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

Speaker: Masynctech | Notebook: Node.js (JavaScript) Projects

---

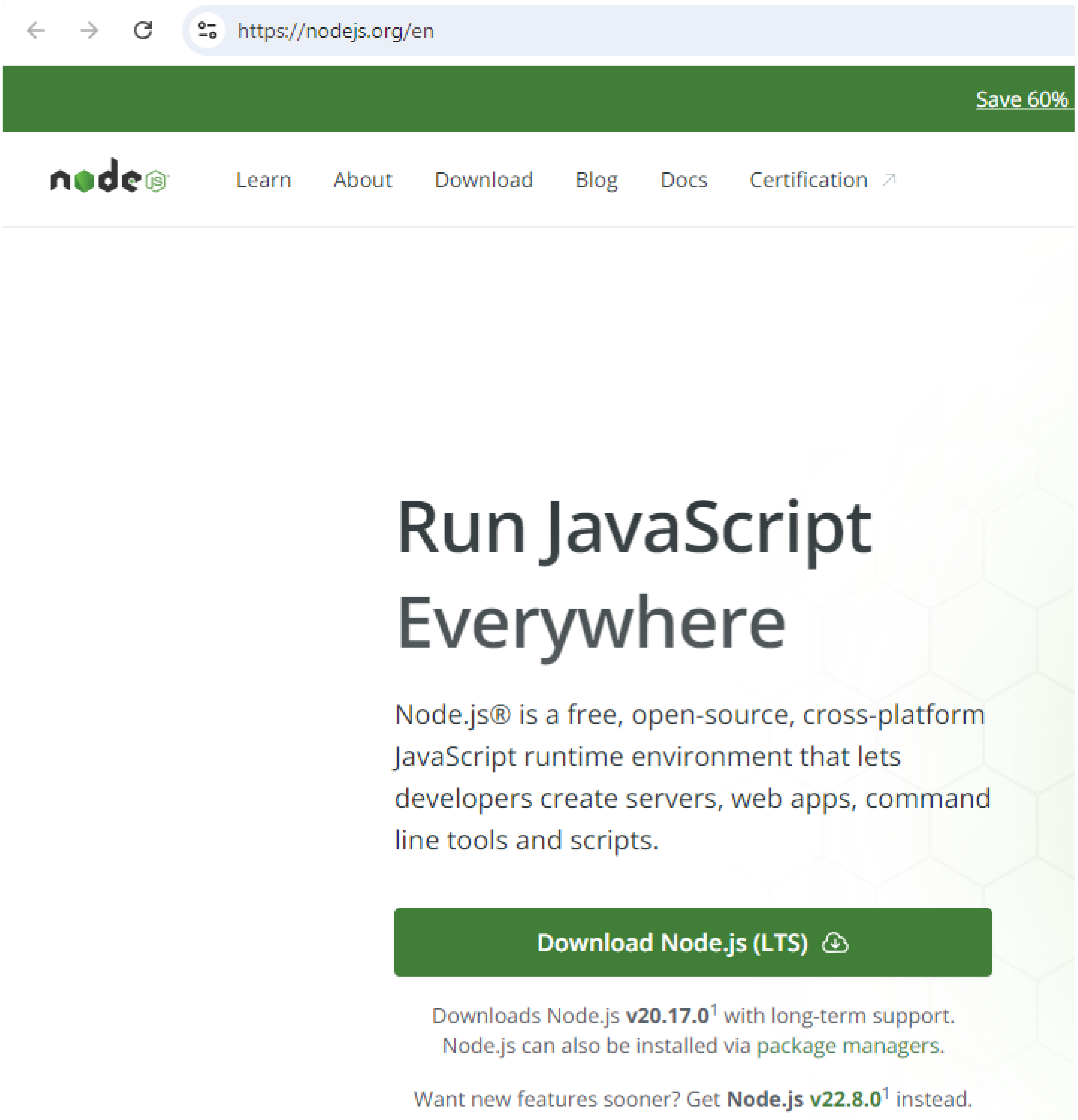
We used NODE.JS (Javascript) and NVM.

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.


https://nodejs.org/en

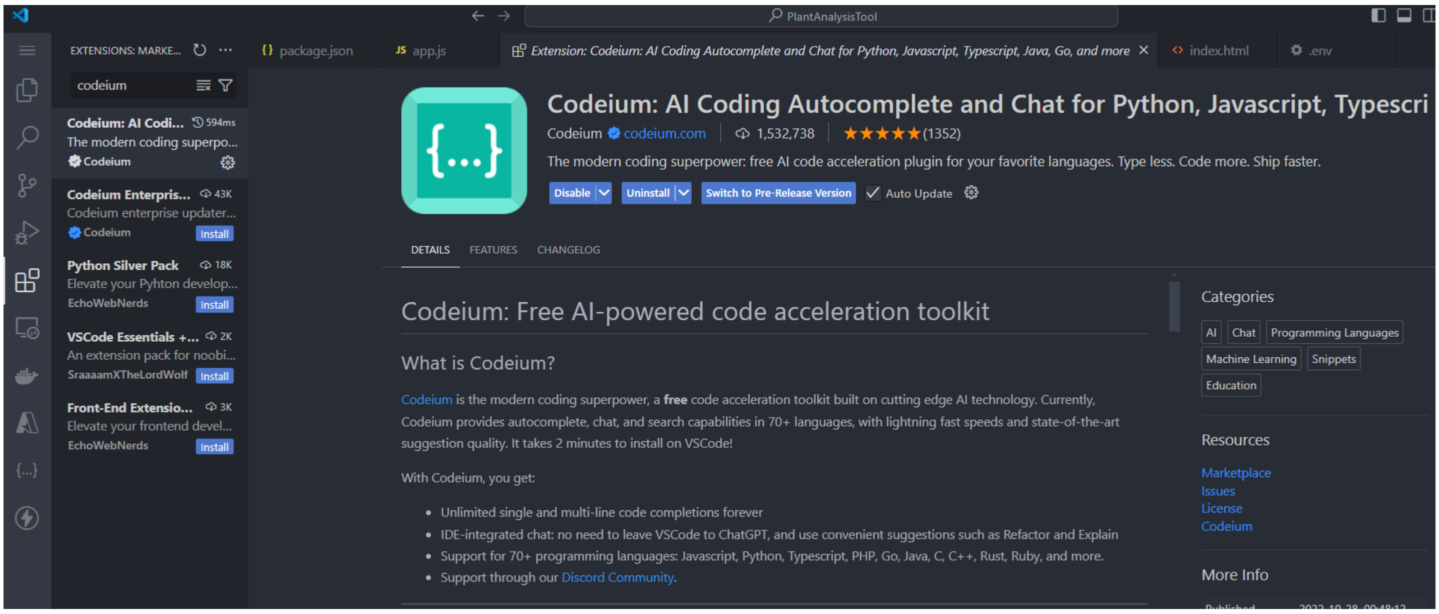
Save 60%

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.

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.


PlantAnalysisTool
EXTENSIONS: MARKE...
codeium
{} package.json
JS app.js
Extension: Codeium: AI Coding Autocomplete and Chat for Python, Javascript, Typescript, Java, Go, and more
index.html
.env

Codeium: AI Codi... 594ms
The modern coding superpo...
Codeium

Codeium Enterpris... 43K
Codeium enterprise updater...
Codeium Install

Python Silver Pack 18K
Elevate your Pyhton develop...
EchoWebNerds Install

VSCode Essentials +... 2K
An extension pack for noobi...
SraaaamXTheLordWolf 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.

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.

