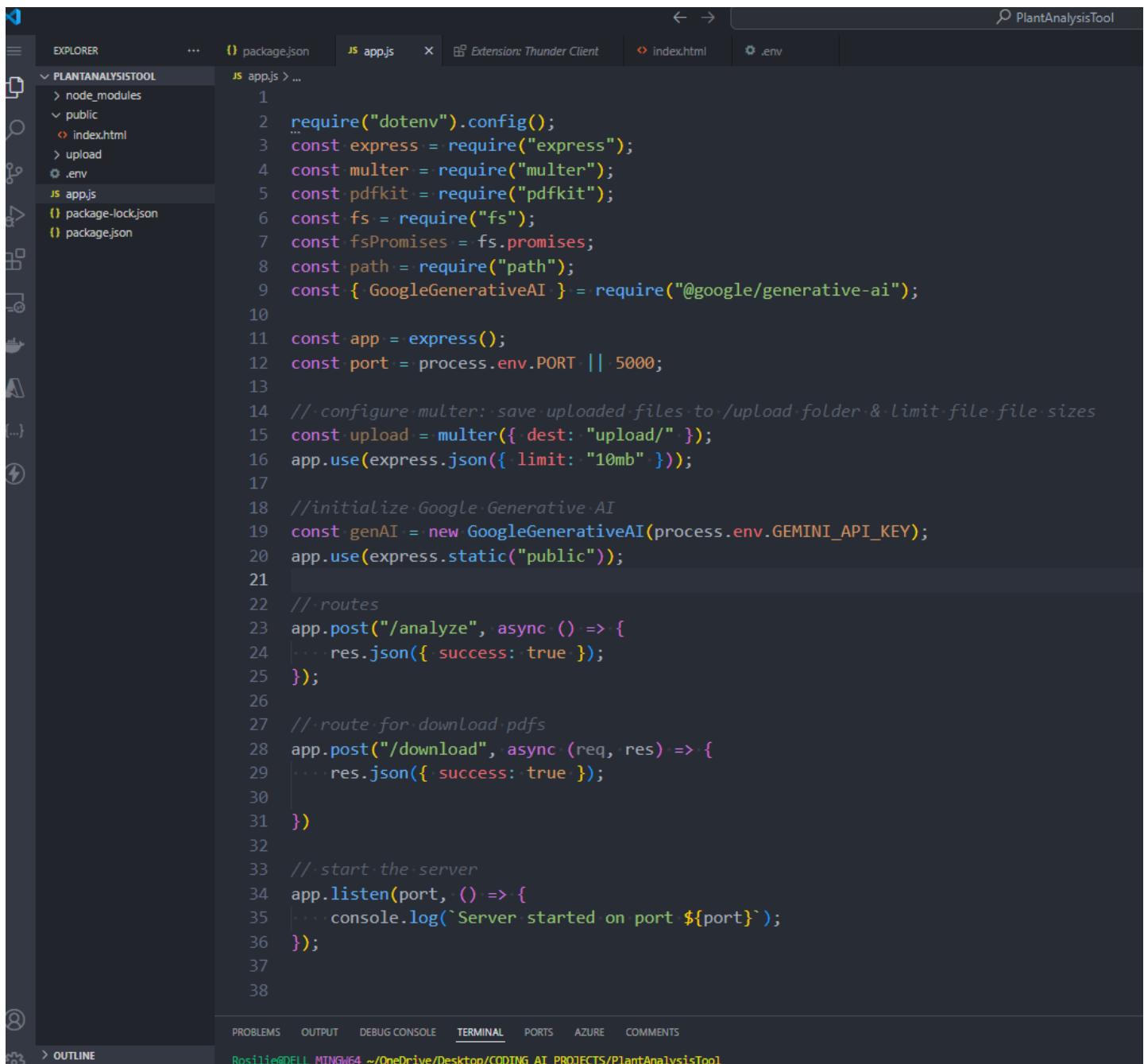
.ENV FILE:



The screenshot shows the VS Code interface with the Explorer sidebar open, displaying a project structure for 'PLANTANALYSISTOOL'. The project contains 'node_modules', 'public' (with 'index.html'), 'upload', and '.env'. The '.env' file is open in the editor, showing the following content:

```
1 GEMINI_API_KEY=AIzaSyBHFTY
2 PORT = 5000
```

