

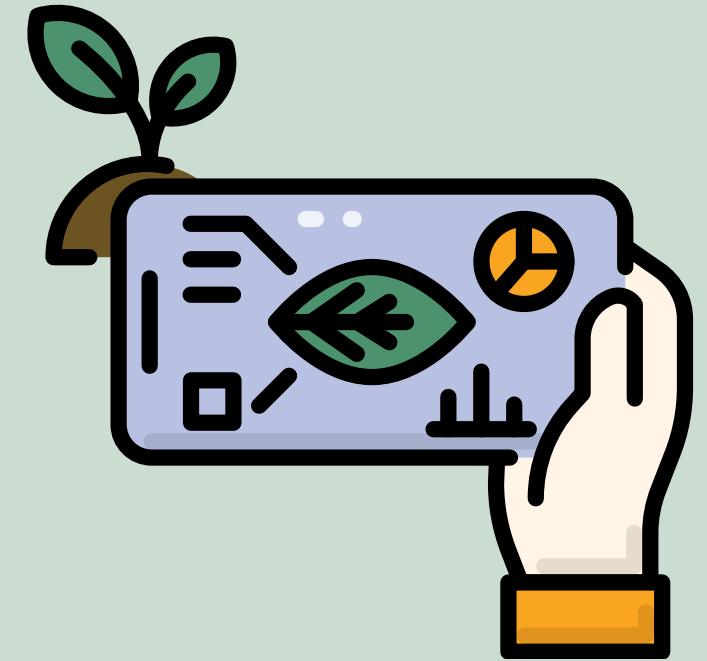
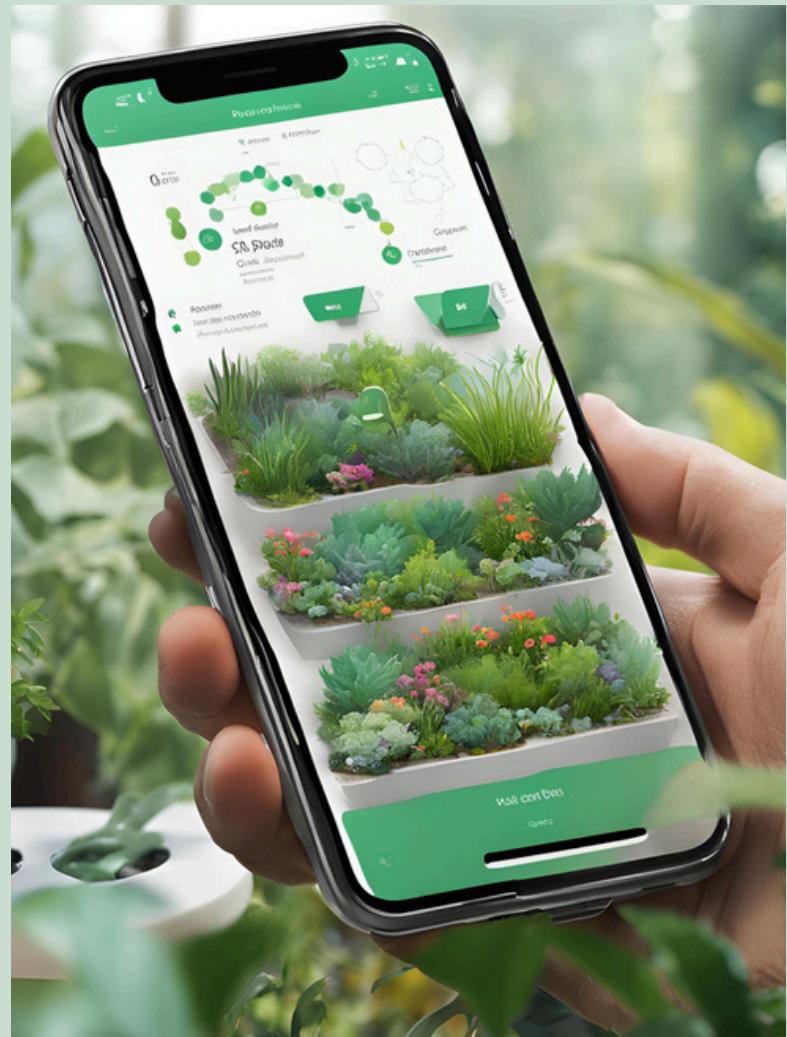


GREEN FINGERS



OUR VISION

Integrating software and hardware tools to manage both indoor and outdoor garden



- AUTOMATION & SMART FEATURES
- GUIDANCE & INFORMATION
- EFFICIENT GARDEN MANAGEMENT

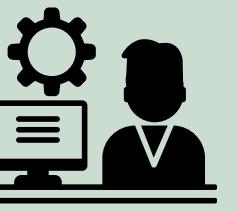


OUR Mission

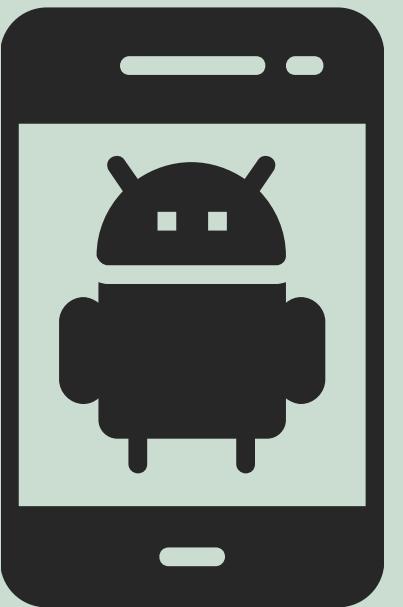
USERS



plants Lover



ADMINS

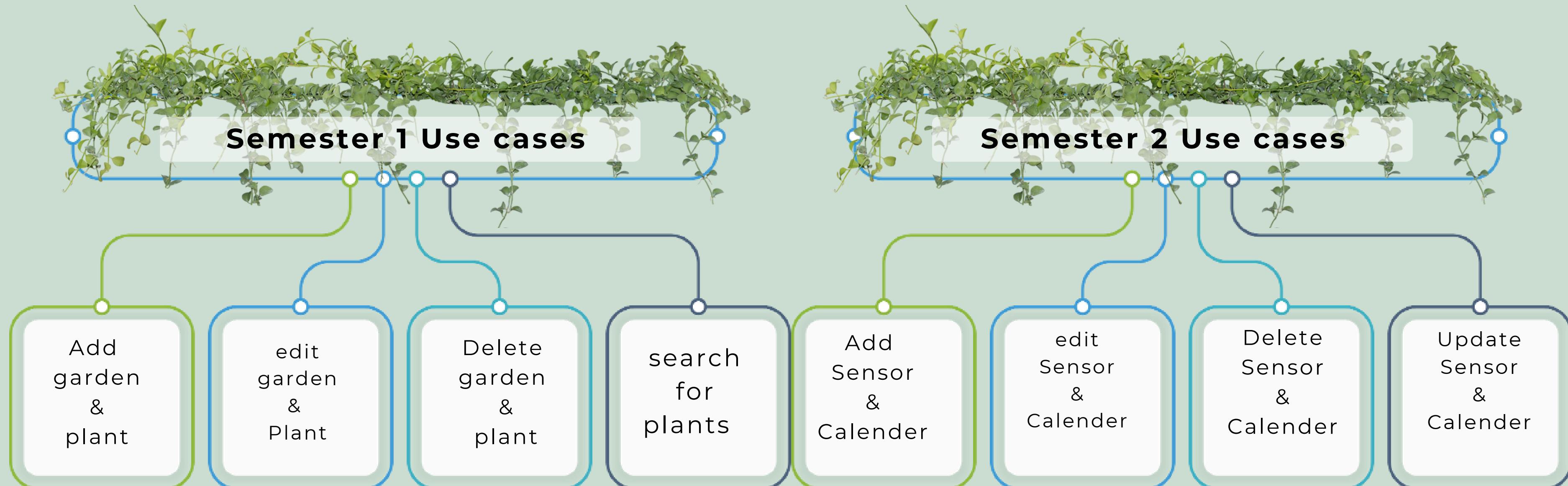


DEVELOP FOR:

