496 Software Project:
Plant Watch

Brenden Bokino and Nishant Gurung

2026-02-11

1 Client Information

By sharing this client information and the rest of this document, you are stating that this client has provided this project as something they want (not something you created and asked if they wanted), and that they are interested in having you complete this project for your capstone.

- Client name: Dr. Maren Blohm
- Client title: Professor of Biology
- Client email address: mblohm@loyola.edu
- Client employer: Loyola University Maryland
- How you know the client: Loyola professor, met in person.

2 Project Description

2.1 Overview

Our project will be to build a mobile application that can be used to manage plant life. The app will allow users to scan plants and assess their condition, suggesting potential watering schedules and nutritional supplements to help their growth. The app will also allow users to manage greenhouses and interact with devices that might be used in them. Some of the devices Dr. Blohm is interested in are devices that can monitor moisture levels in soil, weather conditions, and automatic sprinklers.

These devices would be in the greenhouse and the garden beds we have around the campus. The information provided by our software will help them make better decisions about caring for these plants. It will also save time for the department, requiring them to check on the plants less often. With the Biology Department relocating, having to travel to the greenhouses less would be ideal.

2.2 Key Features

1. Scanning a plant to identify it and possibly note nutrient deficiencies and watering needs.
2. Individualized profiles for users that lets them manage their greenhouses and their devices.
3. Connect with and pull information from devices in the greenhouses such as weather monitors and moisture monitors.
4. Allow users to activate devices remotely or configure devices to activate in certain conditions (ex. Automatically activate sprinklers when moisture in the soil is below a certain threshold)

2.3 Why this Project is Interesting

This project bridges the two fields of Computer Science and Biology by showing how technology can contribute directly to environmental conservation efforts. This capstone gives us a chance to make a real, measurable impact in preserving the biodiversity of Loyola's campus. Additionally, it will build a model that can be adapted by other institutions seeking to monitor and protect the nature around them. The skills that will be learned are interesting as well. Computer vision is something we both do not have worked with and learning to connect different devices together will be a good experience.

2.4 Areas of CS required

1. Mobile app development
2. Computer Vision
3. Networking
4. Database

2.5 Potential Concerns and Questions

1. We are unsure of the difficulty of developing with computer vision. There are likely resources out there on it, but we are concerned with how accurate it can get.
2. We are unfamiliar with the devices used when monitoring plants and how programmable they will be.

2.6 Summary of Efforts to Find a Project

(Not necessary for 482) [Briefly list out when/how you've discussed with this client, and if you've discussed with other clients who either didn't work out or didn't respond. If you considered a different project and it didn't work out, why didn't it work out?]

We've been discussing about this project with Dr. Blohm since the last semester. She has guided us towards the right direction and answered our questions about the greenhouse, plant health, and the requirements of the app.

2.7 Comparison to Draft

The project was initially proposed by me (Nishant) during the drafting phase. While the core concept of building an app that uses image recognition to inform the users about its water cycle, we've changed the proposal to incorporate more fields of Computer Science. The initial draft was largely centered on Computer Vision, particularly image recognition for species identification. But the present proposal is not limited to a single subfield; instead, it is divided into several subfields of Computer Science that include:

- Computer Vision for automated species detection
- Database design and management for storing and organizing plant observations, weather patterns, and health assessment data to enable long-term tracking of garden performance.
- Networking for real-time data synchronization with the weather monitoring devices and multi-user access, which will send immediate alerts on temperature drops, and soil moisture to prevent crop loss.
- Mobile App Development for creating an intuitive, user-friendly interface.

3 Requirements

3.1 Non-Functional Requirements

Table 1 presents the NFRs for this software project.

ID	NFR Title	Category	Description
NFR1	App Launch Time	Performance	The application must launch and display the home screen in under 3 seconds.
NFR2	Plant Identification Accuracy	Reliability	The plant identification feature must have a minimum accuracy rate of 85% across common plant species.
NFR3	Scan Process Efficiency	Usability	The scan-to-identify process must require a maximum of 3 user taps to complete.
NFR4	Outdoor Accessibility	Usability	The application must use accessible font sizes and appropriate color contrast for outdoor visibility and readability.
NFR5	Visual Design Consistency	Usability	The application must maintain a consistent color scheme with green as the primary color and white as the secondary color.
NFR6	Platform Compatibility	Compatibility	The application must support iOS and Android.
NFR7	Battery Efficiency	Performance	The application must consume less than 5% of device battery per hour during active use.
NFR8	Alert Delivery Speed	Performance	Emergency alerts for critical conditions (temperature drops, low moisture) must be delivered to users within 30 seconds of detection.
NFR9	Password Encryption	Security	Passwords should be encrypted before being stored in the database.

