Analysis Model - PlantPal

Team Name: SproutLab	
Student Number	**Name**
2352280	Hengyu Jin
2352609	Qi Lin
2353409	Baoyi Hu
2352288	Sirui Da

1. Introduction

1.1 Background & Motivation

Indoor plants are increasingly used to improve living spaces and well-being, yet beginners struggle with three pivotal moments: before buying (no assessment of light/time/pets/allergy constraints and plant–home fit), first week after purchase (no actionable onboarding steps), and ongoing care (diagnosing yellowing, drooping, or stalled growth is cognitively heavy, and guidance quality varies). We propose a mobile product, PlantPal, combining AI recognition, a structured care knowledge base, and a lightweight rules/learning engine to cover the full loop from “Is this plant right for me?” to “How do I maintain it?”. Interaction leans on the principle of “the lowest-risk first step” so users can act quickly under uncertainty.

1.2 Project Goals

With measurable product outcomes:

a. deliver stable recognition for common indoor species with top-k + confidence and manual confirmation;

b. build a plant dossier to hold species, photos, events, and a timeline;

c. provide a Week-1 Quick-Start checklist and editable reminders to improve early retention;

d. output fit assessments with alternatives when a plant is unsuitable;

e. in troubleshooting, always provide one lowest-risk, at-home action plus a follow-up reminder. Core KPIs: activation, week-1 retention, reminder completion rate, and user-reported survival/confidence. These metrics drive iterative learning.

1.3 User Segments

Based on the needs breakdown, we target three segments:

- **Before Purchase**
  - **With intent to care**: Primary question is “Does this plant fit me?”. They care about aesthetics, ease of care, pet safety, allergy risk, and if it does not fit—clear **alternatives** with reasons.
  - **No intent to care**: Do not plan to buy now; want quick recognition and a concise **species card** for rare/unknown plants.
- **Bought but Not Started**
  Need a clear **Week-1 How-To** list: placement, whether to repot, whether to water immediately, **minimal toolkit** (medium/pot/watering tool), and **don’t-do** items (e.g., “No frequent watering in the first 48h; avoid direct harsh sun”). Local factors (weather/region/placement) are incorporated into the checklist logic rather than separate modules.
- **Already Caring**
  Focus on forgetfulness (watering/fertilizing/repotting reminders), **auto-adjusting plans**, difficulty judging health (symptom check/illness or not), and **stage tracking** with a timeline.

1.4 Key Usability Targets

Design is constrained by:

1. time to show candidates ≤ 2 s (local cache) / ≤ 5 s (with network);
2. first dossier creation ≤ 60 s (confirm/name/start reminders);
3. 80% of Week-1 tasks in ≤ 3 steps;
4. critical paths reachable in ≤ 3 taps (identify→confirm→create, view checklist, log a watering);
5. plain language for non-experts (short sentences, verb-first, minimal jargon), with bilingual CN/EN and accessibility (contrast, touch targets). Together with performance, privacy, and reliability NFRs, these form MVP acceptance criteria.

2. Architectural Analysis

2.1 High-level Architecture & Sub-Systems Overview

The architectural design of **PlantPal** adheres to the core principles of **"Clear Layering"** and **"Separation of Concerns."** It aims to build a system that is testable, evolvable, and capable of accommodating rapid iterations of AI components. We adopted a **Modified Layered Architecture** centered on the **Ports and Adapters Pattern** (Hexagonal Architecture). This effectively **decouples** volatile "technical implementations" (e.g., AI models, notification services) from stable "business rules" (e.g., suitability assessment, lowest-risk diagnosis principles).

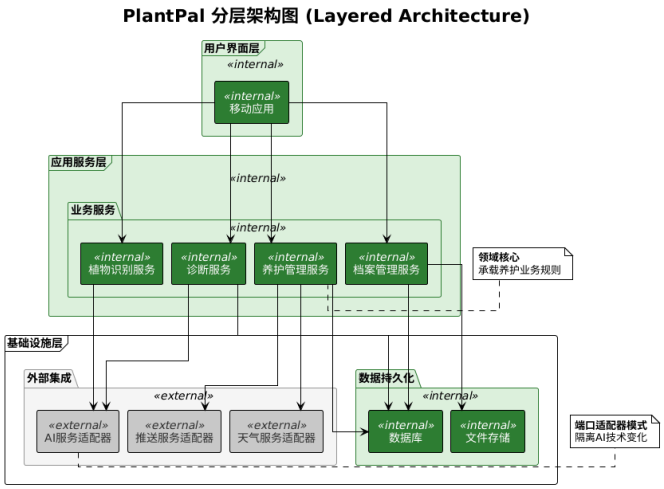
All design decisions are rooted in the system's core positioning of **"Full-Lifecycle Plant Care,"** avoiding the overhead of generic architectures. Furthermore, we prioritize reusing mature technical components (e.g., Flutter, Spring Cloud Gateway) to minimize "reinventing the wheel."

Key Architectural Drivers:

- **User Experience (UX):** Meeting key performance indicators (KPIs), such as recognition response time (≤2s local / ≤5s network).
- **Business Flexibility:** Supporting full-lifecycle management from "Identification" to "Long-term Care," with configurable rules.
- **Technical Adaptability:** AI models can be upgraded or replaced independently without affecting core business logic.
- **Fault Tolerance & Reliability:** The system is capable of **graceful degradation** when the network is unstable or AI services are unavailable.

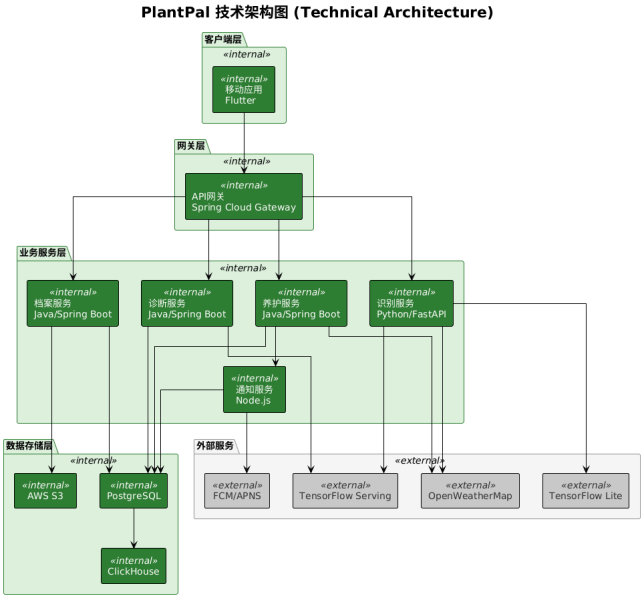
2.2 Logical Architecture & Sub-System Decomposition

The system is logically divided into three main layers. Their collaboration relationships are illustrated in the diagram below, which clearly outlines the system's high-level structure, technical boundaries, and data flow.



This diagram illustrates the high-level architecture of the **PlantPal** system, based on the **Ports and Adapters Pattern** (Hexagonal Architecture). It consists of the following major components:

- **User Interface Layer:** The Mobile Application serves as the sole user entry point, interacting with all core business services.
- **Application Service Layer:** This layer contains **four core business services**, each responsible for a distinct business domain. They process requests from the mobile application, host the system's core business rules, and define their required capabilities through **internal ports**.
- **Infrastructure Layer:** This layer concretely implements the ports defined by the Application Service Layer via adapters, comprising:
  - **Internal Technical Adapters:** Such as database and file storage adapters. Although the technical implementation lies outside (e.g., the actual DBMS), their access interfaces are defined and controlled internally by the system.
  - **External Service Adapters:** These encapsulate the invocation details of third-party services (e.g., AI models, weather data, and push notifications), effectively **decoupling** these volatile external dependencies from the core business logic.
- **Data Flow:** The arrows clearly indicate key interaction relationships within the system.
  - *Example:* The Mobile App calls the **Recognition Service**, which in turn relies on the external **AI Service**.
  - *Example:* The **Care Service** retrieves local environmental data from the **Weather Service** and sends personalized care reminders to the user via the **Push Service**.



2.3 Core Subsystem Responsibilities

2.3.1 Recognition Service (RecognitionAdapter)

- **Responsibilities:** Acts as the **Port Adapter** for AI models, handling plant image recognition requests. It encapsulates the complexity of invoking cloud-based models and implements **fallback strategies** (graceful degradation).
- **Design Rationale:** Isolates volatile AI technical details from stable business logic. By leveraging the **Adapter Pattern**, future changes to model vendors or the introduction of new recognition technologies require **zero changes** to the core business code.
- **Associated Use Cases:** UC1 (Assess Suitability), UC2 (Identify Plant Info), UC3 (Create Plant Dossier).

