

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

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



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

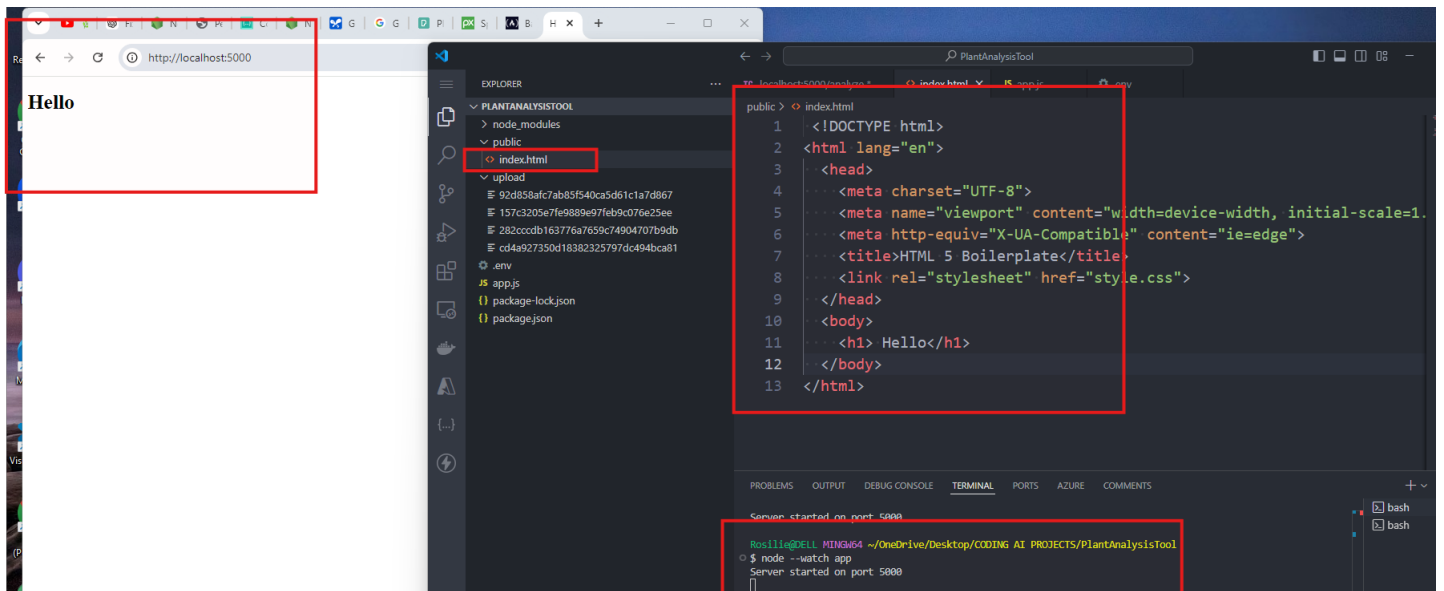
[GITHUB REPO](#)