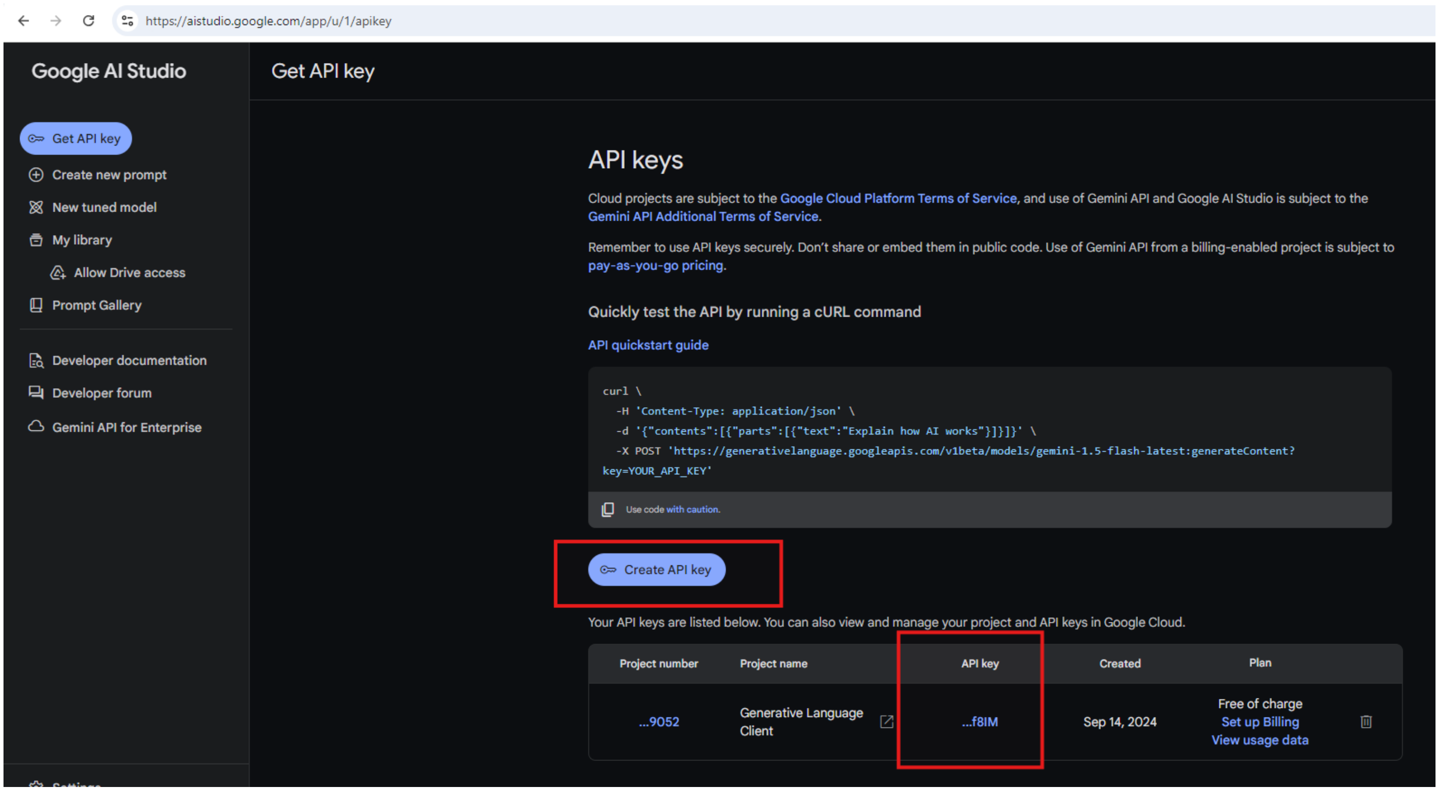
Categories

AI | Chat | Programming Languages | Machine Learning | Snippets | Education

Resources

Marketplace
Issues
License
Codeium

More Info



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

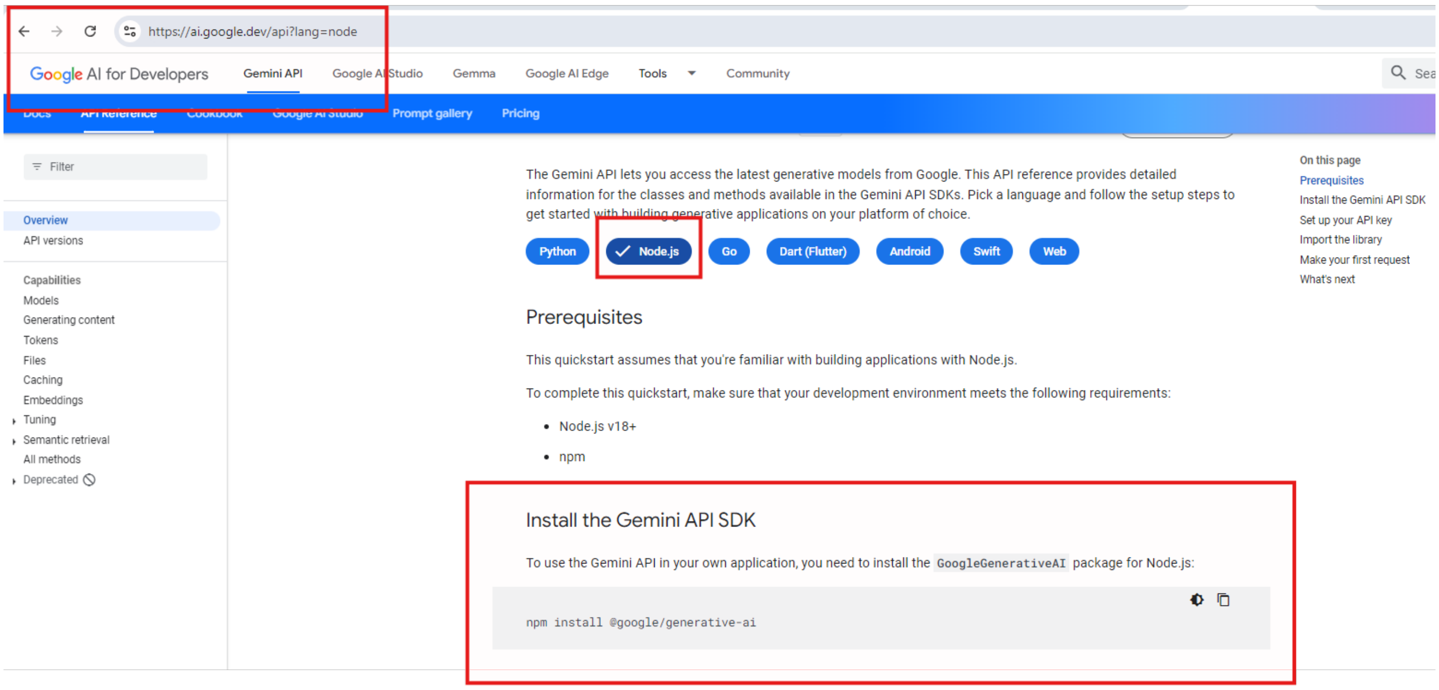
```
curl \
  -H 'Content-Type: application/json' \
  -d '{"contents":[{"parts":[{"text":"Explain how AI works"}]}]}' \
  -X POST 'https://generativelanguage.googleapis.com/v1beta/models/gemini-1.5-flash-latest:generateContent?
key=YOUR_API_KEY'
```

Use code with caution.

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	...f8lM	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:



https://ai.google.dev/api?lang=node

Google AI for Developers | Gemini API | Google AI Studio | Gemma | Google AI Edge | Tools | Community

Docs | API Reference | Cookbook | Google AI Studio | Prompt gallery | Pricing

Filter

Overview
API versions

Capabilities
Models
Generating content
Tokens
Files
Caching
Embeddings
Tuning
Semantic retrieval
All methods
Deprecated

The Gemini API lets you access the latest generative models from Google. This API reference provides detailed information for the classes and methods available in the Gemini API SDKs. Pick a language and follow the setup steps to get started with building generative applications on your platform of choice.

