

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 VSCODE 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 \(LTS\)](#) 

Downloads Node.js **v20.17.0**¹ with long-term support.
Node.js can also be installed via [package managers](#).

Want new features sooner? Get **Node.js v22.8.0**¹ instead.

EXTENSIONS: MARKE... package.json JS app.js

codeium

Codeium: AI Cod... 594ms

The modern coding superpo...

Codeium

Codeium Enterpris... 43K

Codeium enterprise updat...

Codeium

Install

Python Silver Pack 18K

Elevate your Pyhton develop...

SraaamXTheLordWolf

Install

VSCode Essentials +... 2K

An extension pack for noobi...

SraaamXTheLordWolf


Install

Front-End Extensio... 3K

Elevate your frontend devel...

EchoWebNerds

Install



Codeium: AI Coding Autocomplete and Chat for Python, Javascript, Typescri

Codeium [codeium.com](#) | 1,532,738 | ★★★★★(1352)

The modern coding superpower: free AI code acceleration plugin for your favorite languages. Type less. Code more. Ship faster.

Disable Uninstall Switch to Pre-Release Version Auto Update

DETAILS FEATURES CHANGELOG

Codeium: Free AI-powered code acceleration toolkit

What is Codeium?

Codeium is the modern coding superpower, a **free** code acceleration toolkit built on cutting edge AI technology. Currently, Codeium provides autocomplete, chat, and search capabilities in 70+ languages, with lightning fast speeds and state-of-the-art suggestion quality. It takes 2 minutes to install on VSCode!

With Codeium, you get:

- Unlimited single and multi-line code completions forever
- IDE-integrated chat: no need to leave VSCode to ChatGPT, and use convenient suggestions such as Refactor and Explain
- Support for 70+ programming languages: Javascript, Python, Typescript, PHP, Go, Java, C, C++, Rust, Ruby, and more.
- Support through our [Discord Community](#).

Categories

AI Chat Programming Languages

Machine Learning Snippets

Education

Resources

[Marketplace](#)

[Issues](#)

[License](#)

[Codeium](#)