2.3.2 Care Service (CareRoutineService)

- **Responsibilities:** Manages plant care plans and serves as the **holder of core business rules**. It generates care tasks (e.g., watering reminders) based on plant species, environmental data, and user history, while also handling user-initiated **proactive logging**.
- **Design Rationale:** Centralizes the management of care logic to ensure rule consistency. The design features a **lightweight rule engine** (driven by JSON configuration), allowing non-technical personnel to adjust care strategies without redeploying the system. Explicitly supports proactive user recording.
- **Associated Use Cases:** UC5 (Record Care Log).

2.3.3 Plant Dossier Service (PlantDossierService)

- **Responsibilities:** Acts as the **Aggregate Root**, managing all information regarding a single plant, including basic details, photos, care timelines, and diagnosis records.
- **Design Rationale:** Adopts the **Aggregate Pattern** from Domain-Driven Design (DDD) to ensure **consistency boundaries** for data modifications. This provides a clear model foundation for the timeline view and full history traceability required in UC6 (Manage Plant Dossier).
- **Associated Use Cases:** UC3 (Create Plant Dossier), UC6 (Manage Plant Dossier).

2.3.4 Diagnosis Service (DiagnosisService)

- **Responsibilities:** Implements the business principle of **"Lowest Risk First."** It orchestrates the interaction between AI image diagnosis and contextual Q&A (e.g., asking about "Recent Care Operations"). It provides the safest treatment recommendations based on synthesized judgment and automatically sets follow-up reminders.
- **Design Rationale:** Encapsulates the complex diagnosis workflow into an independent service, emphasizing its core status in the business. The service output strictly adheres to safety principles, reflecting how the architecture supports key business goals.
- **Associated Use Cases:** UC4 (Diagnose Problem).

2.3.5 Key Architectural Decisions & Rationale

Decision Point	Core Rationale	Trade-offs / Alternatives
Ports & Adapters (Hexagonal Architecture)	Decouples AI and external services from core business logic; enhances testability and flexibility.	Alternative: Direct embedding of AI logic. Trade-off: Less development effort initially, but results in high coupling and difficulty in maintenance.
Hybrid AI Inference	Combines Cloud Accuracy with Local Availability/Speed (offline capability).	Alternative: Pure Cloud or Pure Local. Trade-off: Simpler architecture, but sacrifices either performance (latency) or service availability/accuracy.
Microservices Decomposition	Enables independent development, deployment, and scaling of modules.	Alternative: Monolithic Architecture. Trade-off: Simpler to start, but difficult to scale and hinders parallel development in larger teams.
Configurable Rule Engine	Allows rapid business adjustments via configuration (JSON) without engineering code changes.	Alternative: Hard-coded Rules. Trade-off: Better runtime performance, but extremely low flexibility for changing business needs.
Audit Logs & Event Sourcing	Enables fault tracing, data loop closure (for AI training), and historical analysis.	Alternative: Storing Current State Only. Trade-off: Lower storage costs, but impossible to trace full history or debug complex logic sequences.
Embedding Mature Cloud VLM	Focuses resources on core business logic and accelerates Time-to-Market.	Alternative: Training Proprietary VLMs. Trade-off: Offers full control, but involves prohibitively high costs and long development cycles.

2.4 Technical Analysis of the AI Recognition Subsystem

2.4.1 Subsystem Objectives & Constraints

The responsibility of the AI Recognition Subsystem is to convert user-provided images (and optional text information) into a set of structured **candidate plant species**. It provides a stable, consistent interface for subsequent use cases (e.g., Plant Suitability Assessment, Plant Info Query, Dossier Creation, and Diagnosis).

The design of this subsystem is driven by the following objectives and constraints:

- **Accuracy & Safety**
  - Provide high **top-k recognition accuracy** for common indoor plants.
  - Avoid making **"arbitrary determinations"** (forcing a single output) for fine-grained or visually similar plants. Instead, reduce the risk of misidentification through a combination of candidate sets, distinguishing features, and user confirmation.
- **Performance & Availability**
  - Primarily rely on **Cloud VLM (Vision-Language Models)** to achieve relatively stable recognition results.
  - Meet the response time requirements defined in the SRS (e.g., **≤ 5s**) under good network conditions.
  - Support **graceful degradation strategies** (e.g., delayed recognition, falling back to on-device lightweight models, or "draft-only" mode) in weak network or offline scenarios.

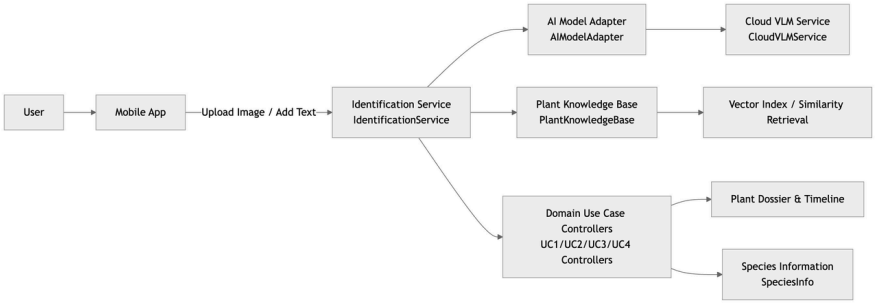
- **Evolvability & Maintainability**
  - Encapsulate recognition capabilities via `IdentificationService` + `AIModelAdapter`, **shielding** upper-layer use cases from specific model types and vendors.
  - Support switching Cloud VLM providers, updating model versions, or expanding the plant knowledge base **non-intrusively**, without affecting the domain model or business use cases.
- **Consistency & Explainability**
  - Uniformly represent all recognition results as: **Candidate List + Confidence Score + Distinguishing Features**.
  - In scenarios involving similar plants, assist users in making the final judgment by providing clear **comparative information** and **key identification points**.

2.4.2 Overall Technical Solution: Cloud VLM + Plant Knowledge Base

The AI Recognition Subsystem adopts a hybrid solution combining **"Cloud VLM + Private Plant Knowledge Base + Similarity Retrieval."** Large Model capabilities are leveraged primarily for two types of tasks:

1. **Embedding Generation:** Encoding images into **semantic vectors (Embeddings)** to perform similarity retrieval within the private plant knowledge base, generating top-k candidate species.
2. **differentiation Description:** Generating **natural language descriptions of key differentiating points** for similar plants when needed, used to guide the user in distinction and confirmation.

The core components and their dependency relationships are illustrated in the diagram below:



Component Descriptions

- **CloudVLMService:** Encapsulates third-party cloud-based **Vision-Language Model (VLM)** APIs, providing `embedImage` capabilities and optional multimodal Q&A functionality.
- **PlantKnowledgeBase:** Maintains plant entries (names, aliases, images, attribute tags, etc.) and their vector representations, enabling **similarity retrieval** via vector indexing.
- **AIModelAdapter: Shields** the upper layers from specific interface discrepancies of different Cloud VLMs, implementing unified Embedding calls and optional description generation.
- **IdentificationService: Orchestrates** the entire recognition workflow, including candidate generation, look-alike disambiguation strategies, confidence calculation, and standardized output for upper-layer use cases.

2.4.3 Recognition Pipeline

The recognition process can be abstracted top-down as a pipeline of **"Candidate Generation -Look -->alike Disambiguation --> User Confirmation."** The main steps are as follows:

1. **Input Acquisition & Preprocessing**

- The user captures or uploads a plant image via the App, with optional supplementary text (e.g., "Indoor, semi-shade, potted").
- The mobile client performs **basic preprocessing** (cropping, scaling, normalization, quality checks) and constructs the recognition request.

2. **Candidate Generation (Cloud VLM + Knowledge Base Retrieval)**

- `IdentificationService` invokes `AIModelAdapter.embedImage(image)`, generating an image vector via the Cloud VLM.
- This vector is used to search the **Vector Index** within the `PlantKnowledgeBase` to retrieve **top-k** similar plant entries.
- An initial candidate list is obtained, where each candidate includes a similarity score and basic attribute information (leaf shape, succulent status, typical environment, etc.).

3. **Look-alike Disambiguation**

- If the similarity gap between the top-1 and top-2 (or more) candidates is small, the system identifies this as a **"Look-alike Set" (Similar Plants Scenario)**.
- The service compares the structured attributes of these candidates to **automatically extract key differentiating features**, such as:
  - Are the leaf margins significantly serrated?
  - Is the flower color common?
  - Plant morphology (trailing vs. upright).
- Optionally, the Cloud VLM is invoked to generate **natural language descriptions** of these differentiating features to enhance understandability.

