

Topic: Plant Analysis Tool using Gemini AI and Express.js Part 2

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



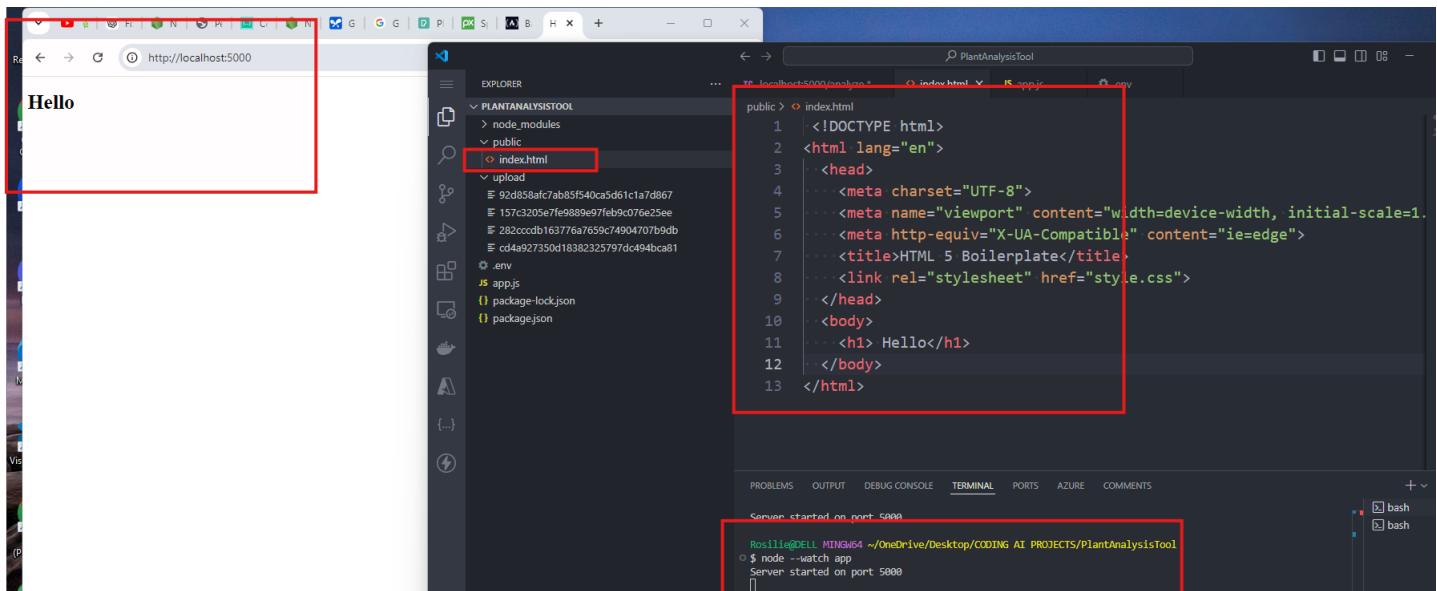
Previously, Gemini and Node.js have created the plant analysis for an uploaded image of a plant.

[GITHUB REPO](#)

REFERENCE: <https://github.com/tweneboah/Full-Stack-Web-Development-Bootcamp-Course/tree/main/PROJECTS/AI-PROJECTS/PLANT-ANALYSIS-TOOL/public>

A screenshot of a Thunder Client interface. The main window shows a POST request to 'http://localhost:5000/analyze'. The 'Body' tab is selected, showing a 'Form Fields' section with a 'file name' field and a 'value' field. Below this is a 'Files' section with a checked 'image' field and a 'Choose File' button. A file selection dialog is open, showing a list of files in the 'Downloads' folder, with 'philodendron-7960228_640.jpg' selected. The 'Response' tab shows a JSON response with a status of 200 OK, size of 1.76 KB, and time of 2.60 s. The response body contains a JSON object with 'results' and 'image' fields. The 'results' field is a detailed description of a Monstera deliciosa plant, and the 'image' field is a base64 encoded image of the plant.

1. Now, create the front end for our project. Update our INDEX.HTML



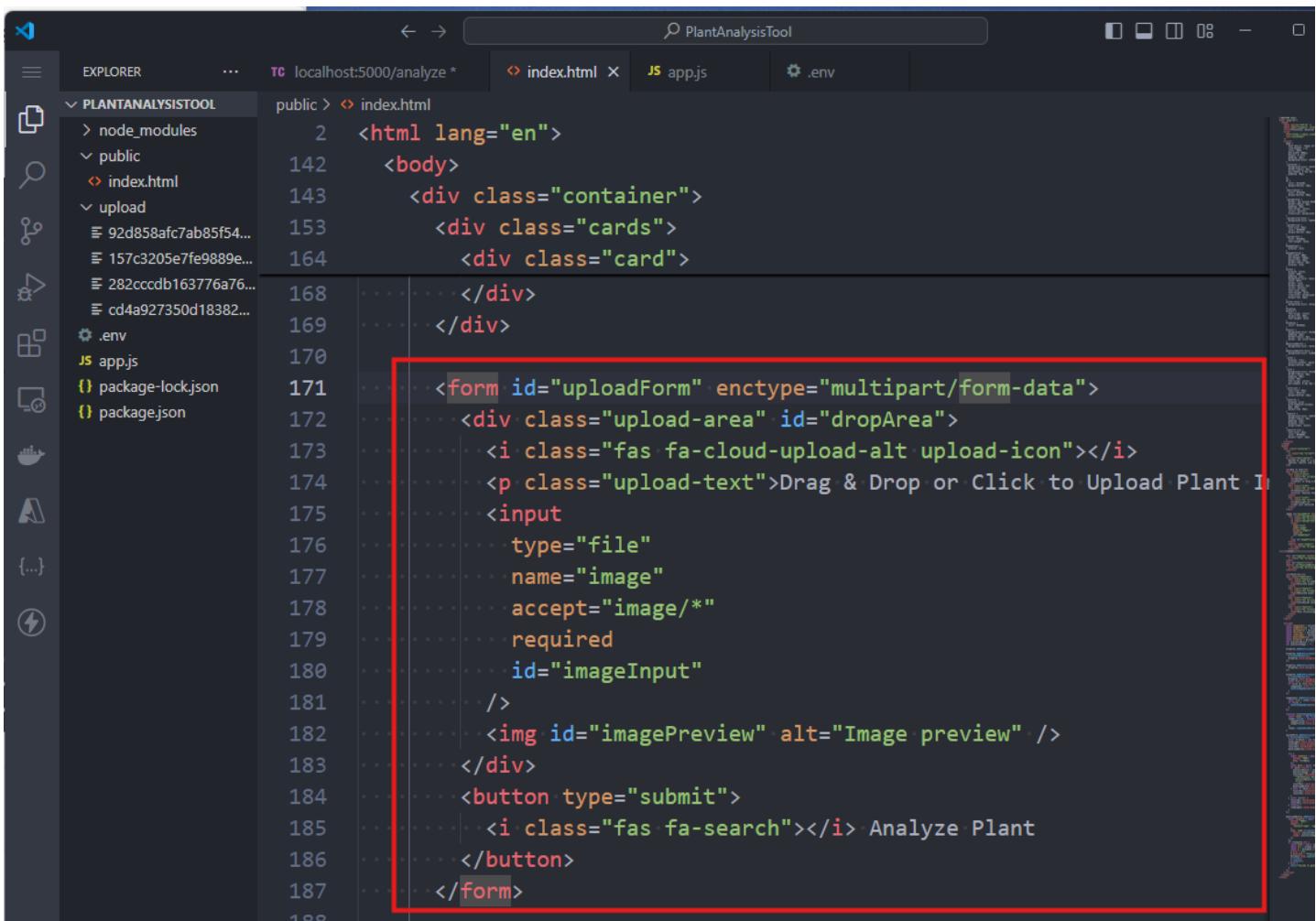
The screenshot shows a browser window with the URL `http://localhost:5000` and the text "Hello". To the right is the VS Code interface. The Explorer sidebar shows a project structure with a red box around the `index.html` file. The code editor shows the content of `index.html` with a red box around the entire code block. The terminal shows the command `$ node --watch app` and the message "Server started on port 5000".

```

public > index.html
1  | <!DOCTYPE html>
2  | <html lang="en">
3  |   <head>
4  |     <meta charset="UTF-8">
5  |     <meta name="viewport" content="width=device-width, initial-scale=1">
6  |     <meta http-equiv="X-UA-Compatible" content="ie=edge">
7  |     <title>HTML 5 Boilerplate</title>
8  |     <link rel="stylesheet" href="style.css">
9  |   </head>
10 |   <body>
11 |     <h1>Hello</h1>
12 |   </body>
13 | </html>

