Business Requirements Document (BRD)

Project Name: GreenEye

Prepared by: Majd Fares Adnan Al Hatoum

Role: Requirements Engineer & QA Lead

Date: May 1st 2025

1. Introduction

GreenEye is an intelligent plant identification web application that allows users to upload one or more images of plant leaves in order to receive accurate species predictions. The system leverages pre-trained machine learning models to classify plants based on visual characteristics and returns results with confidence scores. It is designed for casual users, hobbyist gardeners, and researchers who require quick species identification using only visual input.

2. Actors (Users)

Actor	Description
End User	A user who uploads leaf images to receive plant species predictions.

3. System Modules in Scope

- - Image Upload Module
- - Prediction Engine (Species Classification Only)
- - Double-Model Results
- - Error Handling

4. Functional Requirements

4.1 Image Upload Module

- FR-1.1 - Upload Single Image

The system shall allow users to upload a single leaf image in JPEG or PNG format.

- FR-1.2 - File Validation

The system shall validate uploaded images by type (JPEG/PNG), size (max 10MB), and minimum resolution (e.g., 300x300 px).

- FR-1.4 - Show Preview

The system shall display a preview of each uploaded image prior to submission.

- FR-1.5 - Remove Uploaded Image

The system shall allow users to remove an image from the preview list before submitting the prediction request.

4.2 Prediction Engine (Species Classification Only)

- FR-2.1 - Predict Plant Species

The system shall analyze uploaded leaf images and return the predicted plant species.

- FR-2.2 - Display Confidence Score

The system shall display a numeric confidence score (e.g., 92%) for each prediction.

- FR-2.3 - Handle Multi-leaf Fusion

The system shall aggregate prediction results from multiple uploaded images and return the most confident or consistent result.

- FR-2.4 – Two Models

The system shall run submitted images through both the fine-tuned model and the newly trained model, and return two separate predictions.

- FR-2.5 – Separate Predictions

The system shall display the plant species name and confidence score from each model side-by-side for user comparison.

- FR-2.6 – Tag Predictions

The system may provide short labels or descriptions identifying which model made each prediction (e.g., “Model A – Fine-tuned” and “Model B – Custom Trained”).

4.3 Error Handling

- FR-3.1 - Unsupported File Types

The system shall notify users when uploaded files are not valid image formats.

- FR-3.2 - Image Quality Warnings

The system shall warn users if images are too small, blurry, or low-quality.

- FR-3.3 - Prediction Failure Handling

If no prediction can be made, the system shall return a user-friendly message indicating the failure and advising next steps.

5. Traceability Note

Each of the functional requirements above will be linked to corresponding user stories (defined in Sprint 1), UI wireframes (provided by the Frontend Developer), backend and ML endpoints (defined in Sprint 2), and test cases (prepared during Sprint 2 and executed in Sprint 3 & 4).

Functional Requirements - User Stories & Acceptance Criteria

US-1: Upload Single Image

As an end user, I want to upload a single plant leaf image so that I can get a species prediction.

Acceptance Criteria:

- - Given I am on the image upload screen, When I select a valid image file (JPEG/PNG), Then it should appear in the preview area.
- - Given I have uploaded one image, When I click the submit button, Then the system should analyze the image and return a prediction.

US-2: Upload Multiple Images

As an end user, I want to upload multiple images of the same plant so that the system can make a more accurate prediction.

Acceptance Criteria:

- - Given I am on the upload screen, When I select multiple images (up to 3), Then they should all appear in the preview area.
- - Given I uploaded 2–3 images, When I submit the prediction request, Then the system should return a single species prediction using all uploaded images.

US-3: Remove Uploaded Image

As an end user, I want to remove an image before submitting so that I can correct any mistaken uploads.

Acceptance Criteria:

- - Given I have uploaded multiple images, When I click the “remove” or “X” button on a preview, Then that image should disappear from the upload list.

US-4: Display Confidence Score

As an end user, I want to see how confident the system is in its prediction so that I can assess the reliability of the result.

Acceptance Criteria:

- - Given the system completes a prediction, When it displays the result, Then it should show a confidence score as a percentage (e.g., 92%).