APPS.JS

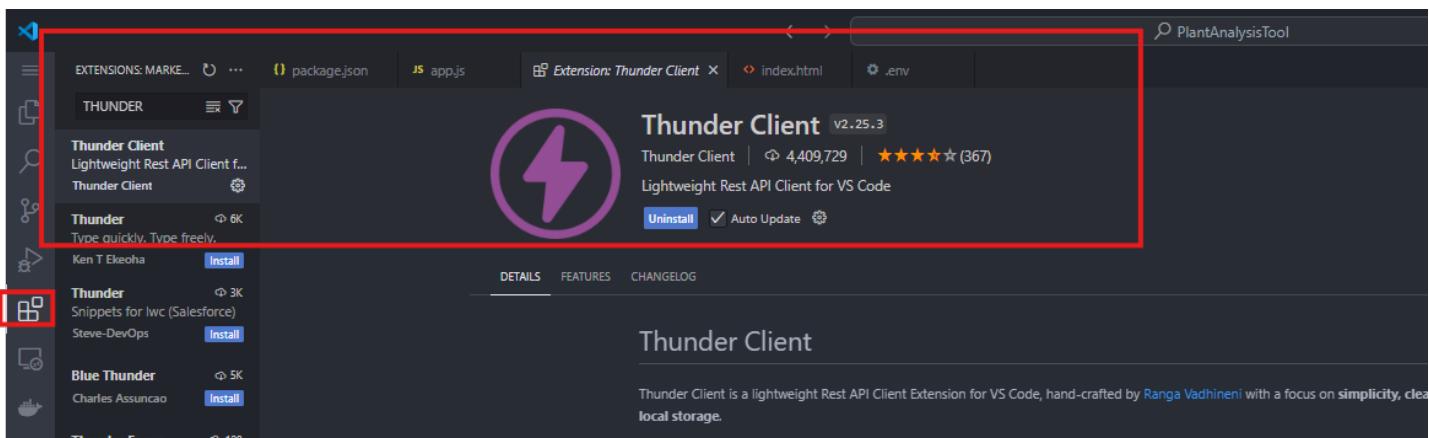


```

1
2  require("dotenv").config();
3  const express = require("express");
4  const multer = require("multer");
5  const pdfkit = require("pdfkit");
6  const fs = require("fs");
7  const fsPromises = fs.promises;
8  const path = require("path");
9  const { GoogleGenerativeAI } = require("@google/generative-ai");
10
11 const app = express();
12 const port = process.env.PORT || 5000;
13
14 // configure multer to save uploaded files to ./upload folder & limit file sizes
15 const upload = multer({ dest: "upload/" });
16 app.use(express.json({ limit: "10mb" }));
17
18 // initialize Google Generative AI
19 const genAI = new GoogleGenerativeAI(process.env.GEMINI_API_KEY);
20 app.use(express.static("public"));
21
22 // routes
23 app.post("/analyze", async () => {
24   res.json({ success: true });
25 });
26
27 // route for download pdfs
28 app.post("/download", async (req, res) => {
29   res.json({ success: true });
30 });
31
32
33 // start the server
34 app.listen(port, () => {
35   console.log(`Server started on port ${port}`);
36 });
37
38

```

5. To test our ENDPOINT, we will use POSTMAN (you used INSOMNIA) or we can install the VS CODE EXTENSION, THUNDER CLIENT.



