

Topic: Creating Plant Analysis Tools Using Gemini AI and Express.js

Speaker: / Notebook: Django Project: Car Listing



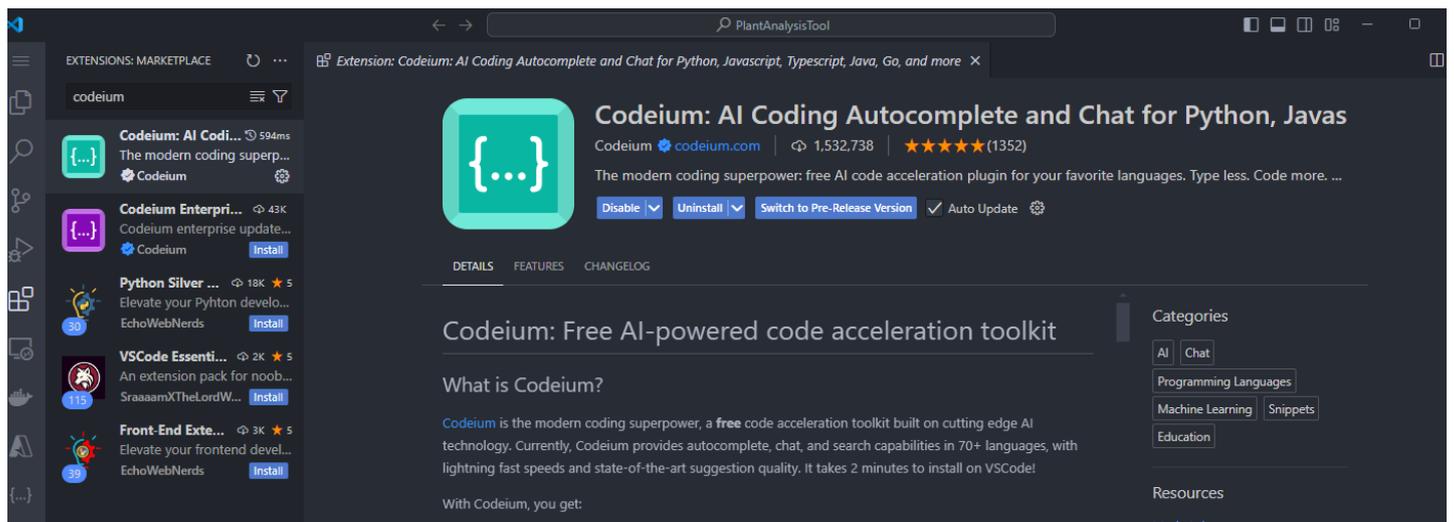
We installed the NODE.JS (Javascript) for this project. You need to [download from here](#).

Node.js for Javascript and Django for Python are 2 different platforms.

Project's Main Resource Page: [YouTube video here](#).

