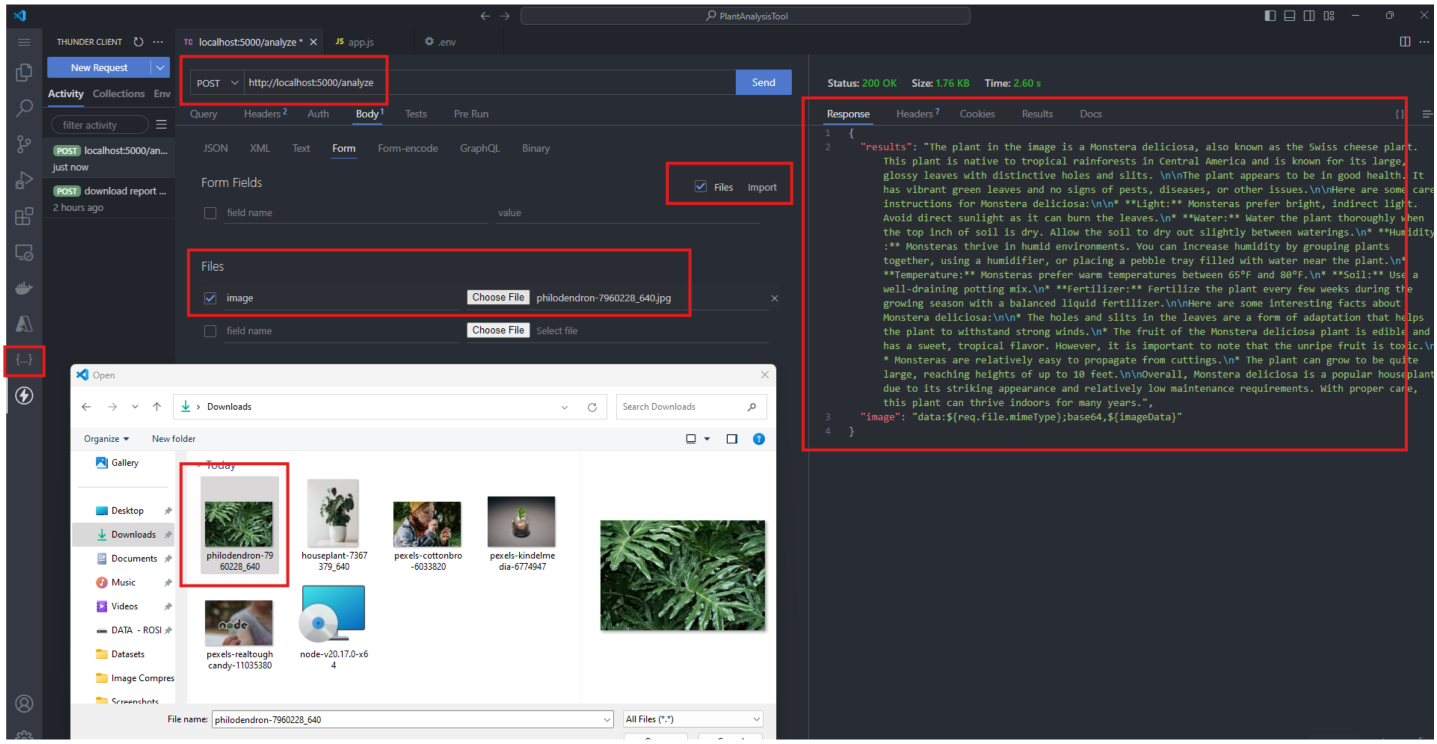
Topic: Plant Analysis Tool using Gemini AI and Express.js Part 2

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](https://github.com/tweneboah/Full-Stack-Web-Development-Bootcamp-Course/tree/main/PROJECTS/AI-PROJECTS/PLANT-ANALYSIS-TOOL/public)



POST http://localhost:5000/analyze

Form Fields: ☑ Files

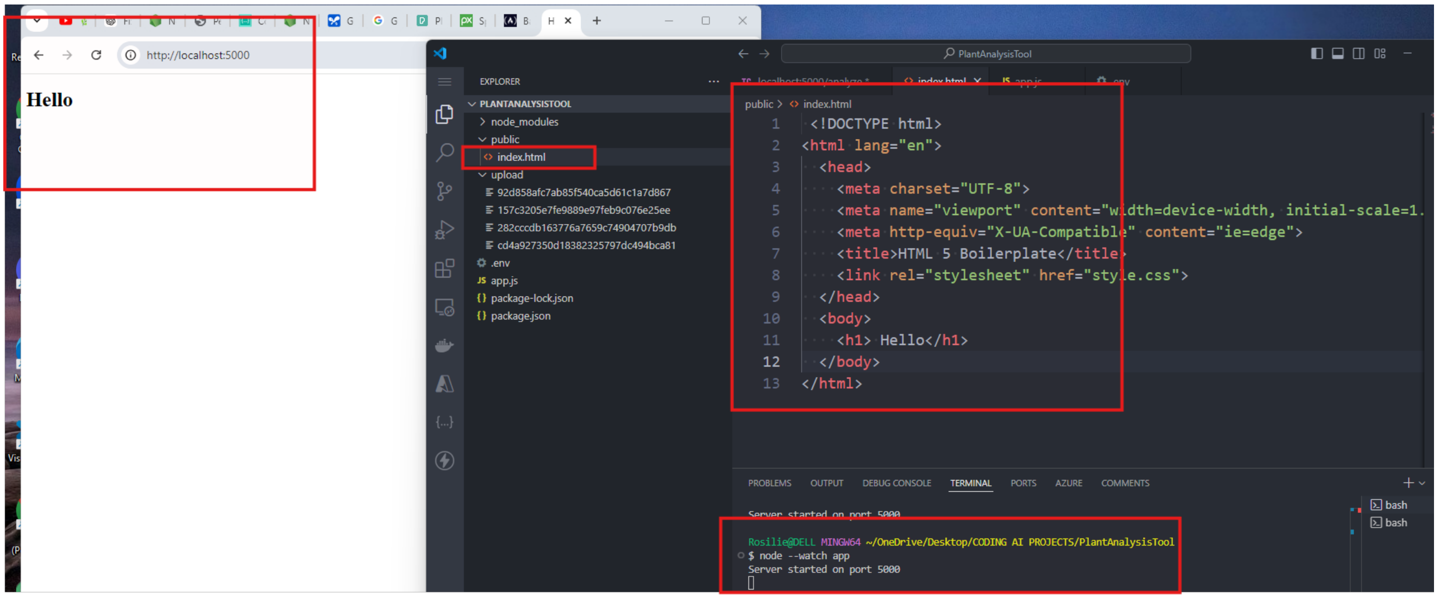
Files: ☑ image philodendron-7960228_640.jpg

