TED UNIVERSITY

CMPE 491 – Senior Project I
High Level Design Report

Team Members

Tolga AKYOL

Yiğit AYDIN

Supervisor

Dr. Evren Coşkun

Jury Members

Prof. Dr. Tansel Dökeroğlu, Dr. Deniz Cantürk

Document Version: v1.0, Date: 30/12/2025

SmartLeaf

1. Introduction

1.1 Purpose of the System

SmartLeaf is a software-intensive decision-support system designed to assist users in analyzing plant health through leaf image analysis. The primary purpose of the system is to allow users to upload an image of a plant leaf and receive an automated assessment that includes disease classification, a numerical health score, and indicative nutrient deficiency information.

The system aims to bridge the gap between advanced machine learning–based plant disease detection methods and practical access without requiring technical expertise through a web-based interface. SmartLeaf does not aim to replace professional agricultural diagnosis; rather, it provides preliminary insights that can support early detection and awareness of potential plant health issues.

This High-Level Design Report translates the previously defined analysis and requirements models into a concrete system design model. It focuses on architectural decisions, subsystem decomposition, deployment strategies, and design principles that guide the implementation of SmartLeaf.

1.2 Design Goals

The design goals of SmartLeaf reflect the problem definition, system requirements, and constraints identified throughout the project.

• Modularity

The system is decomposed into subsystems with explicitly specified responsibilities and interfaces (frontend, backend API, machine learning model, documentation/reporting) to allow independent development, testing, and future extension.

• Usability and Accessibility

SmartLeaf is designed to be usable by non-expert users. The interaction flow is intentionally designed with a limited number of steps, requiring only image upload and result visualization.

• Scalability and Extensibility

The architecture allows future integration of additional plant species, disease classes, and advanced models without major changes to the overall system.

• Separation of Concerns

Machine learning logic, business logic, and presentation logic are strictly separated to improve maintainability and clarity of the system design.

• Ethical and Responsible AI Usage

The system explicitly communicates uncertainty (confidence scores) and avoids presenting results as definitive diagnoses. Human oversight and responsible interpretation are emphasized.

• Platform Independence

The system is designed to run on standard web and cloud platforms, enabling deployment with minimal configuration effort limited to environment-specific parameters, and supporting portability.

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
AI	Artificial Intelligence
ML	Machine Learning
UI	User Interface
API	Application Programming Interface
CNN	Convolutional Neural Network
PDF	Portable Document Format
Health Score	A numerical value (0–100) indicating overall plant health
Smart Leaf	The proposed plant health analysis system

1.4 Overview

This report presents the high-level system design of SmartLeaf.

Section 2 briefly describes the current software architecture that exists at the time of writing.

Section 3 introduces the proposed software architecture, including subsystem decomposition, deployment strategy, data management, security, and control flow.

Section 4 details the services provided by each subsystem.

Finally, a glossary and references are provided to support clarity and traceability of the terminology and design decisions presented in this report.

The focus of this document is on design decisions and architectural structure, not on detailed implementation or low-level algorithms, which are addressed in later project phases.

2. Current Software Architecture

At the time of writing this report, SmartLeaf does not yet have a fully implemented and operational software architecture. However, a preliminary and partial architecture exists in the form of early prototypes, architectural decisions, and initial implementation artifacts developed during the analysis and specification phases of the project.

The current state of the system can be characterized as an early-stage prototype architecture, consisting of the following elements:

2.1 Frontend Prototype

An early-stage web-based frontend prototype has been developed and deployed using a modern JavaScript framework. This prototype serves as a proof-of-concept and is limited to core user interactions, including:

• An initial landing page structure

• UI components required for image upload

• A constrained interaction flow focusing on image submission and result display

At this stage, the frontend does not yet communicate with a fully functional backend or machine learning pipeline. Its primary purpose is to validate user interaction assumptions and demonstrate feasibility rather than provide complete functionality.

2.2 Backend and API Skeleton

The backend architecture currently exists only at a conceptual and skeleton level. Preliminary design decisions include:

• The use of a RESTful API architecture

• A backend service responsible for handling HTTP requests, image uploads, and responses

• Planned separation between request handling and machine learning inference logic

While a health-check endpoint that verifies service availability and structural placeholders have been identified, the backend does not yet perform real inference or persistent data management.

2.3 Machine Learning Model Status

The machine learning component of SmartLeaf has not yet been fully integrated into the system architecture. However:

• The target dataset (PlantVillage) has been identified and analyzed

• The intended model type (Convolutional Neural Network–based classifier) has been selected

• Preprocessing and training strategies have been conceptually defined in the Analysis Report

At this stage, the ML model exists as a design-time component, not as a deployed or callable service.

2.4 Deployment and Integration Status

No complete end-to-end deployment currently exists. Partial deployment has been performed only for the frontend prototype. The system components are not yet integrated, and no automated data flow between frontend, backend, and ML model is operational.