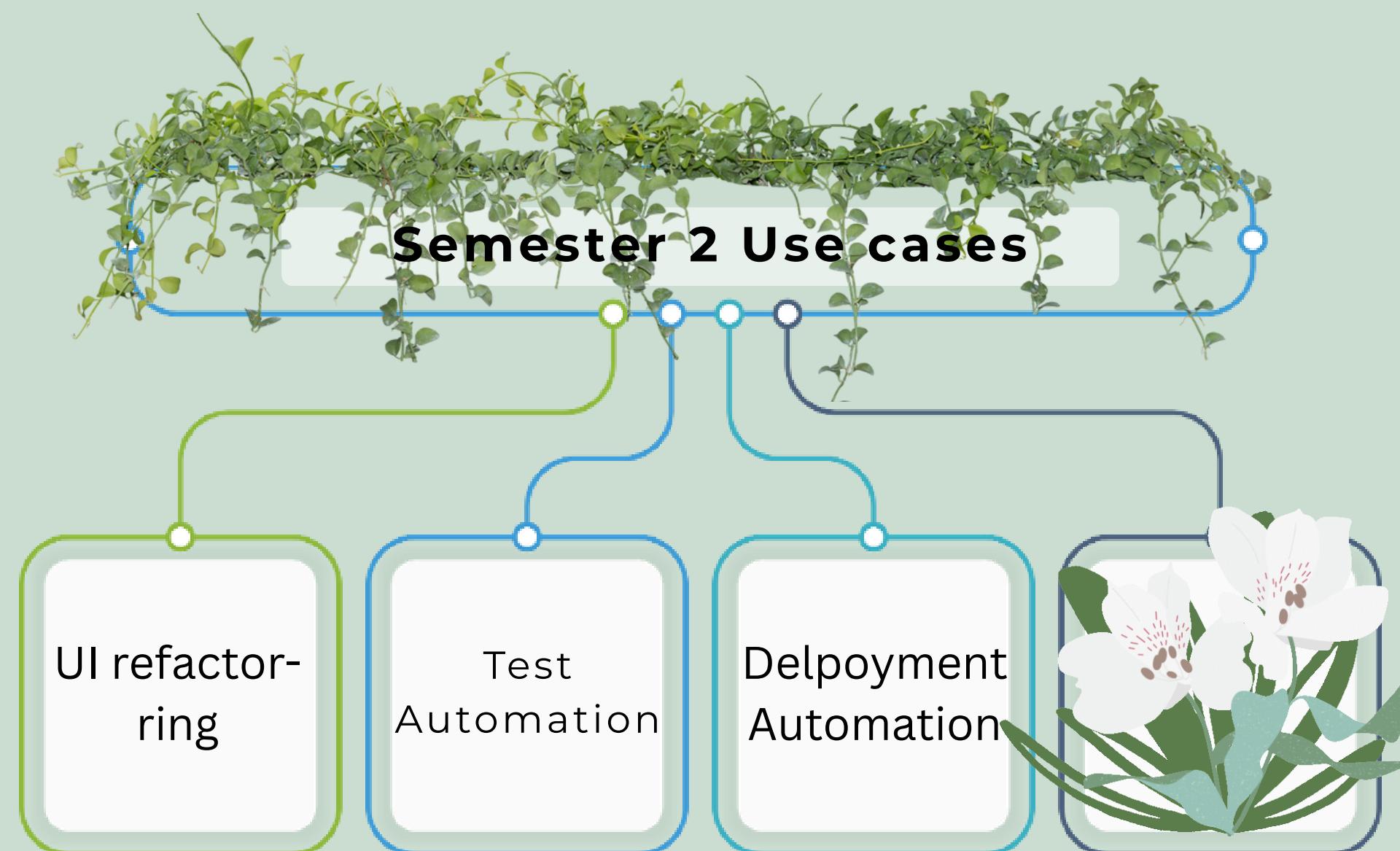


We aim to assist gardeners in tracking plant care through features like adding plants, managing watering schedules, receiving notifications, and integrating sensor data.