More Info

Published 2022-10-28 09:49:13

Google AI Studio

Get API key

Create new prompt

New tuned model

My library

Allow Drive access

Prompt Gallery

Developer documentation

Developer forum

Gemini API for Enterprise

Get API key

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/v1beta/models/gemini-1.5-flash-latest:generateContent?key=YOUR_API_KEY'
```

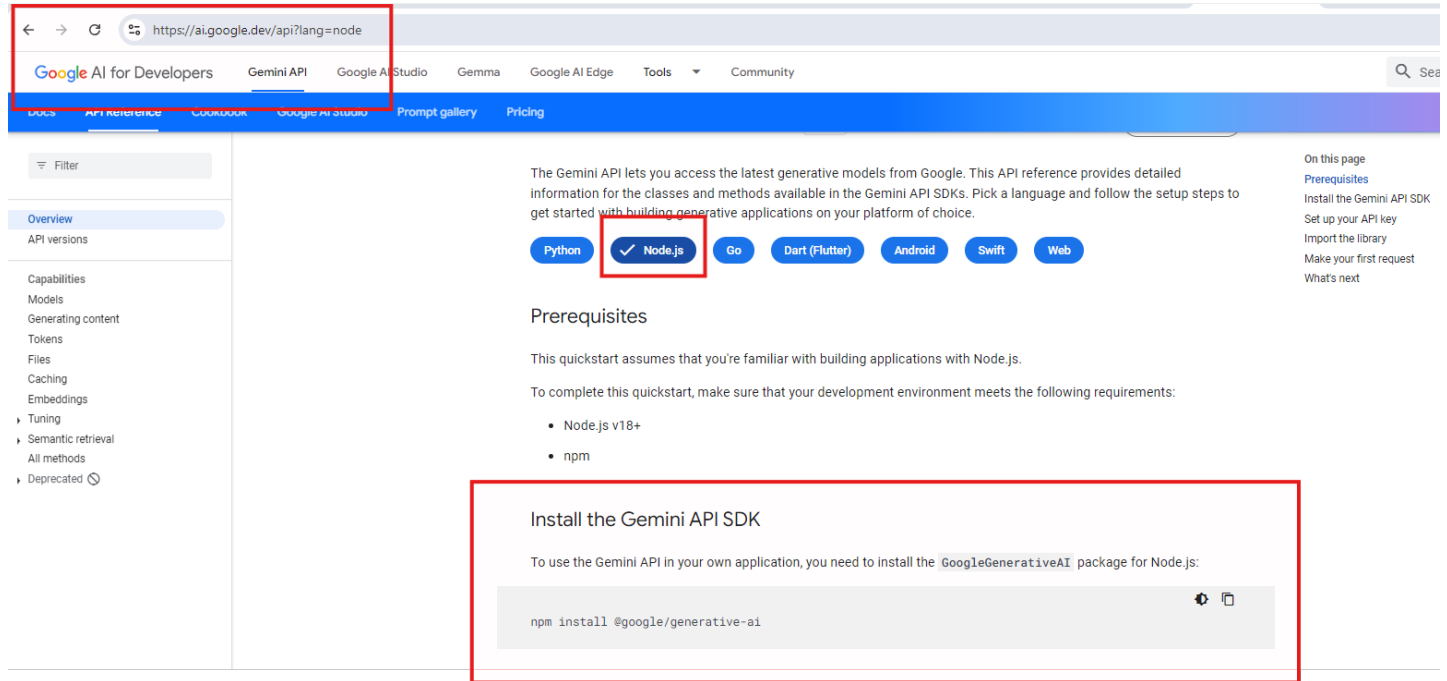
Use code with caution.

Create API key

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

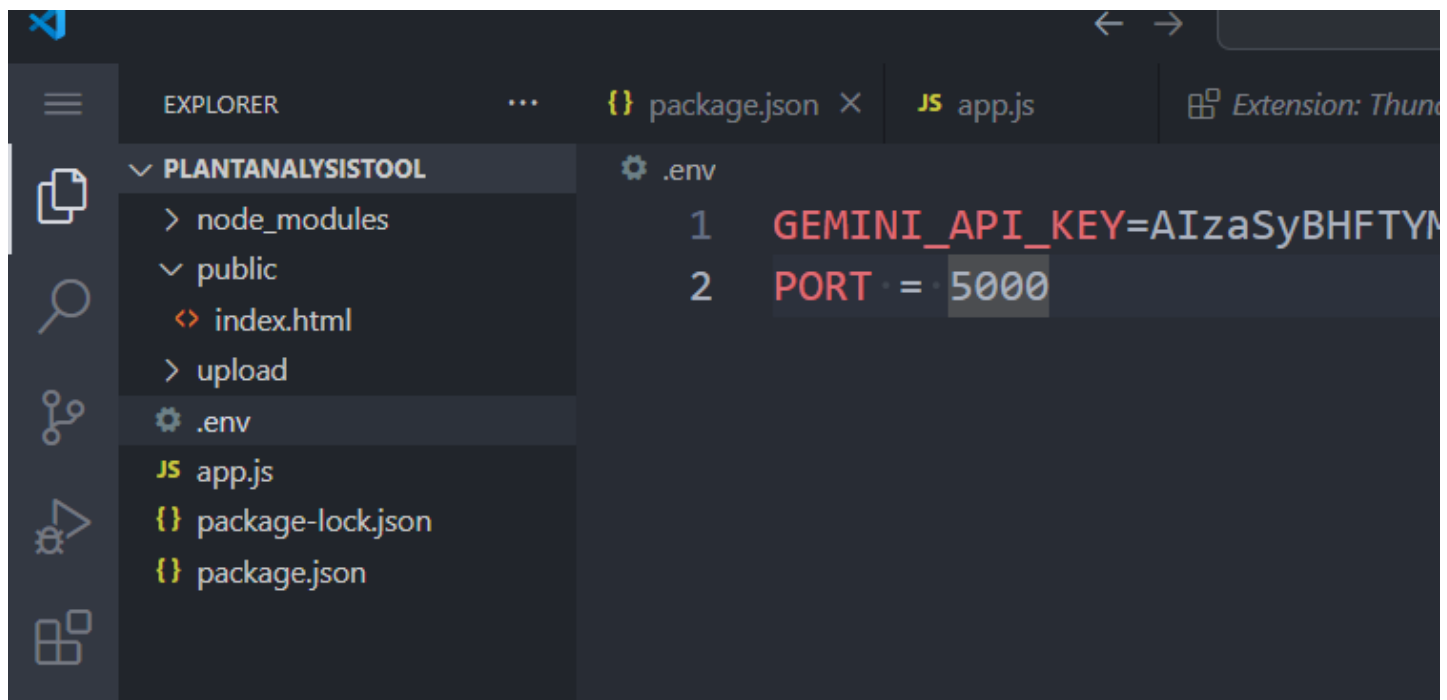
Project number	Project name	API key	Created	Plan