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

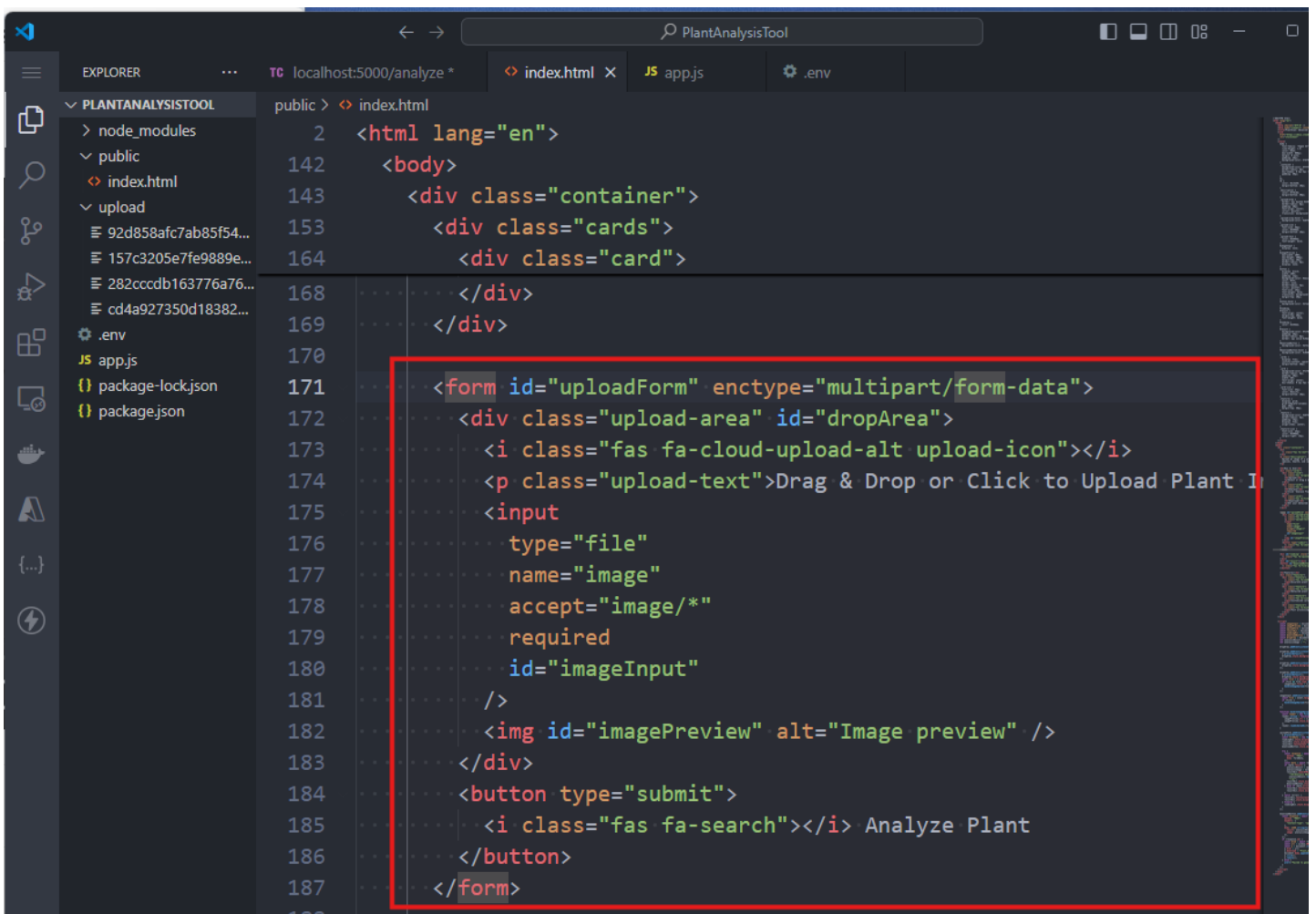
The screenshot displays a web application interface for plant analysis. The interface is divided into several sections:

- Form Section:** A form titled "Form Fields" with a "Files" section. The "Files" section has a checkbox labeled "image" which is checked. Below it, there is a "Choose File" button and a text input field containing the filename "philodendron-7960228_640.jpg".
- Status Bar:** A status bar at the top right indicates a successful POST request to "http://localhost:5000/analyze" with a status of "200 OK", a size of "1.76 KB", and a time of "2.60 s".
- Response Section:** A section titled "Response" displays the output from the Gemini AI model. The response is a JSON object with a "results" field containing a detailed analysis of the plant. The analysis includes information about the plant's native habitat (Central America), its characteristics (large, glossy leaves with distinctive holes and slits), and care instructions (watering, humidity, light, temperature, soil, and fertilizer). The response also includes a base64 encoded image of the plant.