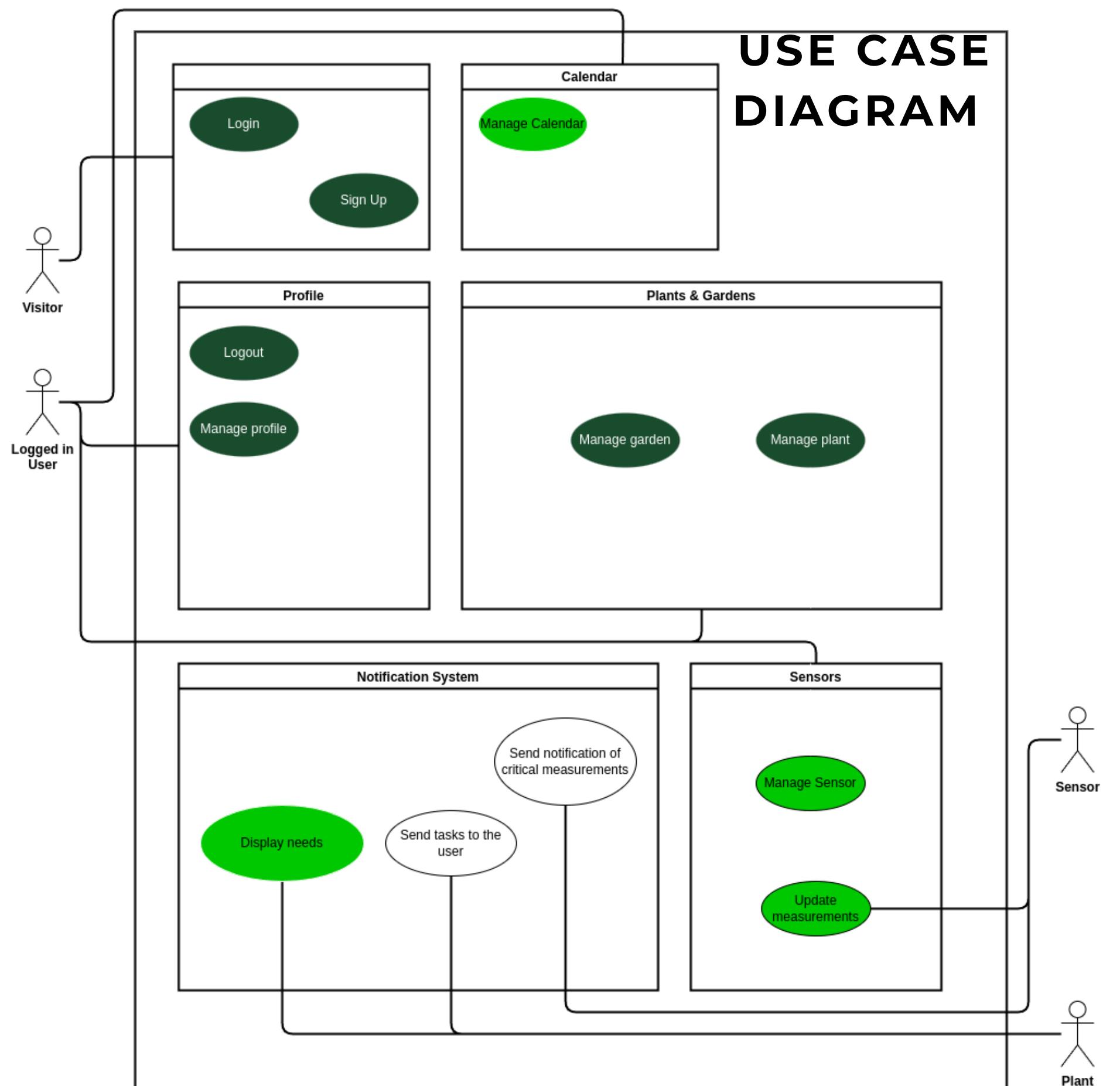
FUNCTIONAL USE CASES



Non functional Use cases



USE CASE DIAGRAM



manage Sensor

Add a sensor to your plant

show moisture data

Update the name and the corresponding plant

Delete the data for the deleted plant

create a water plan along with adding a plant

show the water plan and frequency in calendar page an in detail page

Update the water day plan manually

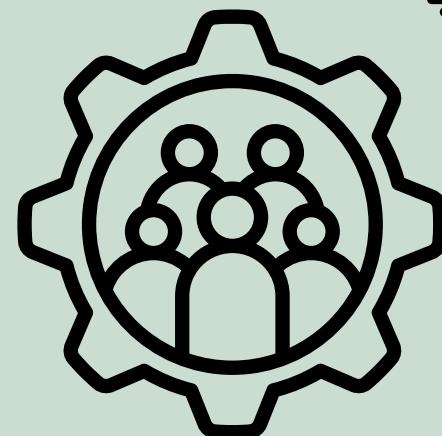
delete the associated information when the corresponding plant is deleted.