6. Run the app by issuing this code.

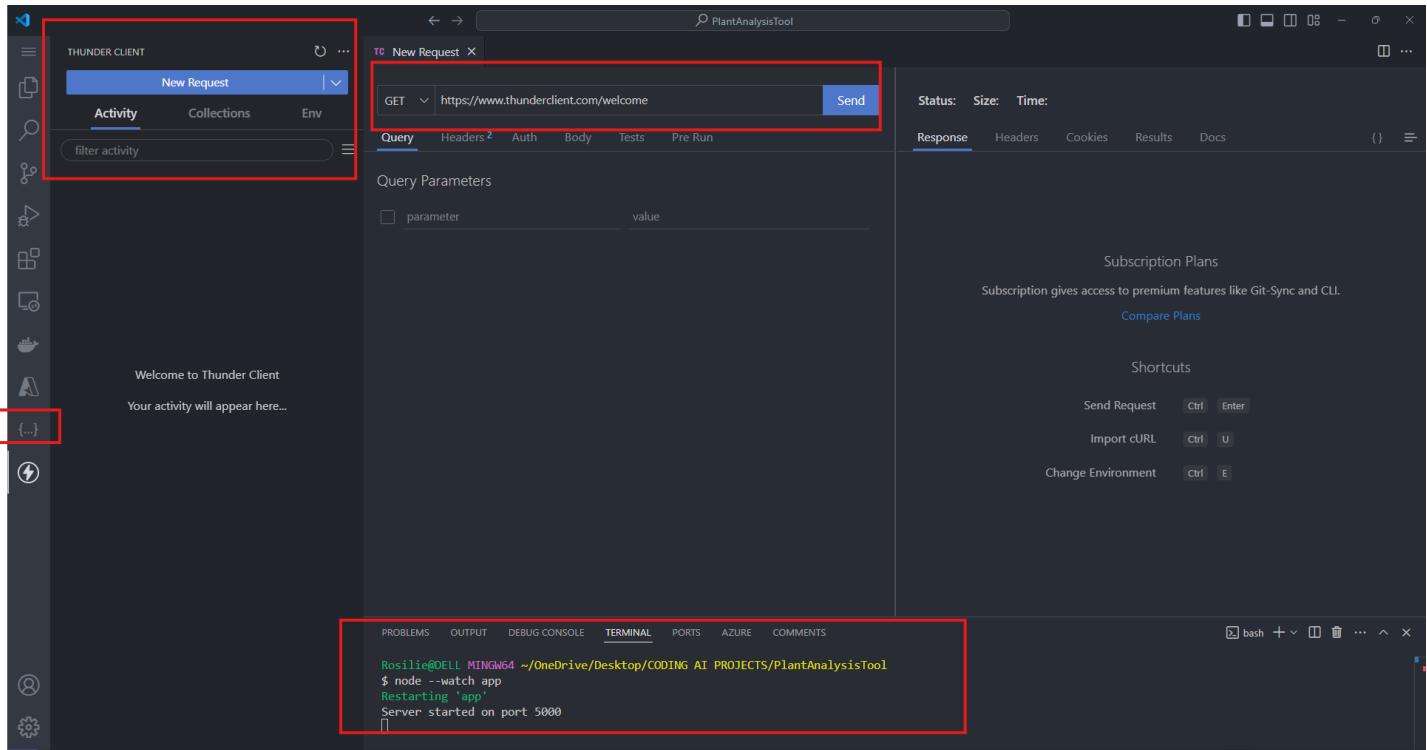
```
$ node --watch app (where app is our APPS.JS)
```

```

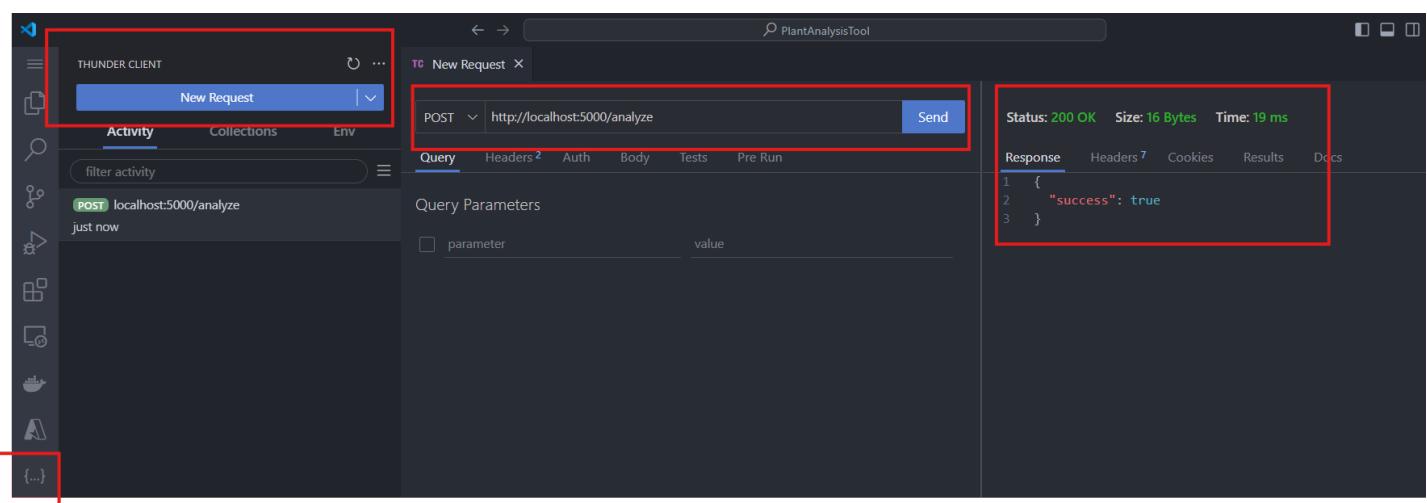
PROBLEMS OUTPUT TERMINAL
Rosilie@DELL MINGW64 ~/OneDrive/Desktop
ysisTool
$ node --watch app
Restarting 'app'
Server started on port 5000