Python | ✓ Node.js | Go | Dart (Flutter) | Android | Swift | Web

On this page
Prerequisites
Install the Gemini API SDK
Set up your API key
Import the library
Make your first request
What's next

Prerequisites

This quickstart assumes that you're familiar with building applications with Node.js.

To complete this quickstart, make sure that your development environment meets the following requirements:

- Node.js v18+
- npm

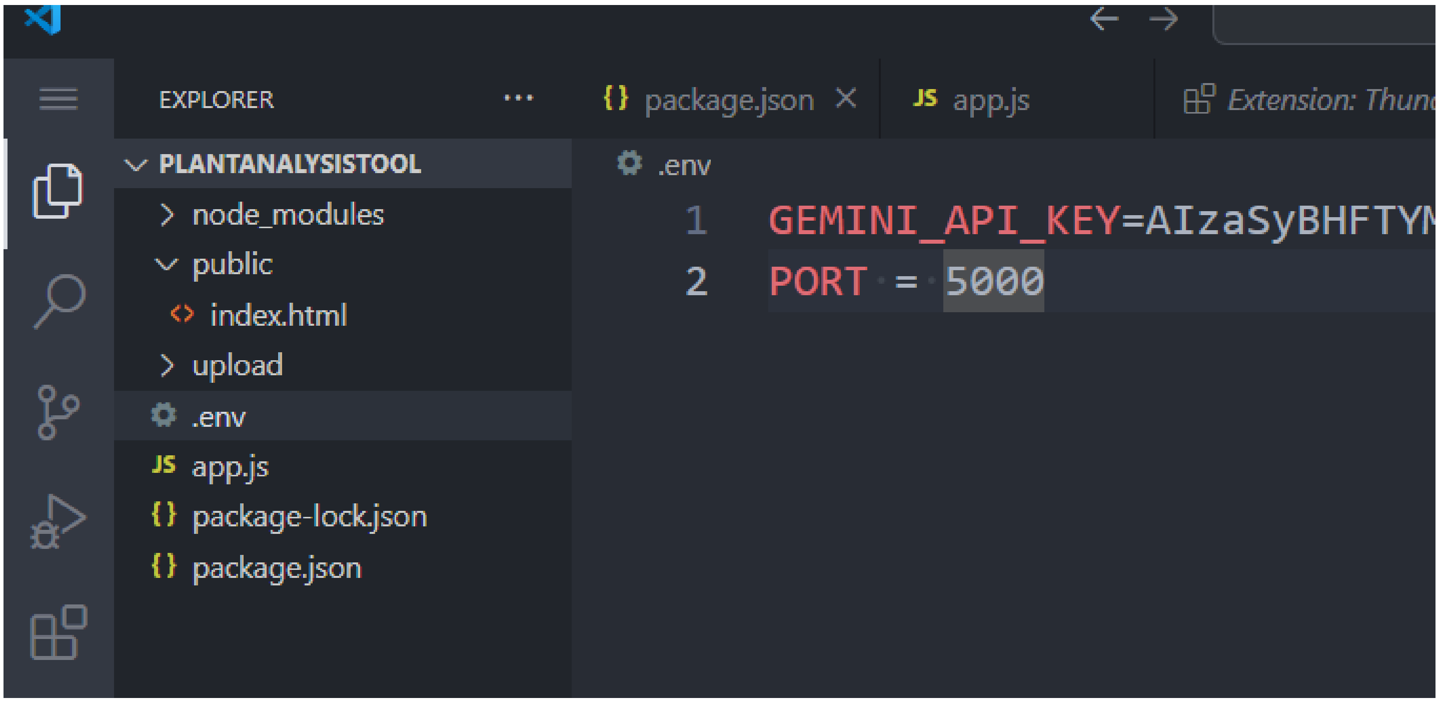
Install the Gemini API SDK

To use the Gemini API in your own application, you need to install the `GoogleGenerativeAI` package for Node.js:

```
npm install @google/generative-ai
```

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

.ENV FILE:



EXPLORER

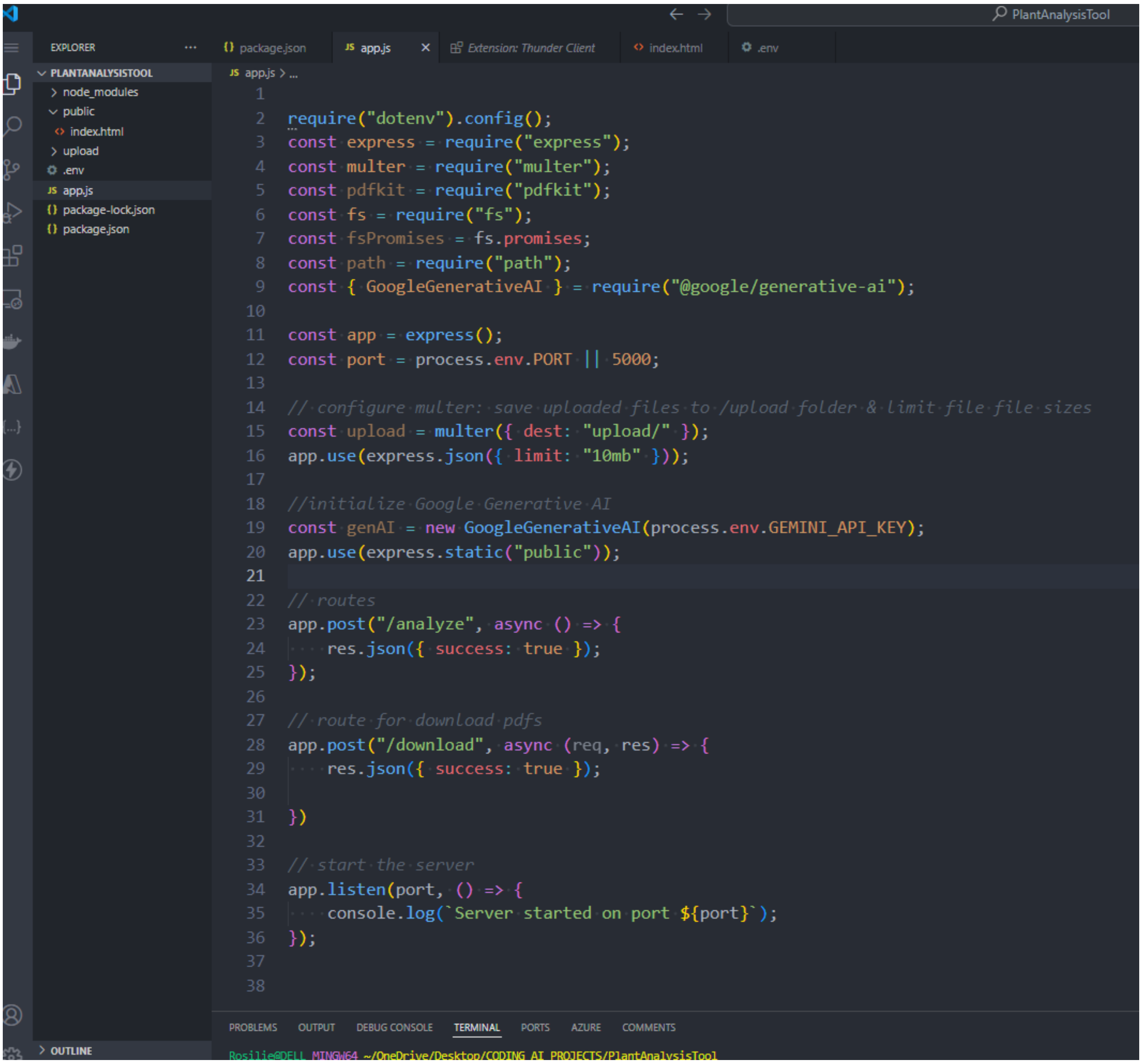
PLANTANALYSISTOOL
- node_modules
- public
  - index.html
- upload
- .env
- app.js
- package-lock.json
- package.json