2.5 Summary of Current Architecture

In summary, SmartLeaf currently consists of:

• A deployed frontend prototype

• A conceptual backend API structure

• A planned but not yet integrated machine learning subsystem

This incomplete architecture serves as the foundation upon which the proposed software architecture, described in the next section, will be built. The transition from the current state to the proposed architecture is a primary goal of the upcoming development iterations.

3. Proposed Software Architecture

3.1 Overview

The proposed software architecture of SmartLeaf follows a modular, service-oriented, and layered design approach. The system is designed to separate concerns between user interaction, application logic, and machine learning inference, in order to support scalability, maintainability, and extensibility.

At a high level, the architecture consists of four main layers:

1. Presentation Layer (Frontend)

2. Application Layer (Backend API)

3. Machine Learning Layer

4. Infrastructure and Deployment Layer

Each layer has explicitly defined responsibilities and communicates with other layers through explicitly specified interfaces. This architectural style is consistent with modern web-based, ML-driven software systems and supports incremental development, which is essential for a two-semester senior design project.

3.2 Subsystem Integration

The SmartLeaf system is decomposed into the following subsystems:

3.2.1 Frontend Subsystem

The frontend subsystem is responsible for all user-facing interactions. Its main responsibilities include:

• Providing a web-based user interface

• Allowing users to upload plant leaf images

• Displaying classification results and confidence information

• Handling client-side validation for file type, file size, and required input fields

The frontend is implemented as a single-page web application using a JavaScript framework. It communicates with the backend exclusively through RESTful API calls, ensuring loose coupling between presentation and business logic.

3.2.2 Backend API Subsystem

The backend subsystem acts as the central coordinator of the system. Its responsibilities include:

• Receiving image upload requests from the frontend

• Validating input data

• Forwarding images to the machine learning subsystem

• Returning classification results to the frontend

• Managing application-level logic and error handling

The backend exposes RESTful endpoints and is designed to remain independent of the frontend implementation details. This separation enables future extension, such as mobile clients or third-party integrations.

3.2.3 Machine Learning Subsystem

The machine learning subsystem is responsible for plant disease classification. Its core responsibilities include:

• Image preprocessing

• Model inference

• Returning predicted disease labels and confidence scores

This subsystem is designed as an isolated component that can be invoked by the backend. Such isolation allows independent experimentation, retraining, or replacement of the ML model without impacting other parts of the system.

3.2.4 Documentation and Reporting Subsystem

In addition to runtime subsystems, SmartLeaf includes a documentation-oriented subsystem responsible for:

• Generating structured outputs for reporting purposes

• Supporting future extensions such as PDF-based reports

• Ensuring traceability between system outputs and analysis results

This subsystem is particularly relevant for academic and demonstrative use cases of the project.

3.3 Hardware/Software Mapping

The SmartLeaf system is deployed on a cloud-based infrastructure using a layered client–server architecture.

Client-Side (User Device):

• A standard user device (desktop or mobile) running a web browser.

• The browser provides access to the SmartLeaf frontend through HTTPS.

Frontend Hosting Platform (Vercel):

• The SmartLeaf frontend application, implemented using React/Next.js, is deployed on Vercel as a managed cloud hosting service.

• Vercel serves static and dynamic frontend content and forwards user requests to the backend via secure HTTPS connections.

Cloud Backend Server:

• The SmartLeaf REST API is deployed on a cloud backend server.

• This component is responsible for request handling, business logic, orchestration of services, and communication with external subsystems.

• The backend acts as the central coordination point of the system.

Machine Learning Inference Service:

• The machine learning inference component is deployed as a separate service node.

• It executes the trained CNN-based model using a Python runtime environment.

• The backend communicates with this service through internal REST or gRPC calls to perform image classification tasks.

Persistent Data Storage:

• Structured application data such as user information and analysis results are stored in a database system (e.g., PostgreSQL or MongoDB) managed by the backend service.

• Large binary data, such as uploaded plant images, are stored in an object storage service (S3-like bucket).

• This separation improves scalability and avoids unnecessary database load.

