Happy Plants GmbH

Happy Plants
Software Architecture Document

Version <1.1>

Revision History

Date	Version	Description	Author
<03/12/22>	<1.0>	<1-4 + beschreibender Text>	<Emili>
<06/12/22>	<1.1>	<Bilder eingefügt>	<Emili>

Table of Contents

Software Architecture Document

1. Introduction

1.1 This document provides a quick architectural overview of the system.

1.2 Purpose

This document is using several different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.3 Scope

This document describes the architecture of the HappyPlants Project.

1.4 Definitions, Acronyms, and Abbreviations

Abbreviation	Explanation
MVC	Model-Viel-Controller
UC	Use Case
n/a	Not applicable

1.5 References

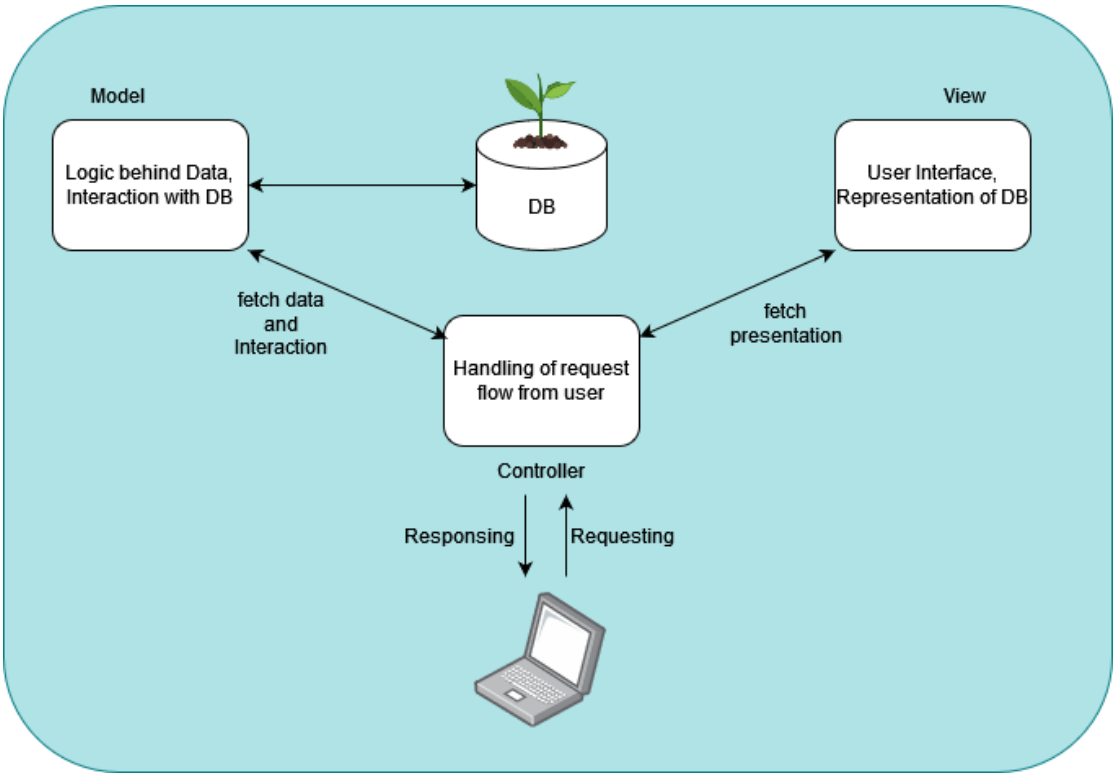
n/a

1.6 Overview

The architectural details will be described in the following sections. This includes the class diagrams which give an overview about the whole project.

2. Architectural Representation

This project we are trying to implement according to the MVC Pattern for the front end (Website) and for the back end (Spring). The model (data model, domain specific classes), the view (user interface) and the controller (controls the Application) are separated. The MVC Pattern can be seen in the next pictures:



3. Architectural Goals and Constraints

We decided to use Spring MVC as our main framework. It allows us to unite backend, frontend development and the database integration in one code base.

3.1.1 MVC

As mentioned in chapter two frontend and backend are using the MVC pattern. This enables a clean software architecture with separate model view and controller.