```

2. The focus of INDEX.HTML shall be the form for UPLOADFORM



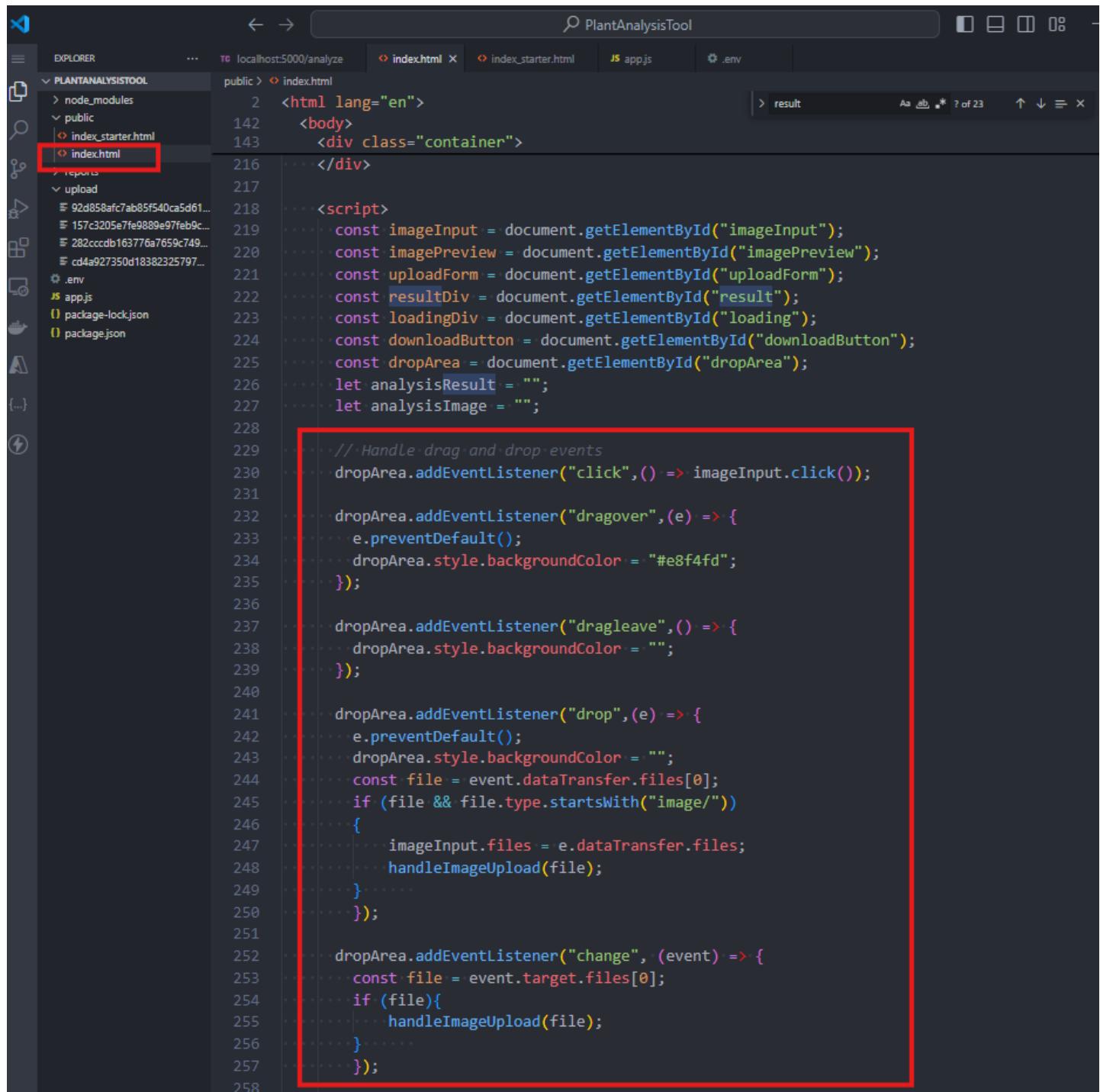
The screenshot shows the VS Code interface with the `index.html` file open. The code editor has a red box highlighting the `form` tag and its contents, which define a drag-and-drop upload area. The Explorer sidebar shows the project structure.

```

<form id="uploadForm" enctype="multipart/form-data">
  <div class="upload-area" id="dropArea">
    <i class="fas fa-cloud-upload-alt upload-icon"></i>
    <p class="upload-text">Drag & Drop or Click to Upload Plant Image</p>
    <input type="file" name="image" accept="image/*" required id="imageInput" />
    <img id="imagePreview" alt="Image preview" />
  </div>
  <button type="submit">
    <i class="fas fa-search"></i> Analyze Plant
  </button>
</form>