```

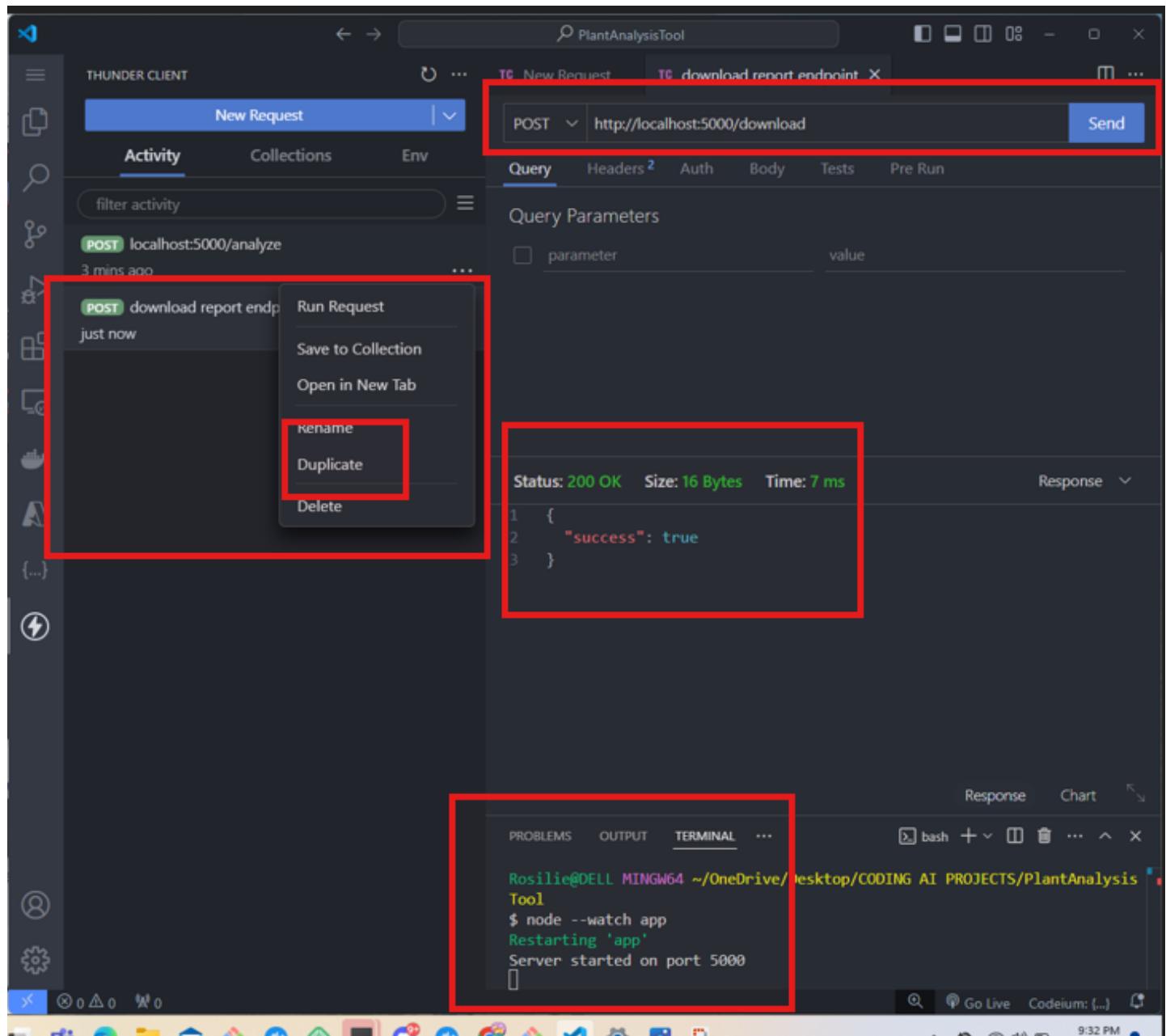
7. Close all your tabs in VS Code. Right click on the THREE DOTS where the EXTENSION button is, and select THUNDER POINT. Select NEW REQUEST.