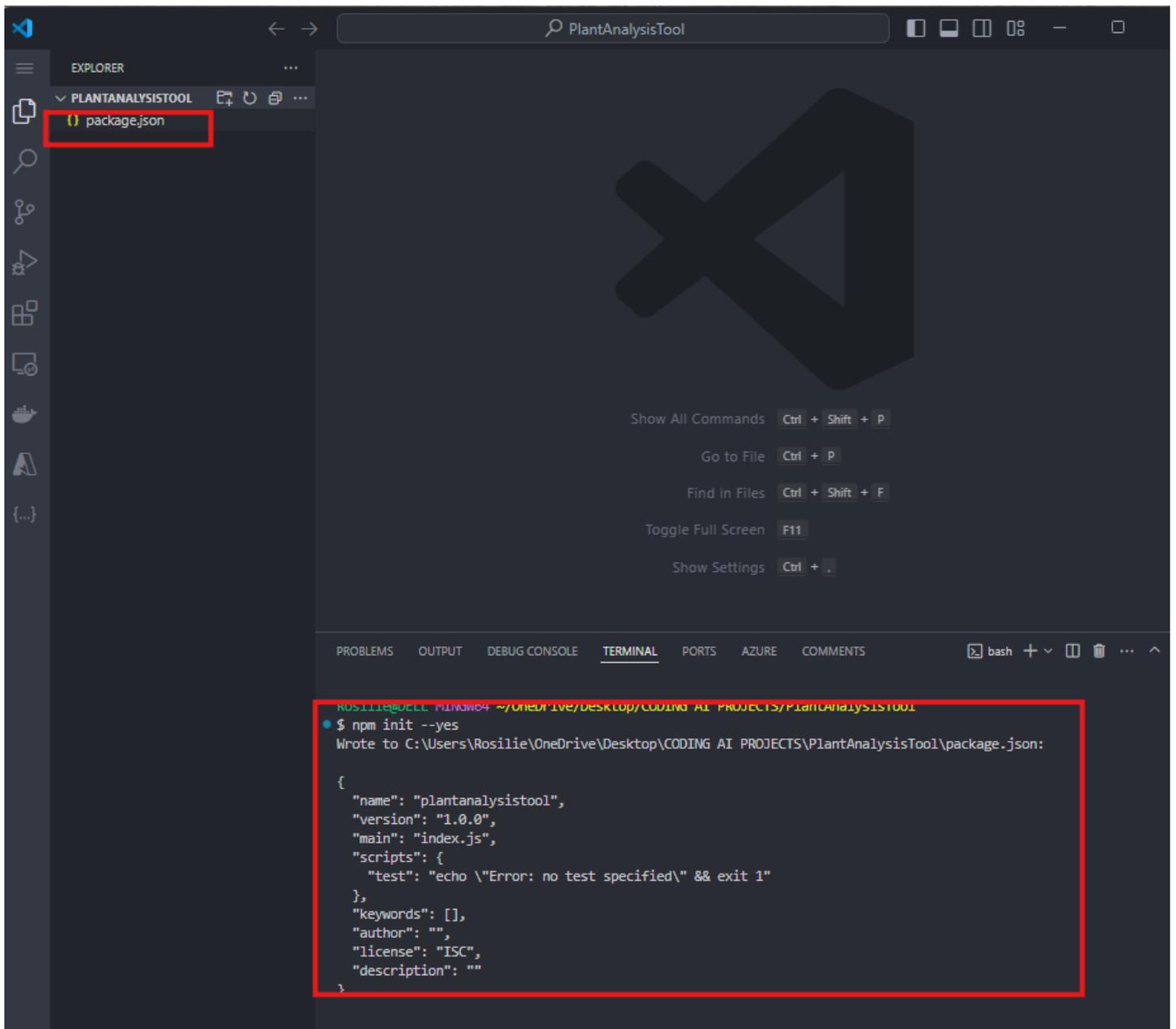
1. Open a new folder, PLANTANALYSISTOOL. Open a new Git bash terminal, and we issue the command, CODE . (code dot) to open the folder in VS Code.

We install the extension CODEIUM AI.



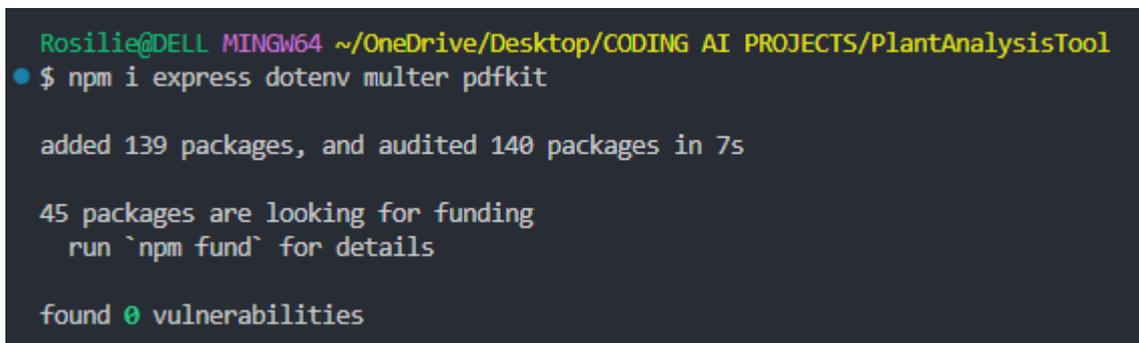
2. We issue the command for Node.js the NPM (which is PIP in Django) to create the PACKAGE.JSON file.

```
$ npm init --yes
```