```

3. This is the code for UPLOAD using DRAG AND DROP OF IMAGE. INDEX.HTML is written as:



PlantAnalysisTool

localhost:5000/analyze

index.html

index_starter.html

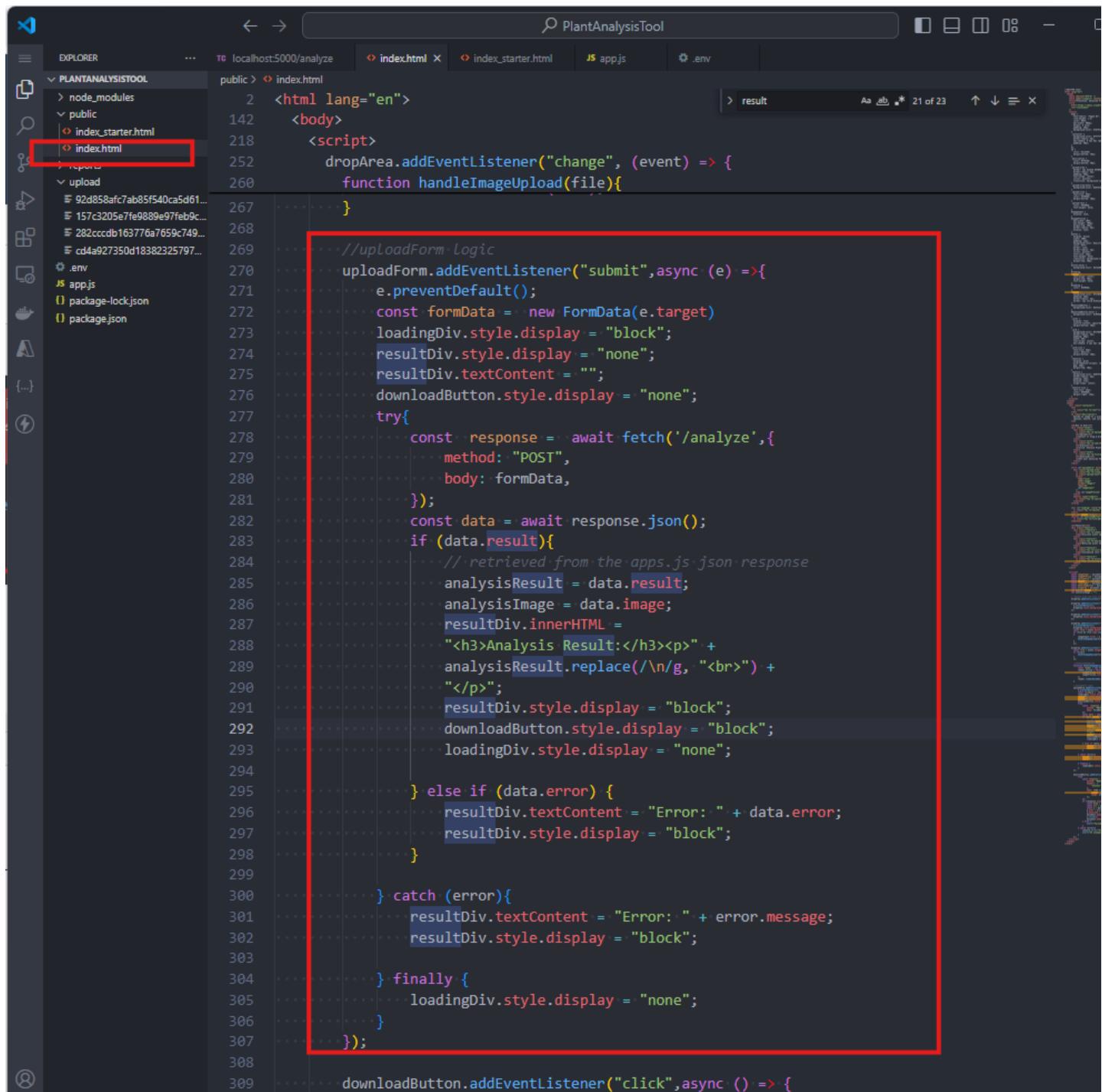
app.js

.env

index.html

```
2  <html lang="en">
142  <body>
143  <div class="container">
216  </div>
217
218  <script>
219  const imageInput = document.getElementById("imageInput");
220  const imagePreview = document.getElementById("imagePreview");
221  const uploadForm = document.getElementById("uploadForm");
222  const resultDiv = document.getElementById("result");
223  const loadingDiv = document.getElementById("loading");
224  const downloadButton = document.getElementById("downloadButton");
225  const dropArea = document.getElementById("dropArea");
226  let analysisResult = "";
227  let analysisImage = "";
228
229  // Handle drag and drop events
230  dropArea.addEventListener("click", () => imageInput.click());
231
232  dropArea.addEventListener("dragover", (e) => {
233  e.preventDefault();
234  dropArea.style.backgroundColor = "#e8f4fd";
235 });
236
237  dropArea.addEventListener("dragleave", () => {
238  dropArea.style.backgroundColor = "";
239 });
240
241  dropArea.addEventListener("drop", (e) => {
242  e.preventDefault();
243  dropArea.style.backgroundColor = "";
244  const file = event.dataTransfer.files[0];
245  if (file && file.type.startsWith("image/"))
246  {
247  imageInput.files = e.dataTransfer.files;
248  handleImageUpload(file);
249  }
250 });
251
252  dropArea.addEventListener("change", (event) => {
253  const file = event.target.files[0];
254  if (file){
255  handleImageUpload(file);
256  }
257 });
258
```

