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 diagnosis, analyzing its purpose, key modules, and data model. UML diagrams are beyond the scope of this markdown report, but the component's structure and data flow are clearly described.

1. Purpose:

The `Analyze` component provides a user interface for diagnosing plant health. Users upload or capture an image of a plant, select the plant type, water frequency, and preferred language. The component then sends this data to a backend server (specified as `https://backend-lj86.onrender.com/analyze`) for analysis. The server returns a prediction (e.g., "Healthy\_Plants", "DiseaseX") and a recommendation, which are displayed to the user. If the plant is unhealthy, a button links to relevant cure information.

2. Key Modules/Classes/Functions:

React Modules: The component leverages core React functionalities like `useState` for managing component state, `useEffect` for lifecycle management (cleaning up URL objects), and `useRef` for accessing the webcam instance. `useNavigate` from `react-router-dom` is used for navigation.

Framer Motion: `motion` from `framer-motion` is used for animations, providing smooth transitions and hover effects on UI elements.

React Icons: Several icons from `react-icons/fa` (Font Awesome) are used to enhance the user interface.

Webcam.js: `react-webcam` is integrated for capturing images directly from the user's webcam.

Axios: `axios` is used for making HTTP requests to the backend server for plant analysis.

Key Functions:

`handleImageChange`: Handles image uploads from the user's device.

`dataURLtoFile`: Converts a data URL (from the webcam) into a File object suitable for form data submission.

`handleCapture`: Captures an image from the webcam.

`handleAnalyze`: Sends the form data to the backend server, handles loading state, and manages errors.

3. Data Models/Entities:

The component interacts with the following data:

Frontend State:

`image`: A File object representing the uploaded or captured plant image (null initially).

`plantType`: String representing the type of plant (default: "neem").

`waterFreq`: String representing the watering frequency in days (default: "1").

`language`: String representing the preferred language for the recommendation (default: "english").

`result`: Object containing the prediction and recommendation from the server (null initially).

`error`: String containing any error messages (empty string initially).

`loading`: Boolean indicating whether the analysis is in progress (false initially).

`preview`: String representing the URL of the image preview (null initially).

`useCamera`: Boolean indicating whether the webcam is active (false initially).

Backend Data (inferred from API response):

`{ prediction: string, recommendation: string }`: The server's response contains a `prediction` (plant health status) and a `recommendation` (advice for care).

4. Component Structure Summary:

The component renders a form with fields for plant type, water frequency, language selection, image upload (from gallery or webcam), and a submit button. It displays a preview of the uploaded/captured image. Upon submission, it sends the data to the backend, displays a loading indicator, and presents the analysis result with appropriate visual cues (green for healthy, red for unhealthy). Error handling is included to inform users about issues like missing images or authentication failures. Navigation back to the home page (`/`) is also provided.

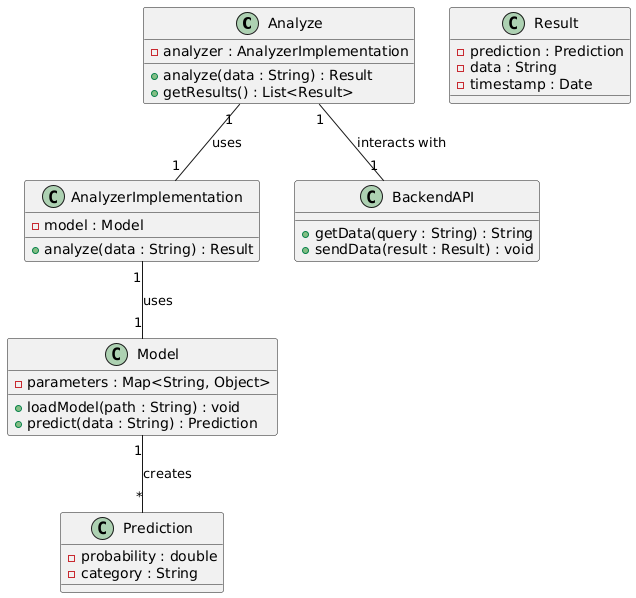
5. Conclusion:

The `Analyze` component is a well-structured React component that effectively handles user input, manages asynchronous operations (backend communication), and presents the results in a user-friendly manner. The use of animations and icons enhances the user experience. The code demonstrates good error handling and resource management (cleaning up object URLs).

# Diagrams

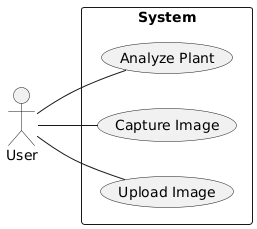
## Class Diagram

Illustrates the classes, attributes, and methods within the system, including the relationship between the `Analyze` component and other classes (e.g., potential backend classes implied by the API interaction).



## Use Case Diagram

Depicts the interactions between the user (actor) and the system, showing the various use cases like "Upload Image," "Capture Image," and "Analyze Plant."



## Sequence Diagram

Shows the sequence of interactions between the React component, the backend API (`/analyze` endpoint), and potentially other components or services involved in the image analysis process.