File name: philodendron-7960228_640

Status: 200 OK Size: 1.76 KB Time: 2.60 s

```
{
  "results": "The plant in the image is a Monstera deliciosa, also known as the Swiss cheese plant.
     This plant is native to tropical rainforests in Central America and is known for its large,
     glossy leaves with distinctive holes and slits. \n\nThe plant appears to be in good health. It
     has vibrant green leaves and no signs of pests, diseases, or other issues.\n\nHere are some care
     instructions for Monstera deliciosa:\n\n* **Light:** Monsteras prefer bright, indirect light.
     Avoid direct sunlight as it can burn the leaves.\n* **Water:** Water the plant thoroughly when
     the top inch of soil is dry. Allow the soil to dry out slightly between waterings.\n* **Humidity
     :** Monsteras thrive in humid environments. You can increase humidity by grouping plants
     together, using a humidifier, or placing a pebble tray filled with water near the plant.\n*
     **Temperature:** Monsteras prefer warm temperatures between 65°F and 80°F.\n* **Soil:** Use a
     well-draining potting mix.\n* **Fertilizer:** Fertilize the plant every few weeks during the
     growing season with a balanced liquid fertilizer.\n\nHere are some interesting facts about
     Monstera deliciosa:\n\n* The holes and slits in the leaves are a form of adaptation that helps
     the plant to withstand strong winds.\n* The fruit of the Monstera deliciosa plant is edible and
     has a sweet, tropical flavor. However, it is important to note that the unripe fruit is toxic.\n
     * Monsteras are relatively easy to propagate from cuttings.\n* The plant can grow to be quite
     large, reaching heights of up to 10 feet.\n\nOverall, Monstera deliciosa is a popular houseplant
     due to its striking appearance and relatively low maintenance requirements. With proper care,
     this plant can thrive indoors for many years.",
  "image": "data:${req.file.mimeType};base64,${imageData}"
}
```

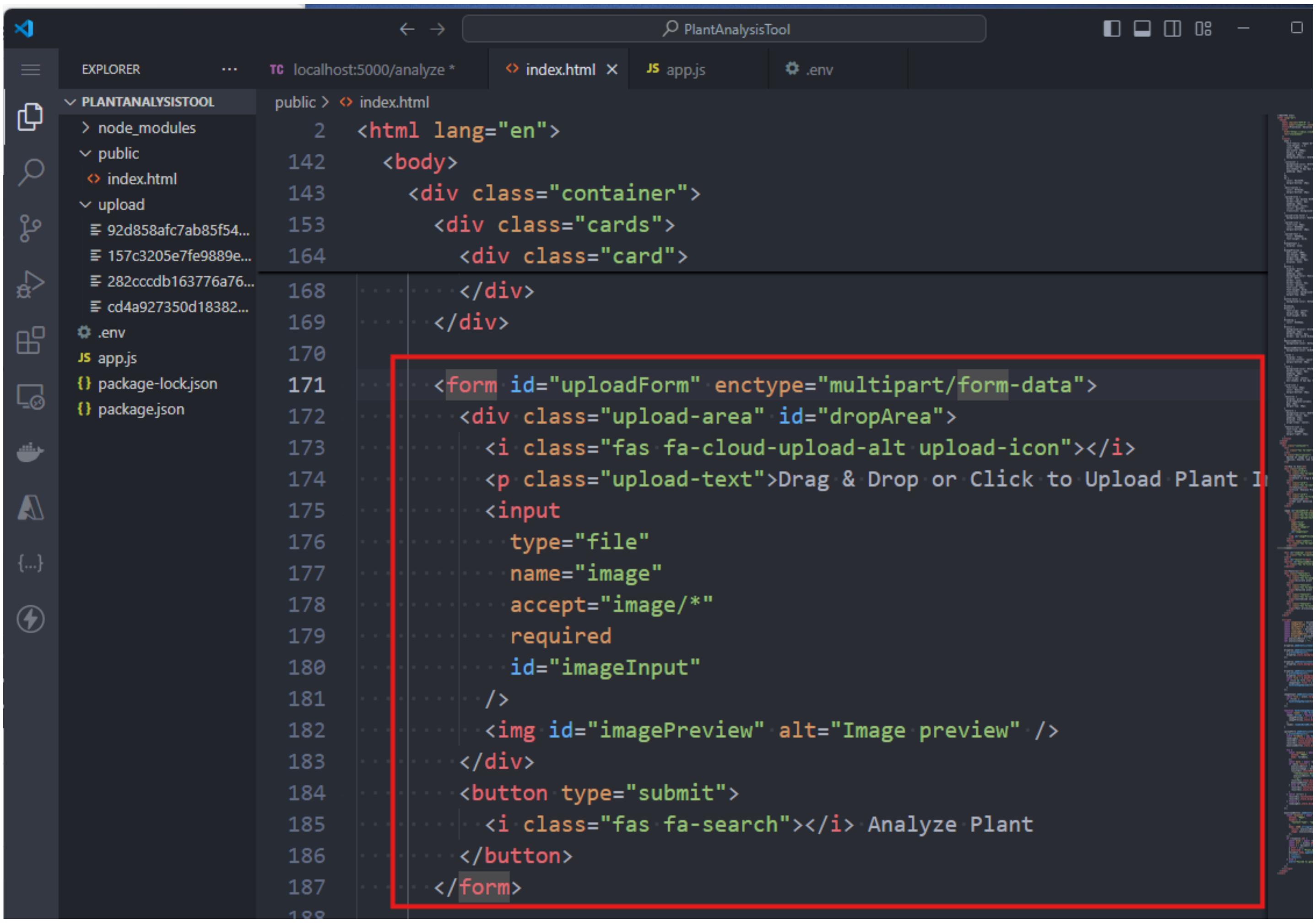
1. Now, create the front end for our project. Update our INDEX.HTML



```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>HTML 5 Boilerplate</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <h1> Hello</h1>
  </body>
</html>
```