4. **User Interaction & Confirmation**

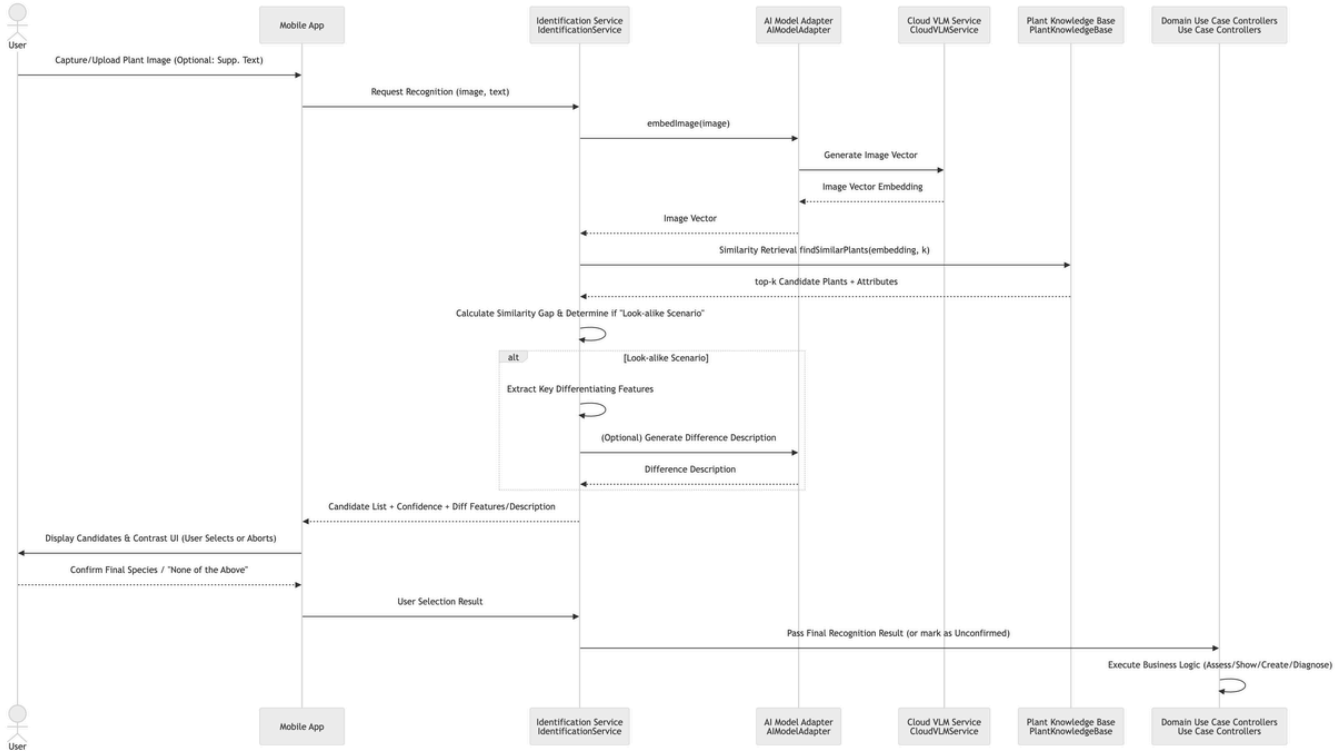
- The App displays the candidate list and key differentiators:
  - **High-Confidence Scenario:** Defaults to highlighting the top candidate, while providing a "One-click Confirm / Change Selection" interaction.
  - **Look-alike Scenario:** Displays multiple candidates in a **Contrast View**, guiding the user to choose based on differentiating features, or select "None of the above."
- The user's selection is returned to the recognition service to update the composite confidence score, record "re-selection" behavior, and support future optimization.

5. **Domain Integration & Post-Processing**

- Once the user confirms the species, the result is passed as input to the relevant **Use Case Controllers**:
  - **Suitability Assessment (UC1):** Calculates suitability combining user preferences and environmental data.
  - **Plant Info Display (UC2):** Loads detailed profiles from the Species Information database.
  - **Create Plant Dossier (UC3):** Creates a dossier and a basic care plan based on the confirmed species.
  - **Problem Diagnosis (UC4):** Uses the species as a prerequisite condition in the diagnosis workflow, combined with symptom recognition/diagnostic rules.
- If the user selects "None of the above" or the composite confidence is too low, a temporary record for an **"Unconfirmed Plant"** is created to avoid writing erroneous species data directly into the dossier.

Sequence Diagram

The process described above is illustrated in the following sequence diagram:



2.4.4 Key Architectural Decision Records (ADR)

To support the workflow above, the AI Recognition Subsystem has made the following key architectural decisions:

- **ADR-AI-01: Retrieval-based Recognition using Cloud VLM + Private Plant KB**
  - Recognition does not rely directly on the open-ended category output of the VLM. Instead, images are encoded into vectors for similarity retrieval within a **Private Plant Knowledge Base**.
  - This controls the species space, allows unified maintenance of attributes and subsequent care/diagnosis rules for each species, and permits the interchangeability of the underlying VLM.
- **ADR-AI-02: Decision Workflow for Look-alikes: "Candidate Set + Differentiating Features + User Confirmation"**
  - In scenarios with small similarity gaps, the system avoids providing a single definitive conclusion. Instead, it presents multiple candidates with **key identification points**, leaving the final choice to the user.
  - Interaction design is used to compensate for VLM uncertainty in fine-grained recognition, mitigating long-term risks associated with misidentification (e.g., incorrect care, toxicity warnings).
- **ADR-AI-03: Encapsulation of Recognition Capabilities as Infrastructure Services**
  - All VLM/model-related calls are completed via `AIModelAdapter` and `IdentificationService`. Upper-layer use cases rely solely on **standardized recognition result objects**.
  - Domain models and use case controllers do not directly depend on specific ML frameworks or cloud services; they depend only on clear interfaces and data structures, supporting future evolution.
- **ADR-AI-04: Extension Points for Degradation Strategies in Weak Network/Offline Scenarios**
  - Although the current version prioritizes Cloud VLM, extension points for **on-device lightweight models** or **delayed recognition** are reserved within the Identification Service.
  - When the cloud is unavailable, the system can choose to: locally cache the image and input for automatic recognition later, or fall back to coarse-grained results from an on-device model.

2.4.5 Risks & Future Optimization Directions

- **Uncertainty in VLM Fine-grained Plant Recognition**
  - Due to limited training data quality and fine-grained labels, large models may exhibit high confusion rates between visually approximate species.
  - **Mitigation:** Currently alleviated via the "Candidate Set + User Confirmation" strategy. Future optimization involves building **annotated evaluation sets** and improving knowledge base images and attribute labels.

- **Risks Regarding Knowledge Base Coverage & Data Quality**
  - Insufficient images or inaccurate attributes for certain species in the private KB will affect retrieval performance and the quality of differentiating feature generation.
  - **Mitigation:** Establish a data maintenance process to support the progressive supplementation of representative images and attributes.
- **Cloud Dependency & Latency Fluctuations**
  - Cloud VLMs introduce risks regarding response time and availability fluctuations.
  - **Mitigation:** Currently managed via timeout controls and degradation strategies. Future optimizations may include data-driven **caching strategies** (e.g., caching vectors and results for high-frequency species).

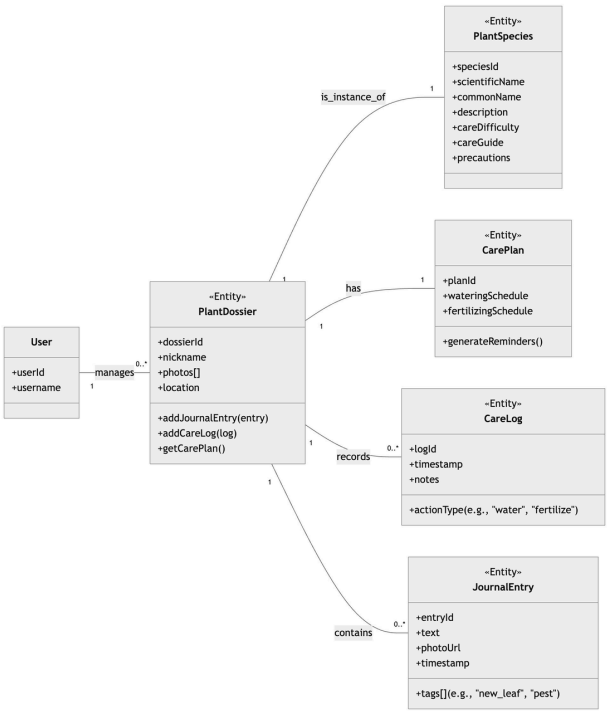
The AI Recognition Subsystem employs a combined solution of **"Cloud VLM + Private Plant Knowledge Base + Look-alike Disambiguation + User Confirmation."** This approach keeps the uncertainty of large models within an acceptable range while ensuring overall availability, providing ample evolutionary space for future model replacements and accuracy improvements.