4. This is the EVENT for UPLOAD BUTTON in INDEX.HTML:

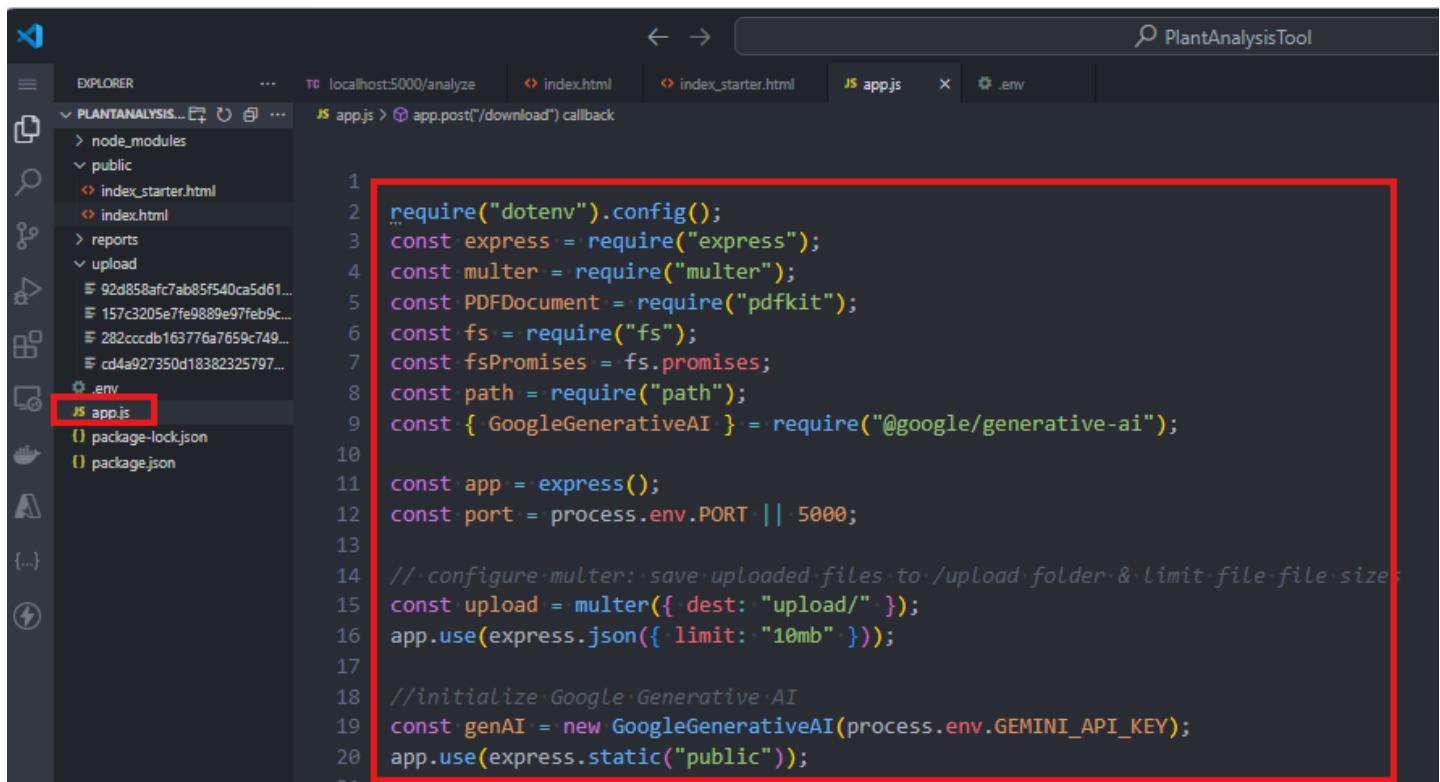


The screenshot shows a code editor interface with the title "PlantAnalysisTool". The left sidebar shows a file tree with the following structure:

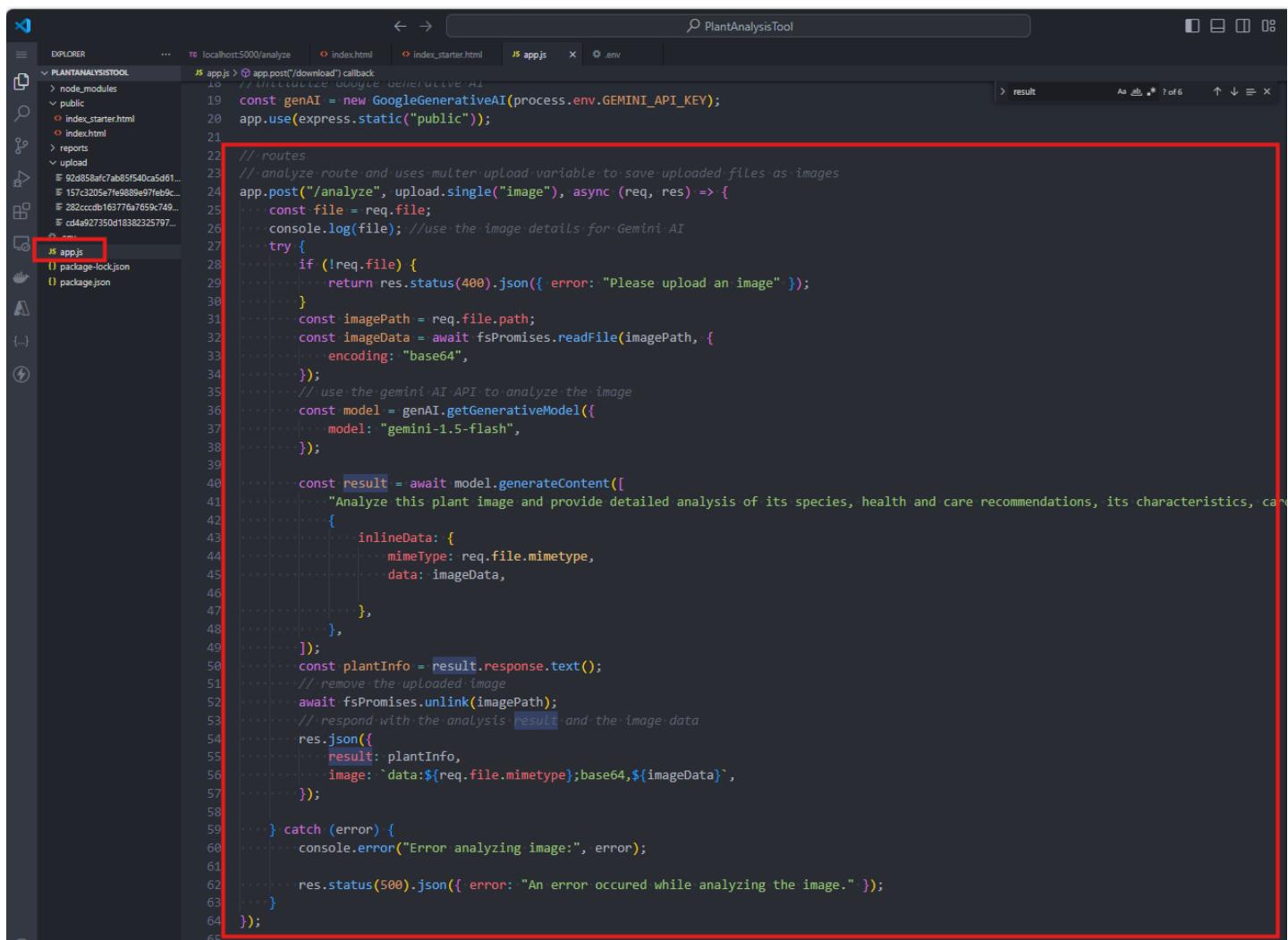
- PLANTANALYSISTOOL
 - node_modules
 - public
 - index_starter.html
 - index.html
 - upload
 - 92d858afc7ab85f540ca5d61...
 - 157c3205e7fe9889e97feb9c...
 - 282ccdb163776a7659c749...
 - cd4a927350d18382325797...
- .env
- app.js
- package-lock.json
- package.json

```
2   <html lang="en">
142  <body>
218    <script>
252      dropArea.addEventListener("change", (event) => {
260        function handleImageUpload(file){
267        }
268
269        //uploadForm Logic
270        uploadForm.addEventListener("submit",async (e) =>{
271          e.preventDefault();
272          const formData = new FormData(e.target);
273          loadingDiv.style.display = "block";
274          resultDiv.style.display = "none";
275          resultDiv.textContent = "";
276          downloadButton.style.display = "none";
277          try{
278            const response = await fetch('/analyze',{
279              method: "POST",
280              body: formData,
281            });
282            const data = await response.json();
283            if (data.result){
284              //retrieved from the app.js json response
285              analysisResult = data.result;
286              analysisImage = data.image;
287              resultDiv.innerHTML =
288                "<h3>Analysis Result:</h3><p>" + 
289                analysisResult.replace(/\n/g, "<br>") +
290                "</p>";
291              resultDiv.style.display = "block";
292              downloadButton.style.display = "block";
293              loadingDiv.style.display = "none";
294
295            } else if (data.error){
296              resultDiv.textContent = "Error: " + data.error;
297              resultDiv.style.display = "block";
298            }
299
300          } catch (error){
301            resultDiv.textContent = "Error: " + error.message;
302            resultDiv.style.display = "block";
303
304          } finally {
305            loadingDiv.style.display = "none";
306          }
307        });
308
309        downloadButton.addEventListener("click",async () => {
```