US-5: File Validation

As an end user, I want the system to only accept valid image files so that I don't waste time uploading incorrect formats.

Acceptance Criteria:

- - Given I attempt to upload an unsupported file type, When the upload is triggered, Then the system should show an error message and reject the file.
- - Given I upload a file over 10MB or under the required resolution, Then the system should reject the file and explain why.

US-6: Error Messages for Prediction Failure

As an end user, I want to receive a helpful error message if the prediction fails so that I understand what went wrong.

Acceptance Criteria:

- - Given the model cannot return a result, When the request is processed, Then the system should display a clear message (e.g., “We couldn’t identify the plant. Try another image.”)

US-7: Receive Predictions from Two Models

As a user, I want to see plant identification results from both models so that I can compare their predictions and confidence scores.

Acceptance Criteria:

- - Given I upload a valid image and submit it, when the results are returned, then I should see two predictions, each labeled with its respective model type.
- - Given both models return predictions, when I view the result, then I should see both species names and their confidence percentages clearly displayed side-by-side.

US-8: Identify Which Model Made Which Prediction

As a user, I want to know which model produced each prediction so that I can understand the source of the results.

Acceptance Criteria:

- - Given the predictions are displayed, when I view the results, then each prediction must be labeled (e.g., 'Pretrained Model' vs. 'Custom Model').
- - Given one model’s confidence score is significantly lower, when I hover or tap on it, then I should see optional tooltip text indicating which model was used and its brief description.

Non-Functional Requirements (NFRs)

1. Performance Requirements

- NFR-1.1: The system shall return predictions from both models within a total of 7 seconds for standard (single) image inputs, 95% of the time.
- NFR-1.2: The system shall support 50 simultaneous active users with an average response time of less than 6 seconds under normal usage.
- NFR-1.3: Both models shall process predictions concurrently (i.e., in parallel) to optimize response time.

2. Usability Requirements

- NFR-2.1: At least 90% of users shall be able to complete the upload and prediction process without assistance on their first attempt.
- NFR-2.2: The system shall support keyboard navigation and screen reader compatibility to meet WCAG 2.1 Level AA accessibility standards.
- NFR-2.3: All labels, buttons, and instructions shall have readability scores equivalent to B2/C1 (Common European Framework) level or lower.
- NFR-2.4: All input actions (e.g., image upload, remove image) shall receive visual confirmation within 500 milliseconds of the action being triggered.
- NFR-2.5: The system shall provide tooltip/help text for at least 100% of functional buttons or inputs not self-explanatory by label.
- NFR-2.6: Each model's prediction shall be visually distinguishable by label or color coding in the frontend UI.
- NFR-2.7: Both models' confidence outputs must be scaled (if needed) to use a consistent 0–100% format for comparison.

3. Reliability & Availability Requirements

- NFR-3.1: The system shall maintain 95% uptime over a rolling 30-day period, excluding scheduled maintenance.
- NFR-3.2: The system shall provide a retry mechanism that automatically retries a failed prediction request up to 2 times before reporting failure.
- NFR-3.3: The system shall handle and log 100% of prediction-related errors with relevant timestamps and status codes.
- NFR-3.4: Image upload failures due to network errors shall trigger a retry prompt or allow the user to re-upload without refreshing the page.
- NFR-3.5: The system shall achieve a prediction success rate of at least 97% under ideal image quality and format conditions.
- NFR-3.6: If one model fails to return a prediction, the system shall still return the other model's result with an appropriate message.

4. Security Requirements

- NFR-4.1: All communication between frontend and backend shall use TLS 1.2 or higher (i.e., HTTPS).

- NFR-4.2: Uploaded images shall be temporarily stored and automatically deleted from the server within 60 minutes unless explicitly retained.

5. Portability & Compatibility

- NFR-5.1: The frontend shall be compatible with Chrome, Firefox, Safari, and Edge (latest 2 versions).
- NFR-5.2: The application shall run inside a Docker container and deploy successfully on AWS, GCP, or Azure using default Kubernetes configurations.