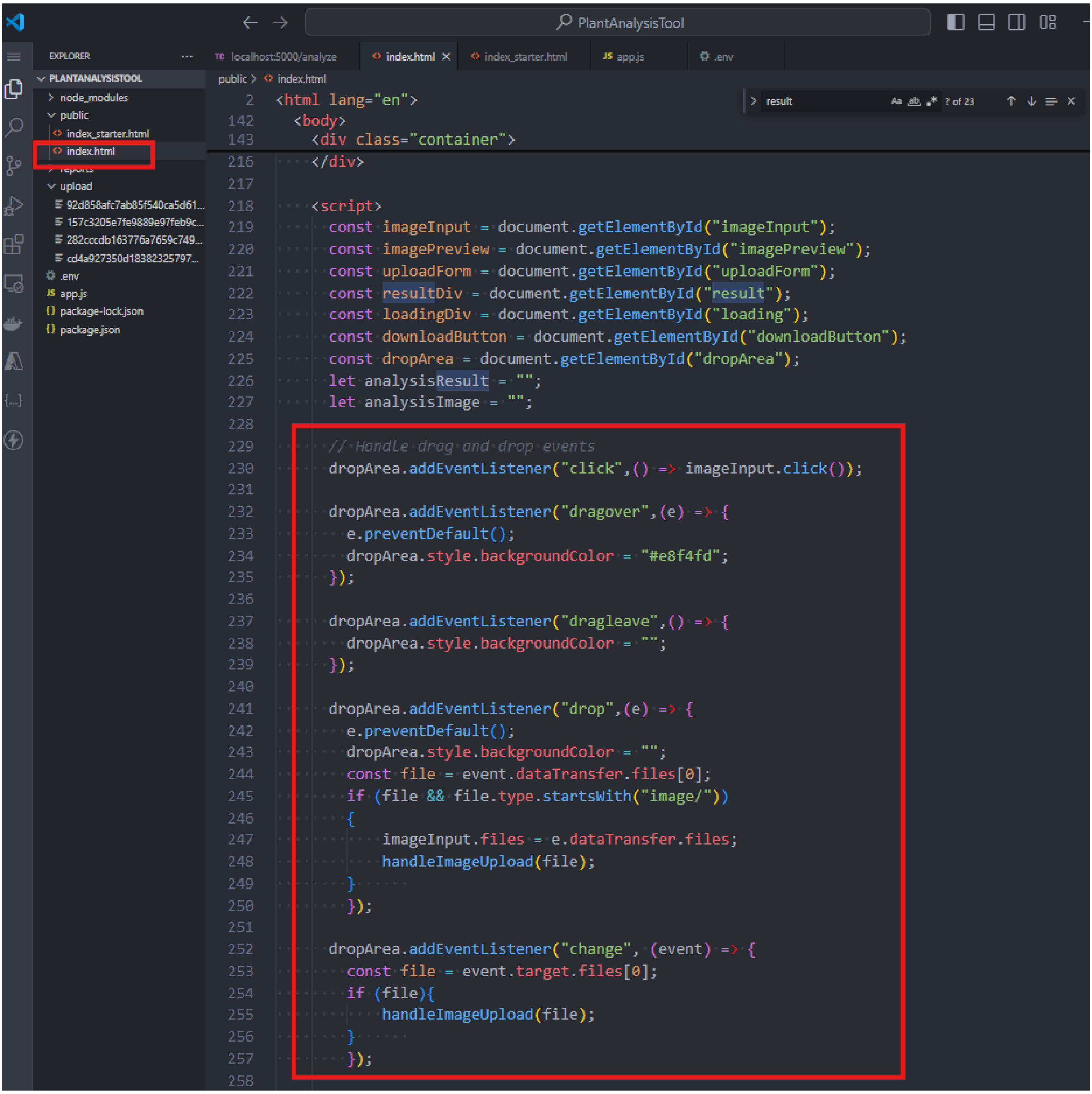
```
Server started on port 5000

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

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



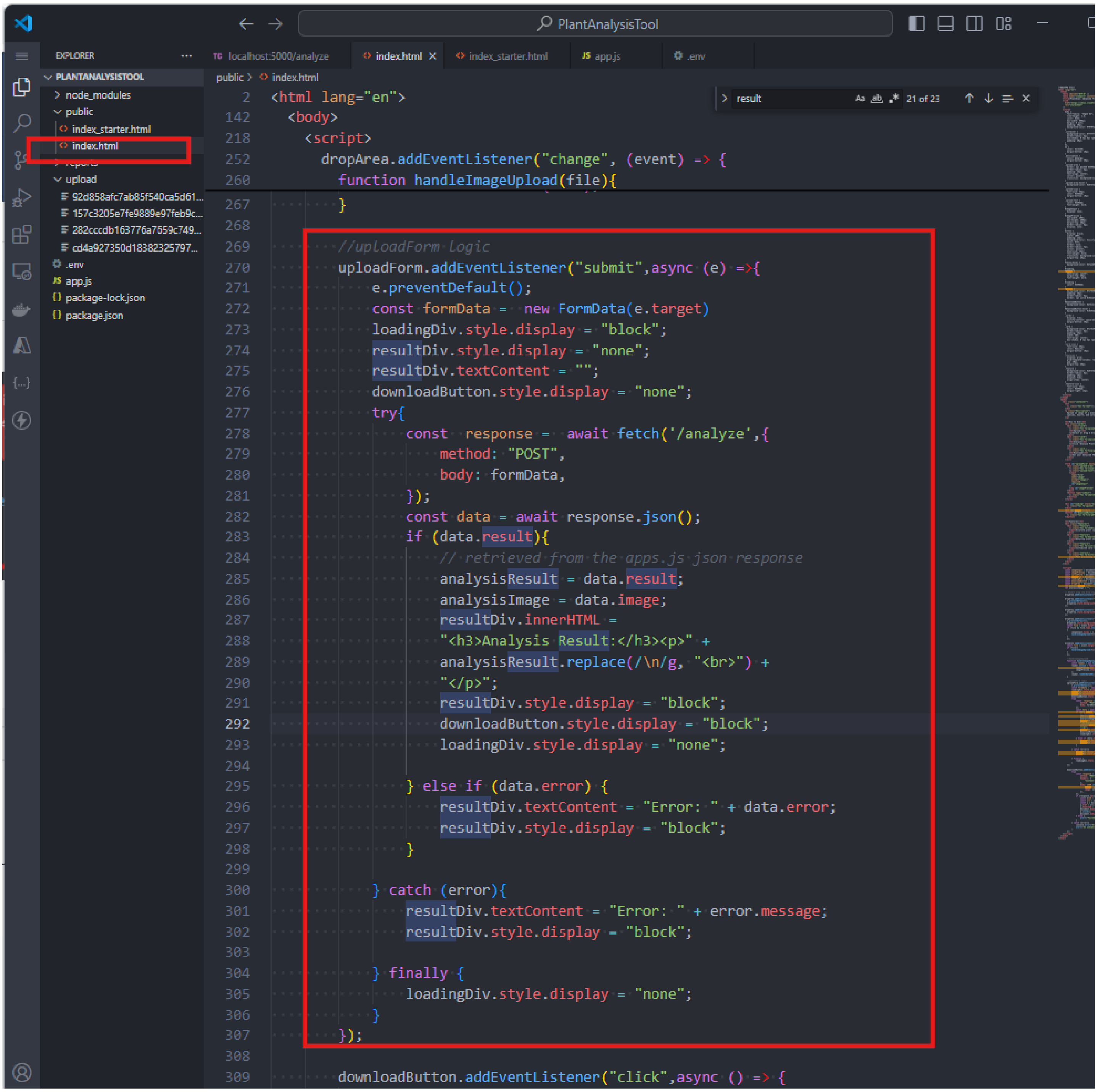
```html
<html lang="en">
  <body>
    <div class="container">
      <div class="cards">
        <div class="card">
        </div>
      </div>

      <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 I
          <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:



