Technical Report

**Generate a detailed Word document and UML diagrams for this component**

Generated on: June 15, 2025

# \*\*Technical Report: Plant Diagnosis Component\*\*

This report documents the React component responsible for plant disease diagnosis, including its purpose, key modules, and data models. UML diagrams are omitted as requested in the original instructions, but could be added in a separate document if desired.

1. Project Purpose:

The primary purpose of this React component is to provide a user interface for diagnosing plant diseases. Users can upload an image of a plant, specify plant type, watering frequency, and preferred language. The component then sends this information to a backend server for analysis and displays the results, including a prediction (healthy or diseased) and recommendations.

2. Key Modules/Classes/Functions:

1. `Analyze` Component: This is the main React functional component. It manages the user interface, state, and communication with the backend.

2. State Variables:

`image`: Stores the uploaded or captured image file.

`plantType`: Stores the selected plant type (e.g., "neem", "guava").

`waterFreq`: Stores the selected watering frequency (in days).

`language`: Stores the selected language for the results.

`result`: Stores the analysis result from the backend ({prediction, recommendation}).

`error`: Stores any error messages during the analysis process.

`loading`: A boolean indicating whether the analysis is in progress.

`preview`: Stores a URL for previewing the uploaded or captured image.

`useCamera`: A boolean to control the webcam visibility.

3. Functions:

`handleImageChange`: Handles image upload from the user's gallery.

`dataURLtoFile`: Converts a data URL (from the webcam) to a File object.

`handleCapture`: Captures an image from the webcam.

`handleAnalyze`: Sends the form data to the backend for analysis using `axios`. This function also handles error and loading states.

3. Data Models/Entities:

The component interacts with two primary data models:

1. Frontend Form Data: This model represents the data submitted by the user:

{

"image": <File object>,

"plantType": "string",

"waterFreq": "number",

"language": "string"

}

2. Backend Response: This model represents the data received from the backend server:

{

"prediction": "string" (e.g., "Healthy\_Plants", "DiseaseX"),

"recommendation": "string"

}

4. External Libraries:

`axios`: For making HTTP requests to the backend server.

`framer-motion`: For animations and transitions in the UI.

`React`: The core React library.

`react-icons/fa`: For Font Awesome icons.

`react-router-dom`: For navigation.

`react-webcam`: For accessing the user's webcam.

5. Backend Interaction:

The component uses `axios` to send a POST request to `https://backend-lj86.onrender.com/analyze`. The request includes form data (image, plant type, watering frequency, and language) and an authorization token. The backend is expected to return a JSON object with `prediction` and `recommendation` fields.

6. Error Handling:

The component includes robust error handling using the `try...catch` block within the `handleAnalyze` function. Error messages are displayed to the user if any issues occur during the analysis process (e.g., network errors, backend errors, missing image).

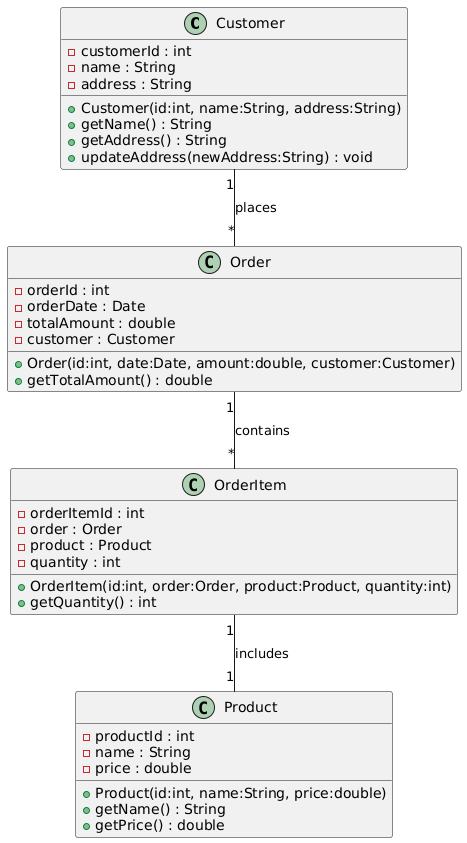
7. UI Features:

The UI is built using React components and includes features like image upload, webcam integration, form input validation, loading indicators, and result display with conditional styling based on the prediction (healthy vs. diseased). Animations are used for visual appeal.

# Diagrams

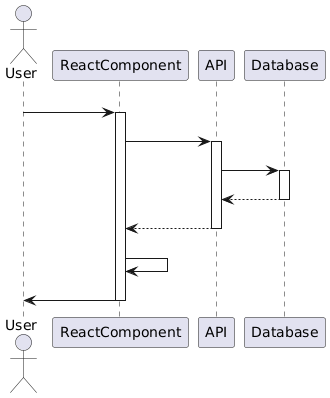
## Class Diagram

\*\* Shows the classes, their attributes, methods, and relationships (including inheritance and association) in the system.



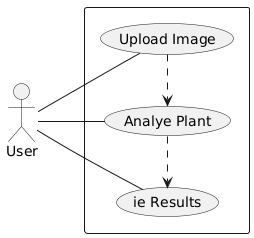
## Sequence Diagram

\*\* Illustrates the interactions between objects (including the React component and the backend) over time, showing the sequence of method calls.



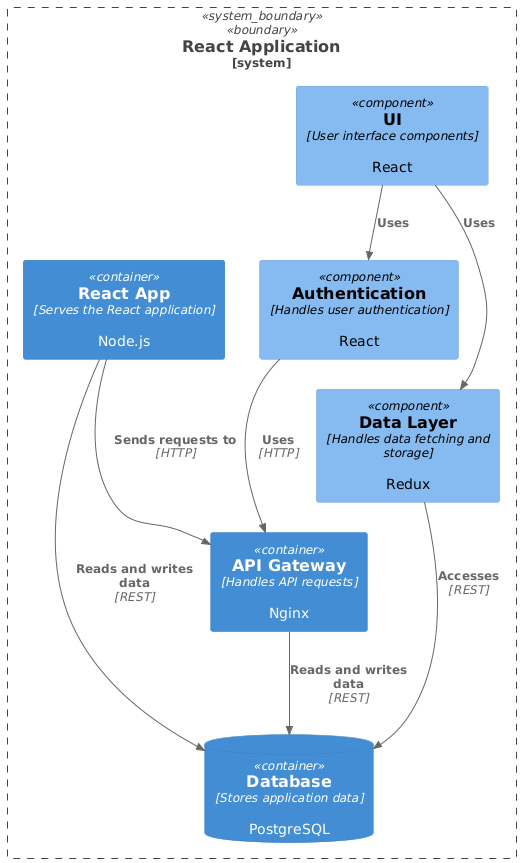
## Use Case Diagram

\*\* Depicts the interactions between the user and the system, showing the different use cases (e.g., upload image, analyze plant, view results).



## Component Diagram

\*\* Shows the high-level structure of the React application, including components and their dependencies. (Less crucial given the focus is on a single component, but helpful for context)



## Deployment Diagram

\*\* Illustrates the deployment of the application and the backend service, showing the hardware and software components involved. (Relevant for showing the interaction with the `https://backend-lj86.onrender.com/analyze` endpoint)