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



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



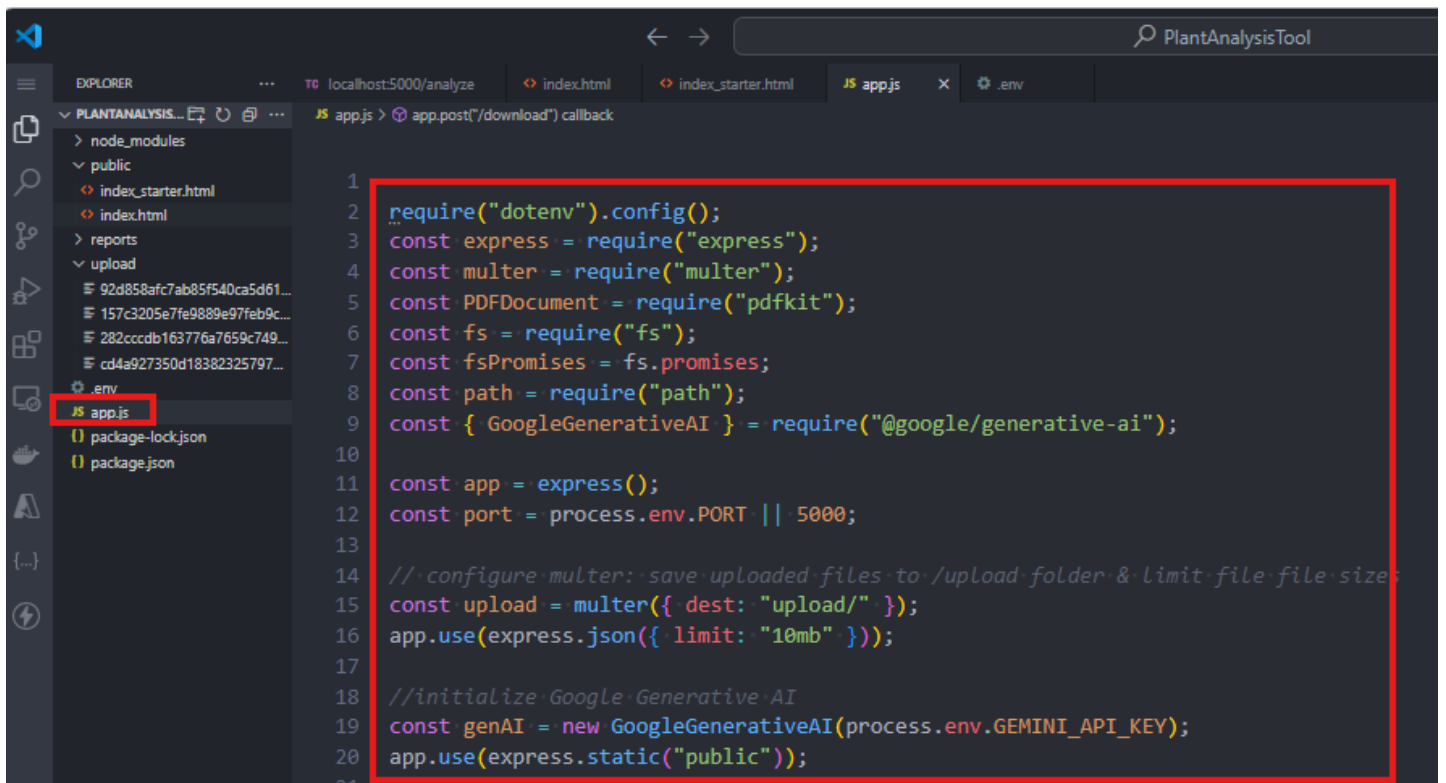
3. This is the code for UPLOAD using DRAG AND DROP OF IMAGE. INDEX.HTML is written as:

```
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
```