3. Domain Model & Interaction Analysis

3.1 Domain Class Diagrams

Design Rationale

- **Core Entity (PlantDossier):** The `PlantDossier` serves as the core entity of our system. Far from being a mere data container, it aggregates the `CarePlan`, `CareLog`, and the newly introduced `JournalEntry`.
- **Introduction of JournalEntry (Addressing Feedback):** To address the "proactivity" requirement in UC5, we introduced the `JournalEntry` entity. Distinct from the `CareLog` —which records standardized care actions (e.g., watering, fertilizing), the `JournalEntry` captures the user's unstructured observations (e.g., photos, notes).
- **PlantSpecies vs. PlantDossier:** We explicitly distinguish between general "species information" (`PlantSpecies`, acting as a template) and the specific "individual plant" (`PlantDossier`, acting as an instance).



3.2 Key Classes: Responsibilities & Attributes

PlantDossier: Plant Dossier (`<<Entity>>`)

Responsibilities:

- **Aggregates all plant information:** Acts as the **Aggregate Root**, managing all data associated with this specific plant instance.
- **Manages CarePlan:** Holds and updates the corresponding personalized `CarePlan`.
- **Records CareLog:** Associates all completed, structured care actions (e.g., watering, fertilizing).
- **Records JournalEntry:** Associates all unstructured observations (photos/text) proactively recorded by the user.

Key Attributes:

- `+dossierId`: Unique identifier in the system.
- `+nickname`: Personalized name assigned by the user.
- `+photos`: List of plant photos uploaded by the user.
- `+location`: User-specified placement location (e.g., "Living Room Windowsill").

PlantSpecies: Plant Species (`<<Entity>>`)

Responsibilities:

- **Stores static template information:** Provides general data shared by all plants of the same species, serving as a template.

- **Provides species profile:** Includes basic introduction such as care characteristics and morphology.
- **Defines care guides & difficulty:** Contains structured care requirements (light, water, soil, temperature) and provides an overall care difficulty rating.
- **Lists key precautions:** Includes common issues, special care tips, etc.

Key Attributes:

- `+speciesId` : Unique identifier in the species database.
- `+scientificName` : Official scientific name of the species.
- `+commonName` : Commonly used name of the species.
- `+description` : A descriptive text/bio of the species.
- `+careDifficulty` : (e.g., "Easy", "Medium", "Hard").
- `+careGuide` : Structured care information.
- `+precautions` : Textual information regarding care precautions.

CarePlan: Care Plan (`<<Entity>>`)

Responsibilities:

- **Defines specific care schedule:** Stores the personalized care cycle for this plant instance (e.g., "Water every 7 days").
- **Generates reminders based on rules:** Calculates the next reminder time based on the schedule, weather, seasons, and other rules.

Key Attributes:

- `+planId` : Unique identifier for the care plan.
- `+wateringSchedule` : Rules or expressions defining the watering cycle.
- `+fertilizingSchedule` : Rules or expressions defining the fertilizing cycle.

CareLog: Care Log (`<<Entity>>`)

Responsibilities:

- **Records structured care *actions*:** Records **completed** actions (e.g., "Watering") initiated by system reminders or the user.
- **Marks status and timestamp:** Tracks care history and serves as a basis for adjusting the `CarePlan` .

Key Attributes:

- `+logId` : Unique identifier for the care log.
- `+actionType` : Type of action (e.g., "water", "fertilize", "prune").
- `+timestamp` : Specific time when the action was completed.
- `+notes` : Additional notes provided by the user for this action.

JournalEntry: Plant Journal (`<<Entity>>`)

Responsibilities:

- **(Addressing Feedback) Records unstructured user *observations*:** Allows proactive user records, such as "Sprouting new leaves" or "Found small bugs".
- **Allows text & photo notes:** Stores observation photos taken by the user and related textual notes.

Key Attributes:

- `+entryId` : Unique identifier for the journal entry.
- `+text` : User's textual notes.
- `+photoUrl` : URL of the photo uploaded with the note.
- `+timestamp` : Time when the user created the entry.

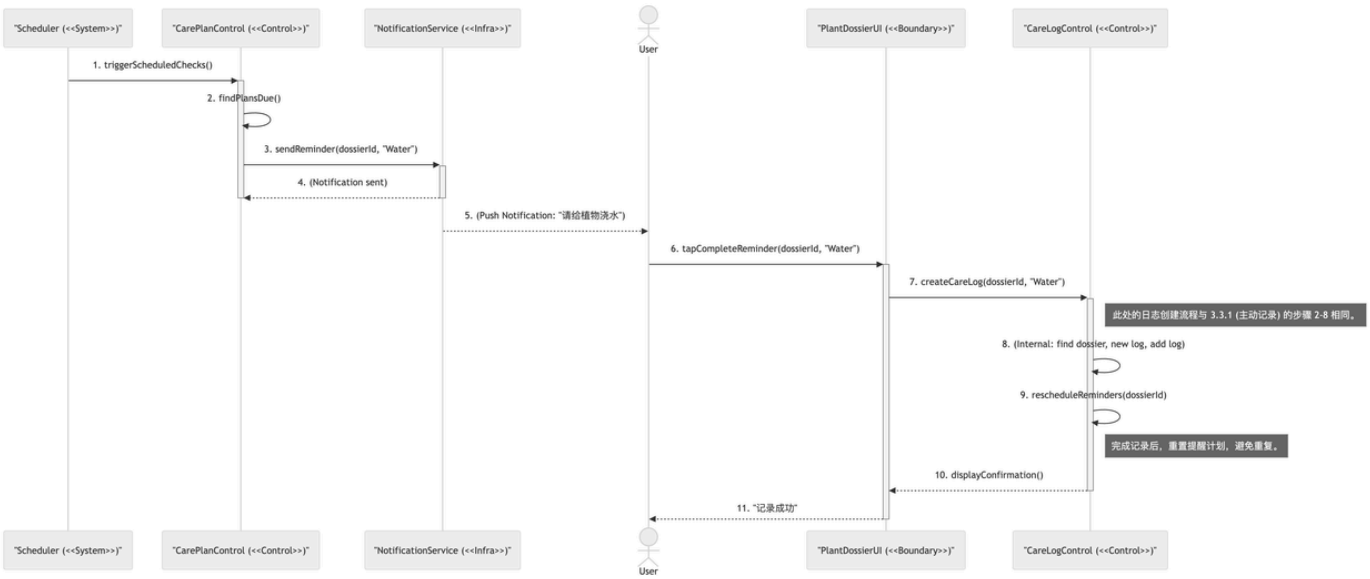
3.3 Core Use Case Analysis

3.3.1 UC5 Care Recording

3.3.1.1 UC5-1: Interaction Diagram for System Reminders & Automatic Recording

Design Rationale:

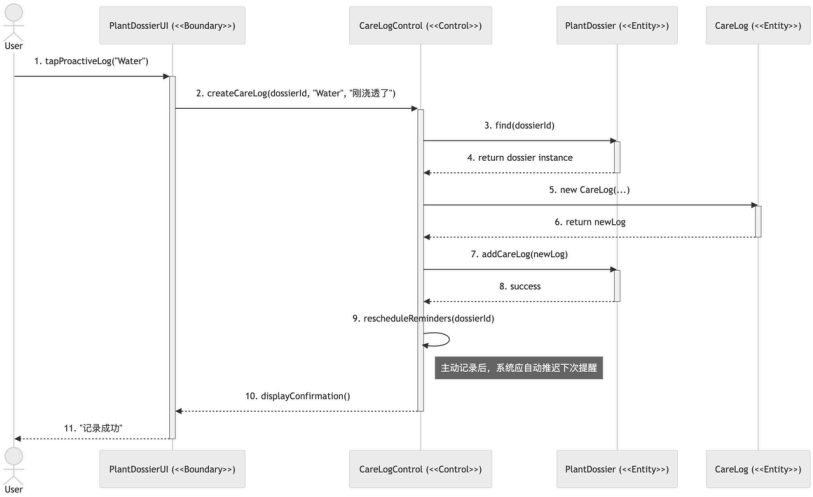
- **Preserving Original Functionality:** This diagram illustrates the original workflow initiated by the **system**. It complements the user-initiated workflow in UC5-2, and together they constitute the complete UC5 scenario.
- **Logic Reuse:** Whether the action stems from a system reminder or a user-initiated record, the process ultimately invokes the same `CareLogControl` . This guarantees consistency in business logic (e.g., both scenarios trigger `rescheduleReminders`), avoids code duplication, and demonstrates sound software design principles.
- **Separation of Concerns & Intelligent Strategy:**
  - `CarePlanControl` (or `CarePlanService`) employs an **adaptive strategy** to determine *when* to send reminders. It does not rely solely on fixed intervals. Instead, it utilizes LLMs to fine-tune reminder cycles based on feedback from the user's "proactive records" (JournalEntries) and "completion records" (CareLogs), combined with contextual information such as location, weather, and plant species.
  - `CareLogControl` is solely responsible for recording the **completed** actions.