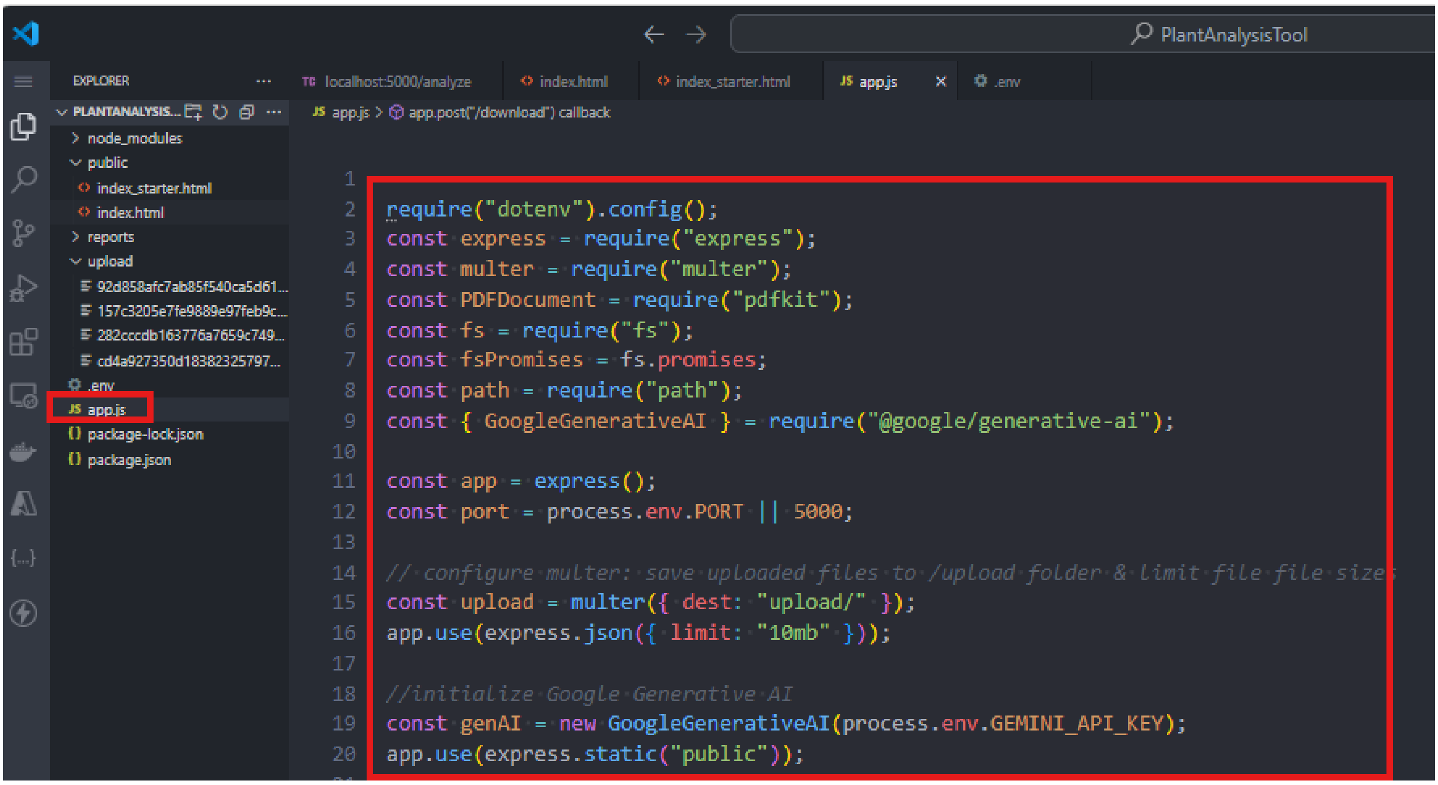
```
<html lang="en">
  <body>
    <div class="container">
    </div>

    <script>
      const imageInput = document.getElementById("imageInput");
      const imagePreview = document.getElementById("imagePreview");
      const uploadForm = document.getElementById("uploadForm");
      const resultDiv = document.getElementById("result");
      const loadingDiv = document.getElementById("loading");
      const downloadButton = document.getElementById("downloadButton");
      const dropArea = document.getElementById("dropArea");
      let analysisResult = "";
      let analysisImage = "";

      // Handle drag and drop events
      dropArea.addEventListener("click",() => imageInput.click());

      dropArea.addEventListener("dragover",(e) => {
        e.preventDefault();
        dropArea.style.backgroundColor = "#e8f4fd";
      });

      dropArea.addEventListener("dragleave",() => {
        dropArea.style.backgroundColor = "";
      });

      dropArea.addEventListener("drop",(e) => {
        e.preventDefault();
        dropArea.style.backgroundColor = "";
        const file = event.dataTransfer.files[0];
        if (file && file.type.startsWith("image/"))
        {
            imageInput.files = e.dataTransfer.files;
            handleImageUpload(file);
        }
        });

      dropArea.addEventListener("change", (event) => {
        const file = event.target.files[0];
        if (file){
            handleImageUpload(file);
        }
        });