Non-Functional Requirements – User Stories & Acceptance Criteria

NFR-1.1: Prediction Time Within Limits

As an end user, I want the system to return predictions from both models quickly so that I don't experience frustrating delays.

Acceptance Criteria:

- Given I upload a single image,
  When I submit the prediction request,
  Then the system shall return both model predictions within 7 seconds, 95% of the time.

- Given I upload three images,
  When I submit the prediction request,
  Then the system shall return both predictions per image within 10 seconds, 95% of the time.

- Given the system is under normal load,
  When multiple users submit prediction requests simultaneously,
  Then logs and metrics must confirm that response time SLAs are consistently met for dual-model predictions.

NFR-1.2: Concurrent Users Support

As a system administrator, I want the system to handle multiple users making dual-model predictions so that the application remains responsive under load.

Acceptance Criteria:

- Given 50 users are actively using the prediction feature,
  When they upload and submit images simultaneously,
  Then the system shall maintain an average dual-model response time of under 8 seconds.

- Given a load test simulating concurrent dual predictions,
  When monitored,
  Then system performance metrics (CPU, memory, latency) shall remain within safe thresholds with no prediction failures.

NFR-2.1: Completion Without Assistance

As an end user, I want to complete the upload and prediction process easily so that I don't need any external help or documentation.

Acceptance Criteria:

- - Given I am a new user, when I open the GreenEye interface, then I should be able to upload and submit images and receive predictions without needing any external guidance.

- - Given a usability test involving 10 users, when they perform the main task without instructions, then at least 9 of them should complete it successfully.

NFR-2.2: Accessibility Compliance

As a user with visual or motor impairments, I want the application to be accessible so that I can use it with assistive technologies.

Acceptance Criteria:

- - Given I navigate the site using a keyboard only, when I tab through interface elements, then all fields, buttons, and images should be reachable and highlighted.
- - Given I use a screen reader, when I focus on input fields and buttons, then the reader should announce their labels and purposes clearly.

NFR-2.3: Readable Language

As a general user, I want the system language to be simple and clear so that I can understand all messages without confusion.

Acceptance Criteria:

- - Given I interact with the system, when messages are displayed (errors, confirmations, tooltips), then each message shall have a readability level not exceeding CEFR B2/C1.
- - Given a linguistic review tool is used, when applied to all user-facing text, then results shall confirm compliance with the target readability level.

NFR-2.4: Immediate Visual Feedback

As a user, I want instant feedback when I take actions so that I know something is happening in the system.

Acceptance Criteria:

- - Given I upload or remove an image, when the action completes, then the interface shall display confirmation within 500 milliseconds.
- - Given I click any actionable button, when the action is triggered, then a loading animation or state change must occur within half a second.

NFR-2.5: Help Text for All Inputs

As a first-time user, I want tooltips or help indicators so that I understand each button or field I interact with.

Acceptance Criteria:

- - Given I hover over or focus on an input or icon, when it is not self-explanatory, then a tooltip or help message should appear with a short description.

- - Given a UI audit, when evaluating all functional inputs and buttons, then 100% of them should have either a descriptive label or a tooltip.

NFR-3.1: 95% Uptime

As an admin, I want the system to be available consistently so that users can access it reliably without frequent downtime.

Acceptance Criteria:

- - Given a 30-day period, when system availability is measured (excluding planned maintenance), then uptime should be ≥ 95% as shown in system logs or monitoring tools.
- - Given the system goes down unexpectedly, when it recovers, then the total downtime must not exceed 36 hours in 30 days.

NFR-3.2: Retry on Prediction Failure

As a user, I want the system to retry failed predictions independently for each model so that I still receive results even if one model encounters an issue.

Acceptance Criteria:

- Given a prediction request is submitted,
  When one model fails due to a temporary error,
  Then the system shall **retry that specific model** up to **2 times**, while continuing to process the other model independently.

- Given both retry attempts fail for one model,
  Then the user shall still receive a response from the other model, along with a **clear error message** explaining that only one result is available.

- Given a system failure affects both models,
  Then the final error shown must clarify that both prediction attempts failed, and suggest retrying the submission.