package.json | app.js | Extension: Thun

.env

```
GEMINI_API_KEY=AIzaSyBHFTY
PORT = 5000
```

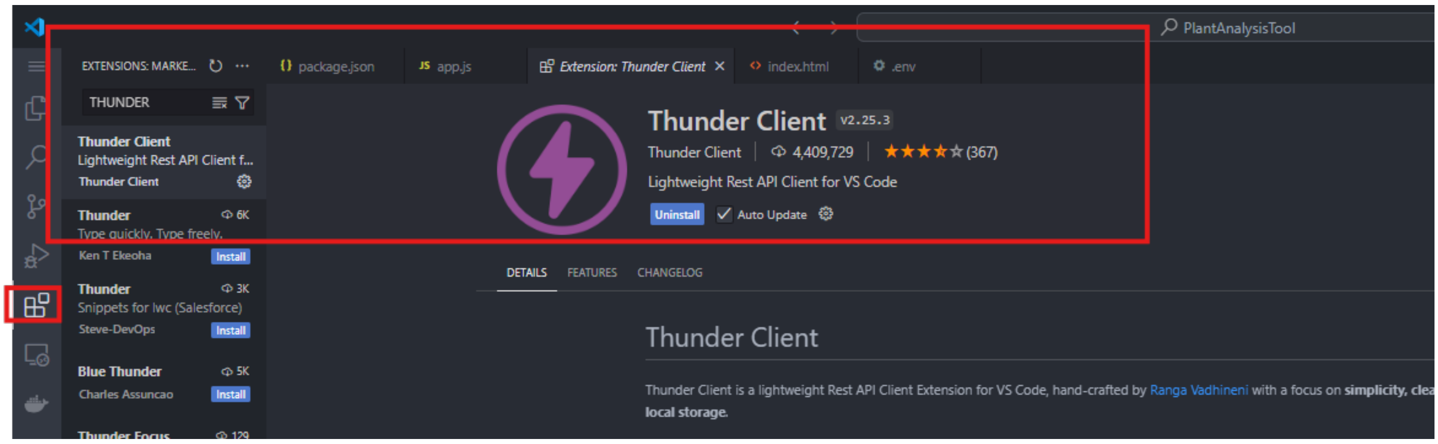
APPS.JS



```

require("dotenv").config();
const express = require("express");
const multer = require("multer");
const pdfkit = require("pdfkit");
const fs = require("fs");
const fsPromises = fs.promises;
const path = require("path");
const { GoogleGenerativeAI } = require("@google/generative-ai");

const app = express();
const port = process.env.PORT || 5000;

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

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

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

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

})

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


```

```
Rosilie@DELL MINGW64 ~/OneDrive/Desktop/CODING AI PROJECTS/PlantAnalysisTool
```

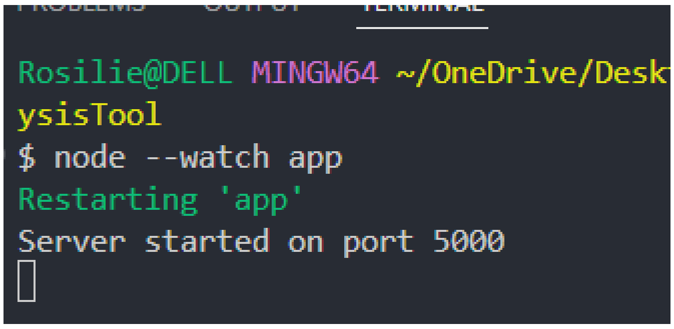
5. To test our ENDPOINT, we will use POSTMAN (you used INSOMNIA) or we can install the VS CODE EXTENSION, THUNDER CLIENT.



Thunder Client is a lightweight Rest API Client Extension for VS Code, hand-crafted by Ranga Vadhineni with a focus on simplicity, clea
local storage.

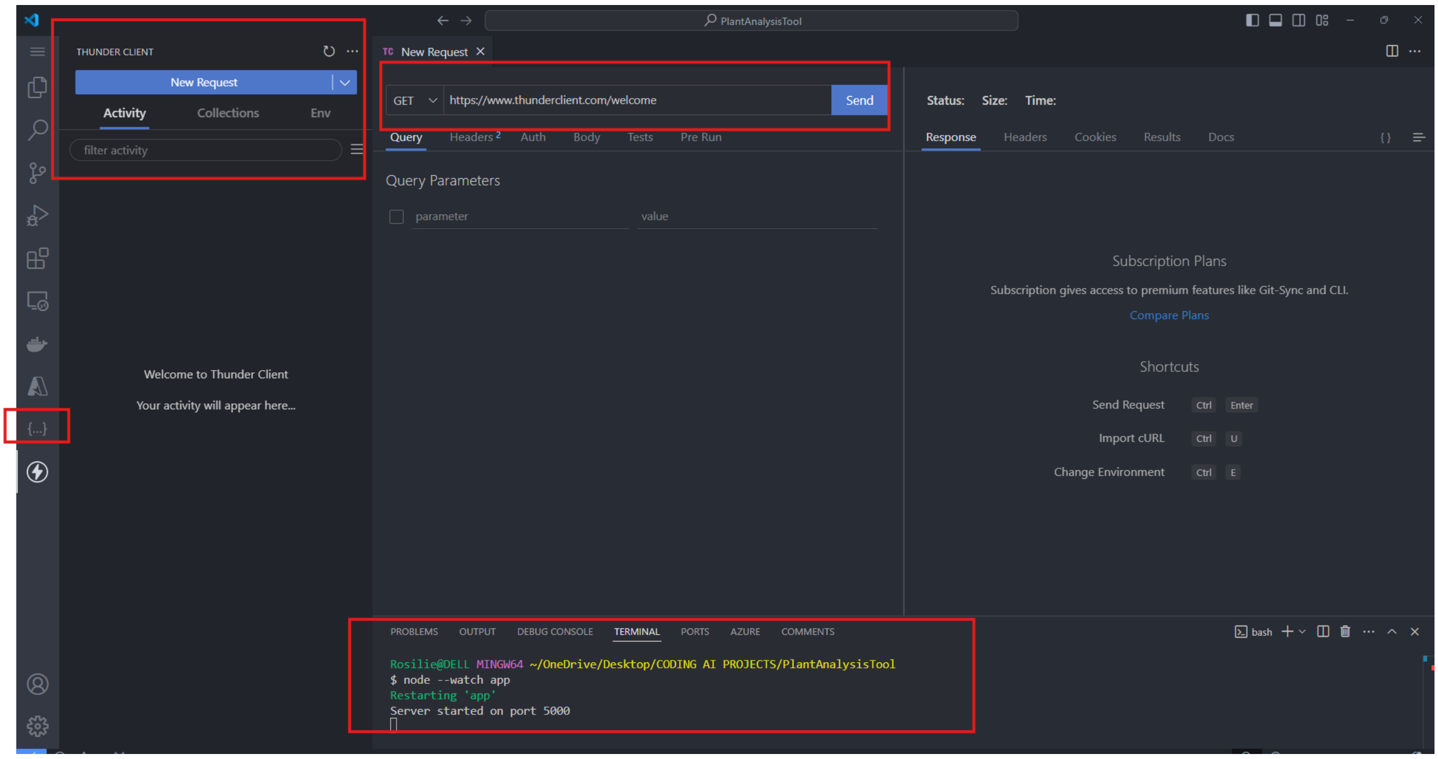
6. Run the app by issuing this code.