```

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

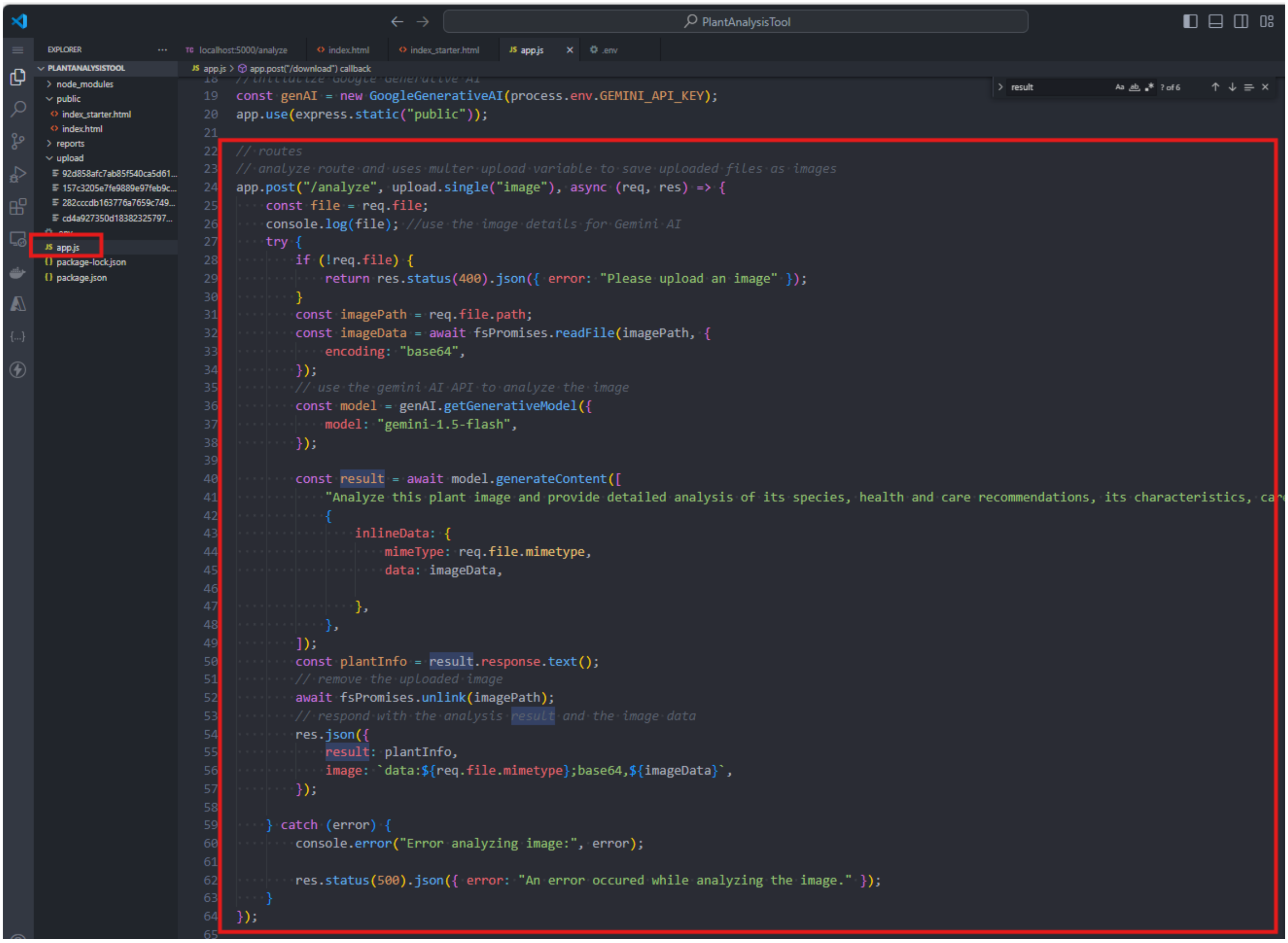


```
<html lang="en">
  <body>
    <script>
      dropArea.addEventListener("change", (event) => {
        function handleImageUpload(file){
        }

        //uploadForm logic
        uploadForm.addEventListener("submit",async (e) =>{
            e.preventDefault();
            const formData =  new FormData(e.target)
            loadingDiv.style.display = "block";
            resultDiv.style.display = "none";
            resultDiv.textContent = "";
            downloadButton.style.display = "none";
            try{
                const  response =  await fetch('/analyze',{
                    method: "POST",
                    body: formData,
                });
                const data = await response.json();
                if (data.result){
                    // retrieved from the apps.js json response
                    analysisResult = data.result;
                    analysisImage = data.image;
                    resultDiv.innerHTML =
                    "<h3>Analysis Result:</h3><p>" +
                    analysisResult.replace(/\n/g, "<br>") +
                    "</p>";
                    resultDiv.style.display = "block";
                    downloadButton.style.display = "block";
                    loadingDiv.style.display = "none";

                } else if (data.error) {
                    resultDiv.textContent = "Error: " + data.error;
                    resultDiv.style.display = "block";
                }

            } catch (error){
                resultDiv.textContent = "Error: " + error.message;
                resultDiv.style.display = "block";

            } finally {
                loadingDiv.style.display = "none";
            }
        });

        downloadButton.addEventListener("click",async () => {
```

The LOGIC is performed in APPS.JS:

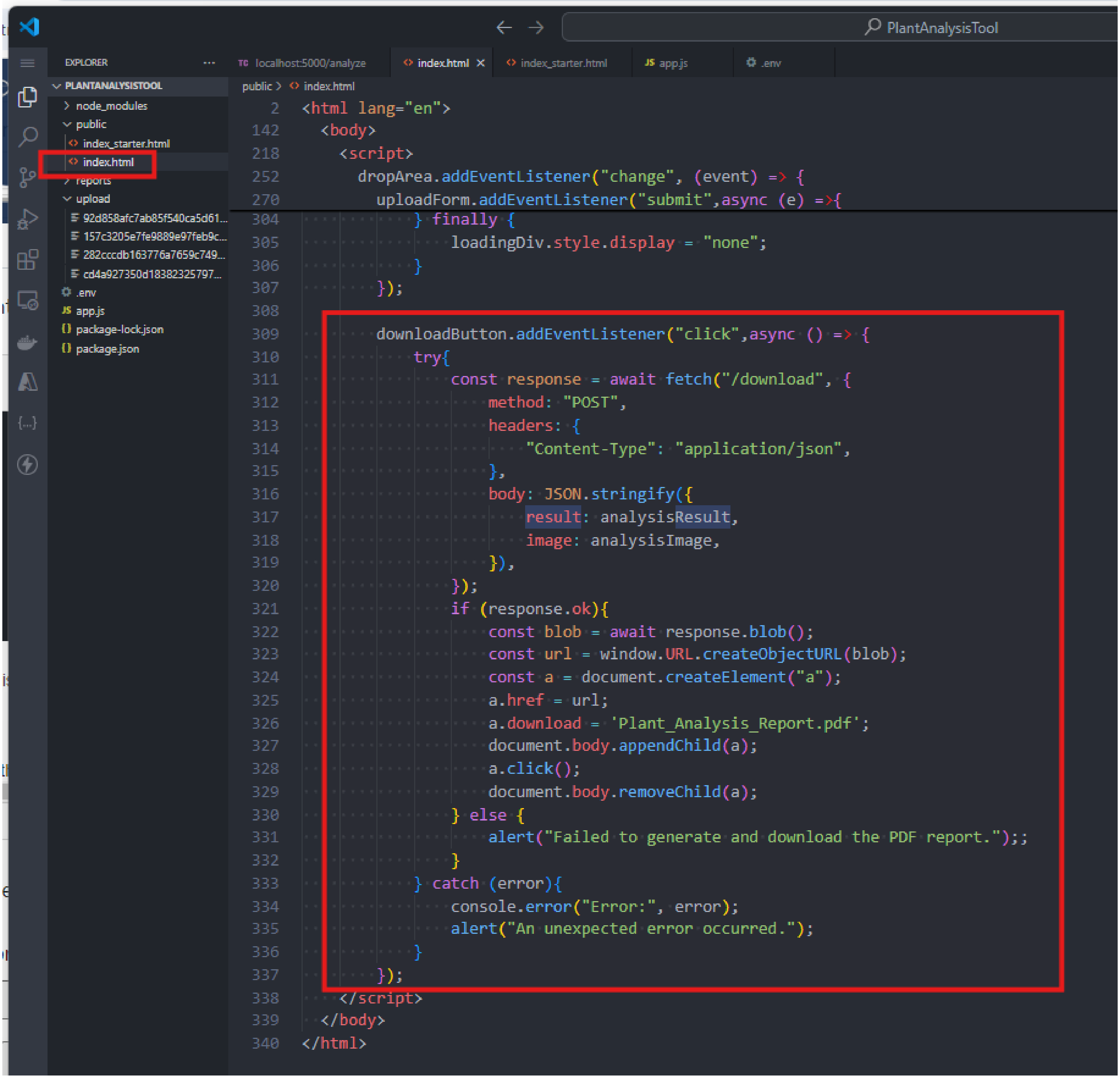


```javascript
require("dotenv").config();
const express = require("express");
const multer = require("multer");
const PDFDocument = require("pdfkit");
const fs = require("fs");
const fsPromises = fs.promises;
const path = require("path");
const { GoogleGenerativeAI } = require("@google/generative-ai");

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

// configure multer: save uploaded files to /upload folder & limit file file size
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"));
```