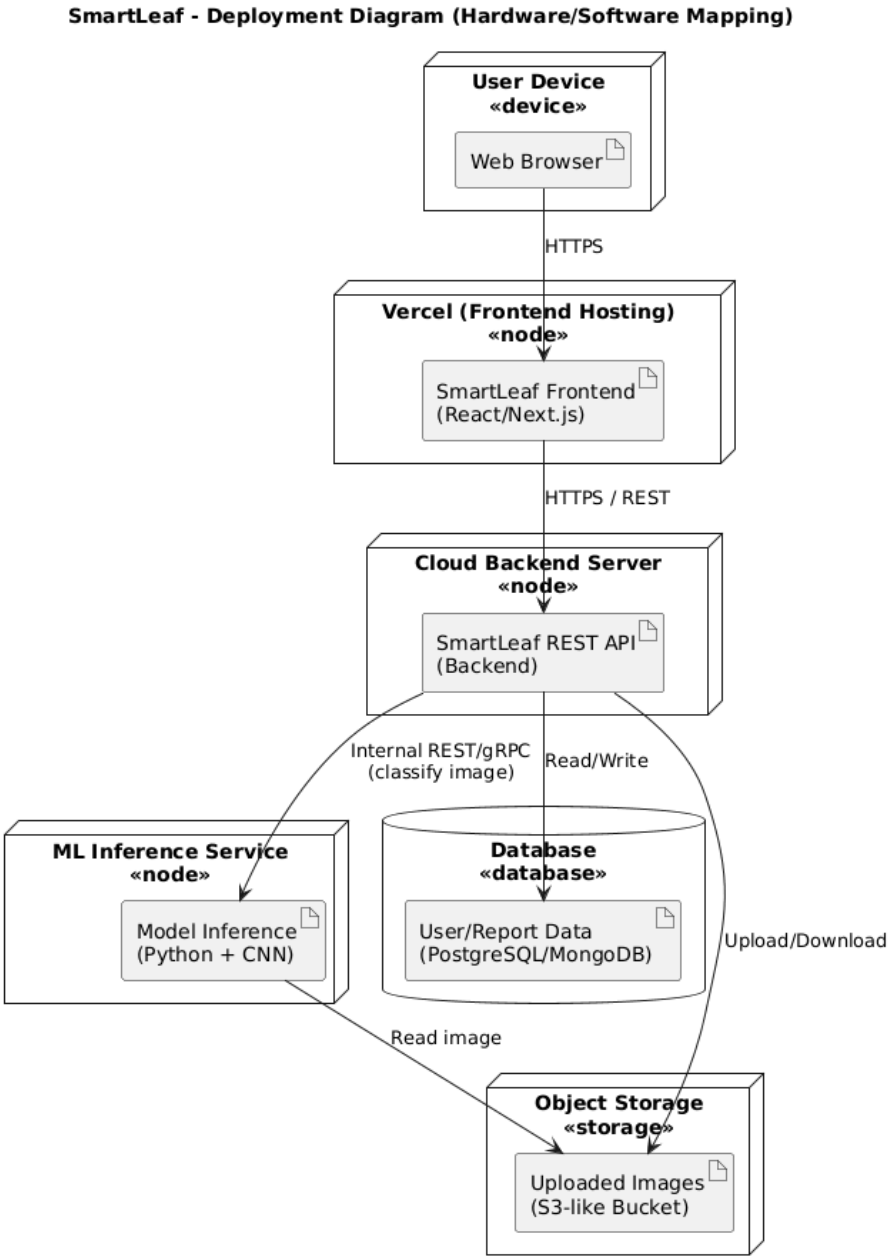
This deployment strategy minimizes client-side hardware requirements, enables centralized updates, and supports scalability by allowing independent deployment and scaling of frontend, backend, and machine learning components.

Design Rationale:

The deployment architecture was designed to separate concerns between presentation, application logic, and machine learning inference. Hosting the frontend on a managed platform such as Vercel simplifies deployment and supports availability, while isolating the backend and ML services

enables independent scaling and maintenance. The use of object storage for image data reduces database load and improves system performance. This design supports scalability, maintainability, and future extensibility of the SmartLeaf system.

Figure X presents the UML deployment diagram illustrating the mapping between software components and the underlying hardware infrastructure of the SmartLeaf system.



(Figure X: Deployment Diagram of the SmartLeaf System)

3.4 Persistent Data Management

In the proposed architecture, persistent data management is limited to essential system metadata. The system does not require long-term storage of uploaded images by default.

Planned data handling includes:

• Temporary storage of uploaded images during inference

• Optional logging of prediction metadata for analysis and debugging

• Configuration and model version metadata storage

This design decision reduces privacy risks and simplifies compliance with data protection considerations.

3.5 Access Control and Security

The initial version of SmartLeaf targets public access without user authentication. However, the architecture is designed to support future access control mechanisms.

Planned security measures include:

• Input validation on all API endpoints

• File type and size validation for image uploads

• Isolation of the ML subsystem from direct client access

• Secure communication channels between system components

These measures help maintain correct system behavior and prevent invalid or unsafe operations, even in the absence of full user authentication.

3.6 Global Software Control

Global control flow in SmartLeaf follows a request-response model:

1. User uploads an image via the frontend

2. The frontend sends the image to the backend API

3. The backend validates and forwards the request

4. The ML subsystem performs inference

5. Results are returned to the backend

6. The backend sends the response to the frontend

This linear and deterministic control flow simplifies debugging, testing, and reasoning about system behavior.