manage calendar

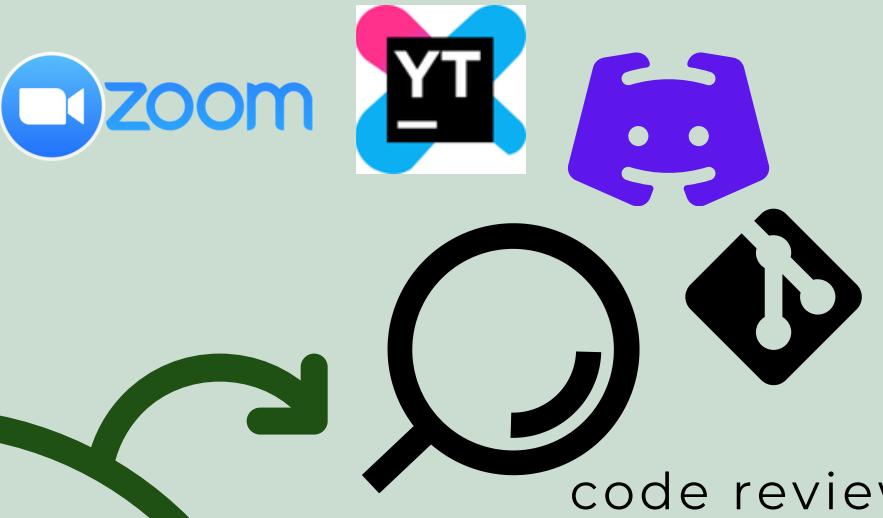


Project Management

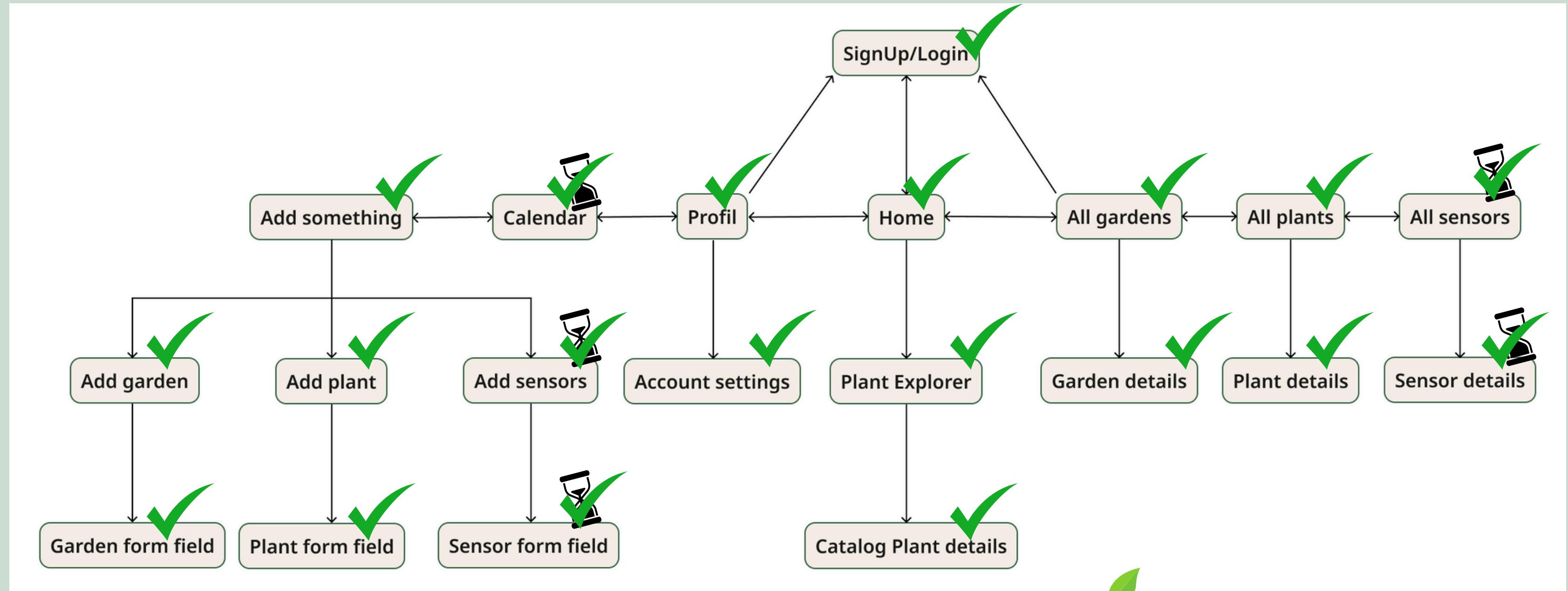
Team working on
the new Ideas

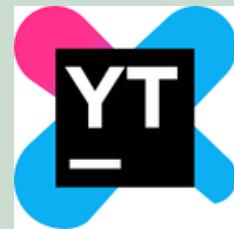


Sprint Review
every week



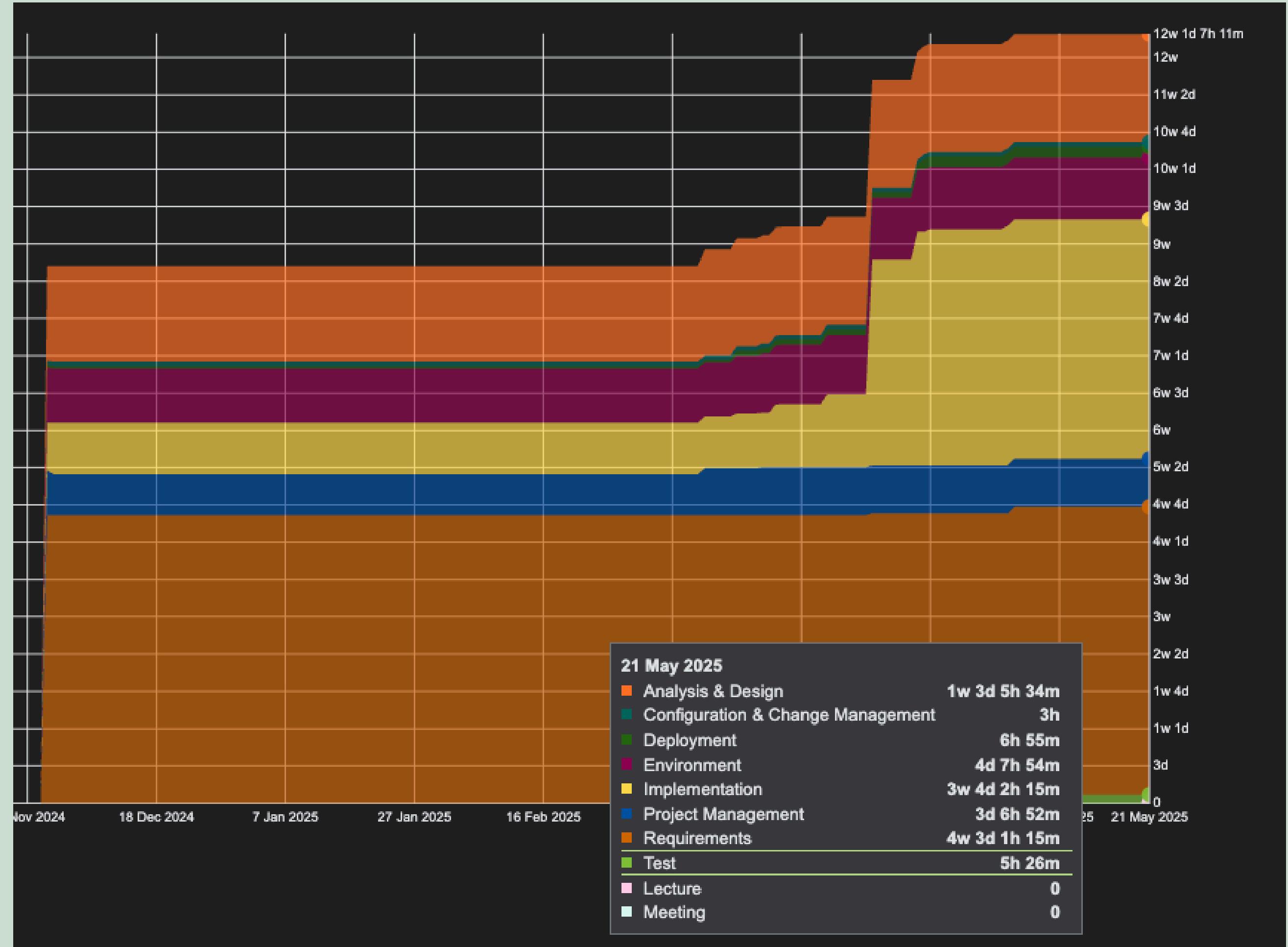
long-term planning





RUP

Our [YouTrack Link](#)