The LOGIC is performed in APPS.JS:

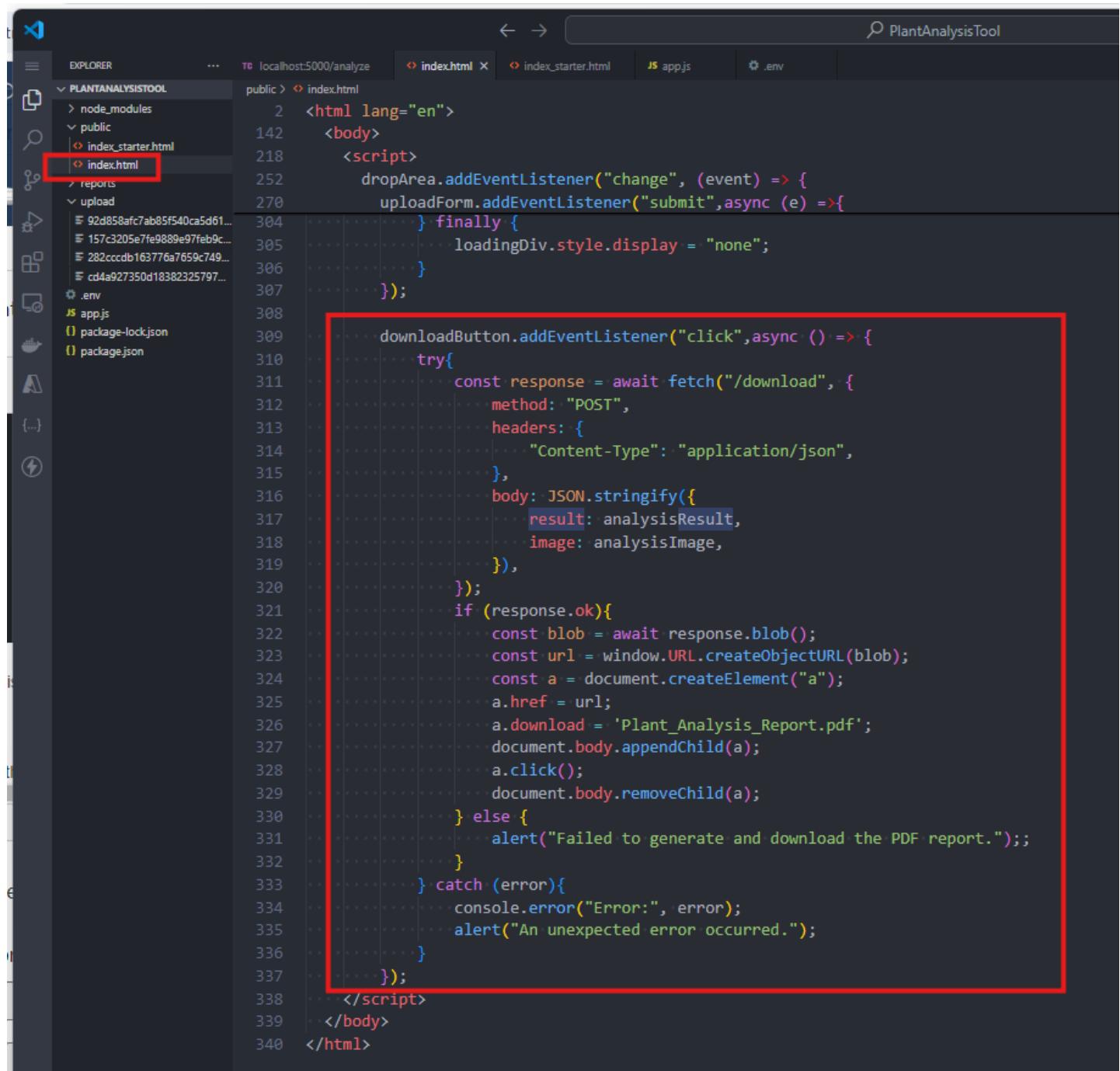


```
1  require("dotenv").config();
2  const express = require("express");
3  const multer = require("multer");
4  const PDFDocument = require("pdfkit");
5  const fs = require("fs");
6  const fsPromises = fs.promises;
7  const path = require("path");
8  const { GoogleGenerativeAI } = require("@google/generative-ai");
9
10 const app = express();
11 const port = process.env.PORT || 5000;
12
13 //configure multer: save uploaded files to /upload folder & limit file size
14 const upload = multer({ dest: "upload/" });
15 app.use(express.json({ limit: "10mb" }));
16
17 //initialize Google Generative AI
18 const genAI = new GoogleGenerativeAI(process.env.GEMINI_API_KEY);
19 app.use(express.static("public"));
20
```



```
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 //define 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 result = await model.generateContent([
41       "Analyze this plant image and provide detailed analysis of its species, health and care recommendations, its characteristics, can",
42       {
43         inlineData: {
44           mimetype: req.file.mimetype,
45           data: imageData,
46         },
47       },
48     ]);
49     const plantInfo = result.response.text();
50     //remove the uploaded image
51     await fsPromises.unlink(imagePath);
52     //respond with the analysis result and the image data
53     res.json({
54       result: plantInfo,
55       image: `data:${req.file.mimetype};base64,${imageData}`,
56     });
57
58   } catch (error) {
59     console.error("Error analyzing image:", error);
60
61     res.status(500).json({ error: "An error occurred while analyzing the image." });
62   }
63 });
64
```