3.3.1.2 UC5-2: Interaction Diagram for User "Proactive Recording"

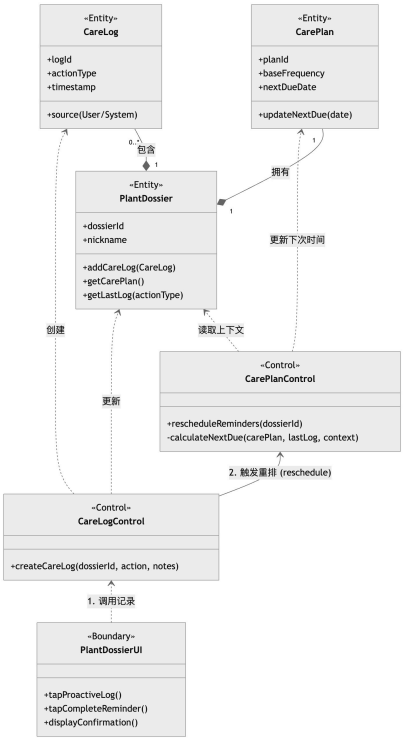
Design Rationale:

- **Proactivity:** This workflow is initiated by the **User**, rather than being prompted by a system reminder.
- **Intelligent Adjustment:** Upon successfully recording the log, the `CareLogControl` invokes the `CarePlanControl` to execute `rescheduleReminders`. This resolves the issue where the system might awkwardly "remind the user to water immediately after they have just finished watering," thereby demonstrating the system's intelligence.
- **Clear Responsibilities (Separation of Concerns):**
  - The **UI** is solely responsible for dispatching the call.
  - The **CareLogControl** handles coordination (creating the Log, associating it with the Dossier).
  - The **CarePlanControl** is responsible for updating the plan/schedule.
  - The **Entities** manage the core data operations.



3.3.1.3 UC5 Implementation

1. **Analysis Class Diagram**



2. **Detailed Component Responsibilities**

CareLogControl (Recording Controller - "Write")

- **Role:** The direct entry point for user care actions.
- **Responsibilities:** Handles the process whether the user clicks "Complete Reminder" or "Proactive Record." It converts user actions into **immutable** `CareLog` entities and persists them.
- **Key Action:** Upon successfully saving a log, it must actively trigger the `CarePlanControl` to reschedule. This is the trigger point for achieving the "feedback loop."

CarePlanControl (Planning Controller - "Schedule")

- **Role:** The system's "Brain," responsible for time calculations.
- **Responsibilities:** It acts as a calculation engine rather than a direct data store. It calculates the `NextDueDate` based on the current `CarePlan` configuration (baseline cycle) and the latest `CareLog` history (actual behavior).
- **Intelligence:** All adaptive algorithms (e.g., weather-based adjustments, user habit fine-tuning) are encapsulated here.

PlantDossier (Plant Dossier - "Context")

- **Role: Aggregate Root**.
- **Responsibilities:** Ensures data consistency. All logs (`CareLog`) and plans (`CarePlan`) belong to the dossier. Through the dossier, controllers can access the complete context required for calculations (e.g., calculating the next watering time requires knowledge of the plant's species, location, and the timestamp of the last watering).

3. **Core Logic: Smart Rescheduling**

When `CareLogControl` successfully saves a new log, it immediately invokes `CarePlanControl.rescheduleReminders(dossierId)` . The execution logic is as follows:

1. **Context Fetching:** Retrieve the plant's `CarePlan` (containing the baseline frequency, e.g., "Once every 7 days") and the latest `CareLog` (retrieving the timestamp, i.e., "Just now") from the `PlantDossier` .
2. **Rule Application:**
   - **Base Logic:** `NextDueDate = LastLog.timestamp + CarePlan.baseFrequency` . This ensures that reminders are always postponed based on **"actual behavior,"** thereby resolving the "reminded immediately after watering" issue.
   - **Intelligent Fine-tuning (Extension Point):** Environmental factors can be injected at this step. For example, if the `WeatherService` indicates high temperatures for the coming week, the `baseFrequency` can be temporarily shortened by a coefficient of 0.8.
3. **State Update:** Update the `nextDueDate` attribute of the `CarePlan` entity and invoke the infrastructure layer (e.g., `NotificationService`) to update or cancel the underlying scheduled tasks in the system.

3.3.2 UC3: Create Plant Dossier

3.3.2.1 UC3: Interaction Diagram for Creating Plant Dossier

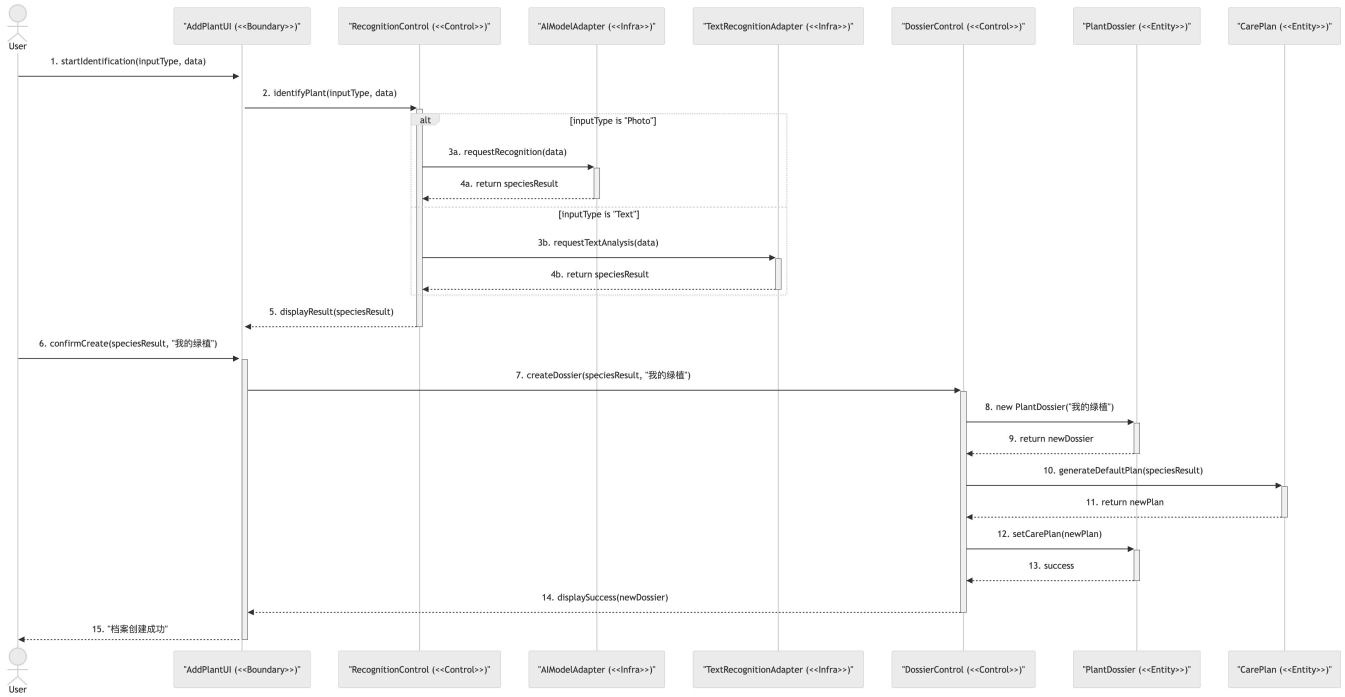
Process Description: The core workflow where the user identifies a plant by capturing images of the plant itself, or scanning plant tags/manuals/descriptions, and subsequently creates a dossier.

Design Rationale:

- **Multimodal Input (Steps 2-4):** To enhance flexibility, the `RecognitionControl` acts as a **Strategy** coordinator. It invokes different infrastructure adapters (`AIModelAdapter` or `TextRecognitionAdapter`) based on the `inputType` (image or text). This enables the easy addition of further recognition sources (e.g., QR codes) without modifying the `DossierControl` .
- **Clear Phase Division:** The workflow is distinctly divided into two phases:
  - **Steps 1-5:** The "Identification Phase," orchestrated by `RecognitionControl` .

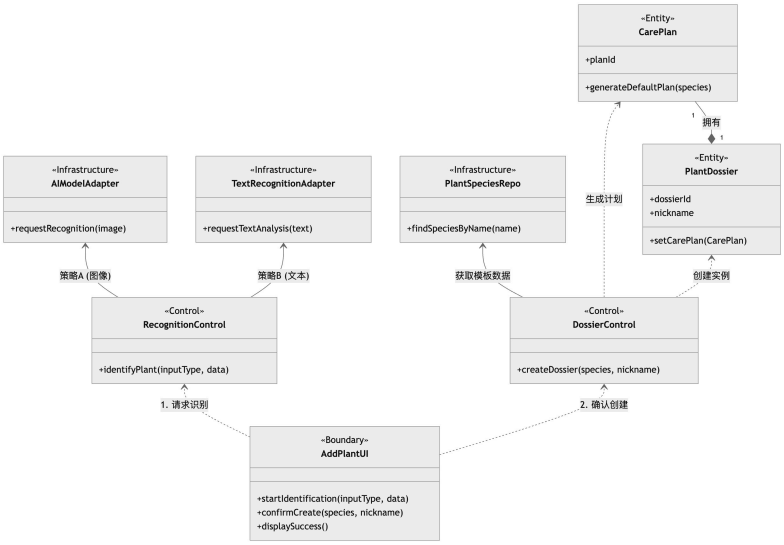
- **Steps 6-15:** The "Creation Phase," orchestrated by `DossierControl`.