Table 1: Non-Functional requirements

3.2 Functional Requirements (User Stories)

Table 2 presents the functional requirements for this software project.

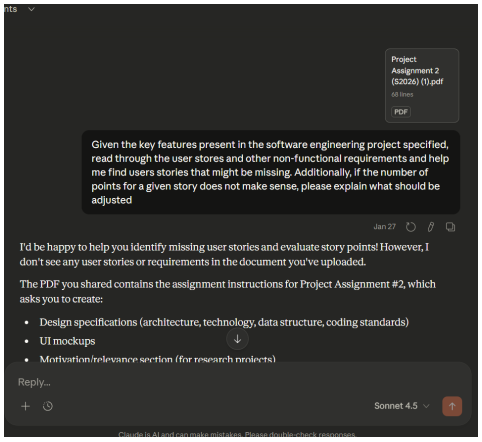
Table 2: Functional requirements as User Stories.

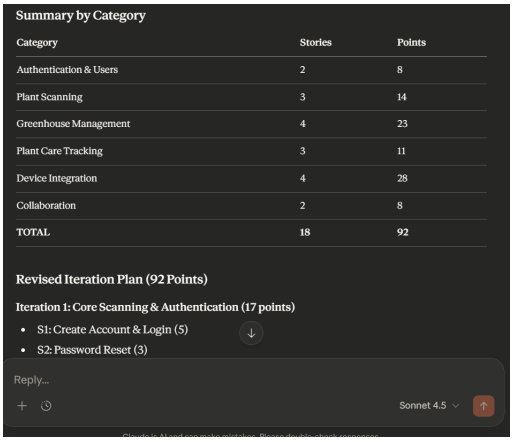
ID	Story Title	Points	Description
S1	Scan Plants for Detailed Information	8	As a user I want to scan a plant with my camera and get information about what plant it is, what watering schedules would be best, and what nutritional deficiencies it might have so I can learn more about plants and how to care for them.
S2	Password Reset	3	As a user I want to be able to reset my password if I forget it so that I can configure a new password and login.
S3	Add Plants to Greenhouse by Scan	2	As a user I want to add a plant that I have scanned to my greenhouse profile so that I do not manually have to input the information I learned from the scan.
S4	View Plant Scan History	3	As a user I want to view the history of plants I have scanned so I don't have to rescan the same plant repeatedly.
S5	Create Account	3	As a user, I want to be able to create an account so I can have my own personal profile.
S6	Greenhouse Profile Management	8	As a user, I want to create, view, edit, and delete a greenhouse profile so I can manage my plants and plant monitoring devices.