5. This is the code for DOWNLOAD REPORT in INDEX.HTML



PlantAnalysisTool

localhost:5000/analyze

index.html

index_starter.html

app.js

.env

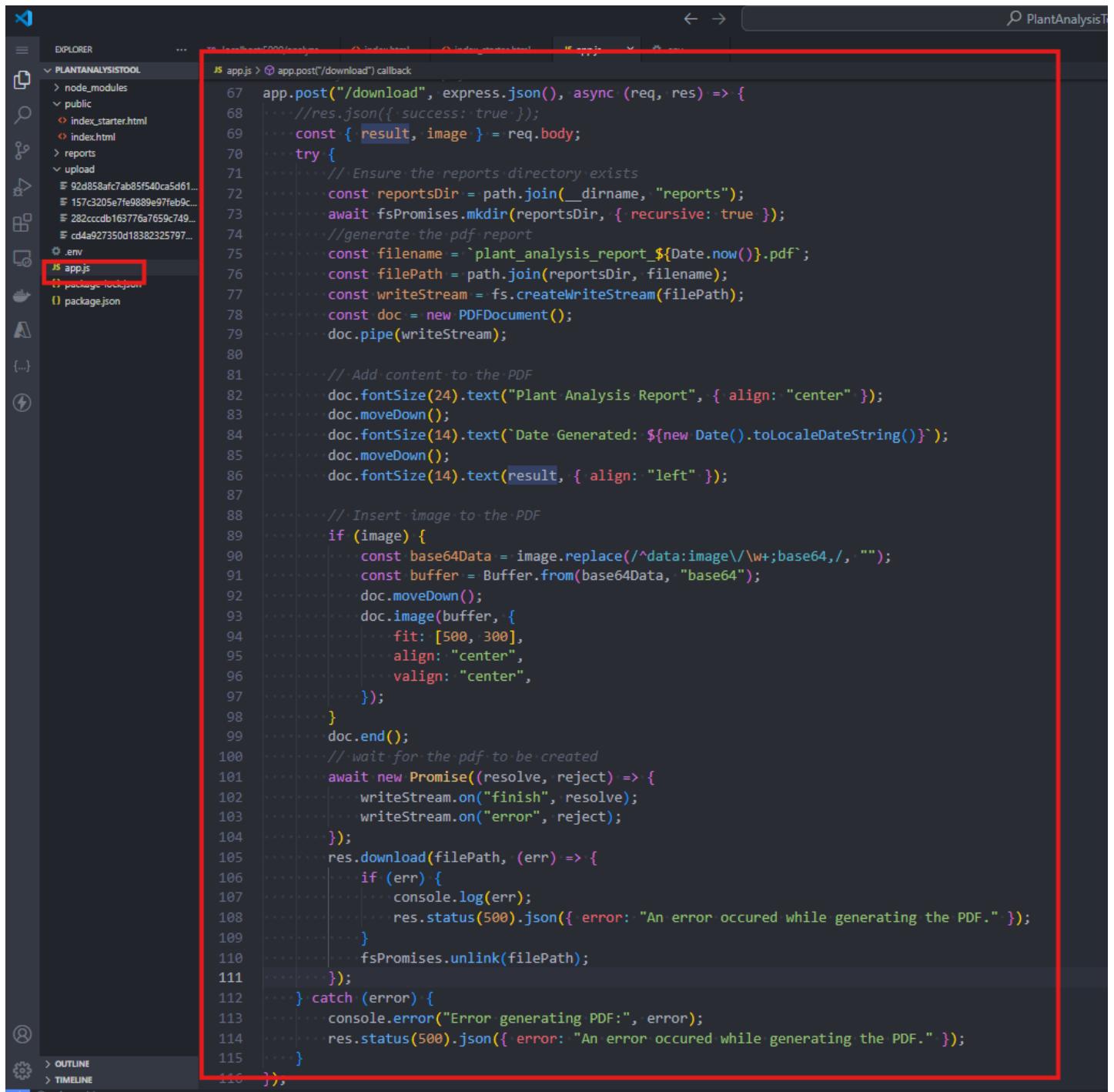
EXPLORER

PLANTANALYSISTOOL

- node_modules
- public
 - index_starter.html
 - index.html
- reports
- upload
 - 92d858afc7ab85f540ca5d61...
 - 157c3205e7fe9889e97feb9...
 - 282cccd163776a7659c749...
 - cd4a927350d18382325797...
- .env
- app.js
- package-lock.json
- package.json

```
public > <html lang="en">
  2   <body>
142     <script>
218       dropArea.addEventListener("change", (event) => {
270         uploadForm.addEventListener("submit",async (e) =>{
304           e.preventDefault();
305           dropArea.innerHTML = "Uploading file... Please wait";
306           loadingDiv.style.display = "block";
307         });
308       });
309       downloadButton.addEventListener("click",async () =>{
310         try{
311           const response = await fetch("/download", {
312             method: "POST",
313             headers: {
314               "Content-Type": "application/json",
315             },
316             body: JSON.stringify({
317               result: analysisResult,
318               image: analysisImage,
319             }),
320           });
321           if (response.ok){
322             const blob = await response.blob();
323             const url = window.URL.createObjectURL(blob);
324             const a = document.createElement("a");
325             a.href = url;
326             a.download = 'Plant_Analysis_Report.pdf';
327             document.body.appendChild(a);
328             a.click();
329             document.body.removeChild(a);
330           } else {
331             alert("Failed to generate and download the PDF report.");
332           }
333         } catch (error){
334           console.error("Error:", error);
335           alert("An unexpected error occurred.");
336         }
337       });
338     </script>
339   </body>
340 </html>
```