...9052	Generative Language Client	...f81M	Sep 14, 2024	Free of charge Set up Billing View usage data

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

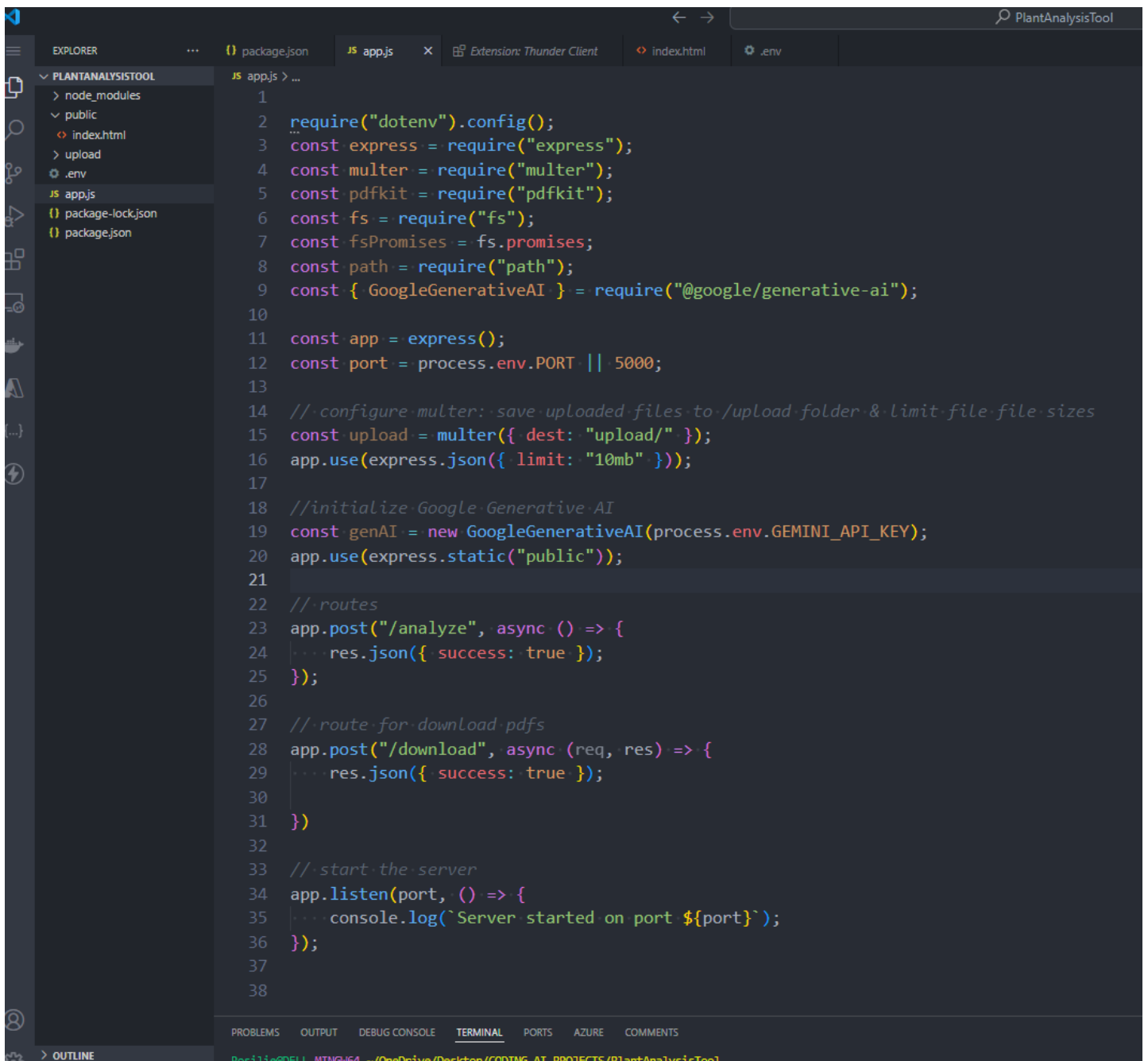


4. We created new folders like UPLOAD, PUBLIC and created APPS.JS and .ENV files

.ENV FILE:

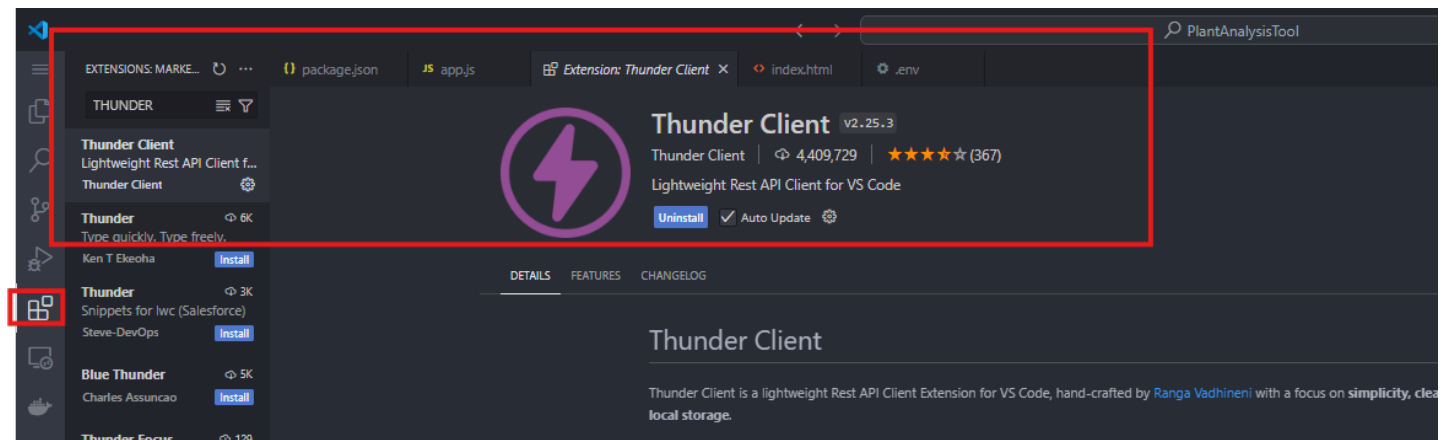


APPS.JS



