PlantPal Web Application

System Requirements Documentation

Prepared by: Tiana Maea

Table of Contents

1. Customer Problem Statement

Plant owners often struggle to remember care schedules for their houseplants and lack a centralized tool to manage care tasks, especially for different plant species with unique needs. PlantPal addresses this gap by providing a web-based solution for tracking care routines, saving notes, and receiving reminders tailored to each plant in a user-friendly interface.

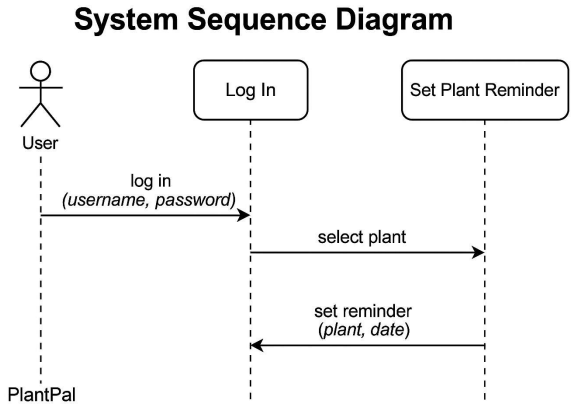
2. System Requirements

- Platform: Web-based, PHP & MySQL
- Environment: Localhost (XAMPP)
- Browser support: Chrome, Firefox
- Backend: MariaDB/MySQL for structured data storage

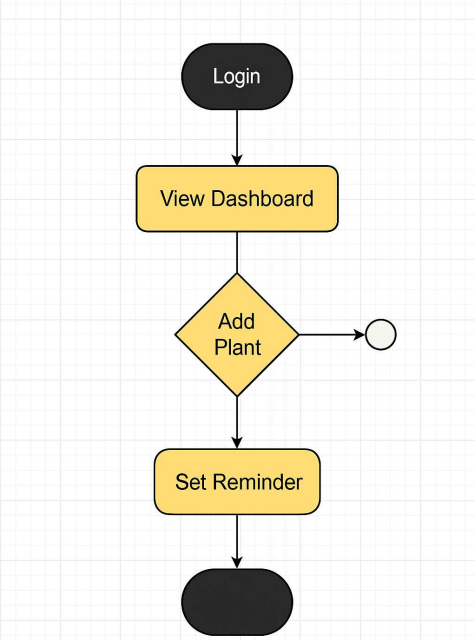
3. Functional Requirements Specification

- Users can register, login, and logout
- Users can browse 50 houseplants with detailed care information
- Users can add plants to a personal library
- Users can write and save care notes
- Users can schedule and view reminders
- Calendar view shows upcoming tasks

4. System Sequence Diagram



5. Activity Diagram



6. User Interface Specification

- Clean dashboard with user stats

- "My Care" section with notes, calendar, and reminders

- Responsive plant library with images and care guides

- Secure login/register forms

7. Traceability Matrix

Problem Statement:

- Many plant owners often struggle with keeping track of their plants' specific care needs, especially when they have several plants in their home. This can include everything from watering schedules, sunlight requirements, to soil preferences. It is challenging to manage multiple plants with different care routines and needs, especially when someone has a busy schedule. To have a simple, yet interactive, system that helps with organizing plants, receive reminders for their care, and access reliable information on best care practices for each plant type, would benefit a household plant owner. A digital platform that acts like a personal plant library would ensure their plants can thrive and reduce the stress of plant care management.

Glossary of Terms:

- Plant Library: A digital collection of plants owned by the user, including plant types, care routines, and other details.
- Watering Alert: A notification sent to the user to remind them of when to water their plant(s).
- Care Database: A built-in repository containing best care practices for different types of plant types.
- Soil Preferences: Specific types of soil required for different plants to thrive in.
- Sunlight Requirements: The amount of sunlight each plant needs daily for healthy growth.

System Requirements:

Functional Requirements

No.	Priority Weight	Description
REQ-1	High	The system should allow users to add, edit, and delete plant entries within their library.
REQ-2	High	The system should send watering alerts based on the plant's watering schedule.
REQ-3	High	Users can input plant details, including name, type, soil preferences, and sunlight needs.
REQ-4	High	System should provide care suggestions to improve or correct plant care data.
REQ-5	Medium	Users should be able to submit suggestions to improve or correct plant care data.
REQ-6	Medium	Users should be able to search and filter their plant library by type, care needs, or name.
REQ-7	Low	The system should allow users to upload images of their plants.

Nonfunctional Requirements

No.	Priority Weight	Description
NFR-1	High	The system should have an intuitive interface for easy navigation.
NFR-2	High	The system should reliably send timely alerts without delays.
NFR-3	Medium	The system should securely store plant data and user contributions.
NFR-4	Medium	The system should perform efficiently even with large

		amounts of data.