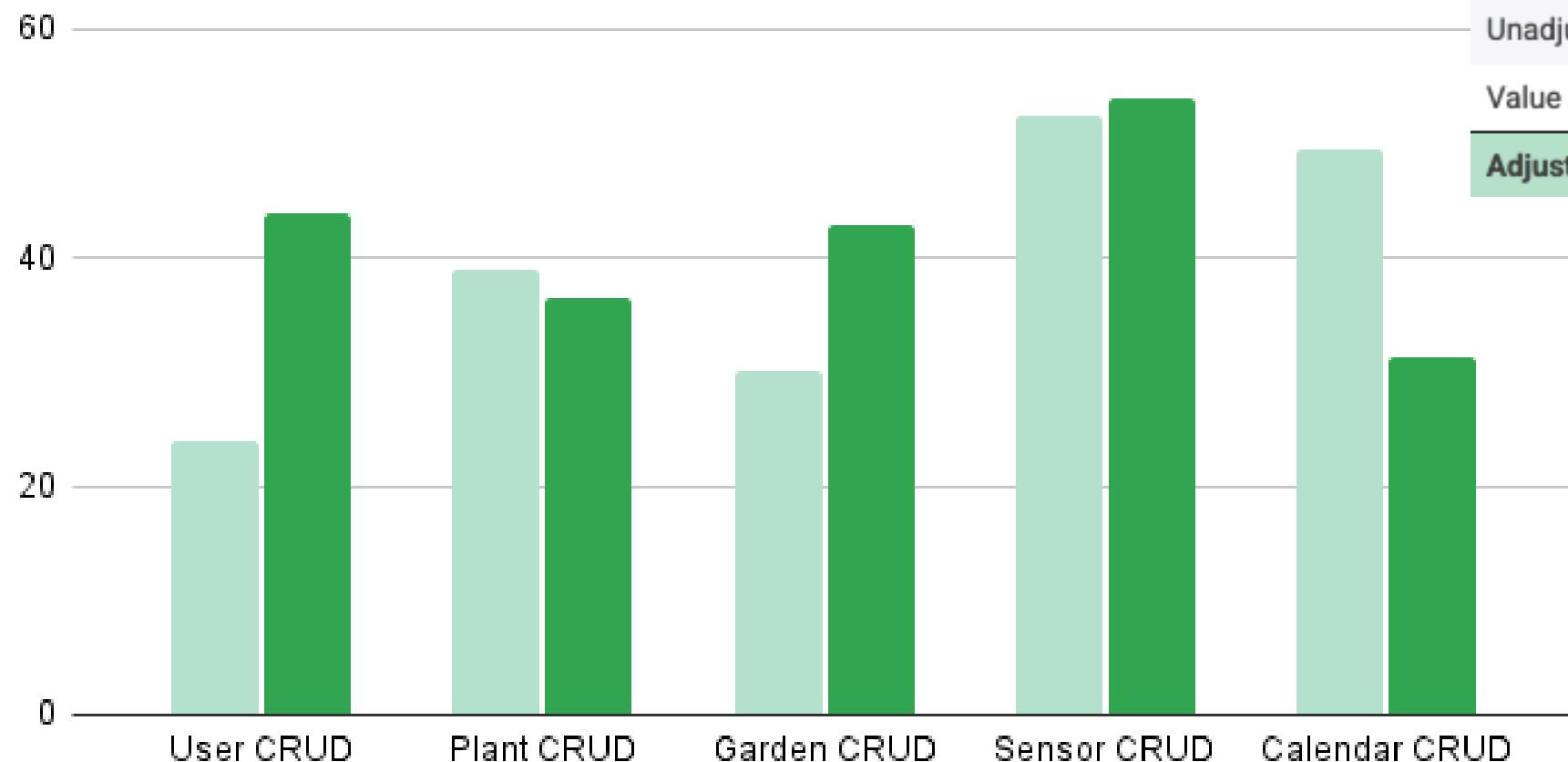


Cost Estimation

link to Function point /time spent

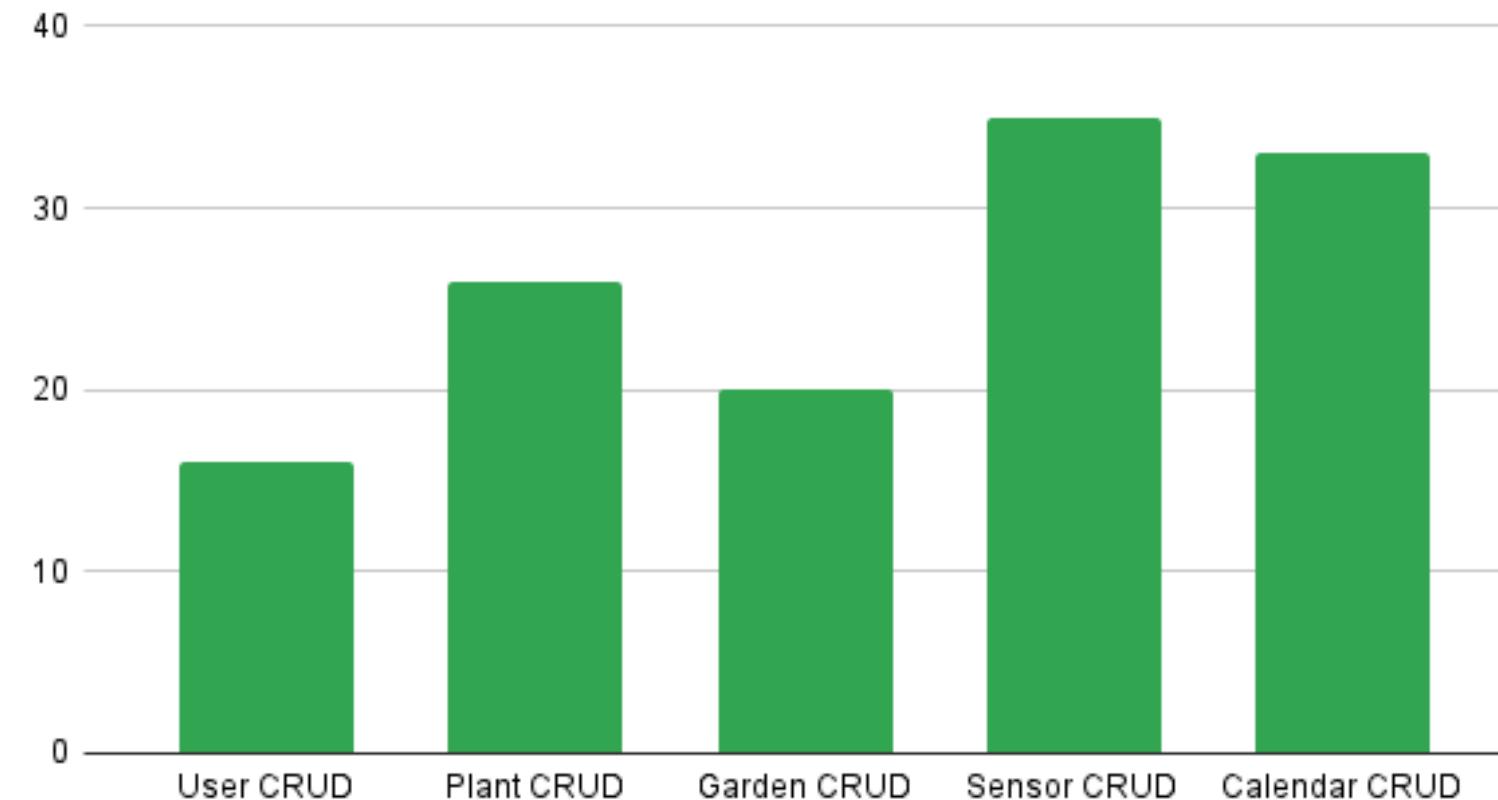
Estimated vs. Actual Hours per CRUD Type

■ CRUD Estimated Hours ■ CRUD Spent Hours



CRUDS	CRUD FP Points	CRUD Estimated Hours	CRUD Spent Hours
User CRUD	16	24	43,75
Plant CRUD	26	39	36,5
Garden CRUD	20	30	42,75
Sensor CRUD	35	52,5	54
Calendar CRUD	33	49,5	31,3
Unadjusted Total	130	195	208,3
Value Adjustment Factor (VAF)	1,01		
Adjusted Total	131,3	196,95	208,3