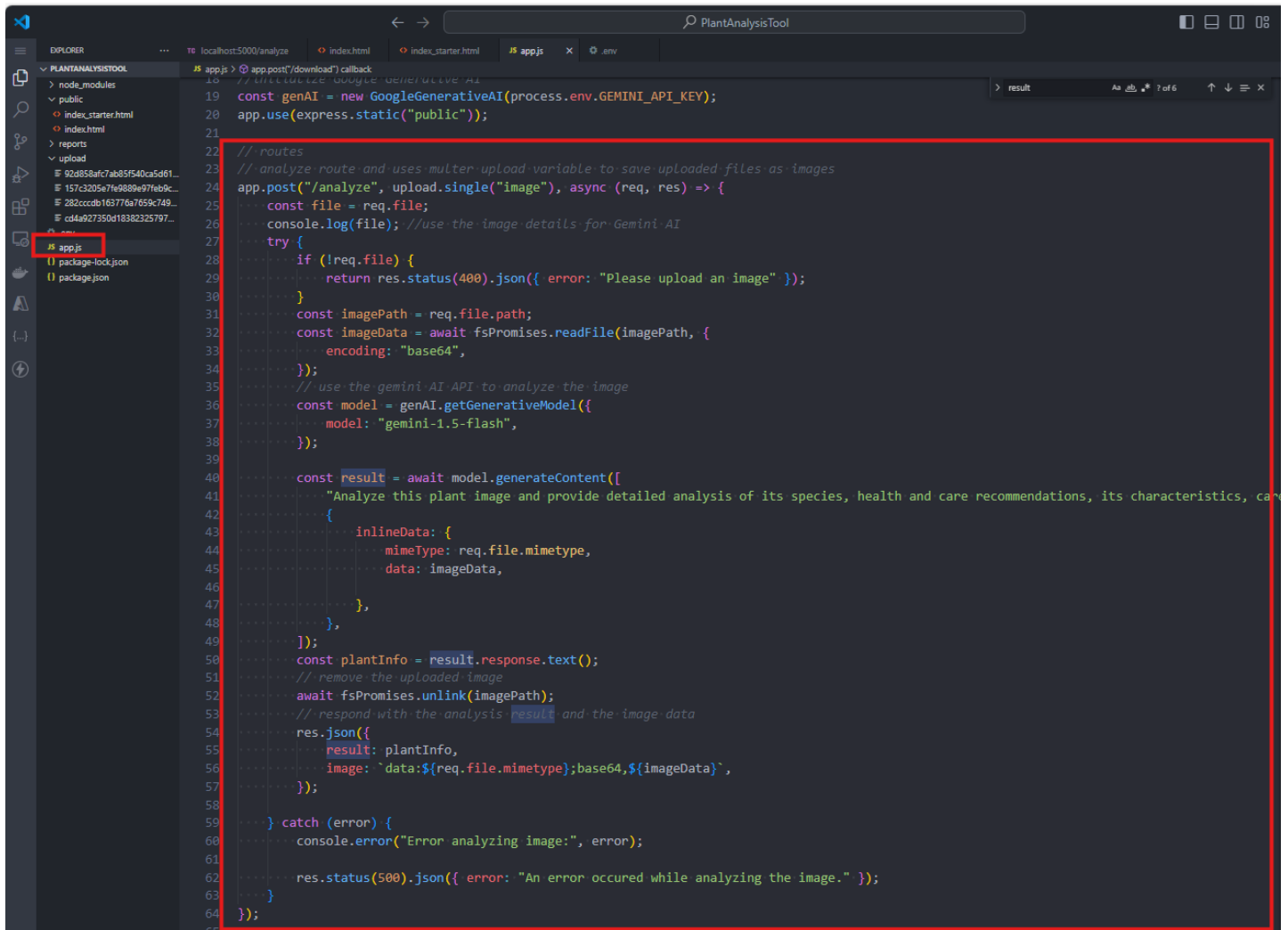
4. This is the EVENT for UPLOAD BUTTON in INDEX.HTML:

```
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 apps.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         } else if (data.error) {
295           resultDiv.textContent = "Error: " + data.error;
296           resultDiv.style.display = "block";
297         }
298       }
299     } catch (error){
300       resultDiv.textContent = "Error: " + error.message;
301       resultDiv.style.display = "block";
302     } finally {
303       loadingDiv.style.display = "none";
304     }
305   });
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"));
```