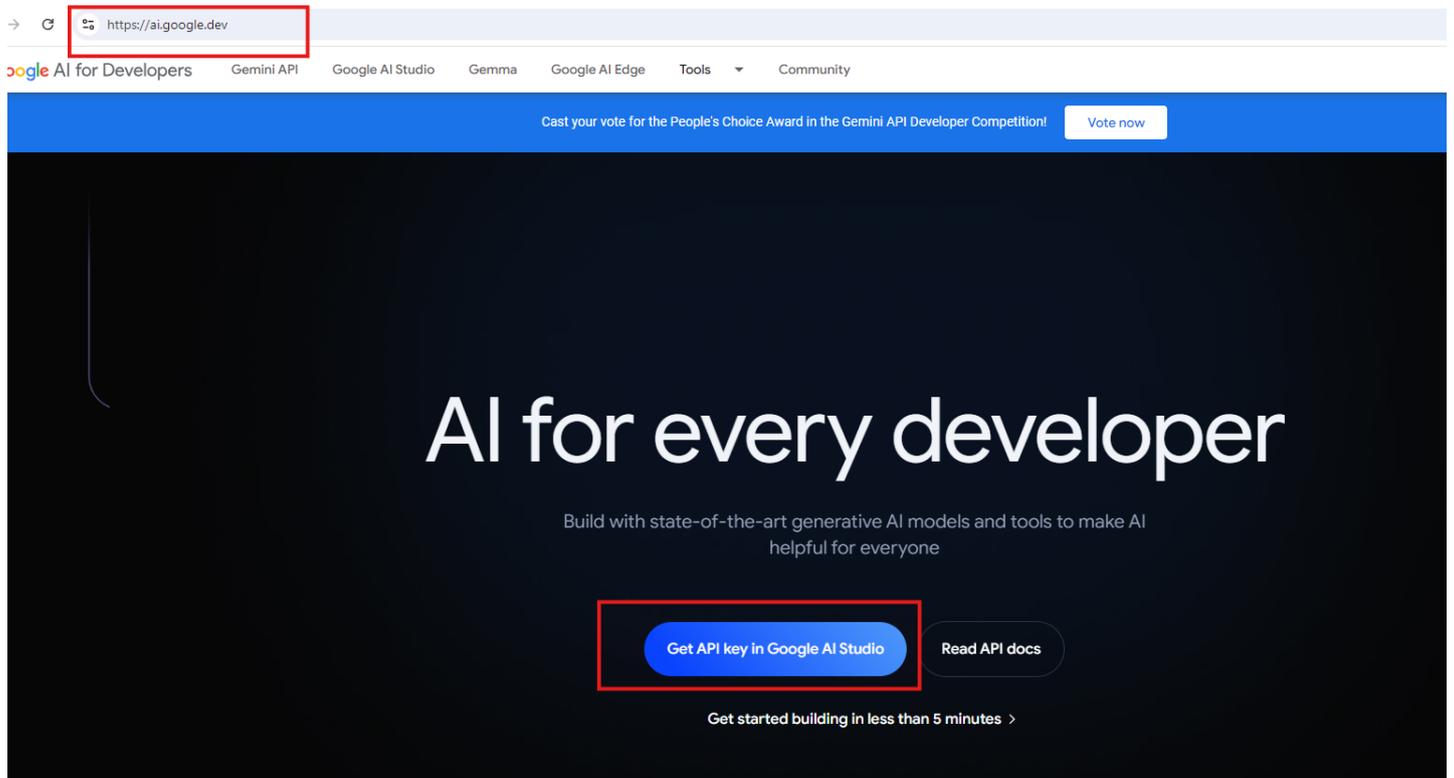


3. We install the packages we need.

```
$ npm i express dotenv multer pdfkit
```



4. Get GEMINI API KEY. Simply in the Google search, type GEMINI API.



5. Update the PACKAGE.JSON:

FROM:

```
1 {
2   "name": "plantanalysistool",
3   "version": "1.0.0",
4   "main": "index.js",
5   "scripts": {
6     "test": "echo \\\"Error: no test specified\\\" && exit 1"
7   },
8   "keywords": [],
9   "author": "",
10  "license": "ISC",
11  "description": "",
12  "dependencies": {
13    "dotenv": "^16.4.5",
14    "express": "^4.21.0",
15    "multer": "^1.4.5-lts.1",
16    "pdfkit": "^0.15.0"
17  }
18 }
19
```

TO:

6. We created files APP.JS and .ENV files in the root directory.

In the APPS.JS:

```
JS app.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");
```

7. On Google AI Dashboard, select API REFERENCE, choose NODE.JS and get the code to install GEMINI API

Google AI for Developers

API Reference

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 `@google/generative-ai` package for Node.js:

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

Set up your API key

To use the Gemini API, you'll need an API key. If you don't already have one, create a key in Google AI Studio.