```
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: save uploaded files to /upload folder & limit file 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.

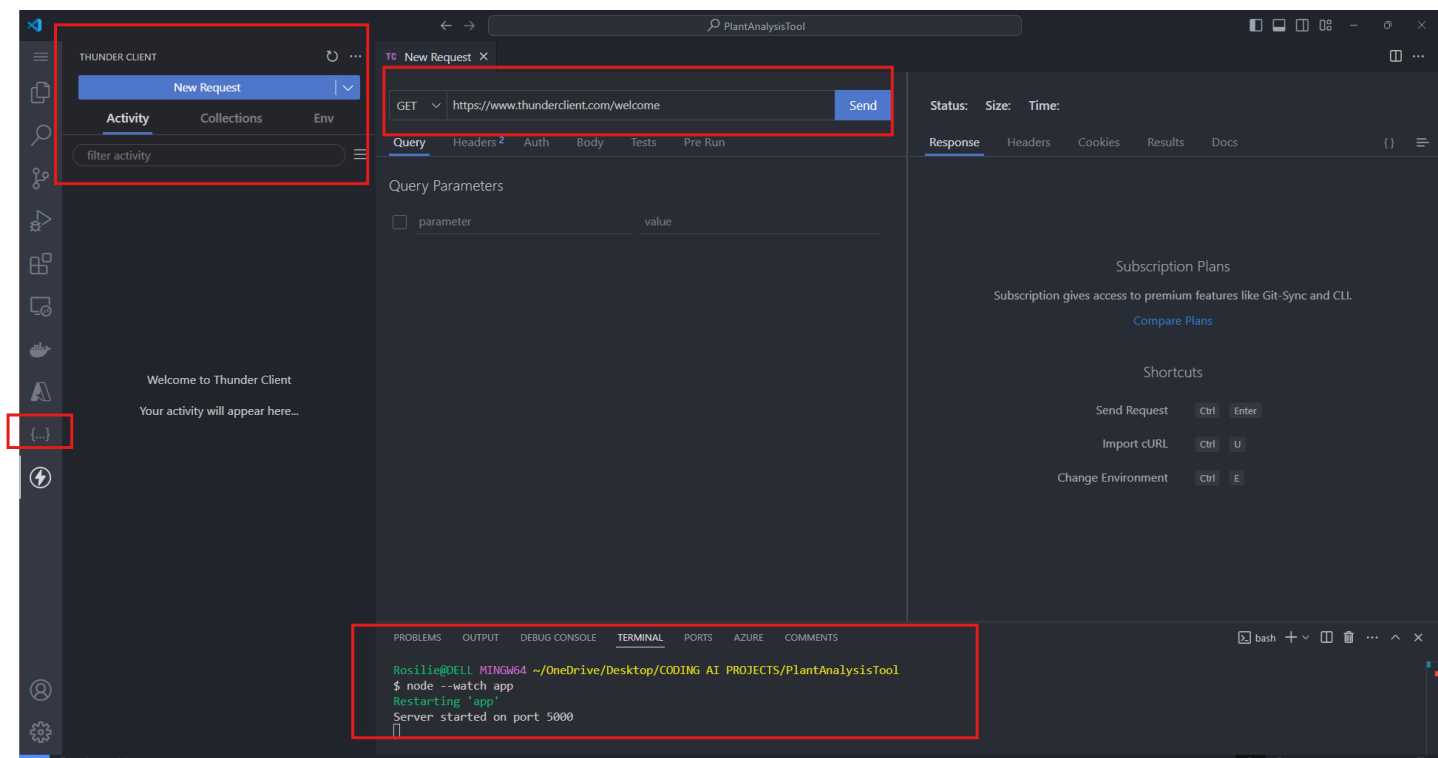


6. Run the app by issuing this code.

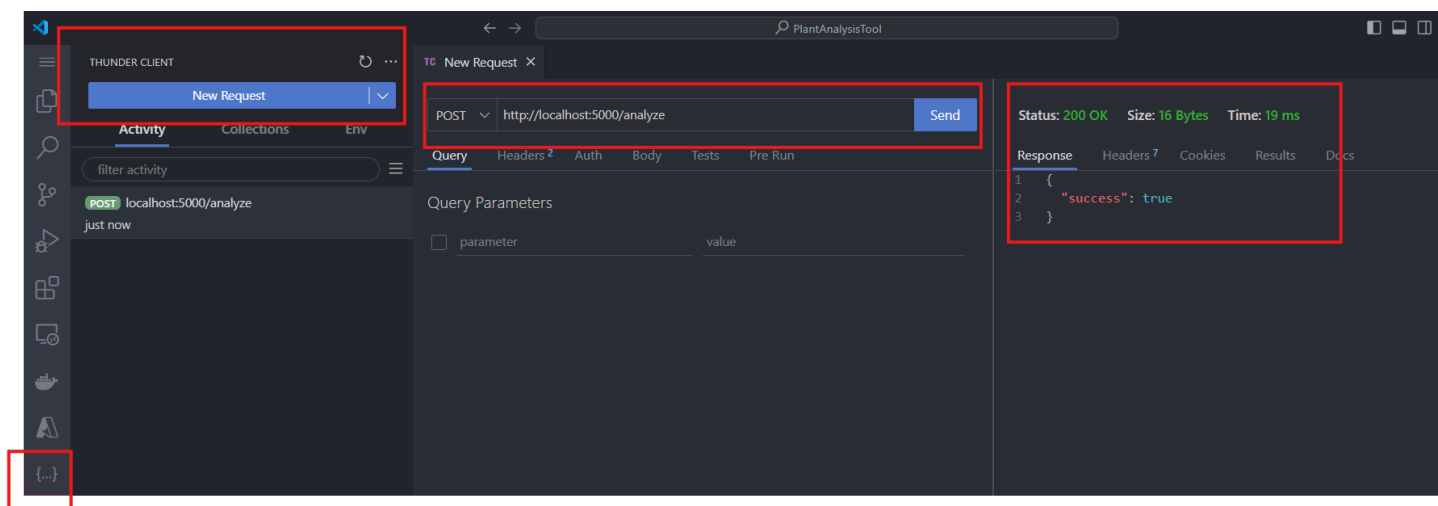
\$ node --watch app (where app is our APPS.JS)