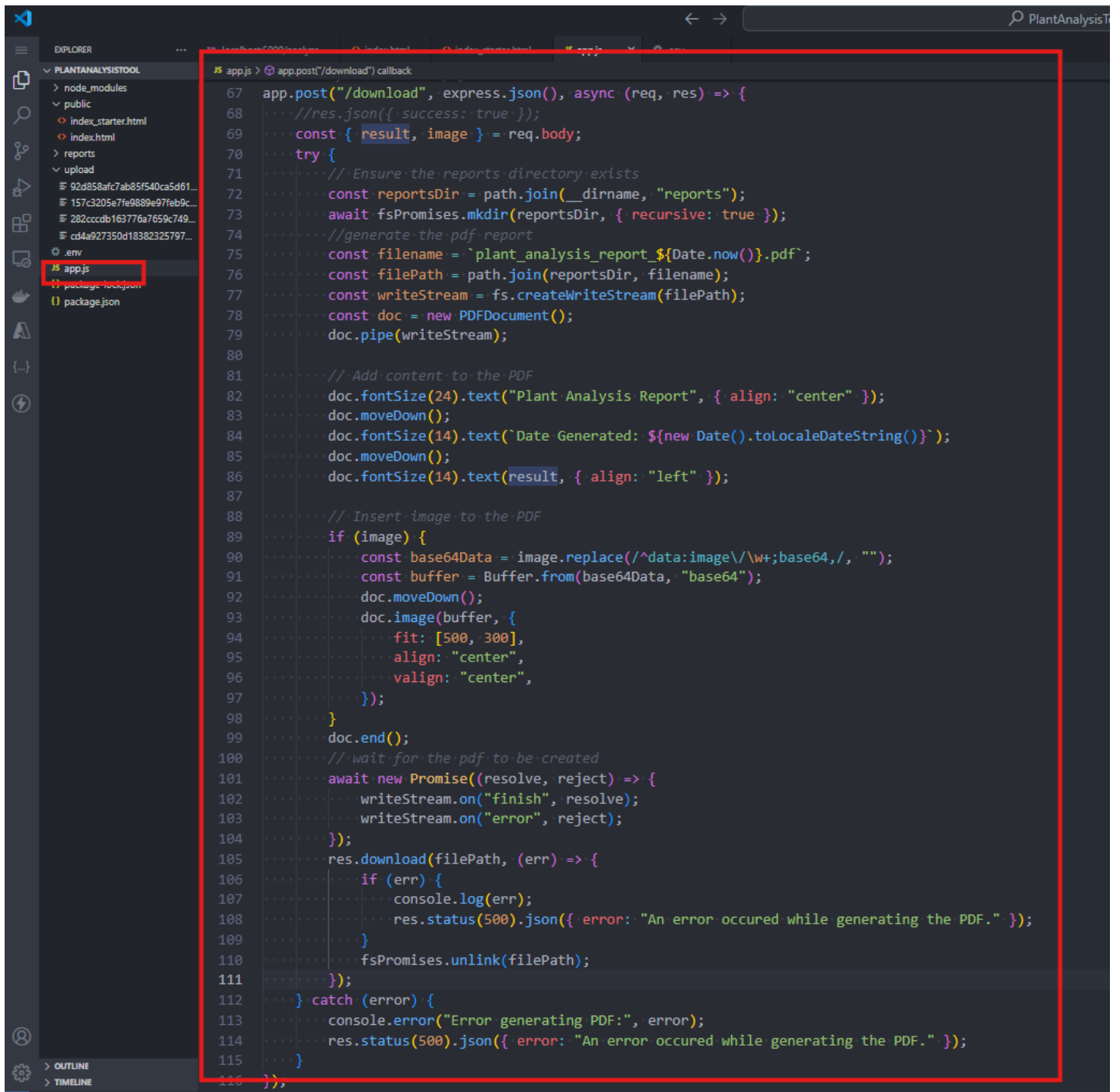


```
18 // initialize Google Generative AI
19 const genAI = new GoogleGenerativeAI(process.env.GEMINI_API_KEY);
20 app.use(express.static("public"));
21
22 // routes
23 // analyze route and uses multer upload variable to save uploaded files as images
24 app.post("/analyze", upload.single("image"), async (req, res) => {
25   const file = req.file;
26   console.log(file); // 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, care",
42     ]);
43     const { inlineData } = result.response.data[0].candidates[0].content.parts[0].inlineData;
44     const { mimeType, data } = inlineData;
45
46     // remove the uploaded image
47     await fsPromises.unlink(imagePath);
48     // respond with the analysis result and the image data
49     res.json({
50       result: plantInfo,
51       image: `data:${req.file.mimetype};base64,${imageData}`,
52     });
53   } catch (error) {
54     console.error("Error analyzing image:", error);
55     res.status(500).json({ error: "An error occurred while analyzing the image." });
56   }
57 });
```