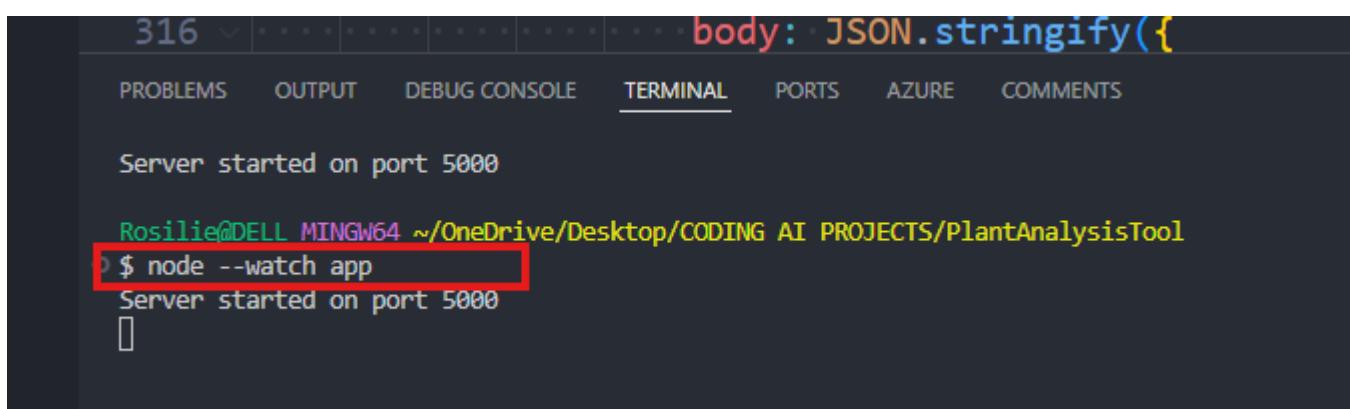
This is the logic in APPS.JS



```
JS app.js > app.post("/download") callback
67 app.post("/download", express.json(), async (req, res) => {
68   //res.json({ success: true });
69   const { result, image } = req.body;
70   try {
71     // Ensure the reports directory exists
72     const reportsDir = path.join(__dirname, "reports");
73     await fsPromises.mkdir(reportsDir, { recursive: true });
74     //generate the pdf report
75     const filename = `plant_analysis_report_${Date.now()}.pdf`;
76     const filePath = path.join(reportsDir, filename);
77     const writeStream = fs.createWriteStream(filePath);
78     const doc = new PDFDocument();
79     doc.pipe(writeStream);
80
81     //Add content to the PDF
82     doc.fontSize(24).text("Plant Analysis Report", { align: "center" });
83     doc.moveDown();
84     doc.fontSize(14).text(`Date Generated: ${new Date().toLocaleDateString()}`);
85     doc.moveDown();
86     doc.fontSize(14).text(result, { align: "left" });
87
88     //Insert image to the PDF
89     if (image) {
90       const base64Data = image.replace(/^data:image\/\w+;base64,/, "");
91       const buffer = Buffer.from(base64Data, "base64");
92       doc.moveDown();
93       doc.image(buffer, {
94         fit: [500, 300],
95         align: "center",
96         valign: "center",
97       });
98     }
99     doc.end();
100    //wait for the pdf to be created
101    await new Promise((resolve, reject) => {
102      writeStream.on("finish", resolve);
103      writeStream.on("error", reject);
104    });
105    res.download(filePath, (err) => {
106      if (err) {
107        console.log(err);
108        res.status(500).json({ error: "An error occurred while generating the PDF." });
109      }
110      fsPromises.unlink(filePath);
111    });
112  } catch (error) {
113    console.error("Error generating PDF:", error);
114    res.status(500).json({ error: "An error occurred while generating the PDF." });
115  }
116},
```

6. When we run our code, type this in your terminal:

```
$ node --watch app
```



```
316 body: JSON.stringify({ PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE COMMENTS
Server started on port 5000
Rosilie@DELL MINGW64 ~/OneDrive/Desktop/CODING AI PROJECTS/PlantAnalysisTool
$ node --watch app
Server started on port 5000
```

When you run in your browser:

→ <http://localhost:5000>

PlantScan: Advanced Plant Analysis Tool

Upload an image of a plant to receive a detailed analysis of its species, health, and care recommendations.

How to Use



Upload

Select or drag & drop a plant image



Analyze

Click 'Analyze Plant' to process the image



Download

Get your detailed PDF report



Drag & Drop or Click to Upload Plant Image



Q **Analyze Plant**

Analysis Result:

The plant in the image is an air plant, specifically a *Tillandsia ionantha*, also known as the Pink Quill. It is a popular choice for indoor plant enthusiasts due to its ease of care and unique appearance.

****Health:**** The plant appears to be healthy and well-hydrated. The leaves are a vibrant green, and there are no signs of pests or diseases.

****Care Recommendations:****

- * **Light:** Bright indirect light is best. Avoid direct sunlight as it can burn the leaves.
- * **Water:** Water your air plant by soaking it in a bowl of room temperature water for 30 minutes once a week. Allow the plant to completely dry before returning it to its pot.
- * **Humidity:** Air plants prefer moderate humidity. If your home is dry, you can mist the plant regularly.
- * **Fertilizer:** You can fertilize your air plant every few weeks with a diluted liquid fertilizer.
- * **Potting:** While air plants are epiphytes (meaning they do not need soil), they often look attractive when displayed in a pot. You can use a decorative pot with drainage holes and a base of gravel or stones for support.

****Characteristics:****

- * **Appearance:** The leaves are narrow, pointed, and grow in a rosette formation. The leaves can turn a vibrant pink color when they are about to bloom.
- * **Size:** *Tillandsia ionantha* can grow up to 6 inches tall and wide.
- * **Flowers:** The pink quill produces small, pink flowers in the center of the rosette. The flowers are short-lived, but the pink coloration of the leaves can persist for several weeks.

****Interesting Facts:****

- * Air plants are epiphytes, meaning they grow on other plants for support.
- * Air plants absorb moisture and nutrients from the air through their leaves.
- * Air plants are native to the tropical and subtropical regions of the Americas.
- * The Pink Quill is known for its beautiful pink coloration, which is more pronounced when the plant is about to bloom.
- * Air plants are relatively easy to care for, making them an excellent choice for beginner plant owners.

→ <http://localhost:5000>

Analysis Result:

The plant in the image is an air plant, specifically a *Tillandsia ionantha*, also known as the Pink Quill. It is a popular choice for indoor plant enthusiasts due to its ease of care and unique appearance.

****Health:**** The plant appears to be healthy and well-hydrated. The leaves are a vibrant green, and there are no signs of pests or diseases.