8. To access our work, we issue our URL path: `HTTP://localhost: 5000`. This should show a SUCCESS MESSAGE



9. We test our other endpoint, `HTTP://LOCALHOST:DOWNLOAD/` We duplicate our first request and name it. Then, we change our URL PATH.



10. Just like in Django where we test our paths using Django's views.py, the logic for Node.js is this:

```

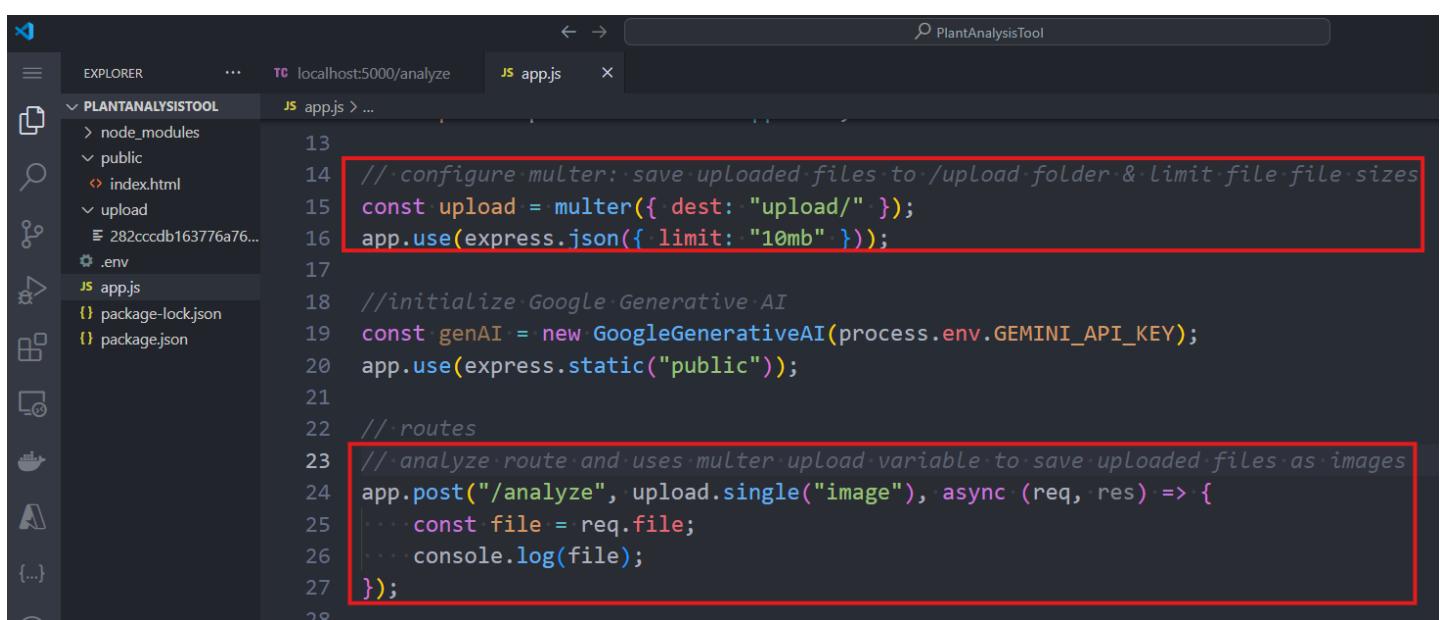
// routes
// analyze route
app.post("/analyze", async (req, res) => {
  res.json({ success: true });
});

// route for download pdfs
app.post("/download", async (req, res) => {
  res.json({ success: true });
}
)

```

11. To test the upload function, we can use the THUNDER BODYFORM and add the variable we used 'IMAGE' and upload a file from our local device. We should be able to see the details of this image.