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

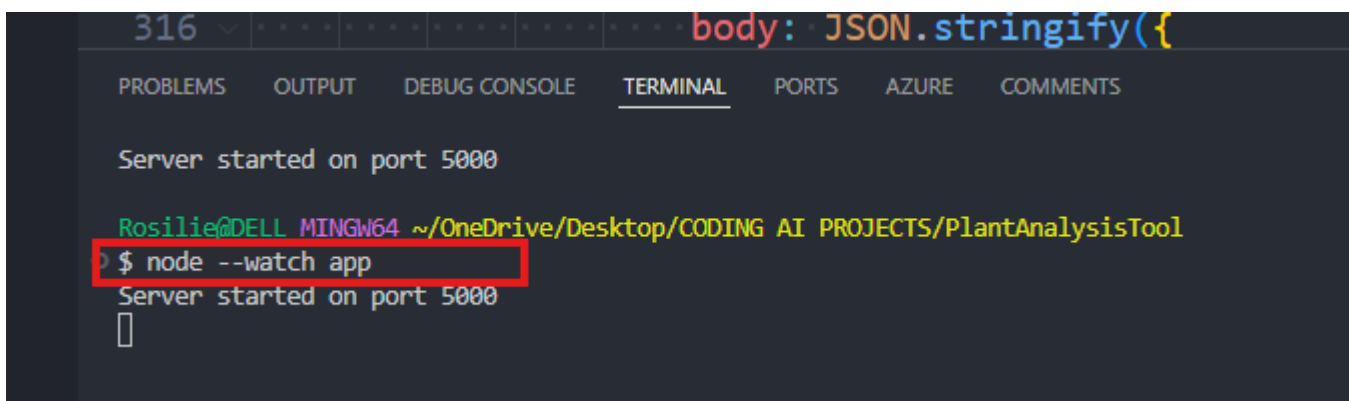
```
2 <html lang="en">
142 <body>
218 <script>
252     dropArea.addEventListener("change", (event) => {
270         uploadForm.addEventListener("submit", async (e) => {
304             ... } finally {
305                 ... loadingDiv.style.display = "none";
306             ... }
307         ... });
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         ... }
339     </script>
340 </body>
</html>
```