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

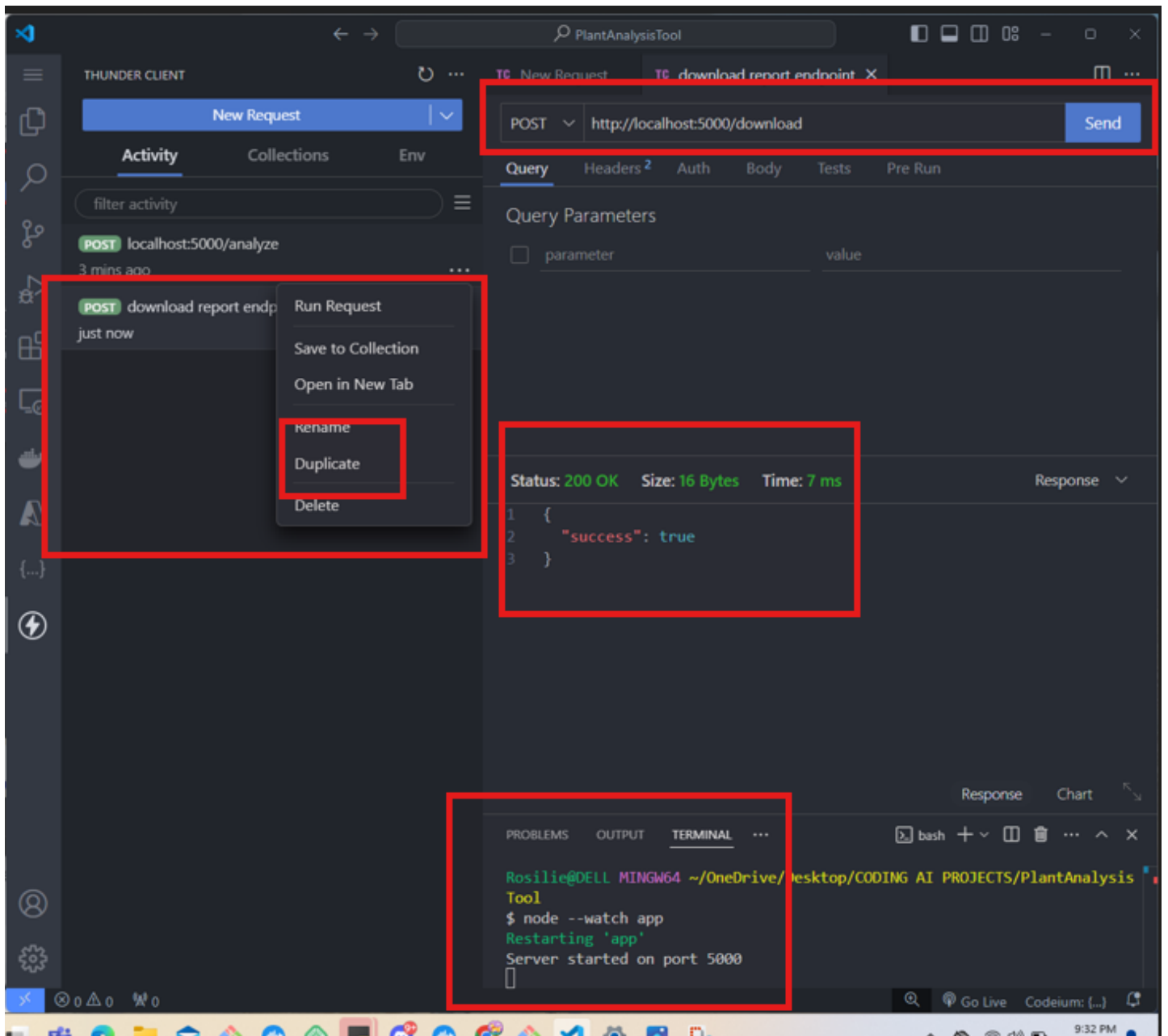
7. Close all your tabs in VS Code. Right click on the THREE DOTS where the EXTENSION button is, and select THUNDER POINT. Select NEW REQUEST.