Function Points per CRUD Type



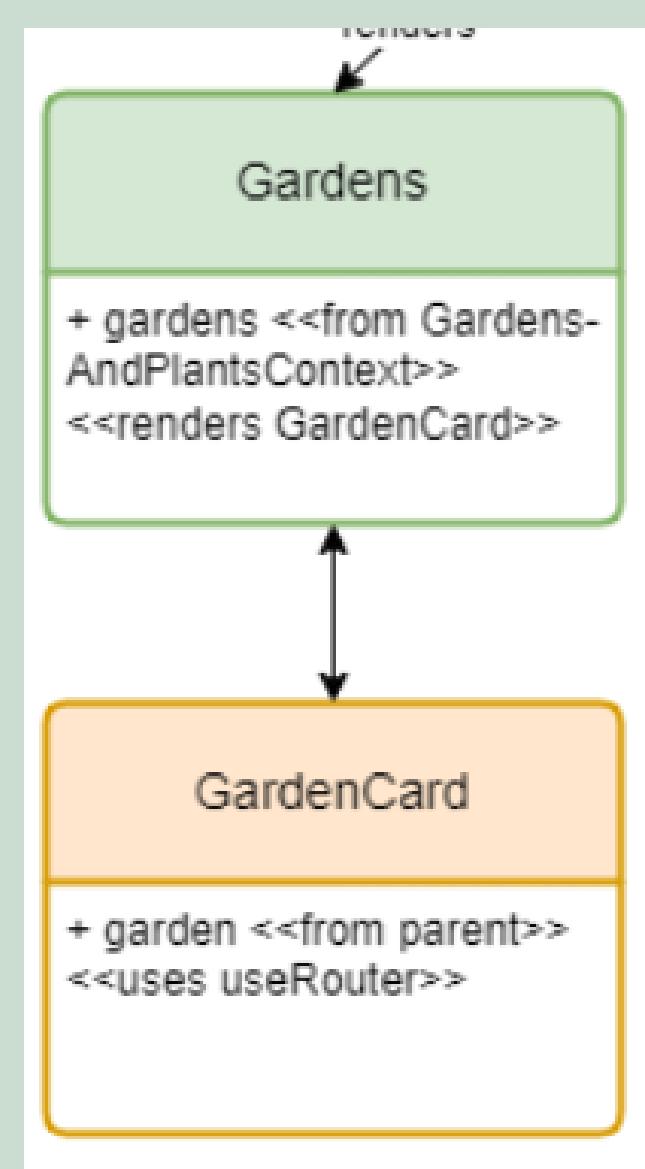
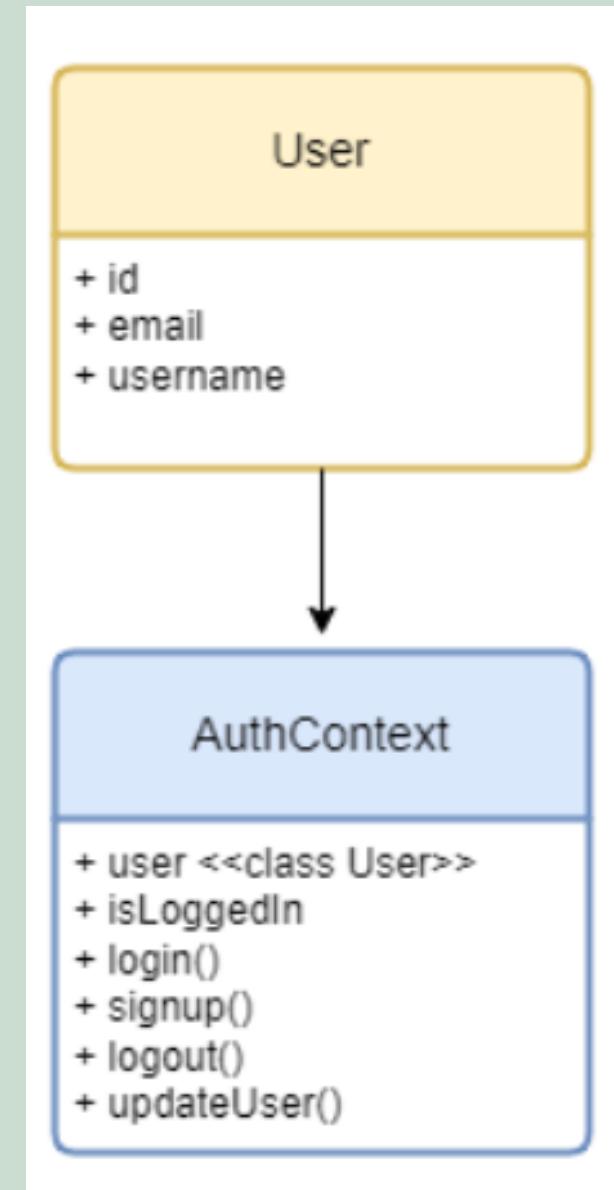
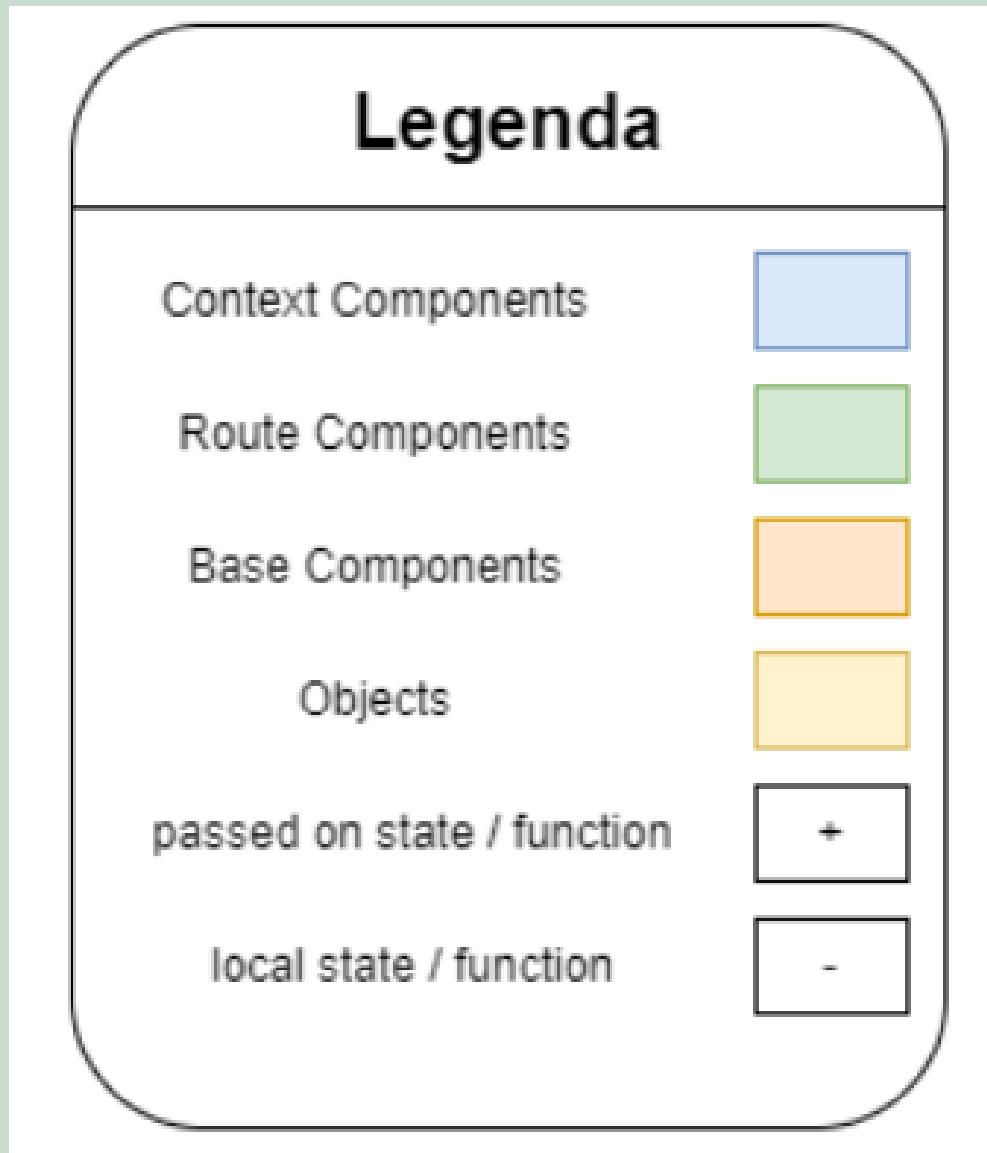
Risk Management

[Link to Risk Management Sheet](#)

1	Risk	Description	Risk Manager	Risk Category
2	Bad code quality	Bloated functions, patternless coding, so called "spaghetti code"	Casimir Bomans	Complexity
3	Lack of Time of Tracking	Forgetting to update Youtrack regulary after work is done.	mal.opderbeck.23@l...	Organizational Environment
4	Miscommunication	not being able to communicate in a team can cause a lack of trust between team members and lead to bigger problems	sim.golrokh.23@lehr...	Team
5	False time estimation	Wrong estimation of time for a issue	mal.opderbeck.23@l...	Estimation
6	Sickness	Team members get sick	sim.golrokh.23@lehr...	Team
7	Unbalanced work distribution	An unbalanced contribution of code to the project	Casimir Bomans	Team
8	Missing deadlines	Sprints fall behind schedule, resulting in the entire project deadline being missed	mal.opderbeck.23@l...	Estimation
9	Lack of project management tools	not using a proper tools for following the workflow can cause chaos in project	mal.opderbeck.23@l...	Requirement
10	Misunderstanding the requirements	cause not delivering the product in time and properly		Requirement
11	Loss of assets and code	Local developed code and assets can get lost		Organizational Environment
12	Team conflicts	Disputes between team members can halt progress and also lead to an unproductive environment		Team