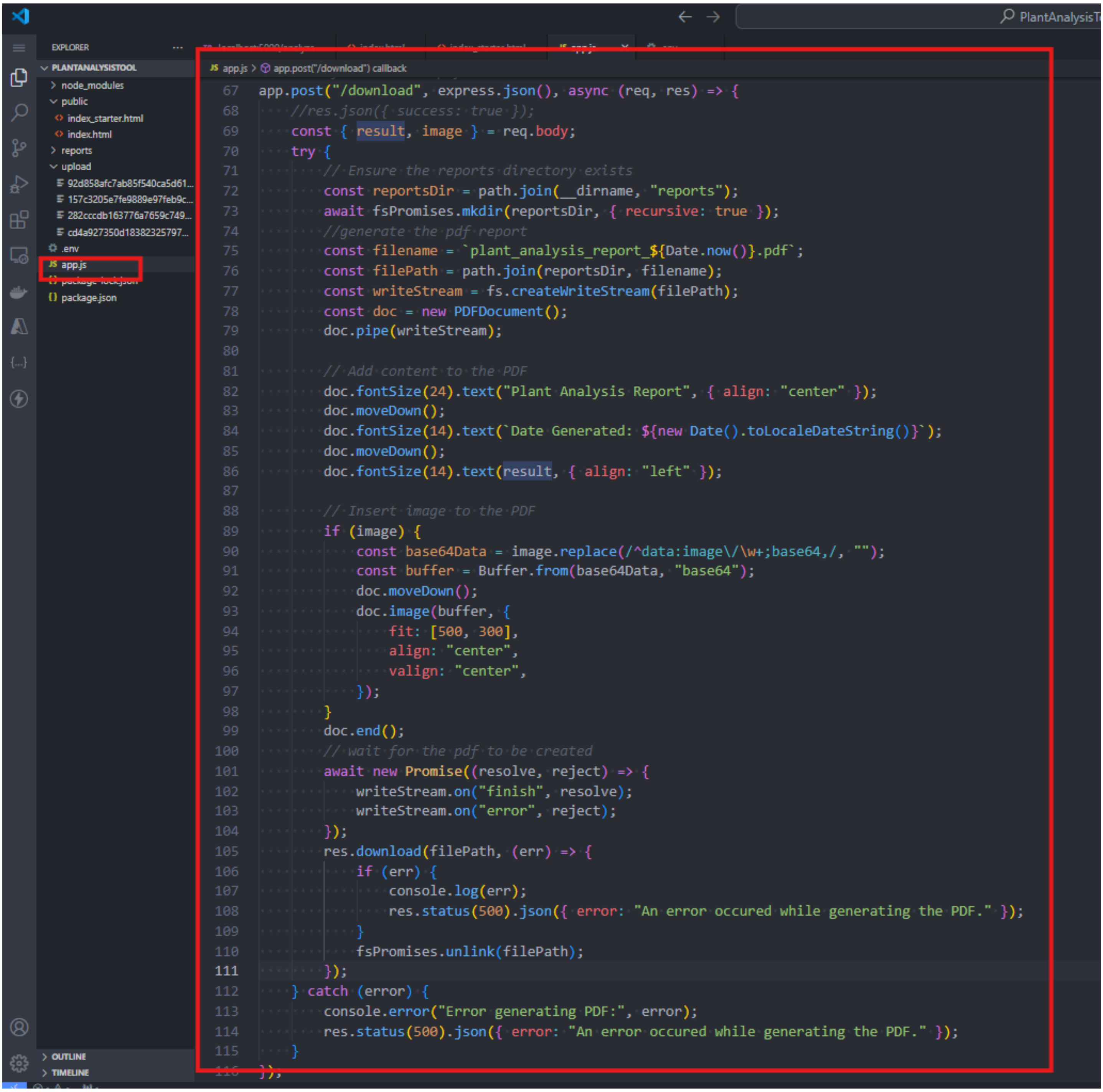
```javascript
const genAI = new GoogleGenerativeAI(process.env.GEMINI_API_KEY);
app.use(express.static("public"));

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

                },
            },
        ]);
        const plantInfo = result.response.text();
        // remove the uploaded image
        await fsPromises.unlink(imagePath);
        // respond with the analysis result and the image data
        res.json({
            result: plantInfo,
            image: `data:${req.file.mimetype};base64,${imageData}`,
        });

    } catch (error) {
        console.error("Error analyzing image:", error);

        res.status(500).json({ error: "An error occured while analyzing the image." });
    }
});
```

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



```
<html lang="en">
  <body>
    <script>
      dropArea.addEventListener("change", (event) => {
        uploadForm.addEventListener("submit",async (e) =>{
            } finally {
                loadingDiv.style.display = "none";
            }
        });

        downloadButton.addEventListener("click",async () => {
            try{
                const response = await fetch("/download", {
                    method: "POST",
                    headers: {
                        "Content-Type": "application/json",
                    },
                    body: JSON.stringify({
                        result: analysisResult,
                        image: analysisImage,
                    }),
                });
                if (response.ok){
                    const blob = await response.blob();
                    const url = window.URL.createObjectURL(blob);
                    const a = document.createElement("a");
                    a.href = url;
                    a.download = 'Plant_Analysis_Report.pdf';
                    document.body.appendChild(a);
                    a.click();
                    document.body.removeChild(a);
                } else {
                    alert("Failed to generate and download the PDF report.");;
                }
            } catch (error){
                console.error("Error:", error);
                alert("An unexpected error occurred.");
            }
        });
    </script>
  </body>
</html>
```