3.1.2 Front end

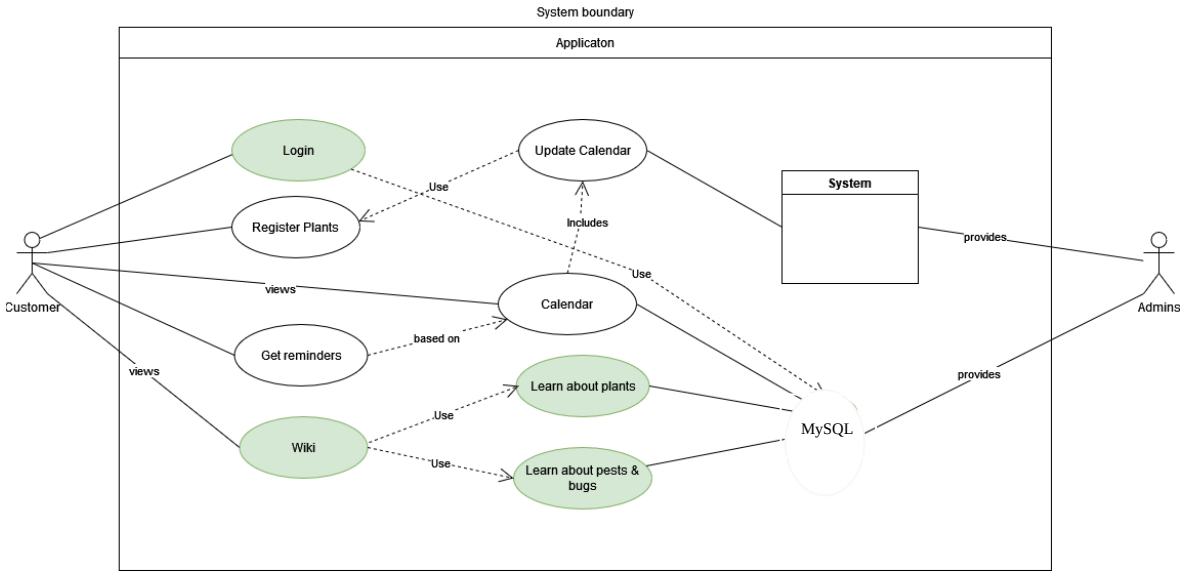
For the Front End we use HTML, CSS and JavaScript.

3.1.3 Back end

The back end is also written in Java. As MVC tool we use Spring Boot. For the account system we will use Spring security. As a database we use MySQL.

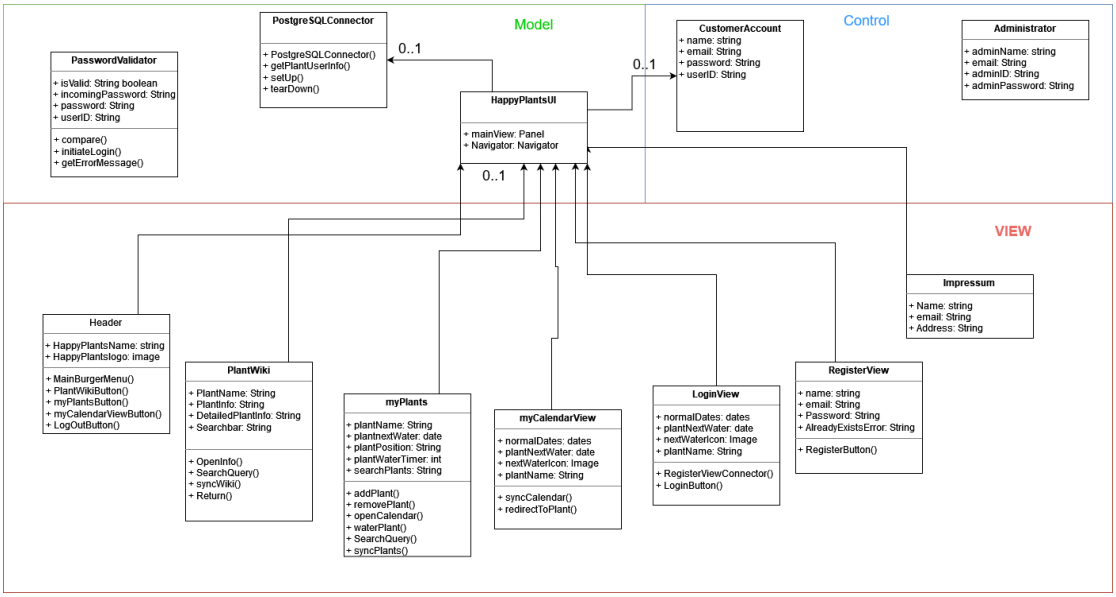
4. Use-Case View

Our overall UC-Diagram:



5. Logical View

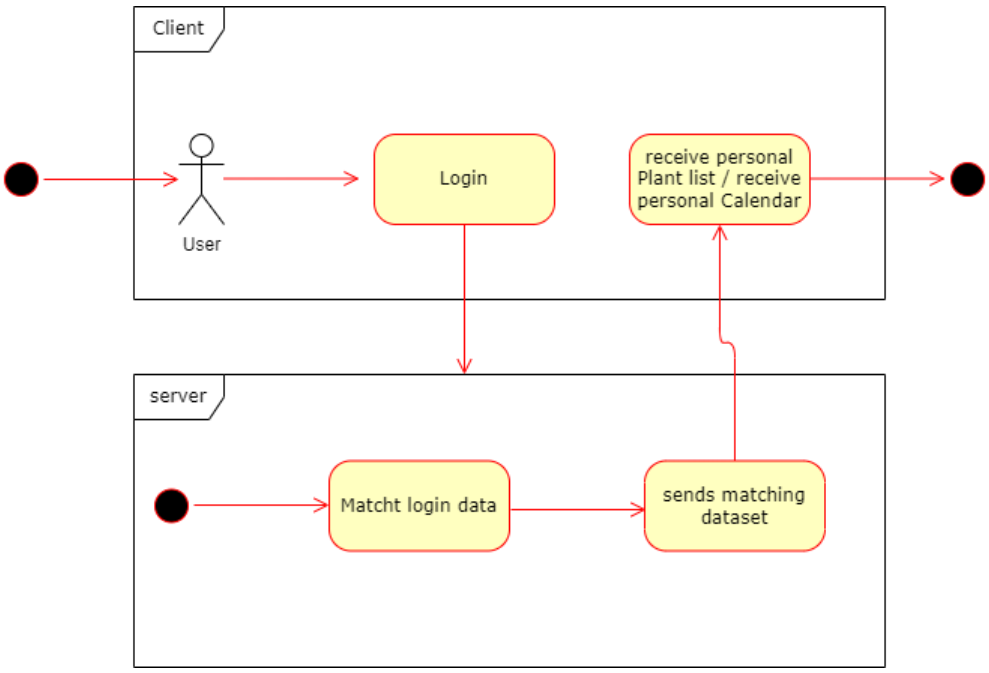
The following image shows an UML diagram of our project whose elements are categorized by model, view and controller. The class diagram contains all the Data Access Object, Models and Controllers, that we will need to finish the basic functionality:



6. Process View

(n/a)

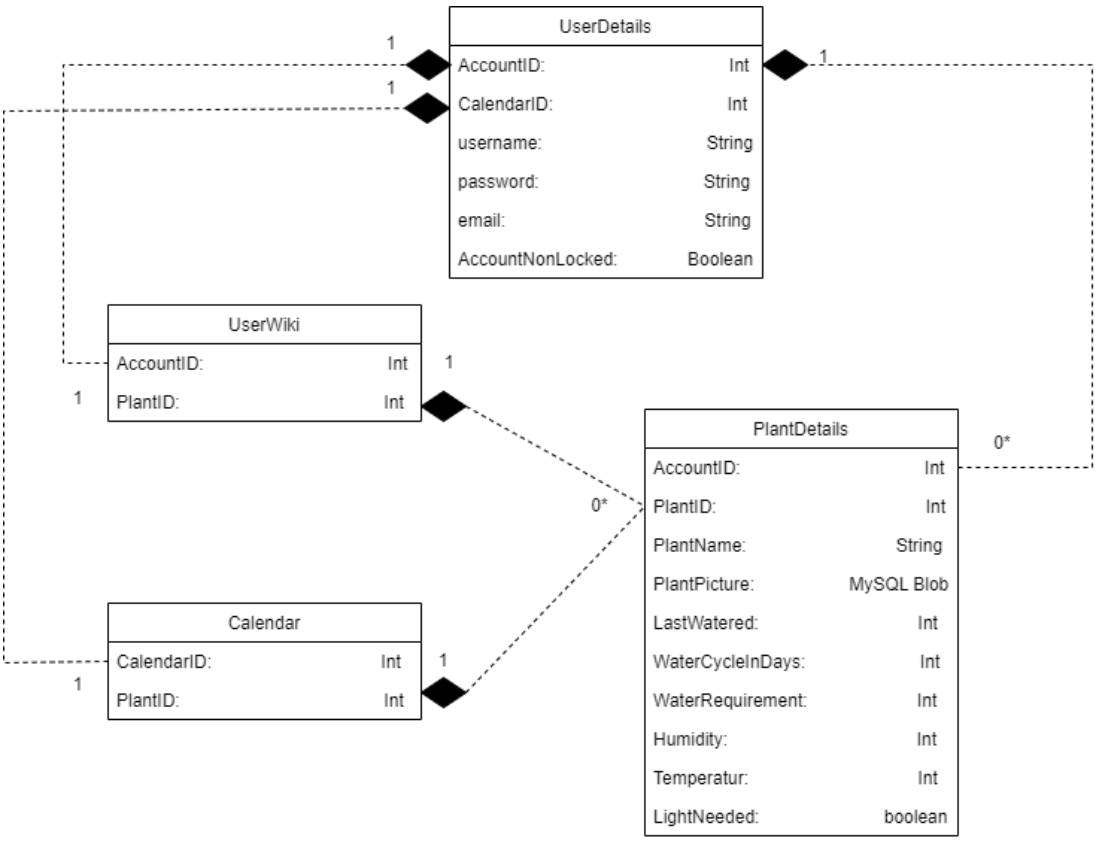
7. Deployment View



8. Implementation View

(n/a)

9. Data View



10. Size and Performance

(n/a)

11. Quality

(n/a)