- **Separation of Entity & Control:** The `DossierControl` only concerns itself with the `speciesResult` (identification result) and remains **agnostic** to *how* the identification was performed. It coordinates the creation and association of the `PlantDossier` and `CarePlan` entities, strictly adhering to the principle of **"High Cohesion, Low Coupling."**



3.3.2.2 UC3 Implementation

1. **Analysis Class Diagram**



2. Detailed Component Responsibilities

RecognitionControl (Recognition Controller - "Identify")

- **Role:** Coordinator of recognition strategies.
- **Responsibilities:** It does not contain specific recognition algorithms. Instead, it **dynamically routes** the request to the corresponding infrastructure adapter based on the user's input type (Photo or Text). This **decouples** the business logic from specific recognition technologies.

DossierControl (Dossier Controller - "Create")

- **Role:** The core of the dossier creation business logic.
- **Responsibilities:** Once the species is confirmed, this controller is responsible for **instantiating** the `PlantDossier`. It also coordinates the generation of the `CarePlan`, ensuring that the newly created dossier immediately possesses a standard care plan based on the species template.

AIModelAdapter & TextRecognitionAdapter (Adapters - "Infra")

- **Role:** Technical implementers.
- **Responsibilities:** The `AIModelAdapter` is responsible for invoking vision model APIs; the `TextRecognitionAdapter` handles OCR and NLP services. They expose a **unified interface** externally and return a standardized `SpeciesResult`.

3. Core Logic: Strategy Dispatch & Dossier Construction

   a. **Identification Strategy (Phase)**

      - The `AddPlantUI` passes user input to the `RecognitionControl`.

- The `RecognitionControl` checks the input type: if it is an image, it invokes the `AIModelAdapter`; if it is text, it invokes the `TextRecognitionAdapter`.
- The system returns a list of candidate species for user confirmation.

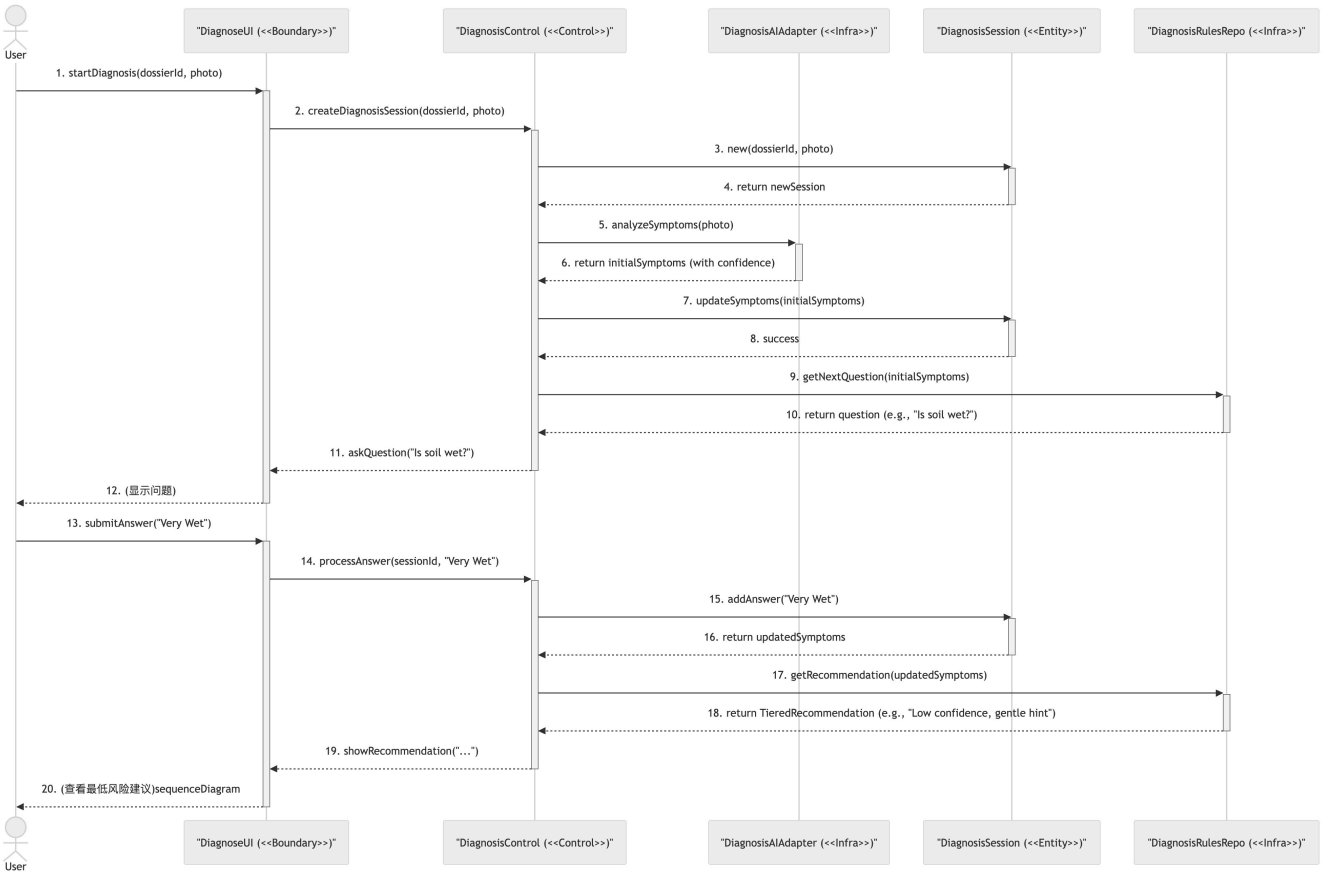
b. **Construction (Phase)**

- After the user confirms the species, `DossierControl` receives the `speciesId`.
- `DossierControl` retrieves full species details via the `PlantSpeciesRepo`.
- `DossierControl` creates a `PlantDossier` instance, invokes `CarePlan.generateDefaultPlan()` to generate the initial plan, and finally associates and saves both entities.

3.3.3 UC4: Problem Diagnosis

3.3.3.1 UC4: Interaction Diagram for Problem Diagnosis

Process Description: A workflow where the system automatically retrieves updated photos from the user's `PlantDossier` to detect potential issues, performs further confirmation via a multi-step Q&A session, and provides "lowest-risk recommendations."



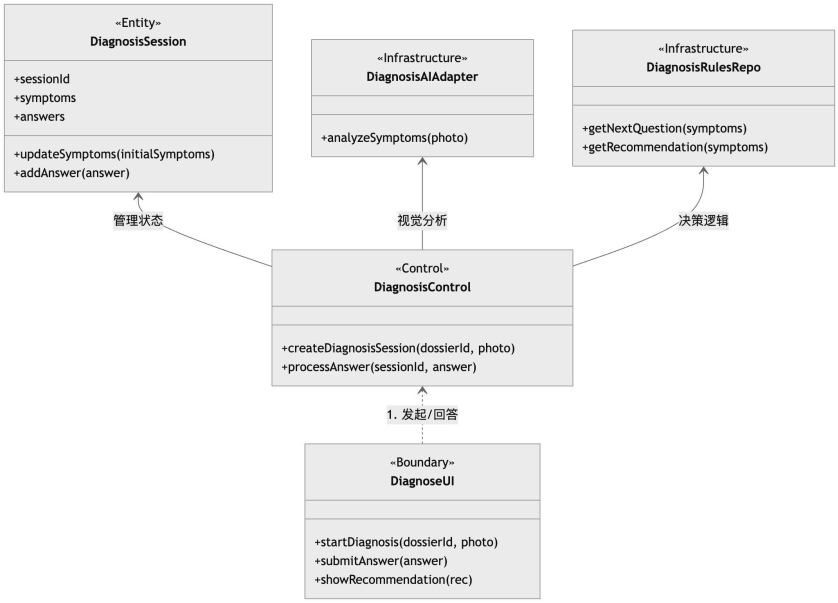
Design Rationale

- **State Management:** Diagnosis is a multi-step, **stateful** process. The `DiagnosisControl` creates a `DiagnosisSession` (Entity) to **persist** the state of this session (Steps 3, 7, 15).
- **Externalized Logic & Tiered Alerting:** `DiagnosisControl` does not contain specific diagnostic knowledge. It relies on the `DiagnosisAIAdapter` (Infrastructure) for image analysis (returning results with confidence scores) and the `DiagnosisRulesRepo` (Infrastructure) to retrieve the next question and the final recommendation.
  - **Tiered Recommendation Strategy:** The recommendation logic is **tiered**. For high-confidence model results, the Control triggers a **strong alert**; for low-confidence results (e.g., potential natural aging or water stains), the Control triggers a **gentle prompt** (e.g., a "System Note"). This strategy avoids **false positives** and effectively manages user expectations.