User Interface Requirements

No.	Priority Weight	Description
UI-1	High	Home screen displaying the plant library with options to add or remove plants.
UI-2	High	Plant detail page showing care requirements, next watering date, and user suggestions.
UI-3	Medium	Contribution page where users can suggest changes or additions to care data.
UI-4	Medium	Search and filter feature for finding plants by type, care needs, or name.

Plan of Work:

- Week 1-2: Set up development environment and project repository on GitHub. (completed)
- Week 3-4: Design database schema for plant entries, care data, and user contributions. (in progress)
- Week 5-6: Implement basic CRUD operations for the plant library and contribution system. (planned)
- Week 7-8: Develop watering alert system and user notification functionality (planned).
- Week 9-10: Integrate care database with user contributions and search/filter options (planned).
- Week 11-12: Design and refine user interface components, including contribution page (planned).
- Week 13-14: Conduct testing and debugging (planned).
- Week 15: Finalize project documentation and submission (planned).

8. System Architecture and Design

Architectural Styles

The Plant Care Database system follows a client-server architecture style. The front-end interface and back-end logic are written in PHP, allowing for better

integration with the MariaDB database and smoother overall flow. The client interacts with the server using HTTP requests. PHP scripts handle both the user interface rendering and business logic processing. Persistent data is managed via a MariaDB relational database.

Identifying Subsystems

The system consists of several subsystems represented as packages in a UML diagram:

- User Interface: Manages user input/output and displays plant-related data.

- Plant Management: Handles the logic for adding, updating, and retrieving plant details.

- Reminder Management: Schedules and retrieves care reminders.

- Care Logging: Logs actions performed on plants.

- Species Information: Provides static data on plant species.

- Data Access Layer: Manages all SQL operations between the application and MariaDB.

- Authentication: Manages users and security (login, registration).

Mapping Subsystems to Hardware

The system is a web application where the client runs on the users browser and the server runs on a web server (localhost or hosted). The database (MariaDB) and the PHP-based application run on the server. Thus, User Interface and all backend subsystems (Plant Management, Reminder Management, Care Logging, Authentication, and Data Access Layer) are implemented in PHP and hosted on the server.

Persistent Data Storage

Yes, the system uses persistent storage. The tables: plants, reminders, users, care_log, and species_infostore information that persists between sessions. The chosen storage strategy is a relational database (MariaDB), suitable for structured data and efficient querying.

Network Protocol

The system uses HTTP for communication between the client (browser) and the PHP server. PHP scripts handle server-side processing and interact with the MariaDB database using standard PHP database connectors (e.g., mysqli or PDO). SQL queries are executed by the PHP backend to retrieve and manipulate data.

Global Control Flow

Execution Orders: The system is event-driven, reacting to user interactions such as button clicks to perform database operations or page navigation.

Time Dependency: The system operates on an event-response model without real-time constraints. Watering or fertilizing reminders are based on date comparisons, and no strict timing is enforced.

Concurrency: Yes, the system can use multiple threads for scheduled background tasks such as sending daily reminder checks or logging actions concurrently. Synchronization may be needed for simultaneous database updates.

Hardware Requirements

- Color Display: Minimum resolution 640x480 pixels

- Computer: Desktop or laptop

- Hard Disk: Minimum 1GB free space

- Network: Basic broadband connection for hosted version

- Server (if hosted): Supports PHP/Java and MariaDB

9. Algorithms and Data Structures

- PHP arrays and loops manage front-end display logic

- Relational structure in SQL with foreign keys to enforce data integrity

10. UI Design and Test Plan

- Manual testing completed for each form and function

- Unit tests created for user registration, note saving, and reminders

- Integration tests verified complete user flows

11. Project Plan

Week 1–2: Initial planning and setup of development environment (Completed)

Week 3–4: Designed the database schema and created SQL scripts for users, plants, reminders, and notes (Completed)

Week 5–6: Developed and tested user registration, login, and plant browsing features (Completed)

Week 7–9: Integrated user library, care note functionality, and care reminders with backend logic (Completed)

Week 10–11: Designed the dashboard layout, calendar view, and refined interface with CSS and JavaScript (Completed)

Week 12: Created functional and integration test scripts, confirmed database and UI connections (Completed)

Week 13–14: Finalized all documentation, brochure design, and added system diagrams (Completed)

Week 15: Prepared presentation slides, zipped project folder, and submitted final project (Completed)

12. References

PHP Manual (https://www.php.net/manual/en/)

W3Schools SQL Tutorial (https://www.w3schools.com/sql/)

MariaDB Docs (https://mariadb.com/kb/en/)