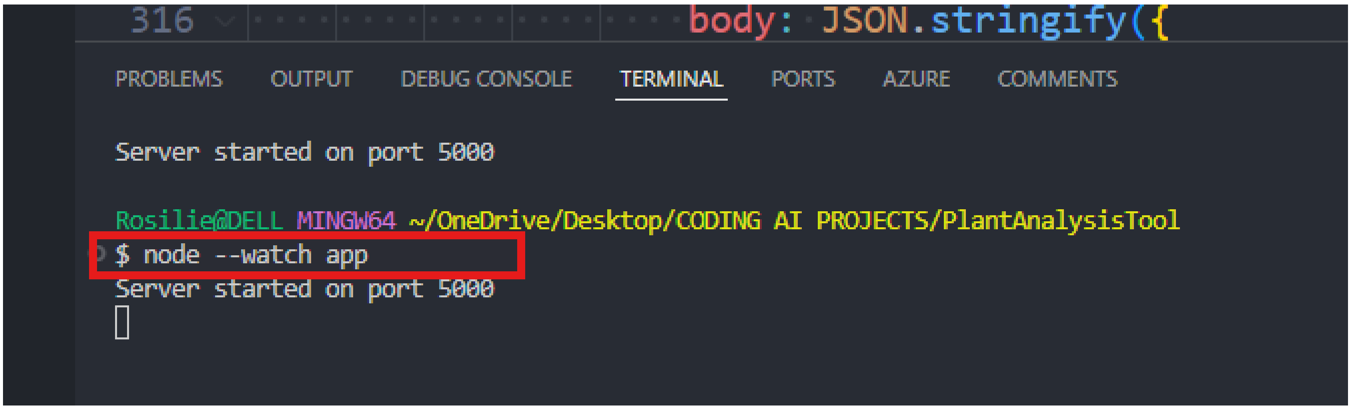
This is the logic in APPS.JS



```javascript
app.post("/download", express.json(), async (req, res) => {
  //res.json({ success: true });
  const { result, image } = req.body;
  try {
    // Ensure the reports directory exists
    const reportsDir = path.join(__dirname, "reports");
    await fsPromises.mkdir(reportsDir, { recursive: true });
    //generate the pdf report
    const filename = `plant_analysis_report_${Date.now()}.pdf`;
    const filePath = path.join(reportsDir, filename);
    const writeStream = fs.createWriteStream(filePath);
    const doc = new PDFDocument();
    doc.pipe(writeStream);

    // Add content to the PDF
    doc.fontSize(24).text("Plant Analysis Report", { align: "center" });
    doc.moveDown();
    doc.fontSize(14).text(`Date Generated: ${new Date().toLocaleDateString()}`);
    doc.moveDown();
    doc.fontSize(14).text(result, { align: "left" });

    // Insert image to the PDF
    if (image) {
      const base64Data = image.replace(/^data:image\/\w+;base64,/, "");
      const buffer = Buffer.from(base64Data, "base64");
      doc.moveDown();
      doc.image(buffer, {
        fit: [500, 300],
        align: "center",
        valign: "center",
      });
    }
    doc.end();
    // wait for the pdf to be created
    await new Promise((resolve, reject) => {
      writeStream.on("finish", resolve);
      writeStream.on("error", reject);
    });
    res.download(filePath, (err) => {
      if (err) {
        console.log(err);
        res.status(500).json({ error: "An error occured while generating the PDF." });
      }
      fsPromises.unlink(filePath);
    });
  } catch (error) {
    console.error("Error generating PDF:", error);
    res.status(500).json({ error: "An error occured while generating the PDF." });
  }
});
```