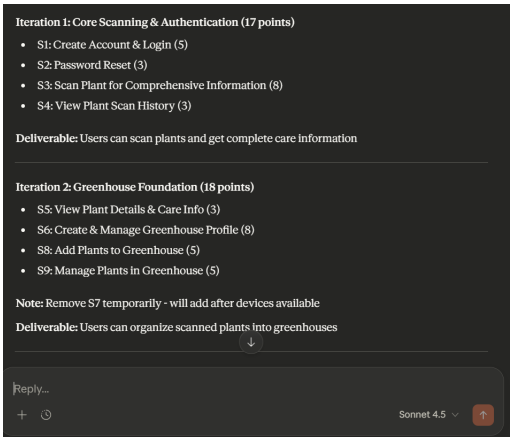
ID	Story Title	Points	Description
S7	Greenhouse Profile Dashboard	5	As a user, I want to view a dashboard of my greenhouse profile with basic information on its plants' and devices' conditions so I can quickly analyze its state.
S8	Sync AcuRite Weather Sensor	5	As a user I want to be able to view information from my AcuRite weather sensor from the app so I know the conditions my plants are in.
S9	Add Plants to Greenhouse	8	As a user I want to be able to create, view, edit, and delete plants in my greenhouse, so that I can keep inventory of plants.
S10	Greenhouse Notifications	5	As a user I want to receive notifications from greenhouses that I am in about conditions that may negatively affect plants and customize what those conditions may be so that I can know when I need to go in to care for them.
S11	Adding Users to Greenhouse	2	As a greenhouse manager I want to add other users to my greenhouse so they can view its information and manage it.
S12	Scheduled Sprinkler Activation	5	As a user I want to sync a sprinkler with the app and configure schedules for automatic sprinkler activation so that I can water plants without having to go to the greenhouse. (Still in discussion with client about these devices)
S13	Greenhouse User Device Use Permissions	2	As a greenhouse manager I want to be able to restrict the usage of devices in my greenhouse to certain users I've invited so I can safely collaborate with other users.
S14	Greenhouse User Plant Edit Permissions	2	As a greenhouse manager I want to restrict the ability to edit plant information to certain users so I can ensure the plant information is coming from a trusted source.
S15	Greenhouse Activity Log	3	As a greenhouse manager I want to be able to view a history of actions taken and edits made to my greenhouse so I can be sure the right users are doing the right things.
S16	Greenhouse Plant Info	2	As a user I want to view detailed information about a plant that I've added to my greenhouse so I can tailor its care and learn more about it.
S17	Sync Soil Moisture Sensor	5	As a user I want to sync information from my soil moisture sensor to the app so that I can view it along with other greenhouse information and make decisions about its care. (Still in discussion with client about these devices)
S18	View Historical Conditions	5	As a user I want to have a graph view of various collected historical information from my greenhouses devices such as humidity, temperature, etc. so I can make informed decisions concerning my plants.
S19	Login/Logout	1	As a user, I want to be able to login and logout of an account so that I can access the app features that require an account.
	Total:	78	