NFR-3.3: Error Logging with Timestamps

As a product owner, I want to ensure that both models produce reliable results under good conditions so that users trust the predictions.

Acceptance Criteria:

- Given ideal image conditions (clear, supported species, correct format),
  When 100 predictions are tested,
  Then each model shall individually meet or exceed **97% accuracy** on test sets.

- Given a post-deployment validation,
  When predictions from both models are logged,

Then confidence scores and output accuracy metrics shall be calculated and compared across both models for reporting.

- Given continuous integration or retraining,
Then the benchmark tests must be rerun to ensure both models consistently meet the performance threshold.

NFR-3.4: Network Upload Resilience

As a user, I want the image upload process to be resilient to network drops so that I don't lose progress during submission.

Acceptance Criteria:

- Given a network error occurs during image upload, when I reconnect or retry, then the system shall allow me to re-upload without page refresh or prompt me to retry the request.
- Given auto-retry is enabled, when the upload fails temporarily, then the system shall attempt at least 1 auto-retry before showing an error.
- If only one model completes prediction after upload recovery, the system shall display that result and notify the user of partial prediction availability.

NFR-3.5: Prediction Accuracy Benchmark

As a product owner, I want to ensure that the model performs reliably with quality images so that the application builds user trust.

Acceptance Criteria:

- Given ideal image conditions (clear, correct resolution, supported species), when 100 predictions are tested, then at least 97 of them shall match expected species labels.
- Given a validation test is performed post-deployment, then model performance logs must show ≥ 97% success rate on the test dataset.

NFR-4.1: Encrypted Communication

As a security-conscious user, I want my data to be transmitted securely so that no one can intercept my uploads or results.

Acceptance Criteria:

- - Given the system is accessed over the internet, when data is sent between frontend and backend, then it shall be encrypted using TLS 1.2 or higher.
- - Given an HTTP request is attempted, when a non-secure (HTTP) connection is made, then the system shall automatically redirect to HTTPS.

NFR-4.2: Temporary Image Storage

As a user, I want to know that my uploaded images are not stored longer than needed so that my data is not misused.

Acceptance Criteria:

- - Given an image is uploaded for prediction, when the prediction is completed, then the image file shall be marked for deletion.
- - Given no explicit retention request is made, when 60 minutes pass after upload, then the system shall automatically delete the image from the server.
- - Given a storage audit is performed, when examining upload directories, then no image older than 60 minutes should be found (unless retention is active).

NFR-5.1: Cross-Browser Compatibility

As a user, I want to use the system on any modern browser so that I can access it from my preferred platform.

Acceptance Criteria:

- - Given I access the system using the latest versions of Chrome, Firefox, Safari, or Edge, when I navigate through all core functionalities, then the system shall work without layout or functionality issues.
- - Given a browser compatibility test, when the test is executed using automation tools or manual QA, then 100% of supported browsers shall pass UI and behavior consistency checks.