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

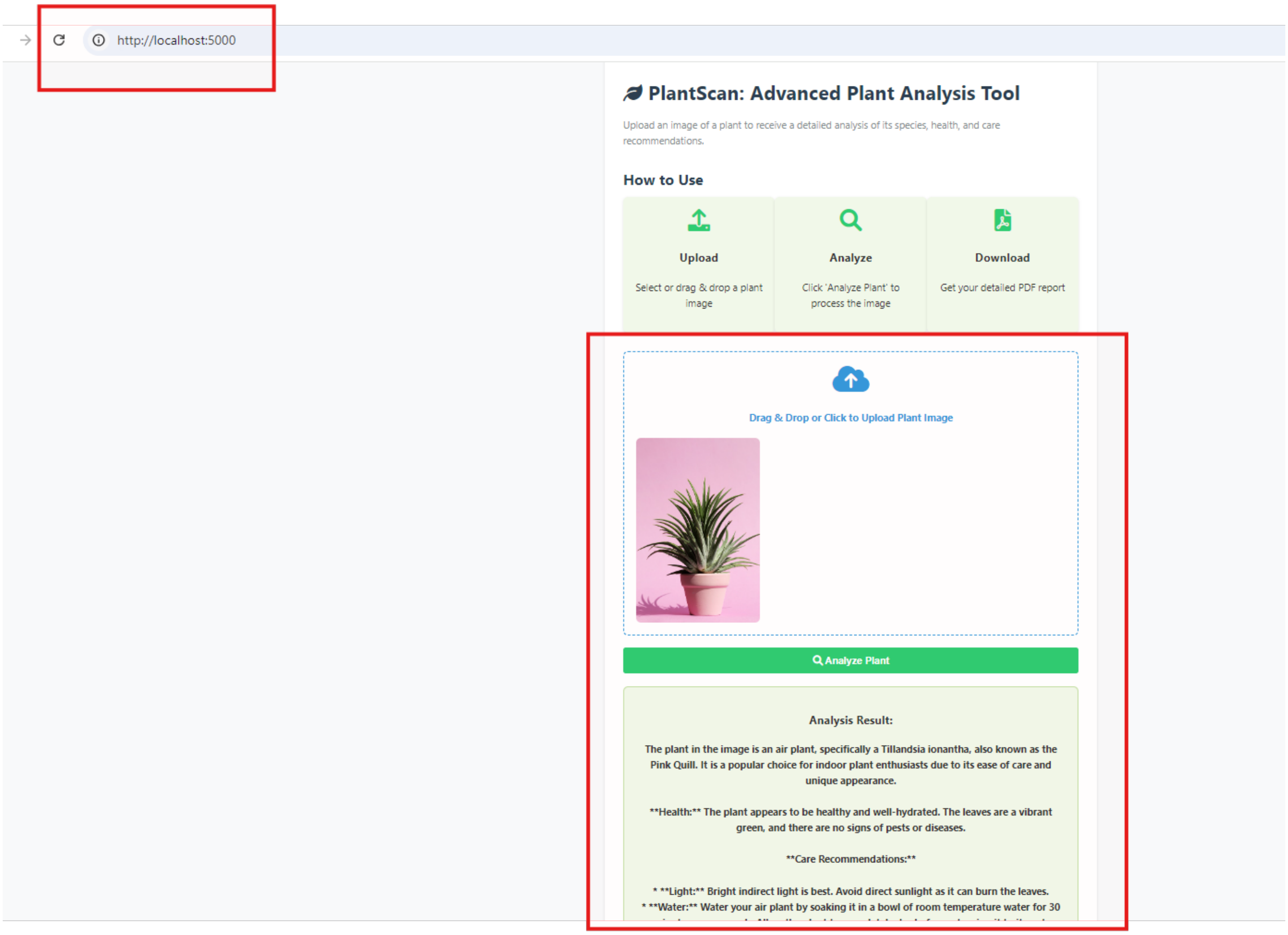
$ node --watch app



```
316   body: JSON.stringify({
```

```
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

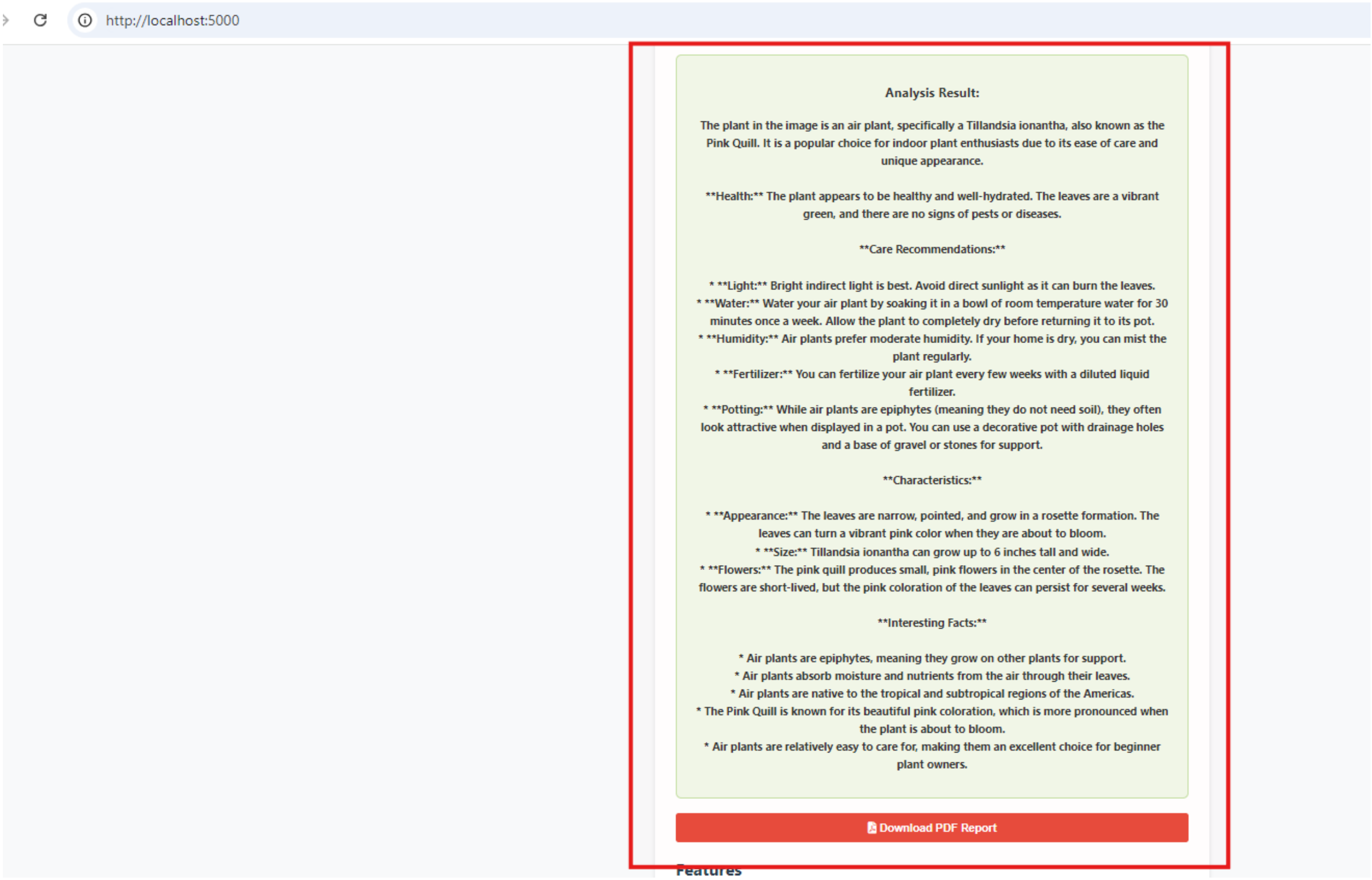
Analysis Result:

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



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.

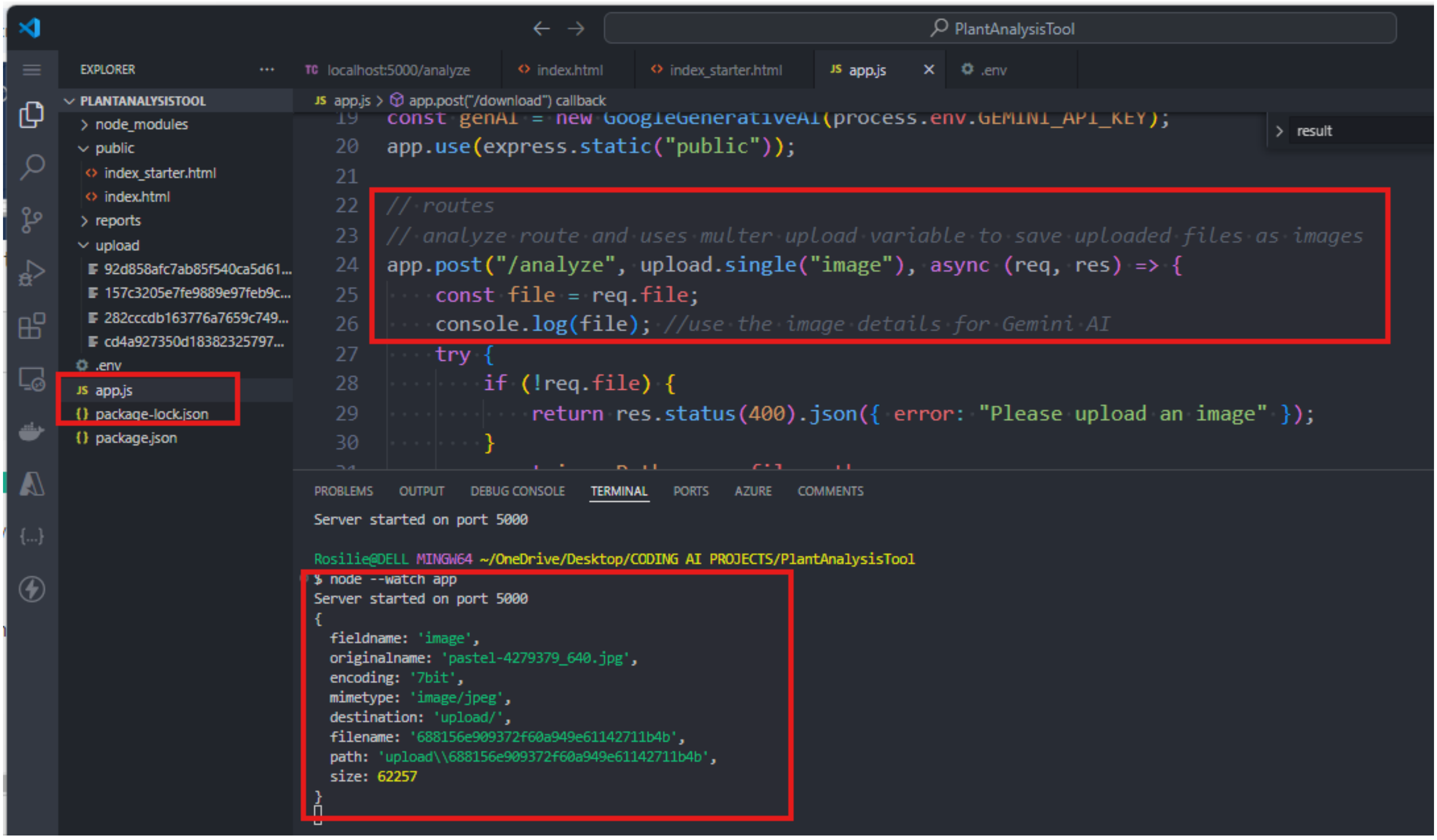
Interesting Facts:

* 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.

Features

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:



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

```
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.