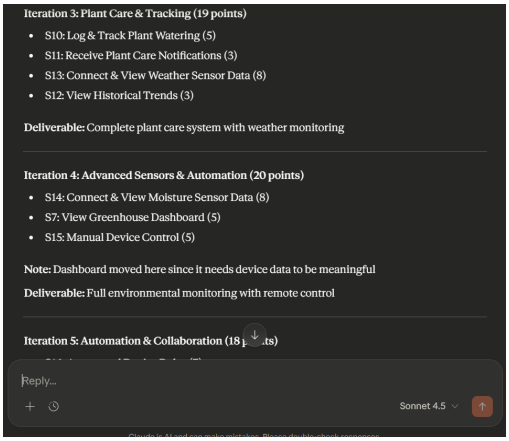
4 AI Usage

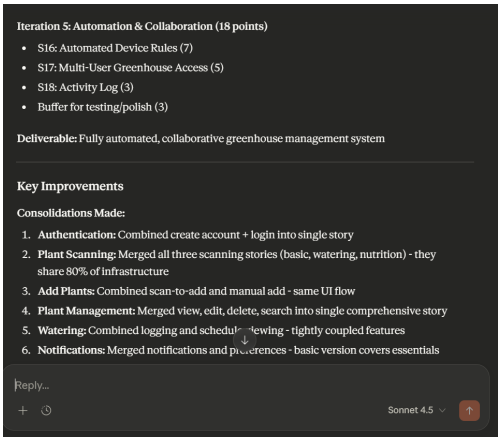
4.1 Conversation

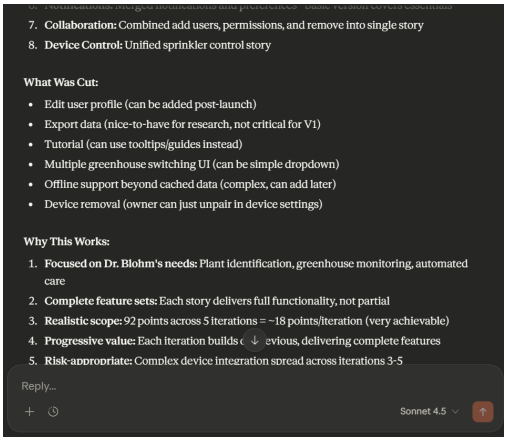


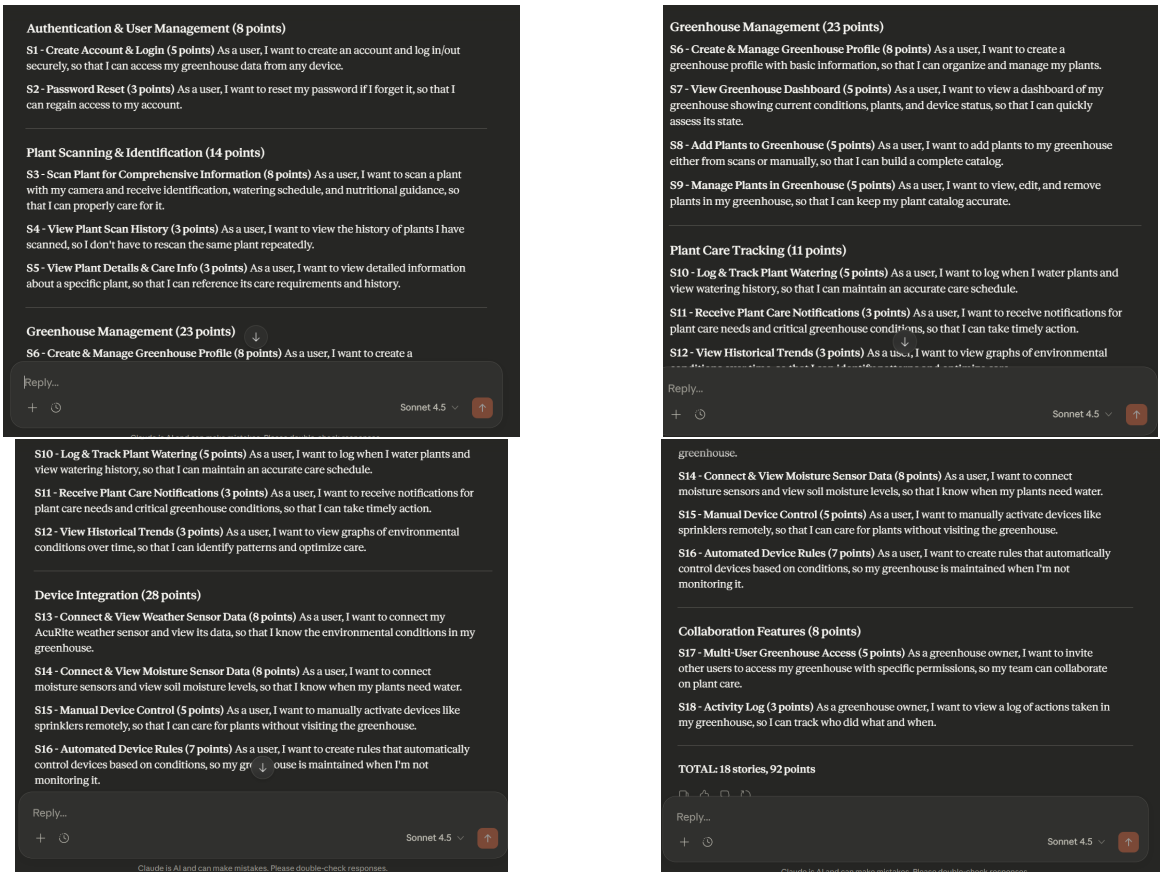












4.2 Reflection

After asking A.I. to help find missing user stories, and reassess the story points, we now have a more detailed list of user stories with a total point tally of 78. A.I. was really helpful for coming up with more ideas, especially since we had a good foundation for it to work off of. There were some stories that it added that were unnecessary and inflated its point totals. There were also some stories it came up with that were more general and would need more specification to be a valid user story. We used some of those as a basis for some new stories. Something especially useful was the way the A.I. grouped the stories into different categories. This inspired us to revise our planning and work on things that were more related together.

Another interesting story was that of syncing the weather sensor data that would be pulled from an API. Dr. Isaacman's guidance was that this story could be overestimating the difficulty of the API calling, but A.I. found no issue with it. This reinforces the idea that A.I. should only be used as a guide. This was a valuable lesson on A.I. usage. The results we got certainly could not have been used immediately and had to be revised, and they were helpful because the A.I. had a lot of context to work with. This shows how A.I. can really shine; as a tool to assist a software engineer with a good idea and expand on it.