[Get an API key from Google AI Studio](#)

8. Install the GEMINI API SDK in the terminal:

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

```
Rosilie@DELL MINGW64 ~/OneDrive/Desktop/CODING AI PROJECTS/PlantAnalysisTool
$ npm install @google/generative-ai
added 1 package, and audited 141 packages in 2s
45 packages are looking for funding
run `npm fund` for details
found 0 vulnerabilities
Rosilie@DELL MINGW64 ~/OneDrive/Desktop/CODING AI PROJECTS/PlantAnalysisTool
$
```

9. We copy this from the GEMINI API REFERENCE into our APPS.JS

Import the library

Import the Google Generative AI library.

```
const { GoogleGenerativeAI } = require("@google/generative-ai");
```

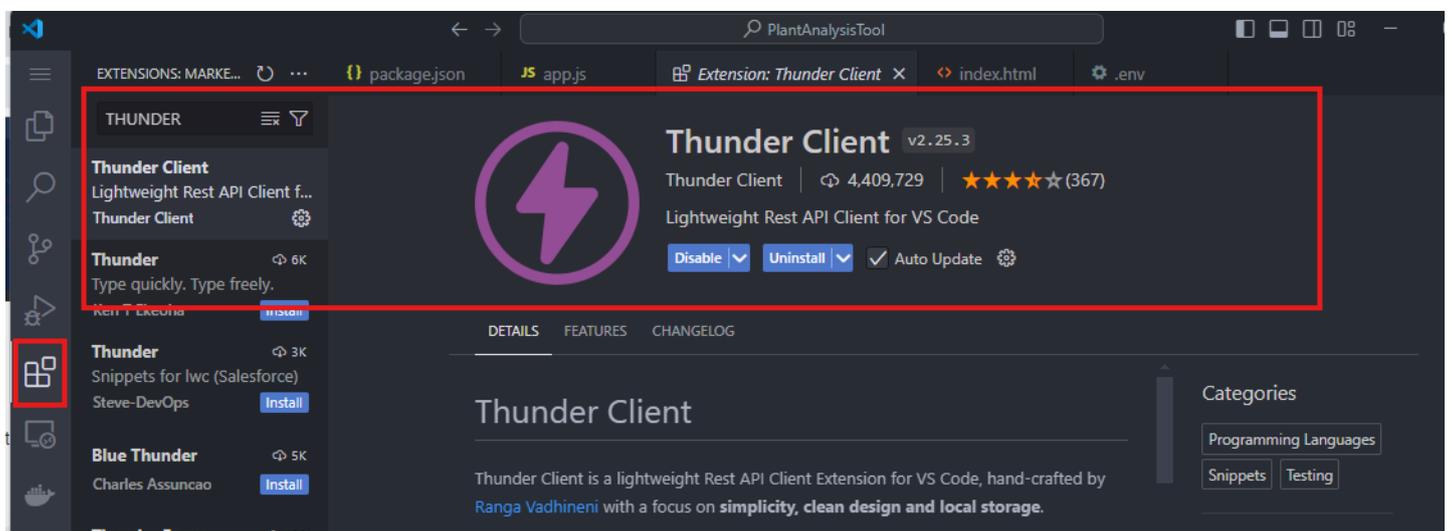
A screenshot of the Visual Studio Code editor. The Explorer sidebar on the left shows a project named 'PLANTANALYSISTOOL' with files like 'node_modules', '.env', 'app.js', 'package-lock.json', and 'package.json'. The 'app.js' file is selected and highlighted with a red box. The main editor area shows the content of 'app.js' with line numbers 1 through 10. Line 9 contains the code `const { GoogleGenerativeAI } = require("@google/generative-ai");`, which is highlighted with a red box. Other code in the file includes `require("dotenv").config();`, `const express = require("express");`, `const multer = require("multer");`, `const pdfkit = require("pdfkit");`, `const fs = require("fs");`, `const fsPromises = fs.promises;`, and `const path = require("path");`. The bottom right corner of the editor shows the text 'Ctrl+I Codeium'.

10. We updated our APPS.JS AS:

```
1 require("dotenv").config();
2 const express = require("express");
3 const multer = require("multer");
4 const pdfkit = require("pdfkit");
5 const fs = require("fs");
6 const fsPromises = fs.promises;
7 const path = require("path");
8 const { GoogleGenerativeAI } = require("@google/generative-ai");
9
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
```

11. To test our work, run `NODE --WATCH APP` (where app is our APP.JS)

Then use POSTMAN (or INSOMNIA) or add an extension THUNDER CLIENT as a VSCODE extension to test the ENDPOINT WITHIN VS CODE.



12.