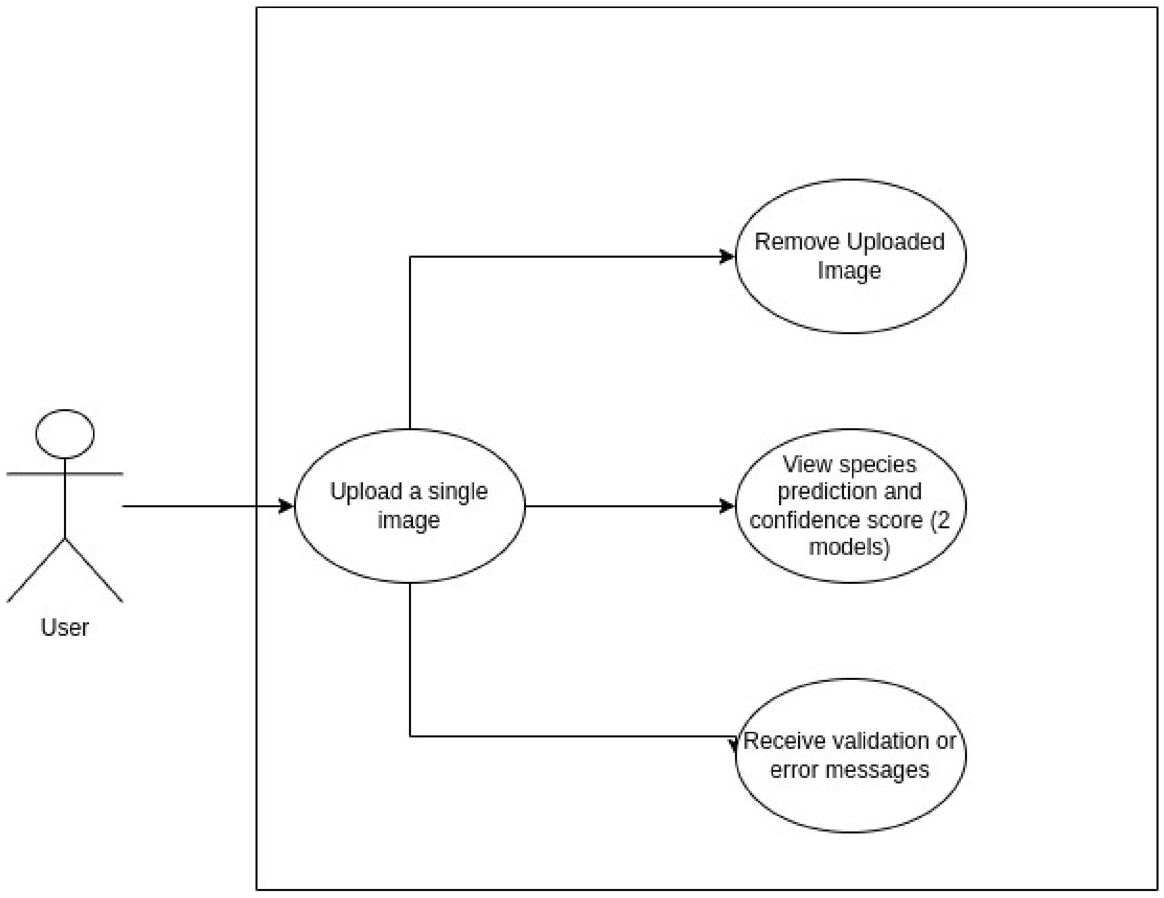
NFR-5.2: Containerized Deployment Across Cloud Platforms

As a DevOps engineer, I want the application to run inside Docker and deploy to any major cloud platform so that I can maintain flexible infrastructure.

Acceptance Criteria:

- - Given the system is containerized using Docker, when the image is deployed to AWS, GCP, or Azure, then the system shall start and operate as expected using default Kubernetes settings.
- - Given CI/CD pipelines are used, when the application is deployed on a fresh cluster, then no platform-specific changes should be required for successful operation.
- - Given infrastructure logs are reviewed, when startup and liveness checks are performed, then the system shall be shown to be fully operational in all three platforms (AWS, GCP, Azure).

Use Case Diagram Explanation – GreenEye System


Remove Uploaded Image
User
Upload a single image
View species prediction and confidence score (2 models)
Receive validation or error messages

In this Use Case Diagram of GreenEye's system, we present a streamlined and focused view of how end-users can interact with the application's core functionality.

The primary actor in this diagram is the User, who may be a gardener, hobbyist, or agriculture enthusiast seeking to identify a plant using visual input. The interaction begins with the user uploading a single image of a plant leaf. Upon upload, the system performs validation checks to ensure the image is in a supported format and of adequate quality. If the user changes their mind, they can remove the image before submission.

Once an image is validated and submitted, the system invokes two different machine learning models — a fine-tuned pre-trained model and a newly trained custom model — to generate predictions. The system then displays the species predicted by each model, along with their respective confidence scores, allowing the user to compare and assess the results.