5 UI Mockups

The application will maintain visual consistency through a green and white color scheme with high contrast for outdoor readability.

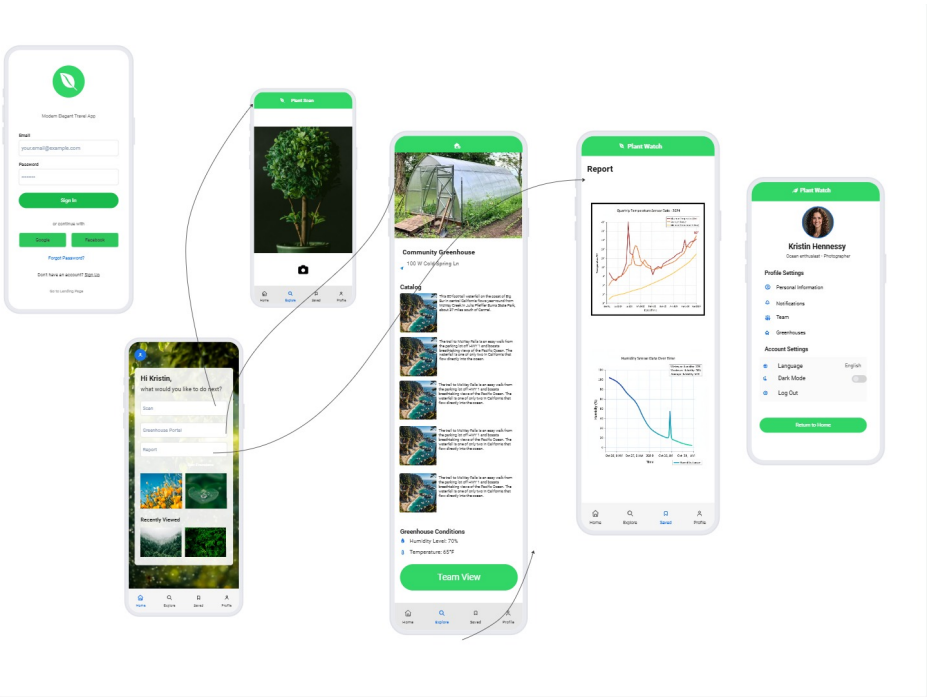


Figure 3: PlantWatch mobile application UI mockups

6 System Design

6.1 Architecture

We will be using a layered architecture. The layers we will have are:

- **Presentation:** Front-end User Interfaces
- **Application:** App logic and API calls
- **Data:** Database

We have chosen this architecture because we plan to use Supabase, a backend service that simplifies client server communications. Logic and processing will be done on the client when possible, and if the load is too strenuous or the server needs to do some processing, Supabase Edge Functions will be used. The server's primary role will be to hold data.
Figure 4 has a simplified class diagram that demonstrates the architecture. The list of front end pages is not exhaustive. The device page is intentionally one class and kept concise because we only know one specific model of device we want to be compatible with our app at the moment. We are still working with our client to decide on what other devices to integrate.