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



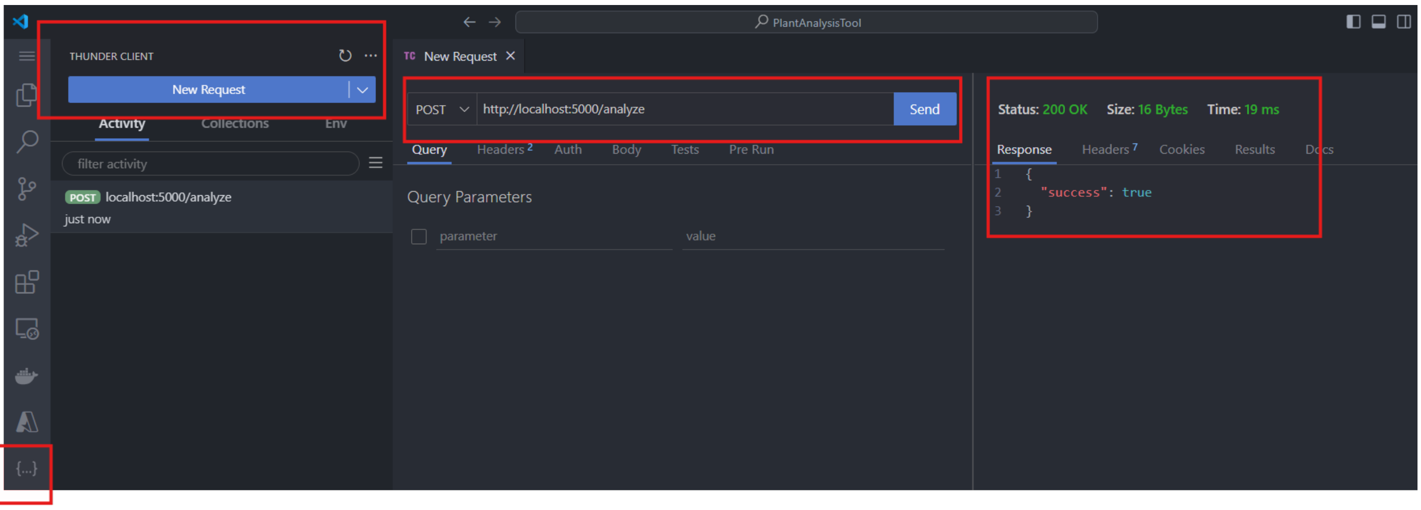
```
Rosilie@DELL MINGW64 ~/OneDrive/Desk
ysisTool
$ node --watch app
Restarting 'app'
Server started on port 5000
```

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.



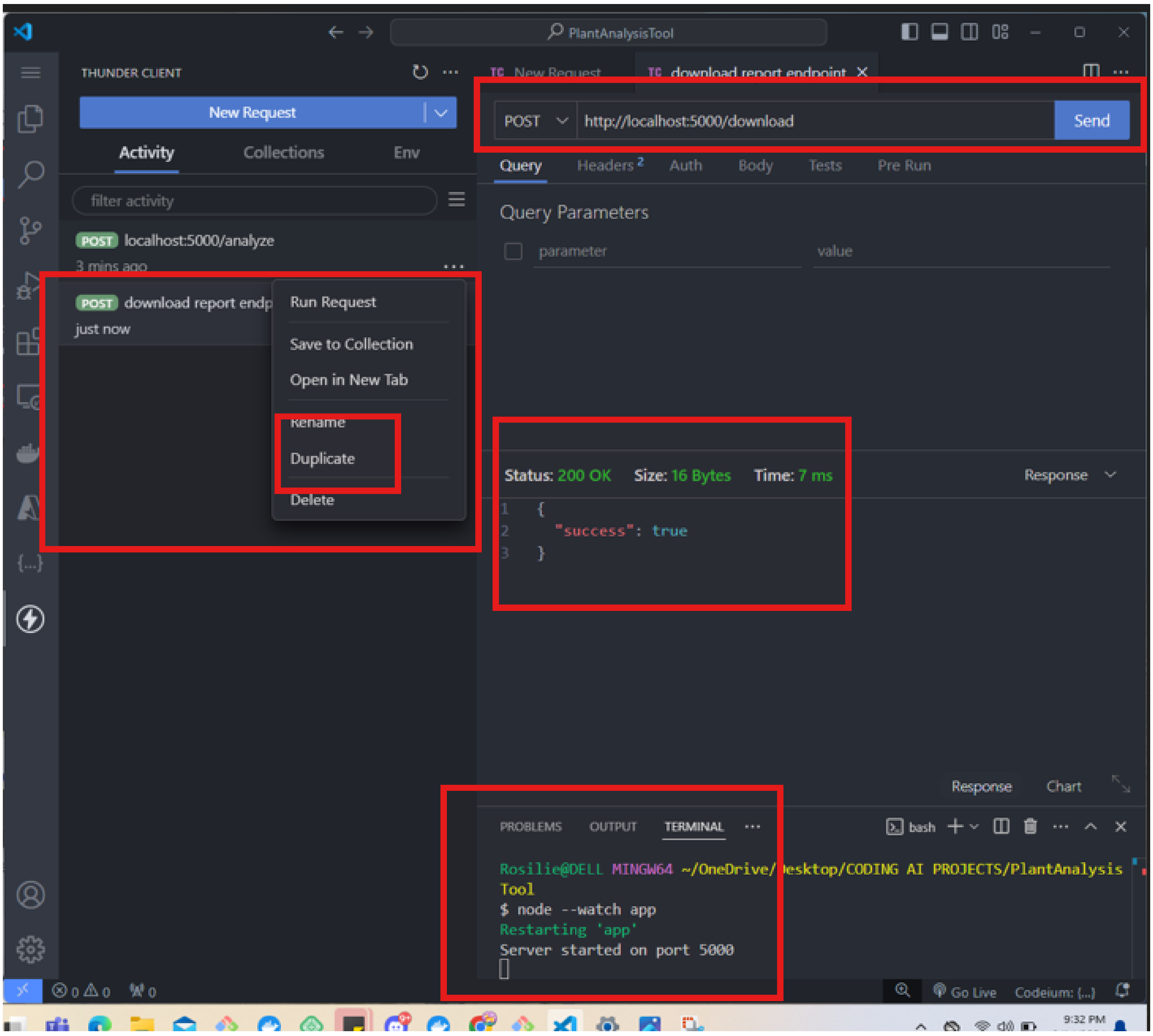
```
Rosilie@DELL MINGW64 ~/OneDrive/Desktop/CODING AI PROJECTS/PlantAnalysisTool
$ node --watch app
Restarting 'app'
Server started on port 5000
```

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



```
{
  "success": true
}
```