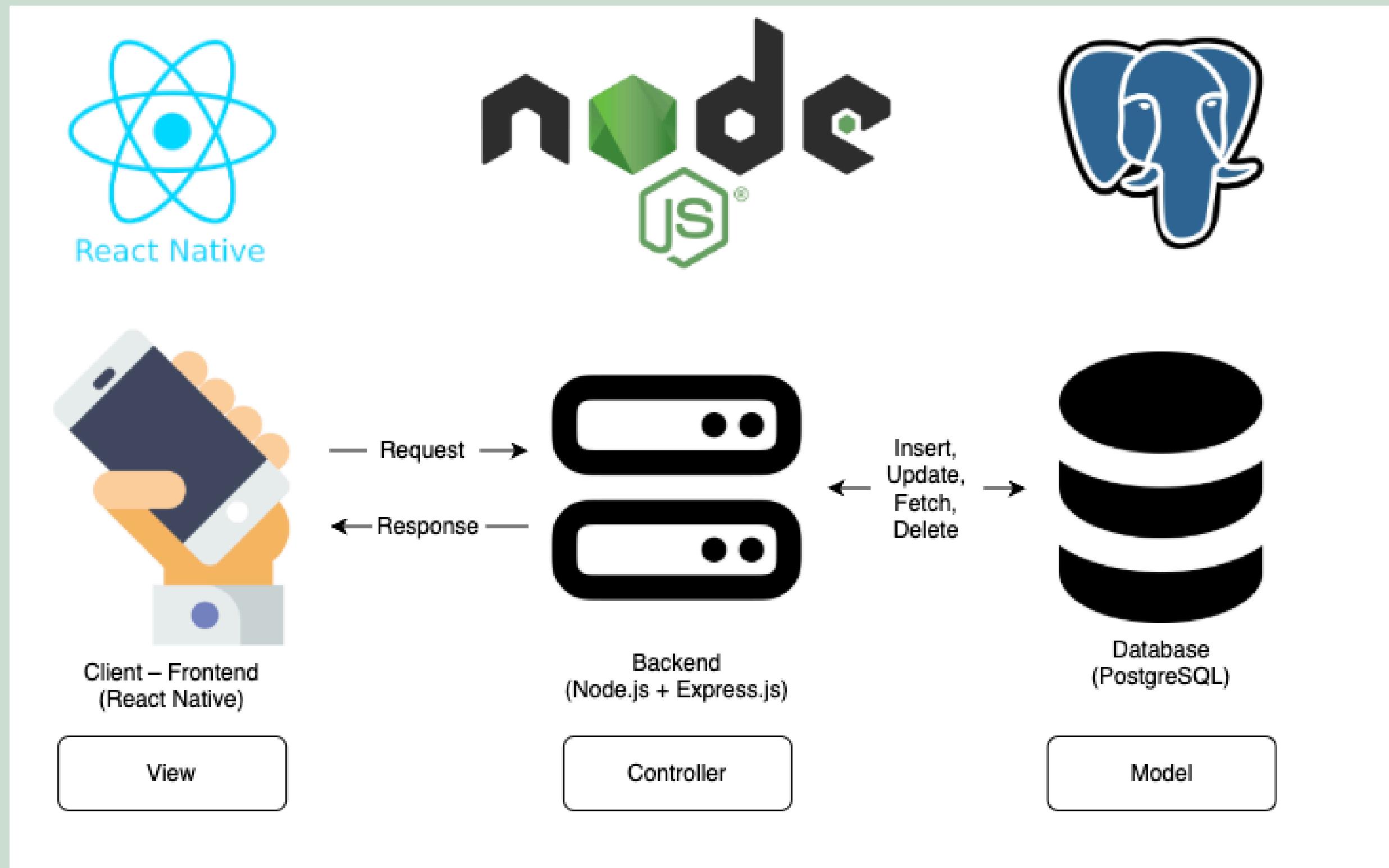
1	Probability	Damage	Priority	Mitigation Strategy
2	2	3	6	refactoring, code reviews
3	3	2	6	make use of YouTrack timer and visit sprint board. Plan time for updating administration at the end of a work phase.
4	2	2	4	try to make the team a trustworthy and safe place for communication between team members
5	2	2	4	Use Youtrack to get a clear view of used time and estimated time to see if our estimations are realistic.
6	2	2	4	Catch up on lost work and information when better again. Keep your code up to date on github, so other people can continue working.
7	1	3	3	Open communication, keep track of blame on Github
8	3	1	3	Keep Youtrack up to date. Have weeklies to keep the team updated on the progress
9	2	1	2	using a proper tools
10	1	1	1	make the requirements clear in the team meetings
11	1	1	1	Make use of Github and push commits regularly.

Class diagram

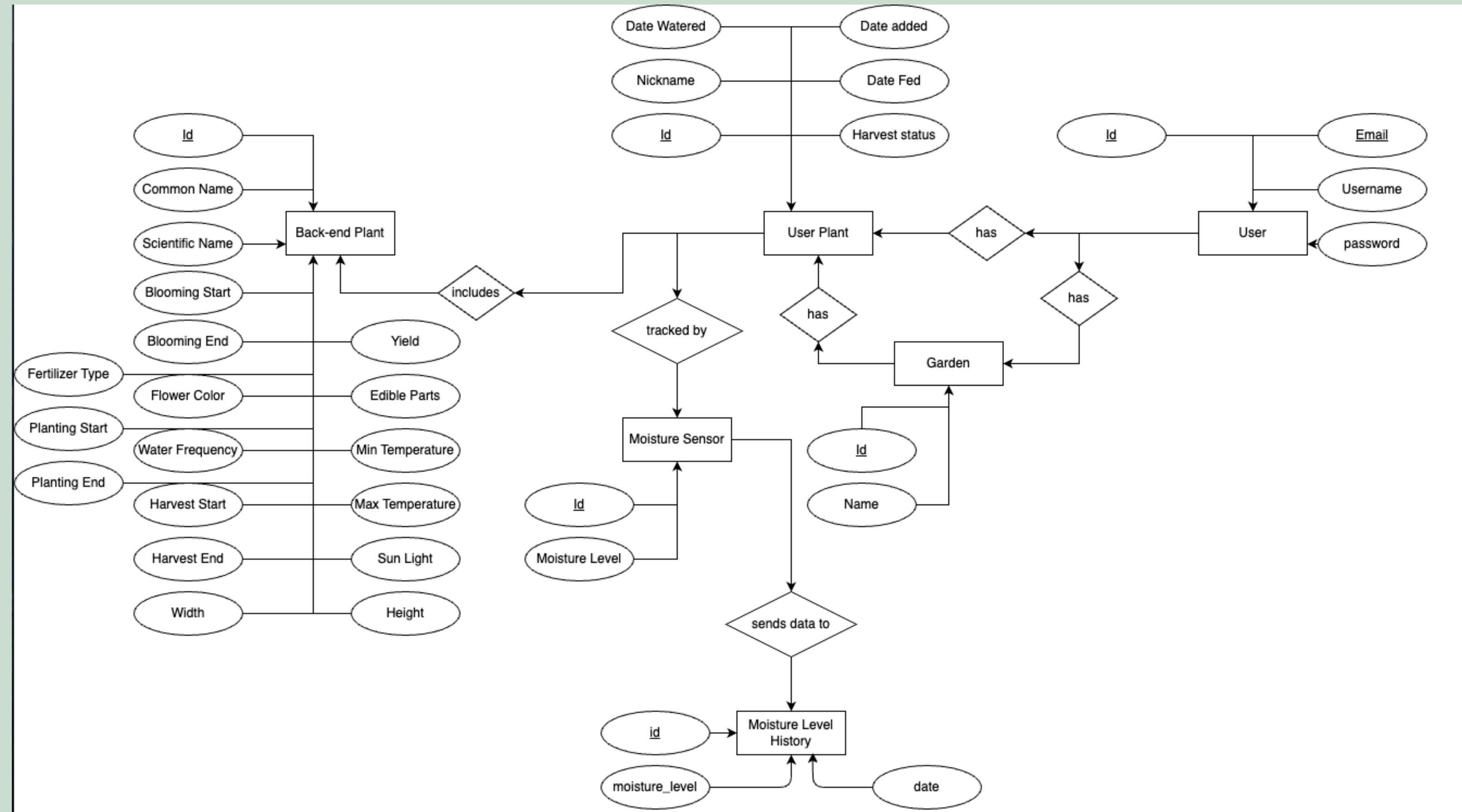


check out the complete class diagram [here](#)