6.2 Diagrams

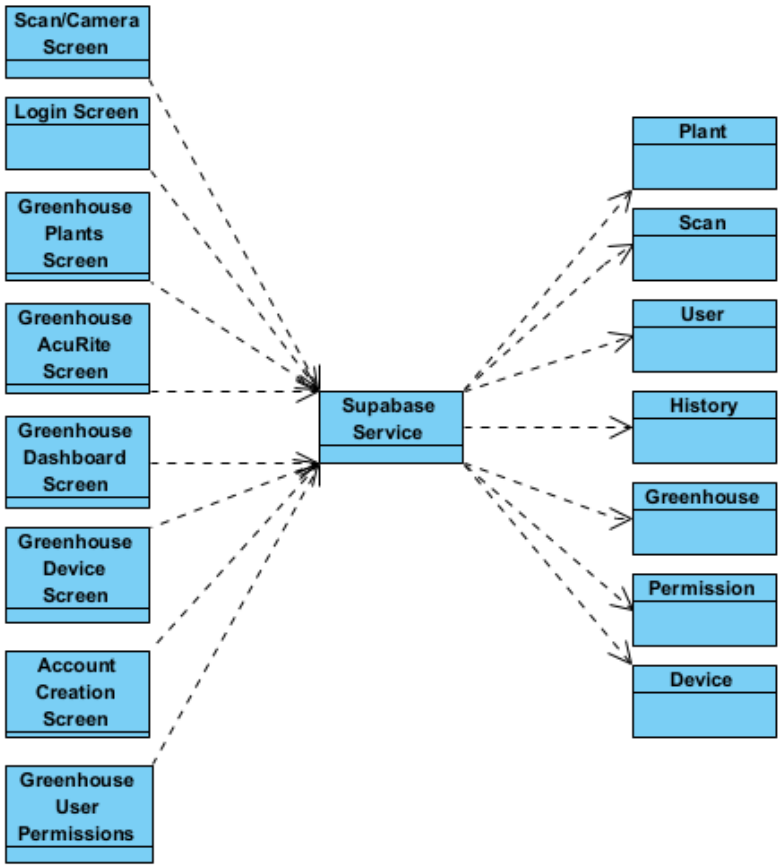


Figure 4: Simplified Class Diagram of App

6.3 Technology

Framework: React Native with Expo
Backend Services: Supabase
Language: JavaScript and Typescript
Primary Database: PostgreSQL 15+
Plant Identification: Pl@ntNet API
Weather Sensor: AcuRite → Weather Underground → Weather Underground API, or use RTL-433 workaround (AcuRite has no direct API)
Testing: Jest

6.4 Coding Standards

We will be following the coding standards below to maintain consistency and quality:

Naming Conventions

- **Database tables:** lowercase (e.g., `user`, `plant`, `greenhouse`)
- **Database Columns:** snake_case (e.g., `created_at`, `moisture_level`)
- **Methods/functions:** snake_case (e.g., `get_plant_info()`, `scan_plant()`)
- **Classes:** PascalCase (e.g., `PlantScanner`, `GreenhouseManager`)
- **Constants:** UPPER_SNAKE_CASE (e.g., `MAX_SCAN_TIMEOUT`)

Testing Requirements

- Minimum 70% code testing coverage for all commits

Code Review Process

- We will review and approve all code changes from our independent branches before they are added to the main project
- The commit messages will include the story number and brief description: `[S3] Added plant scanning`

Quality & Security

- Sensitive information like passwords, API keys will be kept private using environmental variables.
- Encrypt all user passwords before storing them in the database to ensure user passwords are protected.
- Use parameterized database queries to prevent SQL injection attacks where malicious users attempt to manipulate the database
- Include error handling for all external connections like API, database, sensor, so the app doesn't crash when something goes wrong
- Attempt to not repeat codes and instead create reusable functions, so that changes only need to happen in one place, reducing redundancy and potential bugs.

Documentation

- Functions will have comments at the top explaining what it does, explaining the purpose, input, parameters and return values.
- Complex codes will include comments explaining what it does.
- We will document all API endpoints with examples of how to use them.

6.5 Data

Figure 4 contains a basic logical model of how we plan to structure data. So the diagram is not cluttered, some proposed attributes for the tables and relationships will be listed below the diagram. More are likely to be added when development begins.

Scan

- scan_dt
- scan_photo
- scan_description

User

- user_name
- password
- email

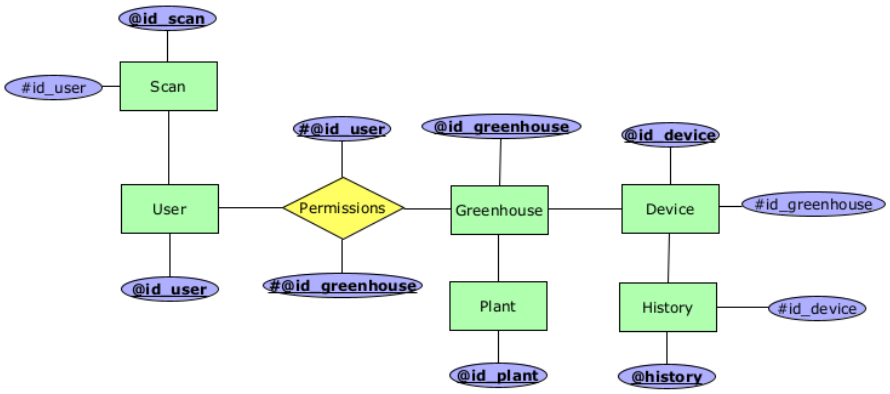


Figure 5: Basic Logical Model of Data

Greenhouse

- greenhouse_name

Permissions

- plant_edit
- device_use

Device

- device_name
- device_api

History

- history_details

Plant

- plant_name
- plant_description
- plant_count

7 Iterations

7.1 Iteration Planning

Table 3 shows the iteration planning.