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 });
});

```

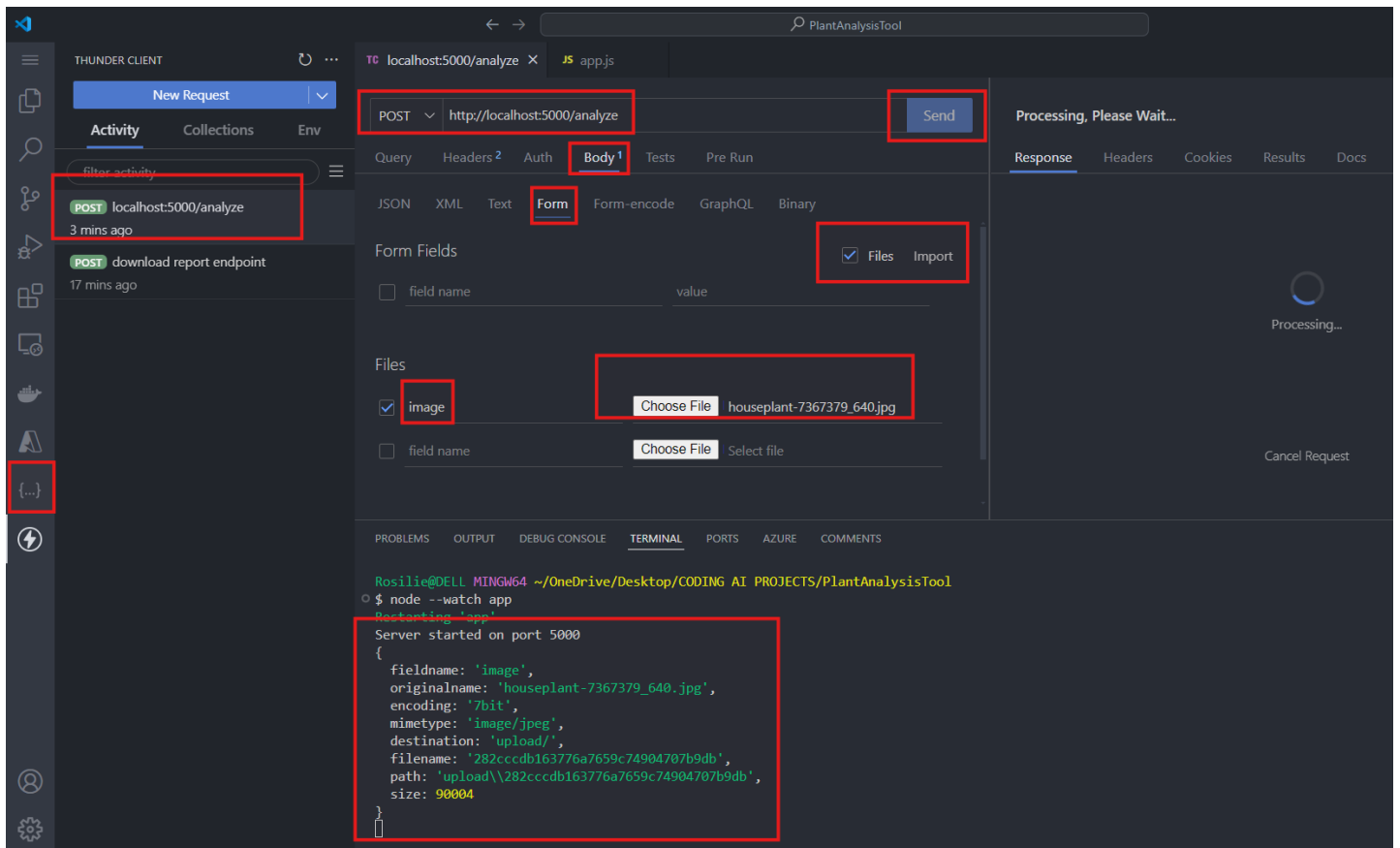
11. To test the upload function, we can use the THUNDER BODY\FORM and add the variable we used 'IMAGE' and upload a file from our local device. We should be able to see the details of this image.

APPS.JS:

```

13
14 //configure multer: 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

```



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

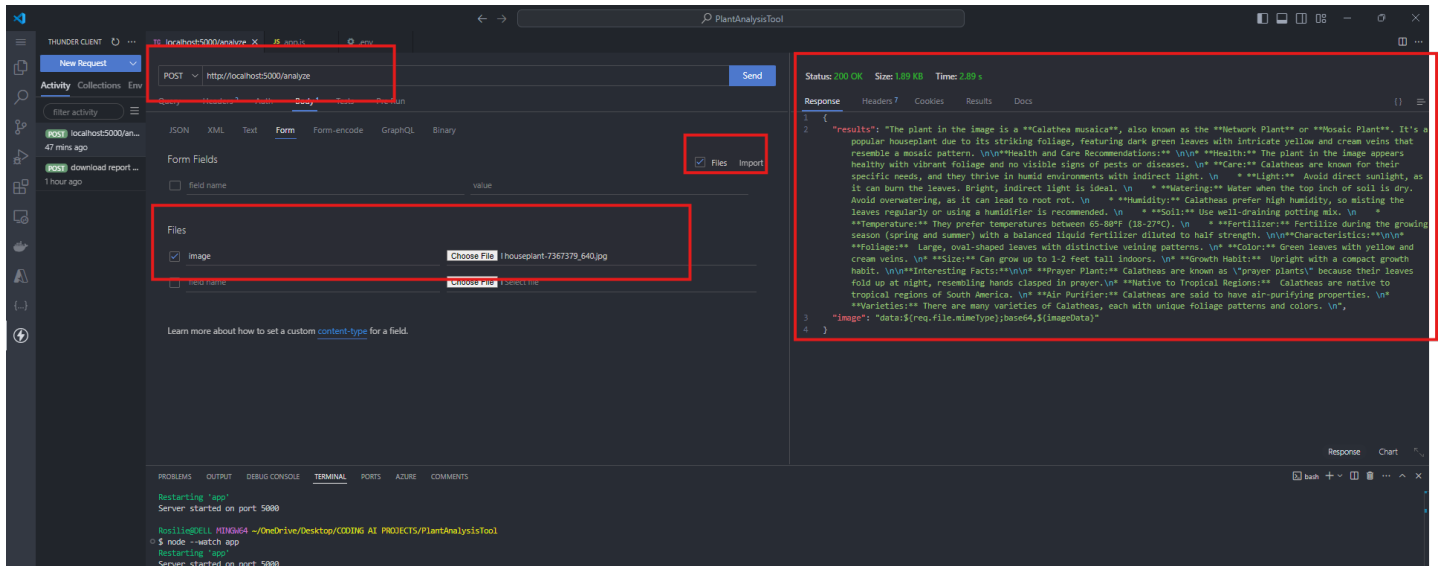
13. We updated our APPS.JS to include GEMINI API.

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     const plantInfo = results.response.text()
49     // remove the uploaded image
50     await fsPromises.unlink(imagePath);
51     // send the response
52     res.json({ results: plantInfo, image: `data:${req.file.mimetype};base64,${imageData}` });
53   } catch (error) {
54     res.status(500).json({ error: error.message });
55   }
56 });
```

14. We run our endpoint using Thunder Client:



15.