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



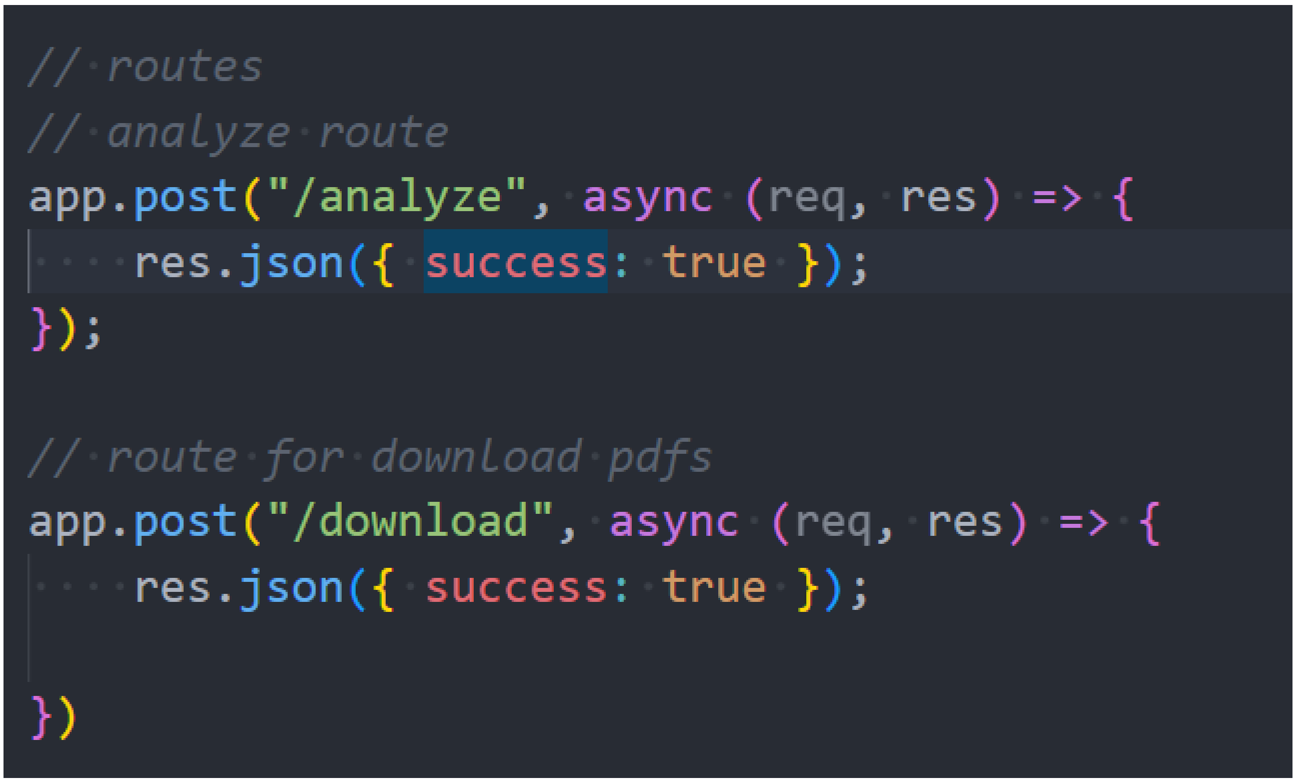
```
POST http://localhost:5000/download
```

Status: 200 OK   Size: 16 Bytes   Time: 7 ms

```
{
  "success": true
}
```

```
Rosilie@DELL MINGW64 ~/OneDrive/Desktop/CODING AI PROJECTS/PlantAnalysis
Tool
$ node --watch app
Restarting 'app'
Server started on port 5000
```

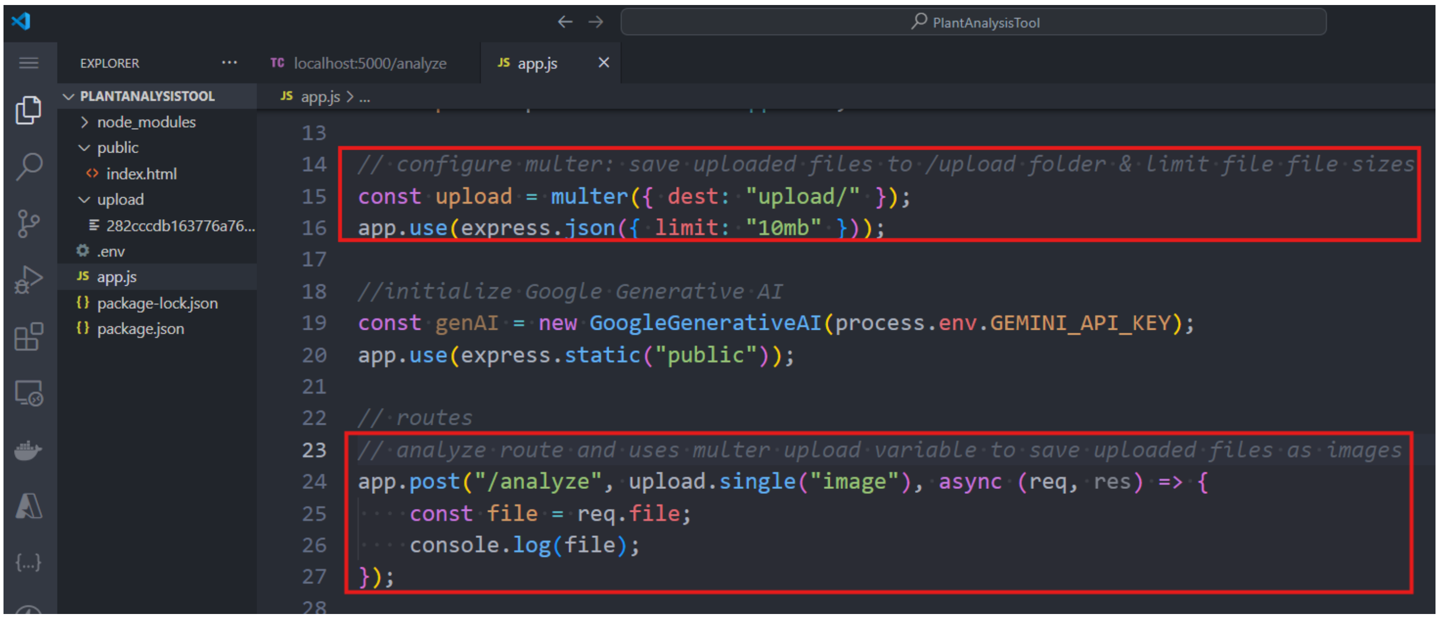
10. Just like in Django where we test our paths using Django's views.py, the logic for Node.js is this:



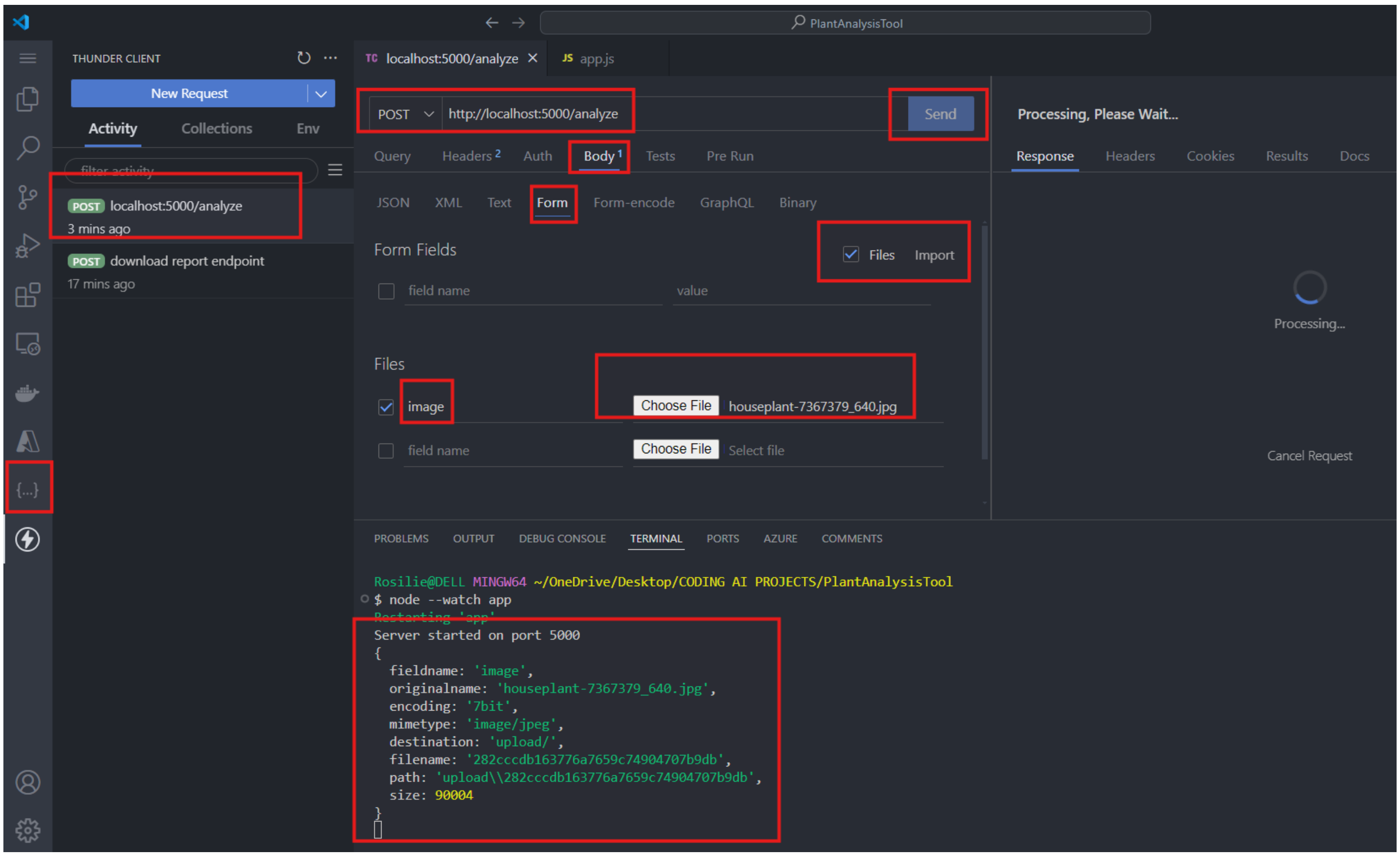
```javascript
// routes
// analyze route
app.post("/analyze", async (req, res) => {
    res.json({ success: true });
});

// 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:

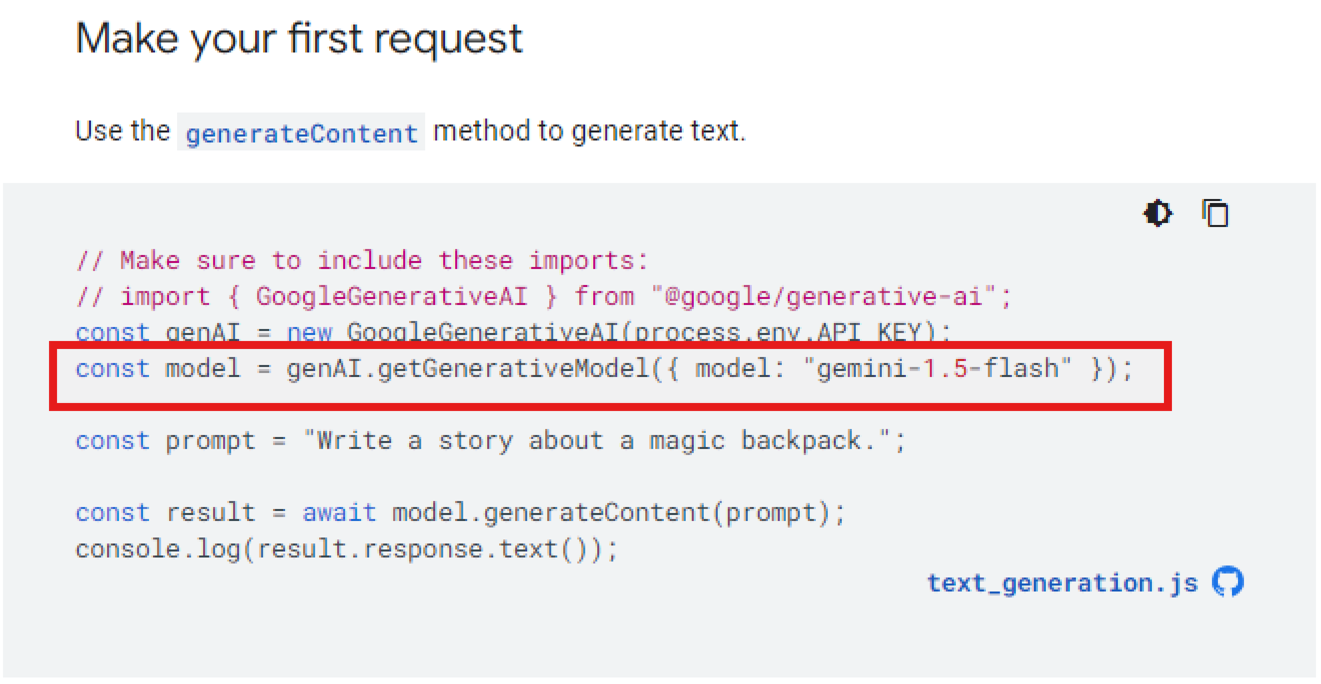


```javascript
// configure multer: save uploaded files to /upload folder & limit file file sizes
const upload = multer({ dest: "upload/" });
app.use(express.json({ limit: "10mb" }));

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

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



```
Rosilie@DELL MINGW64 ~/OneDrive/Desktop/CODING AI PROJECTS/PlantAnalysisTool
$ node --watch app
Restarting 'app'
Server started on port 5000
{
  fieldname: 'image',
  originalname: 'houseplant-7367379_640.jpg',
  encoding: '7bit',
  mimetype: 'image/jpeg',
  destination: 'upload/',
  filename: '282cccdb163776a7659c74904707b9db',
  path: 'upload\\282cccdb163776a7659c74904707b9db',
  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

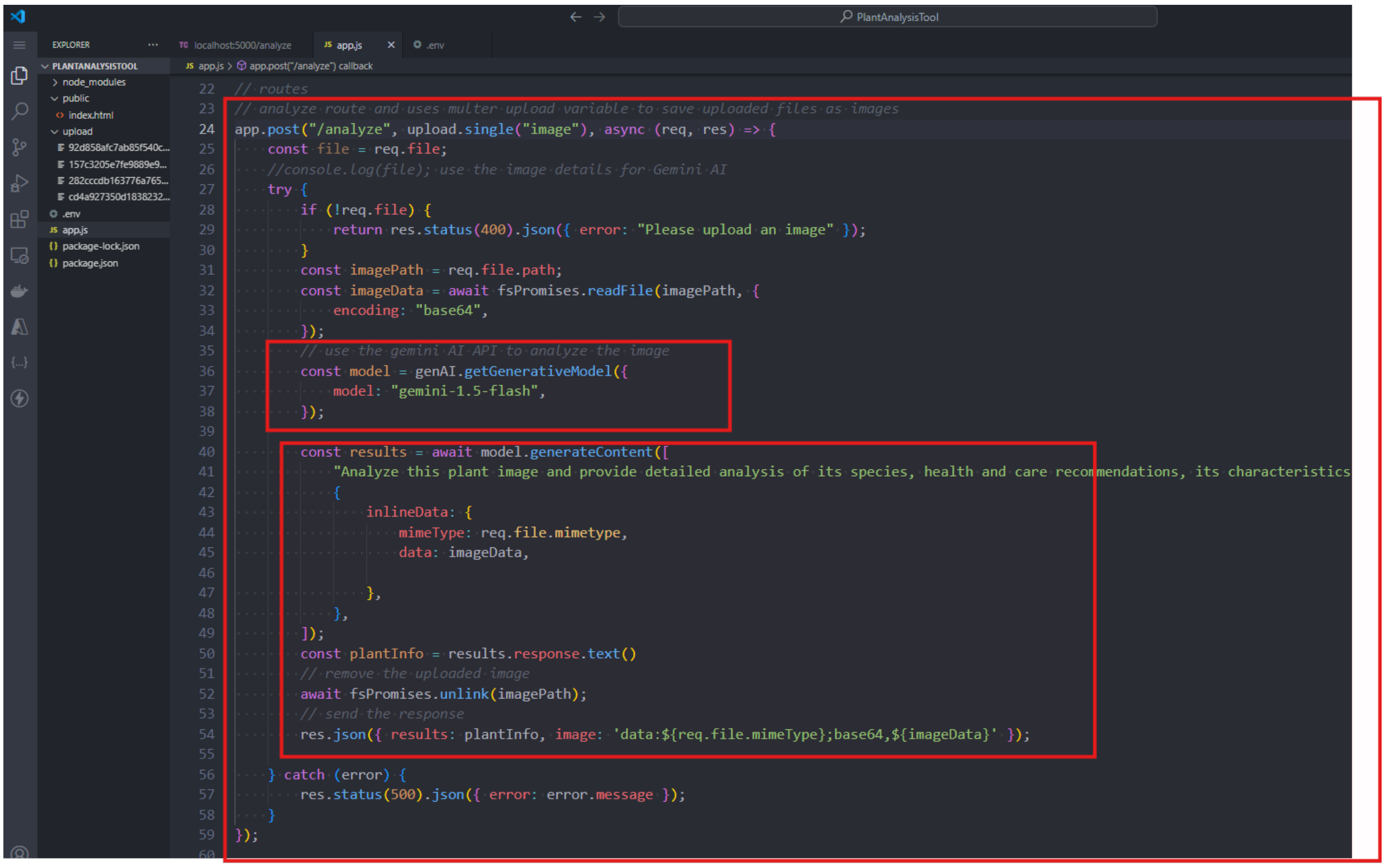
Use the `generateContent` method to generate text.

