Ist Durstig

Plant Care Tracker

Ist Durstig is a plant care tracking application designed to help users manage their plants efficiently with custom watering schedules and collaborative features.

Made with GAMMA

🌱 Key Features

Plant Management

Add, edit, and organize plants with photos, tags, and notes.

Scheduling

Custom watering frequencies (Frequent/Medium/Rare) with automatic reminders.

Care Tracking

Log watering, fertilizing, and transplanting events.

Shared Lists

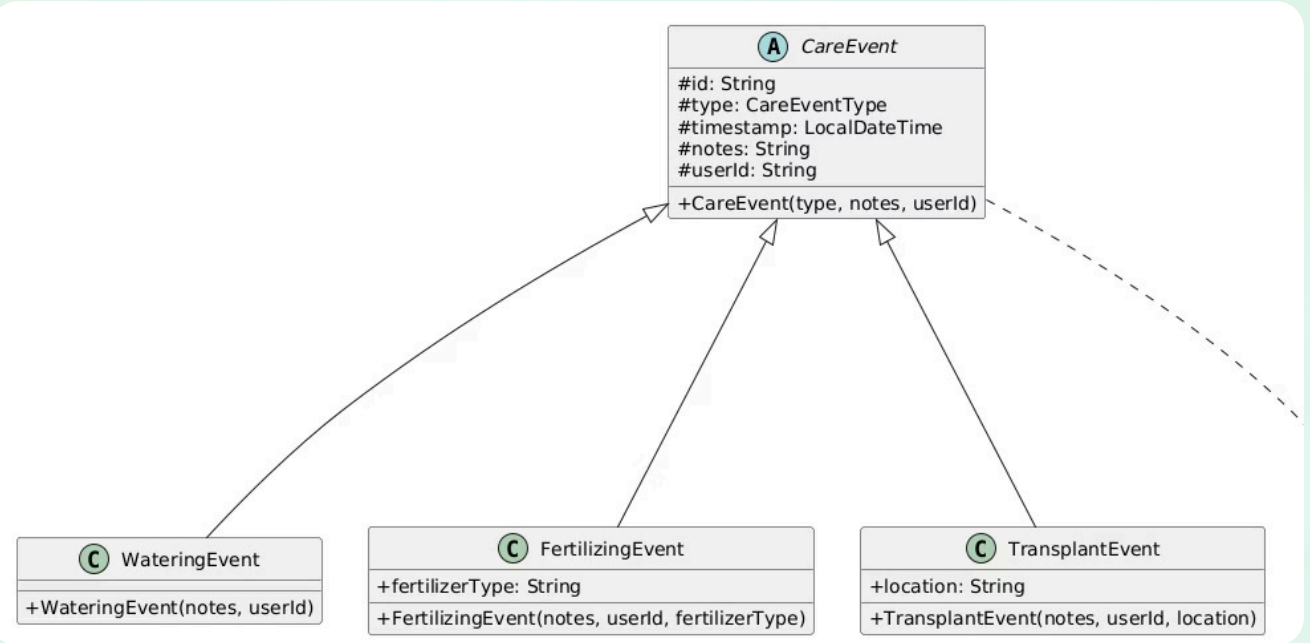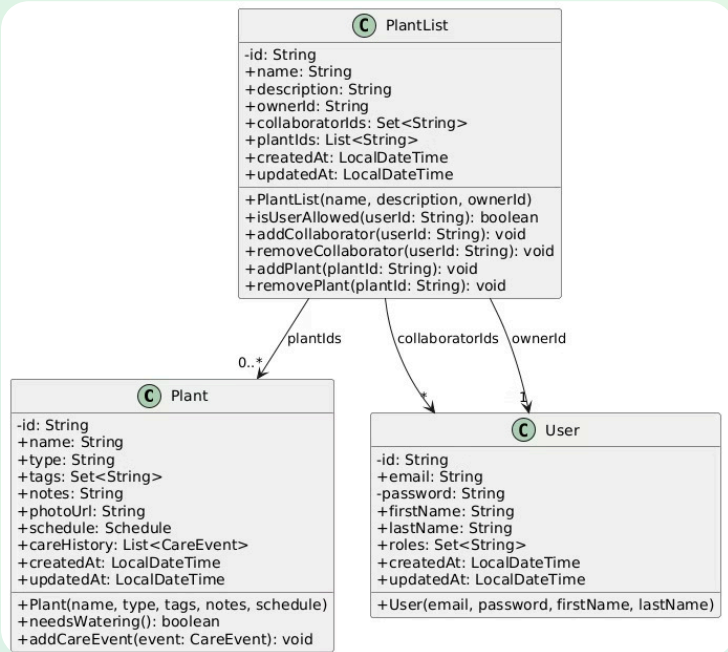Collaborate with roommates or family members on plant care.

Responsive Design

Beautiful interface that works on all devices.

Daily Dashboard

See plants needing attention with beautiful statistics.

🧩 UML Class Diagram

PlantList

-id: String
+name: String
+description: String
+ownerId: String
+collaboratorIds: Set<String>
+plantIds: List<String>
+createdAt: LocalDateTime
+updatedAt: LocalDateTime

+PlantList(name, description, ownerId)
+isUserAllowed(userId: String): boolean
+addCollaborator(userId: String): void
+removeCollaborator(userId: String): void
+addPlant(plantId: String): void
+removePlant(plantId: String): void

plantIds   collaboratorIds   ownerId

0..*

Plant

-id: String
+name: String
+type: String
+tags: Set<String>
+notes: String
+photoUrl: String
+schedule: Schedule
+careHistory: List<CareEvent>
+createdAt: LocalDateTime
+updatedAt: LocalDateTime

+Plant(name, type, tags, notes, schedule)
+needsWatering(): boolean
+addCareEvent(event: CareEvent): void

User

-id: String
+email: String
-password: String
+firstName: String
+lastName: String
+roles: Set<String>
+createdAt: LocalDateTime
+updatedAt: LocalDateTime

+User(email, password, firstName, lastName)

A CareEvent

#id: String
#type: CareEventType
#timestamp: LocalDateTime
#notes: String
#userId: String

+CareEvent(type, notes, userId)

WateringEvent

+WateringEvent(notes, userId)

FertilizingEvent

+fertilizerType: String

+FertilizingEvent(notes, userId, fertilizerType)

TransplantEvent

+location: String

+TransplantEvent(notes, userId, location)

🏗️ Architecture & Tech Stack

Backend

- Framework: Spring Boot 3.2.0 (Gradle)
- Database: MongoDB with Spring Data
- Security: Spring Security + JWT (In progress)
- Architecture: Clean, Repository-Service-Controller

Frontend

- Framework: React 18 + TypeScript
- Styling: Tailwind CSS (responsive)
- Routing: React Router
- API: Axios

📋 Prerequisites

- Java Development Kit (JDK) 17+
- Node.js 18+ and npm
- MongoDB 6.0+ (Community Edition)

▶️ **Demo**

&&

🧬 **Code Exploration**

Take away

I was able to:

- Designe a funcional web-app
- Best practec of OO Programing
- MongoDB integration
- Use of API

I have to improve:

- Implement secure measurs
  Personalize scheduling dates
- Finalize deployment Rendler
- Expand testing coverage

Next steps

- Basic care analytics
- Expand testing coverage
- Fix bugs
- Improve performances