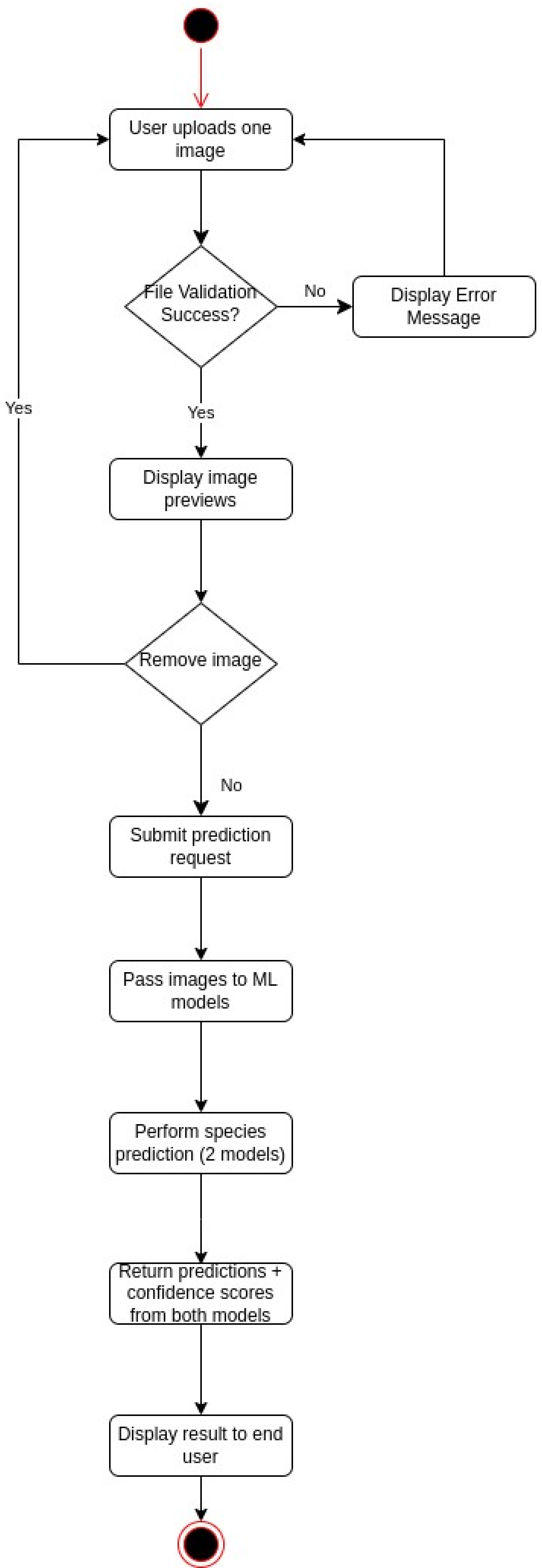
System Process Flow Explanation – GreenEye (Below)

The GreenEye system process flow begins when a user accesses the interface and initiates the species identification feature by uploading a single image of a plant leaf. As soon as the image is submitted, the system performs a validation check to ensure it meets defined requirements such as supported file formats (e.g., JPEG or PNG), minimum resolution, and size limits. If the image fails validation, the system interrupts the process and displays an error message explaining the issue, allowing the user to re-upload a valid file.

If the image passes validation, it is shown to the user in a preview screen. Here, the user has the option to remove the image and upload another one if necessary. This preview stage provides a safeguard, ensuring that the correct image is used for prediction.

Once the user confirms the image and submits the prediction request, the system sends the input to two separate machine learning models: one is a fine-tuned pre-trained model, and the other is a custom model trained specifically for the GreenEye project. Both models operate in parallel to analyze the input image and generate their respective predictions. After processing, the system receives output from both models. It then returns a pair of results: predicted plant species along with a confidence score for each model. This dual-result presentation empowers users to compare outcomes from different learning strategies and gain a better understanding of prediction reliability.

The final step of the process involves displaying the results to the end user in a clear, accessible format — including species names and confidence percentages — thereby completing the prediction cycle.


User uploads one image
File Validation Success?
No
Display Error Message
Yes
Display image previews
Remove image
Yes
No
Submit prediction request
Pass images to ML models
Perform species prediction (2 models)
Return predictions + confidence scores from both models
Display result to end user