3.3.3.2 Use Case Implementation: UC4 (Problem Diagnosis)

This section demonstrates how the system diagnoses plant health issues through a **"Human-Computer Collaboration"** approach.

1. **Initiation:** The process begins when the user uploads or selects a plant photo. The system utilizes AI to perform **preliminary symptom detection** (e.g., yellowing leaves, spots).
2. **Verification:** Subsequently, to rule out misjudgments, the system initiates a **multi-turn Q&A** session based on the AI's confidence level (e.g., asking about soil moisture).
3. **Conclusion:** Finally, the system provides a **"lowest-risk" treatment recommendation**, preventing the user from accidentally harming the plant due to incorrect actions based on wrong assumptions.



1. **Detailed Component Responsibilities**

   DiagnosisControl (Diagnosis Controller - "Coordinate")

   - **Role:** Coordinator of the diagnostic workflow.
   - **Responsibilities:** It orchestrates AI recognition, session state management, and invocations of the rule engine. It **does not encapsulate** specific phytopathology (plant disease) knowledge; instead, it is responsible for wiring together various components to drive the process forward.

   DiagnosisSession (Diagnosis Session - "State")

   - **Role:** State Container.
   - **Responsibilities: Persists** all context information within a single diagnostic run, encompassing uploaded photos, preliminary symptoms identified by AI, and all user responses. This enables the diagnostic process to span multiple interaction steps (**Stateful**).

   DiagnosisAIAdapter & DiagnosisRulesRepo (Infrastructure - "Infra")

   - **Role:** Components of the Expert System.
   - **Responsibilities:**
     - `DiagnosisAIAdapter` is responsible for **"Seeing"**: analyzing visual symptoms.
     - `DiagnosisRulesRepo` is responsible for **"Thinking"**: determining the next question and the final recommendation based on the symptoms.
2. **Core Logic: Tiered Alerting & Multi-step Confirmation**

   Initialization & AI Analysis

   - The user initiates the diagnosis. `DiagnosisControl` creates a session and invokes `DiagnosisAIAdapter` to retrieve a list of symptoms accompanied by **confidence scores**.

   Dynamic Q&A

   - `DiagnosisControl` passes the symptoms to `DiagnosisRulesRepo`.
   - The Rule Engine evaluates the **uncertainty** of the symptoms (e.g., "Leaves are yellowing: is it due to water shortage or root rot?") and generates the next **validating question** (e.g., "Is the soil dry or wet?").

   Decision Output

   - After collecting user responses, the system outputs a recommendation based on the synthesized information.
   - The strategy is **"Tiered Alerting"**:
     - **High Confidence:** Reports specific diseases/pests.
     - **Low Confidence:** Reports environmental issues or merely suggests continued observation.

4. Updated Requirements

4.1 Drivers for Change

The requirement updates focus on three core issues:

- **Responding to Instructor Feedback:** Addressing the passive nature of UC5 (reliance on system reminders) by supplementing it with a "proactive user operation" path.
- **Completing the Business Loop:** Mitigating the risk of misidentifying look-alike species by introducing a **Human-in-the-loop** confirmation mechanism, thereby implementing the "Lowest Risk" principle.
- **Quantifying Technical Metrics:** Transforming qualitative descriptions like "stable recognition" and "fast response" into measurable standards to support acceptance testing.

4.2 Core Functional Requirement Optimizations

4.2.1 UC5 Care Recording: New Proactive Recording Capability

- **Original Issue:** UC5 relied on a passive "Remind-Execute-Mark" workflow. Manual recording was merely an extension point without a clear path or data association.
- **Optimization Solution:** Added **UC5-2 Proactive Recording Workflow**. Users can click "Record Care" on the dossier page, independently select action types, fill in notes, and upload photos without waiting for a reminder. Proactive `CareLogs` automatically sync to the timeline and trigger **Smart Rescheduling** of the `CarePlan` (e.g., postponing the next reminder automatically after watering). Distinct boundaries are set between `CareLog` (structured action records) and `JournalEntry` (unstructured observation diaries, e.g., new leaves, pests).
- **Associated Design:** Matches the `JournalEntry` entity, filling the gap for "recording plant status without performing care actions."

4.2.2 UC2/UC3 Recognition Workflow: Look-alike Confirmation Mechanism

- **Original Issue:** The system only prompted for re-shots when confidence was low. It lacked a fault-tolerance mechanism for similar species (e.g., *Epipremnum aureum* vs. *Philodendron*), leading to risks of incorrect care.
- **Optimization Solution:** When the similarity gap between top-1 and top-2 is ≤ 15%, the system automatically generates a **"Look-alike Contrast View."** It displays key differences (e.g., serrated leaf edges, vein patterns) and AI descriptions to guide user confirmation. Confirmed results carry the "SpeciesID + User Flag" to downstream processes, ensuring data accuracy.
- **Associated Design:** Supports the "Cloud VLM + Knowledge Base Retrieval" solution, using interaction design to compensate for model uncertainty.

4.2.3 UC6 Dossier Management: Function Consolidation & Boundary Clarification

- **Original Issue:** UC6 overlapped with UC3 and UC5, serving merely as a viewing entry point with no independent management value.
- **Optimization Solution:** Refactored into a **"Dossier Aggregation Management Center,"** containing three modules: Basic Info Editing, Timeline Viewing, and Care Plan Adjustment. Duplicate viewing functions were removed. It is now explicitly defined as the operation entry point for the `PlantDossier` **Aggregate Root**, rather than a standalone use case. Added multi-selection tagging and batch export to adapt to multi-plant scenarios.
- **Associated Design:** Aligns with the `PlantDossier` Aggregate Root positioning, reinforcing data consistency.

4.2.4 New Requirement: Multimodal Recognition (Text/OCR)

- **Scenario Background:** Responding to needs like scanning flower shop tags or uploading manuals, supplementing non-image recognition methods.
- **Core Function:** Supports OCR tag scanning and text input to trigger recognition. The logic is unified with image input, outputting the standard "Candidate List + Confidence" format, compatible with UC3/UC1 workflows.
- **Associated Design:** Matches the `TextRecognitionAdapter`, demonstrating the flexibility of the **Ports and Adapters** pattern.

4.2.5 New Requirement: AI Interactive Q&A Recognition

- **Scenario Background:** Targeting blurry images or plants with unclear features where visual recognition alone is prone to bias. AI proactively interacts to supplement key information, improving precision.
- **Core Function:** When recognition confidence is between 50%-80% or status is ambiguous, the AI automatically initiates **Interactive Q&A** (e.g., "Do the leaves have fuzz?", "Are there recent curling yellow leaves?"). Supports user-triggered AI Q&A to describe features or abnormalities. Q&A results fuse with image data to output a more precise SpeciesID and Status Assessment (e.g., "Suspected Epipremnum aureum, Status: Normal, no pests"), syncing to the `PlantDossier` and care advice modules.
- **Associated Design:** Integrates with the AI Q&A Engine and associates with the `PlantDossier` entity, providing reliable data support for subsequent care plan generation.

4.3 Non-Functional Requirements Quantifiable Metrics

Requirement Category	Original Description	Quantifiable Metrics	Explanation
Recognition Performance	Stable recognition of common indoor species	1. Top-1 accuracy for 200+ common species ≥ 90%; 2. Accuracy for look-alikes after user confirmation ≥ 95%; 3. Response: Local ≤ 2s, Cloud ≤ 5s	Covers mainstream plants like Pothos, Succulents, etc.
System Reliability	Offline saving and synchronization	1. Offline support for recording care/drafts; sync success rate ≥ 99%; 2. Retry within 10s after reconnection when degrading	Adapts to weak network scenarios like balconies or elevators.
User Experience (UX)	Key paths ≤ 3 clicks	1. Identify-Confirm-Create ≤ 3 steps; 2. Proactive Record ≤ 2 steps; 3. Initiate Diagnosis-Upload ≤ 2 steps	Optimization based on User Journey Maps.

4.4 Rationalization for Changes

All changes remain within the scope of "Full-Lifecycle Plant Care." Their validity is demonstrated by:

- **Problem-Oriented:** Proactive recording and look-alike confirmation specifically address feedback and pain points; they are not baseless additions.
- **Technically Feasible:** Requirements like multimodal input and AI Q&A are deeply matched with the architecture (Ports & Adapters, AI Encapsulation).
- **Clear Acceptance:** Quantifiable metrics provide distinct standards for development and testing, aligning with **MVP delivery requirements**.