MVC architecture & TeckStack



Database structure



Technical ability

Green Fingers - Software Architecture

Table of Contents

- [1. Introduction](#)
 - [1.1 Purpose](#)
 - [1.2 Scope](#)
 - [1.3 Definitions, Acronyms, and Abbreviations](#)
 - [1.4 References](#)
 - [1.5 Overview](#)
- [2. Architectural Representation](#)
 - [2.1 Model-View-ViewModel \(MVVM\)](#)
 - [2.2 MVVM High-Level Overview](#)
- [3. Architectural Goals and Constraints](#)
- [4. Use-Case View](#)
- [5. Logical View](#)
 - [5.1 High-Level Overview](#)
 - [5.2 Class Diagram](#)
- [6. Process View](#)
- [7. Deployment View](#)
- [8. Implementation View](#)
- [9. Data View](#)
- [10. Size and Performance](#)
- [11. Quality](#)

Green Fingers - Software Requirements Specification

Table of contents

- [Table of contents](#)
- [Introduction](#)
 - [Purpose](#)
 - [Scope](#)
 - [Definitions, Acronyms, and Abbreviations](#)
 - [References](#)
 - [Overview](#)
- [Overall Description](#)
 - [Vision](#)
 - [Use Case Diagram](#)
 - [Used Tools](#)
- [Specific Requirements](#)
 - [Functionality](#)
 - [Usability](#)
 - [Reliability](#)
 - [Performance](#)
 - [Supportability](#)
 - [Design Constraints](#)
 - [Online User Documentation and Help System Requirements](#)
 - [Purchased Components](#)
 - [Interfaces](#)
 - [Licensing Requirements](#)
 - [Legal, Copyright And Other Notices](#)
 - [Applicable Standards](#)

The screenshot shows a GitHub repository page for 'gardeningApp'. The repository is public and has 43 branches and 0 tags. The main branch is selected. The repository contains files such as .expo, databaseStructure, docs, green-fingers, .gitignore, Gemfile, Gemfile.lock, README.md, package-lock.json, and package.json. The README section is titled 'gardeningApp' and describes it as a 'Project for Software Engineering' with a link to <https://dhw-malte.github.io/gardeningApp/>.

[link to SAD](#)

[Link to SRS](#)

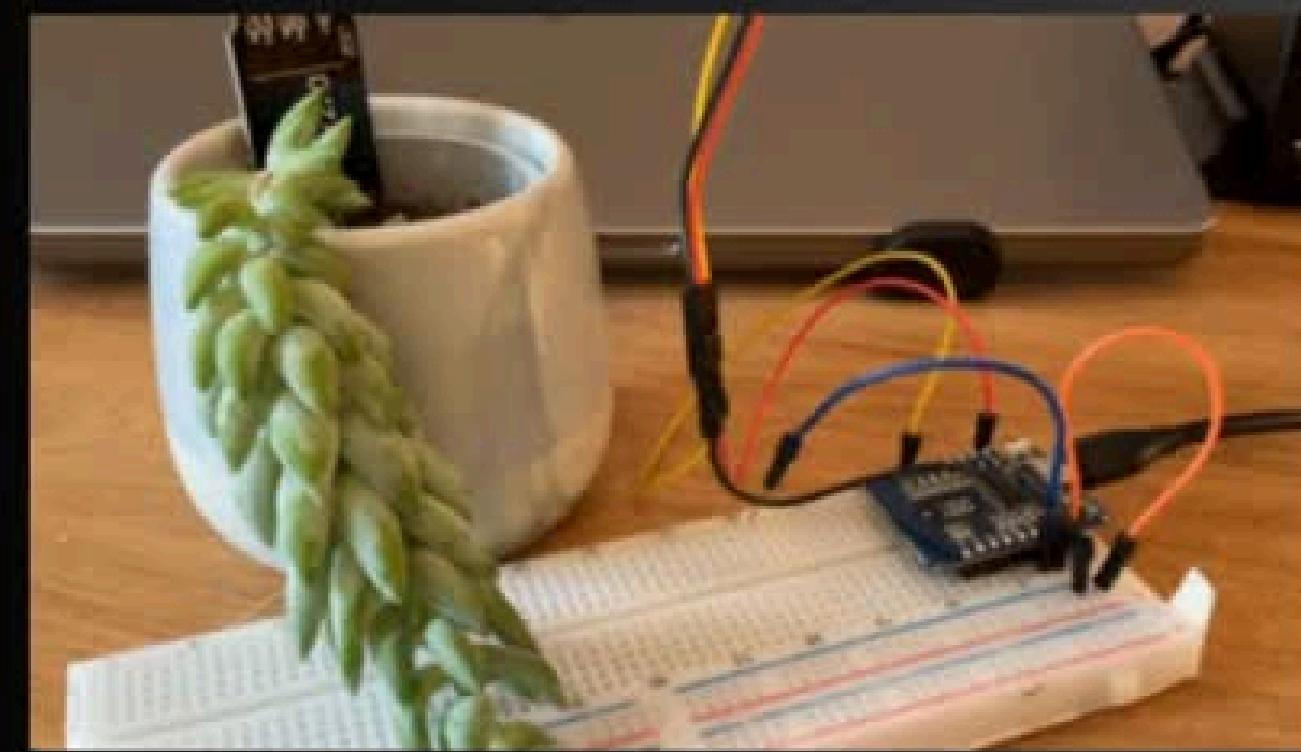
[link to GitHub GardeningApp-FrontEnd](#)

[Link to Github GardeningApp-BackEnd](#)

Testing & Automation - From E2E to Deployment

The image displays a composite view of several software interfaces related to mobile application development and deployment.

- Top Left:** A screenshot of a mobile application titled "Gardens". The screen shows a form for "Add Garden" with fields for "Name" and "Location". A keyboard is visible at the bottom. The interface includes navigation icons like home, gardens, plants, and calendar, along with a top bar with a user icon and a "Hello, malte" message.
- Middle Left:** A screenshot of a mobile UI testing tool. It shows a recorded script for interacting with a mobile application. The script includes actions like "tapOn", "assertVisible", and "inputText", along with coordinates (e.g., "point: 50%,54%").
- Top Right:** A screenshot of a GitHub Actions pipeline for a Node.js project. The pipeline is named "build-and-deploy" and has a status of "succeeded 7 hours ago in 30s". The steps listed are: Set up job, Checkout code, Set up Node.js, Install dependencies, Run tests, Create .env file for Docker Compose, Build and start containers with Docker Compose, Post Set up Node.js, Post Checkout code, and Complete job.
- Bottom Right:** A screenshot of a README page for a repository named "gardeningApp". It features a "codecov" badge showing 77% coverage and the repository name in large, bold, white text.



Output Serial Monitor ×

Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on '/dev/cu.usbserial-130')

```
18:59:42.232 -> {id:0x058c 616c60 00:00} AP started: GreenSensor_16AADF
18:59:42.232 -> IP: 192.168.4.1
18:59:42.232 -> W
```

Ln 20, Col 3 NodeMCU 1.0 (ESP

Sensors Hello, Thilo 🌱

You don't have any sensors yet. Click on the plus button to add one.

+

Home Gardens Plants Sensors Calendar

DEMO