3.7 Boundary Conditions

The proposed architecture explicitly considers boundary conditions, including:

• Invalid or corrupted image files

• Unsupported image formats

• Network failures between subsystems

• ML model inference errors

The backend subsystem is responsible for handling these conditions through explicit error handling mechanisms and returning structured error responses to the frontend.

4. Subsystem Services

This section describes the services provided by each subsystem of the SmartLeaf system. Each service represents an explicitly defined functionality exposed to other subsystems or end users. Explicit definition of these services supports modularity, testability, and maintainability of the system.

4.1 Frontend Subsystem Services

The frontend subsystem provides services related to user interaction and visualization. These services do not contain business logic but act as an interface between the user and the system.

4.1.1 Image Upload Service

• Allows users to upload plant leaf images through a web interface

• Performs basic client-side validation (file type, size)

• Sends validated image data to the backend API

4.1.2 Result Visualization Service

• Displays classification results returned by the backend

• Shows predicted disease label and confidence score

• Handles user feedback and error messages

4.1.3 Navigation and Page Rendering Service

• Provides navigation between system pages (home, upload, result)

• Renders UI components dynamically

4.2 Backend API Subsystem Services

The backend subsystem provides core application services and acts as the central integration point of the system.

4.2.1 Image Processing Request Service

• Receives image upload requests from the frontend

• Performs server-side validation

• Manages request lifecycle and error handling

4.2.2 ML Inference Orchestration Service

• Forwards validated images to the ML subsystem

• Receives prediction results

• Transforms raw ML output into human-readable format responses

4.2.3 Health Check and System Status Service

• Provides a health-check endpoint

• Supports system monitoring and deployment verification

• Allows verification of backend service availability

4.3 Machine Learning Subsystem Services

The machine learning subsystem provides services related to plant disease detection and classification.

4.3.1 Image Preprocessing Service

• Resizes and normalizes images

• Applies preprocessing steps consistent with the input requirements of the trained ML model.

• Handles preprocessing errors

4.3.2 Disease Classification Service

• Performs inference using the trained classification model

• Outputs disease class labels and confidence scores

• Operates as a separate service without direct dependency on frontend or backend UI logic

4.3.3 Model Management Service

• Loads the trained model into memory

• Supports future model updates or retraining

• Maintains version consistency between inference runs

4.4 Documentation and Output Services

This subsystem is responsible for generating, formatting, and delivering the outputs produced by SmartLeaf to the end user. After an image is analyzed, the system presents the prediction results and supporting information in a structured format and allows the user to export the results as a downloadable PDF report.

4.4.1 Result Presentation Service

• Displays the model prediction (e.g., disease/healthy label) and confidence score

• Presents additional information such as brief explanations and recommended actions (if provided by the system)

• Produces outputs in a predefined format suitable for presentation in the web interface

4.4.2 PDF Export Service

• Generates a PDF report containing the uploaded image (optional), prediction output, and related details

• Provides a “Download PDF” capability through the frontend

• Supports reuse of a standard report template to keep exported outputs consistent

5. Glossary

This section defines the key terms, concepts, and abbreviations used throughout the SmartLeaf High-Level Design Report in order to support clarity and consistency.

Artificial Intelligence (AI):

The field of computer science that focuses on creating systems capable of performing tasks that normally require human intelligence, such as image recognition and decision support.

Machine Learning (ML):

A subset of artificial intelligence that enables systems to learn patterns from data and make predictions without being explicitly programmed for each task.

Convolutional Neural Network (CNN):

A class of deep learning models particularly well suited for image processing tasks, such as object recognition and classification.

Frontend:

The client-side component of the system responsible for user interaction and visual presentation, executed in a web browser.

Backend:

The server-side component responsible for application logic, request handling, and communication with the machine learning subsystem.

Application Programming Interface (API):

A defined set of rules and endpoints that allow communication between different software components.

Health Score:

A numerical value between 0 and 100 representing the estimated overall condition of a plant leaf based on visual analysis.

Nutrient Deficiency Indicator:

A suggestive output indicating a potential lack of specific nutrients inferred from visual leaf characteristics. This output is not intended as a definitive diagnosis.

Inference:

The process of applying a trained machine learning model to new input data in order to generate predictions.

Subsystem:

A modular component of the overall system that provides a specific set of services and can be developed and maintained independently.

6. References

1. B. Bruegge and A. H. Dutoit, Object-Oriented Software Engineering: Using UML, Patterns, and Java, 2nd ed., Prentice Hall, 2004.

2. D. P. Hughes et al., “PlantVillage: A Dataset for Plant Disease Detection,” arXiv preprint arXiv:1511.08060, 2015.

3. R. S. Pressman and B. R. Maxim, Software Engineering: A Practitioner’s Approach, 8th ed., McGraw-Hill, 2014.