This is the logic in APPS.JS

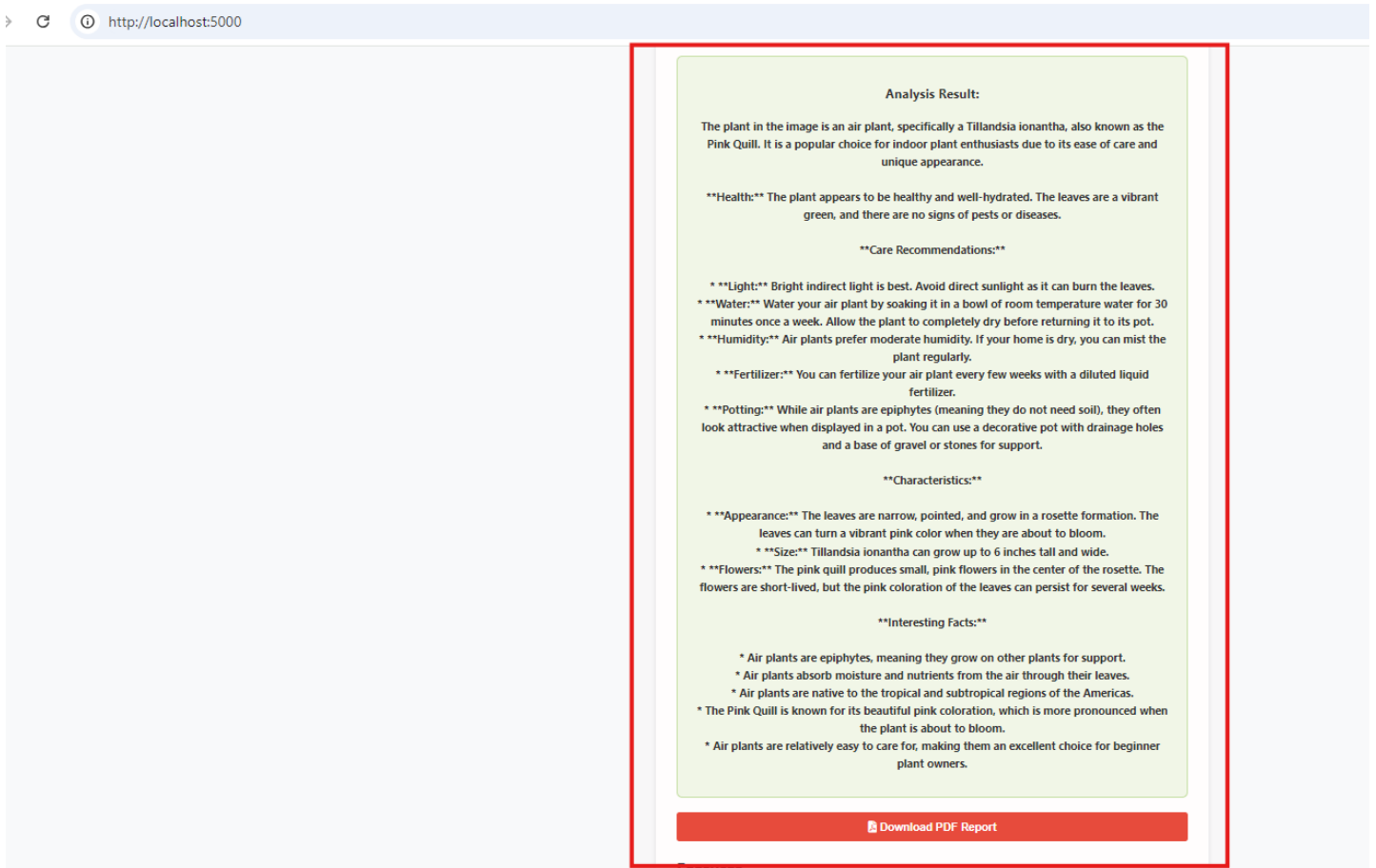
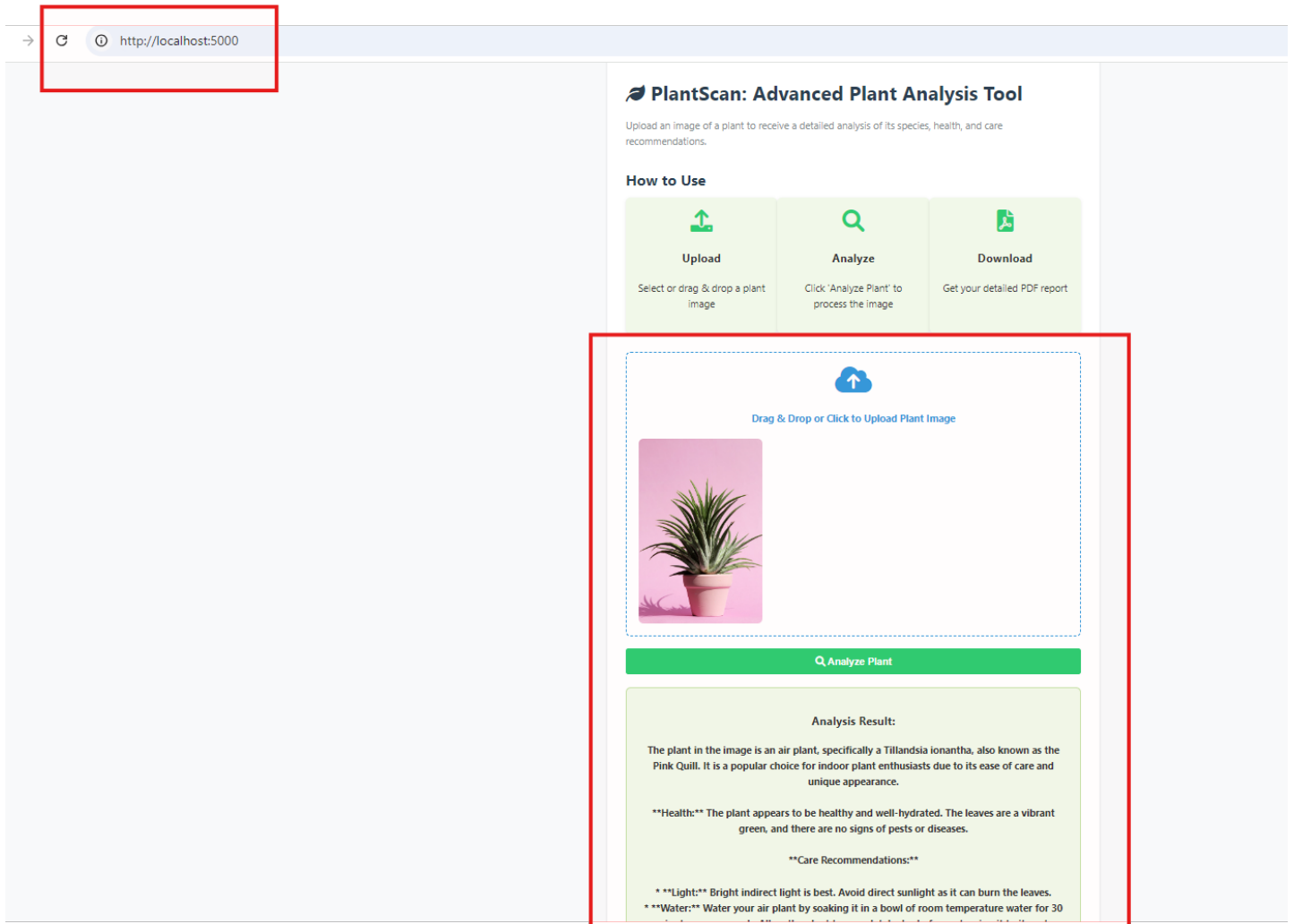


6. When we run our code, type this in your terminal:

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



When you run in your browser:



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:


```
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); // use the image details for Gemini AI
27 }
28 try {
29   if (!req.file) {
30     return res.status(400).json({ error: "Please upload an image" });
31   }
32 }
```

```
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.