APPS.JS:



```

13
14 //configure multer to save uploaded files to /upload folder & limit file sizes
15 const upload = multer({ dest: "upload/" });
16 app.use(express.json({ limit: "10mb" }));

17
18 //initialize Google Generative AI
19 const genAI = new GoogleGenerativeAI(process.env.GEMINI_API_KEY);
20 app.use(express.static("public"));

21
22 //routes
23 //analyze route and uses multer upload variable to save uploaded files as images
24 app.post("/analyze", upload.single("image"), async (req, res) => {
25   const file = req.file;
26   console.log(file);
27 });
28

```

POST localhost:5000/analyze

Body 1

Form

File: houseplant-7367379_640.jpg

field name: image

filename: 'image', originalname: 'houseplant-7367379_640.jpg', encoding: '7bit', mimetype: 'image/jpeg', destination: 'upload/', filename: '282ccdb163776a7659c74904707b9db', path: 'upload\282ccdb163776a7659c74904707b9db', size: 90004

12. To allow Gemini AI to use the details captured from step 11, we have to indicate the GEMINI VERSION:

Make your first request

Use the `generateContent` method to generate text.

```
// Make sure to include these imports:
// import { GoogleGenerativeAI } from "@google/generative-ai";
const genAI = new GoogleGenerativeAI(process.env.API_KEY);
const model = genAI.getGenerativeModel({ model: "gemini-1.5-flash" });

const prompt = "Write a story about a magic backpack.";

const result = await model.generateContent(prompt);
console.log(result.response.text());
```

text_generation.js

13. We updated our APPS.JS to include GEMINI API.

This is the PROMPT we used for Gemini "Analyze this plant image and provide detailed analysis of its species, health and care recommendations, its characteristics, care instructions and interesting facts. Please provide the response in plain text without using any markdown formatting "

Our function:

```

22 // routes
23 // analyze route and uses multer.upload variable to save uploaded files as images
24 app.post('/analyze', upload.single("image"), async (req, res) => {
25   const file = req.file;
26   //console.log(file); use the image details for Gemini AI
27   try {
28     if (!req.file) {
29       return res.status(400).json({ error: "Please upload an image" });
30     }
31     const imagePath = req.file.path;
32     const imageData = await fsPromises.readFile(imagePath, {
33       encoding: "base64",
34     });
35     // use the gemini AI API to analyze the image
36     const model = genAI.getGenerativeModel({
37       model: "gemini-1.5-flash",
38     });
39
40     const results = await model.generateContent([
41       "Analyze this plant image and provide detailed analysis of its species, health and care recommendations, its characteristics
42     {
43       inlineData: {
44         mimeType: req.file.mimetype,
45         data: imageData,
46       },
47     },
48   ]);
49   const plantInfo = results.response.text()
50   // remove the uploaded image
51   await fsPromises.unlink(imagePath);
52   // send the response
53   res.json({ results: plantInfo, image: 'data:' + req.file.mimeType + ';base64,' + imageData });
54
55 } catch (error) {
56   res.status(500).json({ error: error.message });
57 }
58 });
59 });

```

14. We run our endpoint using Thunder Client:

POST http://localhost:5000/analyze

Form Fields

File

image

Response

```

{
  "results": "The plant in the image is a **Calathea musaica**, also known as the **Network Plant** or **Mosaic Plant**. It's a popular houseplant due to its striking foliage, featuring dark green leaves with intricate yellow and cream veins that resemble a mosaic pattern. \n\n**Health and Care Recommendations:** \n\n**Health:** The plant in the image appears healthy with vibrant foliage and no visible signs of pests or disease. \n\n**Care:** Calatheas are known for their specific needs, and they thrive in indirect sunlight with bright light. Bright, direct sunlight can burn the leaves. Bright, indirect light is ideal. \n\n**Watering:** Water when the top inch of soil is dry. Avoid overwatering, as it can lead to root rot. \n\n**Humidity:** Calatheas prefer high humidity, so misting the leaves regularly or using a humidifier is recommended. \n\n**Soil:** Use well-draining potting mix. \n\n**Temperature:** They prefer temperatures between 65-80°F (18-27°C). \n\n**Fertilizer:** Fertilize during the growing season (spring and summer) with a balanced fertilizer. \n\n**Size:** Can grow up to 1-2 feet tall indoors. \n\n**Color:** Green leaves with yellow and cream veins. \n\n**Size:** Can grow up to 1-2 feet tall indoors. \n\n**Growth Habit:** Upright with a compact growth habit. \n\n**Interesting Facts:** \n\n**Prayer Plant:** Calatheas are known as "prayer plants" because their leaves fold up at night, resembling hands clasped in prayer. \n\n**Native to Tropical Regions:** Calatheas are native to tropical regions of South America. \n\n**Air Purifier:** Calatheas are said to have air-purifying properties. \n\n**Varieties:** There are many varieties of Calatheas, each with unique foliage patterns and colors. \n\n"
  "image": "data:" + req.file.mimeType + ";base64," + imageData
}

```

15.