****Care Recommendations:****

- * **Light:** Bright indirect light is best. Avoid direct sunlight as it can burn the leaves.
- * **Water:** Water your air plant by soaking it in a bowl of room temperature water for 30 minutes once a week. Allow the plant to completely dry before returning it to its pot.
- * **Humidity:** Air plants prefer moderate humidity. If your home is dry, you can mist the plant regularly.
- * **Fertilizer:** You can fertilize your air plant every few weeks with a diluted liquid fertilizer.
- * **Potting:** While air plants are epiphytes (meaning they do not need soil), they often look attractive when displayed in a pot. You can use a decorative pot with drainage holes and a base of gravel or stones for support.

****Characteristics:****

- * **Appearance:** The leaves are narrow, pointed, and grow in a rosette formation. The leaves can turn a vibrant pink color when they are about to bloom.
- * **Size:** *Tillandsia ionantha* can grow up to 6 inches tall and wide.
- * **Flowers:** The pink quill produces small, pink flowers in the center of the rosette. The flowers are short-lived, but the pink coloration of the leaves can persist for several weeks.

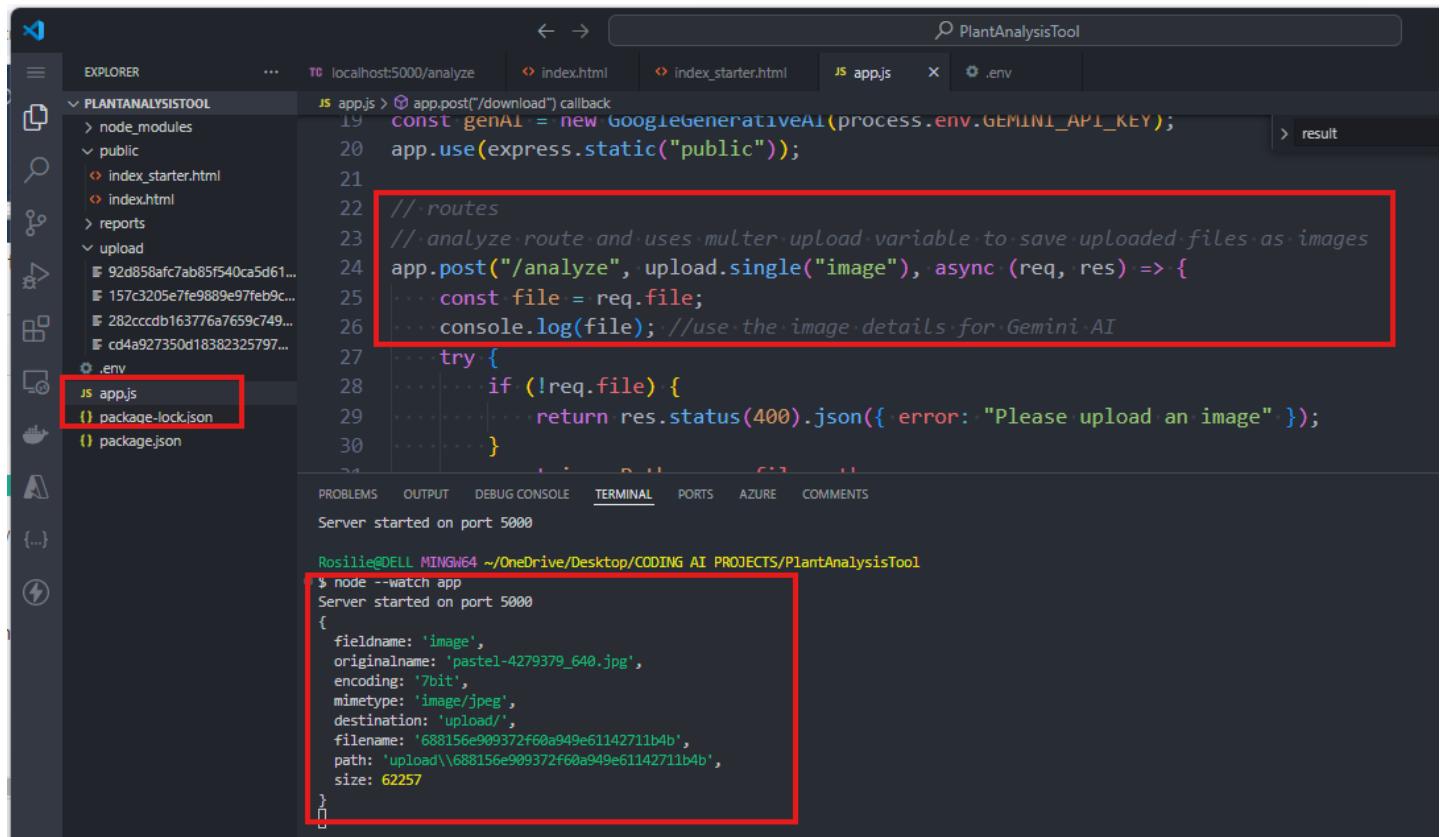
****Interesting Facts:****

- * Air plants are epiphytes, meaning they grow on other plants for support.
- * Air plants absorb moisture and nutrients from the air through their leaves.
- * Air plants are native to the tropical and subtropical regions of the Americas.
- * The Pink Quill is known for its beautiful pink coloration, which is more pronounced when the plant is about to bloom.
- * Air plants are relatively easy to care for, making them an excellent choice for beginner plant owners.

Download PDF Report

See the attached PDF for its sample output.

When we run our app and upload an image for plant analysis, we show our JSON data in our terminal:



PlantAnalysisTool

localhost:5000/analyze index.html index_starter.html app.js .env

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

```
app.js:19:19:19 const genAI = new GoogleGenerativeAI(process.env.GEMINI_API_KEY);
app.js:20:19 app.use(express.static("public"));
app.js:21:
app.js:22:19 // routes
app.js:23:19 // analyze route and uses multer upload variable to save uploaded files as images
app.js:24:19 app.post("/analyze", upload.single("image"), async (req, res) => {
app.js:25:19   const file = req.file;
app.js:26:19   console.log(file); // use the image details for Gemini AI
app.js:27:19   try {
app.js:28:19     if (!req.file) {
app.js:29:19       return res.status(400).json({ error: "Please upload an image" });
app.js:30:19     }

```

TERMINAL

```
Server started on port 5000
Rosilie@DELL MINGW64 ~/OneDrive/Desktop/CODING AI PROJECTS/PlantAnalysisTool
$ node --watch app
Server started on port 5000
{
  fieldname: 'image',
  originalname: 'pastel-4279379_640.jpg',
  encoding: '7bit',
  mimetype: 'image/jpeg',
  destination: 'upload/',
  filename: '688156e909372f60a949e61142711b4b',
  path: 'upload\688156e909372f60a949e61142711b4b',
  size: 62257
}
```

We uninstall NODE.JS after this project to give way to Django projects.