5. Updated UI Snapshots with Descriptions

1. **Registration / Onboarding Page**

- **User Profiling:** Presents a questionnaire of 3-5 items (e.g., prior care experience, environmental conditions, preferred plant types) to construct a preliminary user profile.

2. **AI Recognition Page (Home / Explore)**

- **Global Search Module:**
  - **Scope:** Plant names, disease symptoms, care issues, etc.

- **Prioritization:** Matches in the user's "My Plants" list $\rightarrow$ Plant Database (Guides/Articles) $\rightarrow$ External Search Engine.
- **Features:** Search history and keyword recommendations.

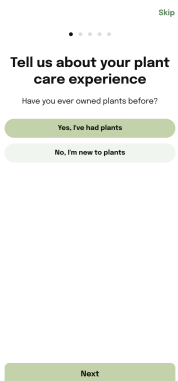
- **Recognition Module:**
  - **Methods:** Camera capture, Photo Library upload, Text Scan (OCR from tags or books).
- **Explore Mode:**
  - **Target Audience:** Users browsing without a specific goal.
  - **Output:** Generates a **Plant Card** after recognition, containing: Name, HD Gallery, Bio, Detailed Care Guide, and Difficulty Rating (e.g., Beginner-friendly).
  - **Post-Recognition Actions:**
    i. **"Add to My Plants":** One-click addition to the user's list, creating a care dossier.
    ii. **"Update My Plant":** Direct jump to the status update page for an existing plant.
- **Workflows:**
  - **Add to My Plants:** Allows users to name the plant, set watering frequency, and record the purchase date.
  - **Update My Plant:** Enables quick recording of current health status, new growth (e.g., new leaves), or issues via photos and text.
  - **Create Profile (Flow):** Targeted at users identifying a plant specifically to create a dossier. Recognition -> Direct entry to "Add to My Plants" flow.
  - **Update Status (Flow):** Targeted at users updating an existing dossier. Recognition - Direct ->entry to "Update My Plant" flow.

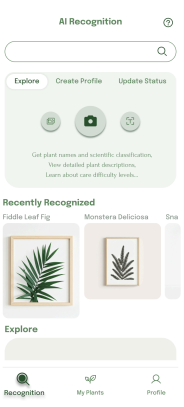
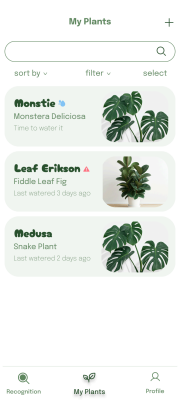
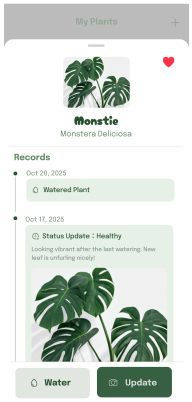
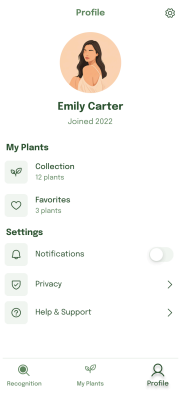
3. **My Plants Page**

- **Global Search Module:** Similar to the AI Page, but strictly scoped to the user's personal plant library and logs.
- **Filter & Sort Module:** Tools for sorting, filtering, and multi-selecting plants based on specific criteria.
- **Record / List Module:**
  - **Display:** Grid or list of plant cards.
  - **Plant Card Detail View:**
    - **Basic Info Section:** Core plant details.
    - **Care Reminder Section:**
      - Settings: Manual setup or **AI Smart Reminders** (auto-scheduling based on needs).
      - Action: Manual "Mark as Completed".
      - **AI Alerts:** System analyzes uploaded status photos to detect potential diseases or water shortages and pushes warning alerts.
    - **Status Timeline:**
      - **Format:** Diary-style, reverse chronological order.
      - **Content:** Date, User Photos, User Notes, System Logs (auto-recorded actions).

3. **Personal Center Page**

- **Profile & Data:** Account data management and display of personal information.
- **App Settings:** General settings including Push Notification management and Language selection.
- **Feedback & Support:** Help Center, User Feedback, and "About Us."

page	UI snapshots
Registration / Onboarding Page	 UI 1.1
AI Recognition Page (Home / Explore)	

	 UI 2.1	 UI 2.2	 UI 2.3	 UI 2.4
My Plants Page	 UI 3.1	 UI 3.2		
Personal Center Page	 UI 4.1			

6. Open Issues & Next-stage Task

1. **Disambiguation Guidance for Non-Expert Users**

- The Scenario: When the system identifies two visually similar plants (e.g., Epipremnum aureum vs. Philodendron hederaceum), the AI might generate a technical distinction, such as "The difference lies in the shedding mechanism of the leaf sheath."
- **Next-stage Task:**
  - **The Problem:** Average users struggle to understand botanical jargon.
  - **The Solution:** We need to design a mechanism where the **VLM (Visual Language Model)** translates professional features into "layman's terms." Furthermore, it should **automatically crop and highlight** the corresponding regions in the image to provide visual proof.
  - **Design Challenge:** This is a User Experience (UX) challenge: How to empower novice users to perform expert-level identification with AI assistance.

2. **Ambiguity of User Delay (Behavioral Data)**

- The Scenario:A core value proposition of the system is "Adaptive Scheduling." Suppose the system reminds the user to water on Monday, but the user clicks "Complete" on Thursday.
- **The Ambiguity:** This delay creates significant noise in the data:
  - **Possibility A (System Error):** The soil was still wet on Monday, so the user intentionally waited until Thursday. ->The model needs correction.
  - **Possibility B (User Constraint):** The soil was dry on Monday, but the user was too busy, causing the plant to suffer. ->The model logic should remain unchanged.
- **Next-stage Task:**
  - **Goal:** Design a **"Low-friction Intent Capture"** mechanism.
  - **The Constraint:** We cannot bombard the user with a popup asking "Why were you late?" every time, as this degrades the user experience.

- **Design Challenge:** How to use **micro-interactions** (e.g., placing a quick "Still Wet?" tag next to the "Complete" button) or analyze behavioral metadata (e.g., App launch frequency) to **implicitly infer** the user's true intent. This is crucial for **cleansing the feedback data** used for Reinforcement Learning (RL), ensuring the model learns from the correct "ground truth."

7. AI Tools Acknowledgment

In preparing this report, our team used Generative AI (ChatGPT, Gemini) primarily to handle repetitive tasks and check for logical gaps. We want to be transparent about how these tools were used versus what was our own work.

How we used AI:

1. **Generating Diagram Code (Mermaid.js):** We adopted **Mermaid.js** for its efficiency and modern "docs-as-code" approach. Our design process started with team discussions and hand-drawn sketches on draft paper to define the core logic and interactions. We then used AI to assist in converting these rough sketches into clean, aesthetic Mermaid diagrams. This workflow allowed us to bridge the gap between our brainstorming and high-quality technical documentation, focusing on architectural design rather than syntax formatting.
2. **Brainstorming & Scenario Exploration:** During the initial analysis phase, we used AI to brainstorm potential user interaction methods. For example, when discussing **UC3 (Dossier Creation)**, we asked AI to list various ways a user might identify a plant in a store. This process inspired us to include "Text Recognition" (scanning labels/manuals) alongside image recognition. We then evaluated the feasibility and designed the specific Strategy Pattern implementation ourselves.
3. **Sanity Checks & Edge Cases:** After designing the core logic for **UC5 (Smart Reminder)**, we asked AI to point out potential logical holes (e.g., "What happens if the user waters the plant before the reminder?"). This helped us identify edge cases, but the solution—the "Smart Rescheduling" logic—was designed entirely by the team.
4. **Proofreading:** We drafted the content based on our analysis. We then used AI to fix grammatical errors and ensure the technical language was concise and professional.

AI was used strictly as a productivity tool for formatting, coding syntax, and proofreading. All architectural patterns, design reasoning, and key decisions presented in this report are the original work of our team.

8. Annotated References

- **Hessayon, D.G. (2011). *The House Plant Expert*.** A classic practical guide covering common indoor species, routines, and troubleshooting—informing beginner-friendly checklists and species profiles (watering, light, toxicity).
- **Ferentinos, K.P. (2018). Deep learning models for plant disease detection and diagnosis. *Computers and Electronics in Agriculture*, 145, 311–318.** Surveys CNN-based disease recognition and informs design/expectations on mobile photos.
- **Mohanty, S.P., Hughes, D.P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, 7, 1419.** Demonstrates high accuracy on PlantVillage and highlights dataset bias/generalization risks.

9. Contributions of Team Members

Owner	Percentage
Hengyu Jin	25%
Qi Lin	25%
Baoyi Hu	25%
Sirui Da	25%