```
// 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());
```

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

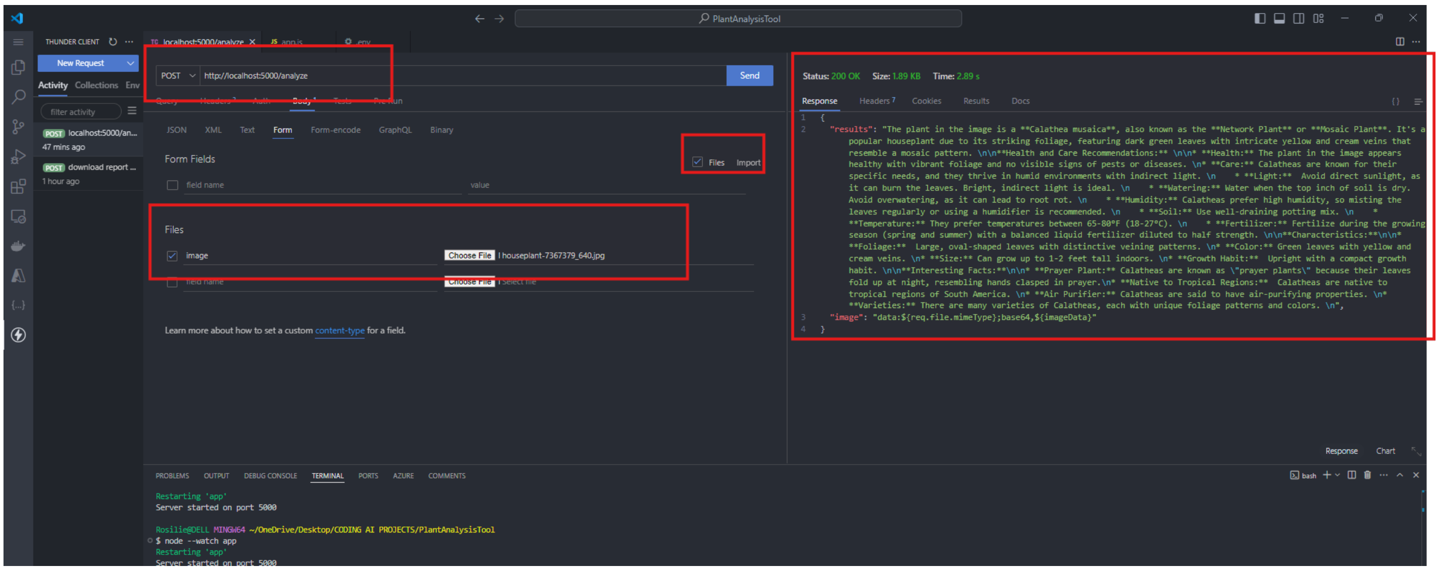
This is the PROMPT we used for Gemini `"Analyze this plant image and provide detailed analysis of its species, health and care recommendations, its characteristics, care instructions and interesting facts. Please provide the response in plain text without using any markdown formatting "`

Our function:



```javascript
// routes
// analyze route and uses multer upload variable to save uploaded files as images
app.post("/analyze", upload.single("image"), async (req, res) => {
  const file = req.file;
  //console.log(file); use the image details for Gemini AI
  try {
    if (!req.file) {
      return res.status(400).json({ error: "Please upload an image" });
    }
    const imagePath = req.file.path;
    const imageData = await fsPromises.readFile(imagePath, {
      encoding: "base64",
    });
    // use the gemini AI API to analyze the image
    const model = genAI.getGenerativeModel({
      model: "gemini-1.5-flash",
    });

    const results = await model.generateContent([
      "Analyze this plant image and provide detailed analysis of its species, health and care recommendations, its characteristics
      {
        inlineData: {
          mimeType: req.file.mimetype,
          data: imageData,

        },
      },
    ]);
    const plantInfo = results.response.text()
    // remove the uploaded image
    await fsPromises.unlink(imagePath);
    // send the response
    res.json({ results: plantInfo, image: 'data:${req.file.mimeType};base64,${imageData}' });

  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```

14. We run our endpoint using Thunder Client:



POST http://localhost:5000/analyze

Form Fields

Files ✓ image — houseplant-7367379_640.jpg

Learn more about how to set a custom content-type for a field.

Status: 200 OK Size: 1.89 KB Time: 2.89 s

```
{
  "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 diseases. \n* **Care:** Calatheas are known for their specific needs, and they thrive in humid environments with indirect light. \n    * **Light:**  Avoid direct sunlight, as it can burn the leaves. Bright, indirect light is ideal. \n    * **Watering:** Water when the top inch of soil is dry. Avoid overwatering, as it can lead to root rot. \n    * **Humidity:** Calatheas prefer high humidity, so misting the leaves regularly or using a humidifier is recommended. \n    * **Soil:** Use well-draining potting mix. \n    * **Temperature:** They prefer temperatures between 65-80°F (18-27°C). \n    * **Fertilizer:** Fertilize during the growing season (spring and summer) with a balanced liquid fertilizer diluted to half strength. \n\n**Characteristics:**\n\n* **Foliage:**  Large, oval-shaped leaves with distinctive veining patterns. \n* **Color:** Green leaves with yellow and cream veins. \n* **Size:** Can grow up to 1-2 feet tall indoors. \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* **Native to Tropical Regions:**  Calatheas are native to tropical regions of South America. \n* **Air Purifier:** Calatheas are said to have air-purifying properties. \n* **Varieties:** There are many varieties of Calatheas, each with unique foliage patterns and colors. \n",
  "image": "data:${req.file.mimeType};base64,${imageData}"
}
```

```
Restarting 'app'
Server started on port 5000

Rosilie@DELL MINGW64 ~/OneDrive/Desktop/CODING AI PROJECTS/PlantAnalysisTool
$ node --watch app
Restarting 'app'
Server started on port 5000
```

15.