			Points	
It.	**Dates**	**Stories**	**Planned**	**Done**
1	01/01 - 02/01	S01, S05, S04, S19	16	–
2	02/01 - 03/01	S16, S06, S09	18	–
3	03/01 - 04/01	S03, S07, S08, S17	17	–
4	04/01 - 05/01	S10, S12, S18	15	–
5	05/01 - 06/01	S2, S11, S13, S14, S15	12	–
		Total:	78	–

Table 3: Iteration Planning for Incremental Deliveries

7.2 Iteration/Sprint 1

7.2.1 Planning

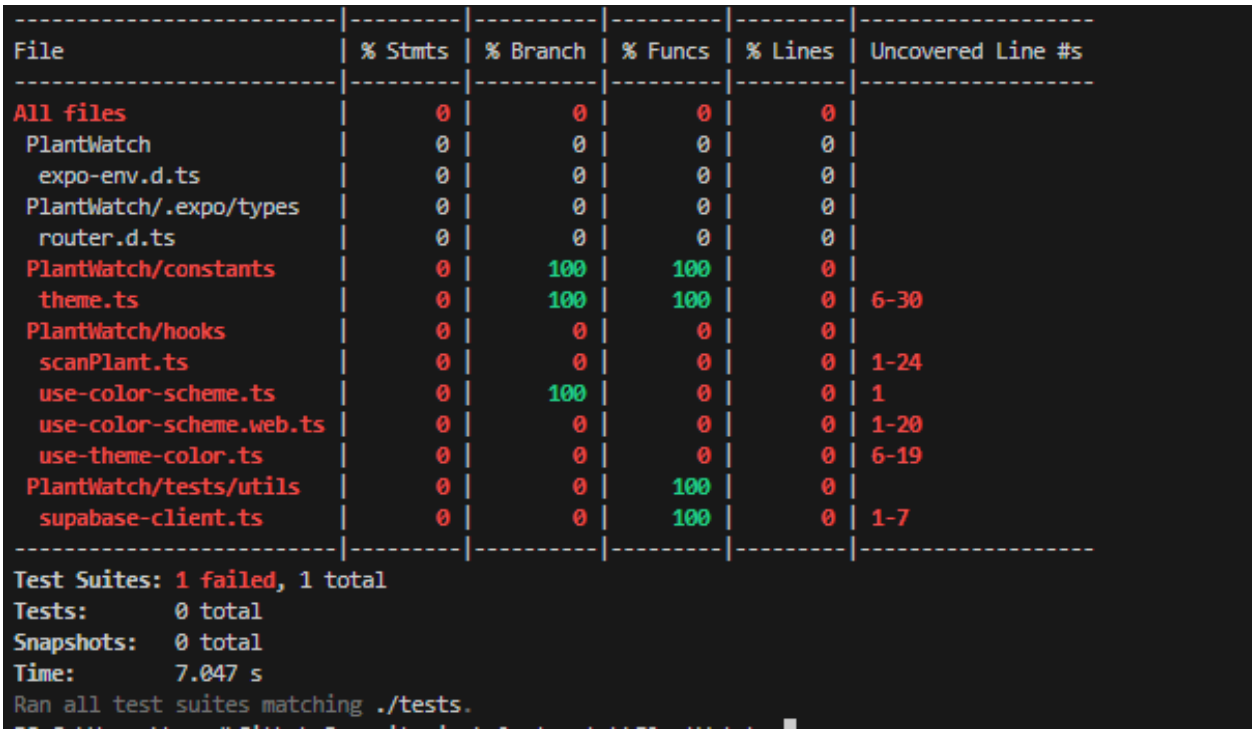
The plan for this iteration was 4 stories that addressed account credentials and plant scanning (S1, S4, S5, S19) for a total of 15 points. These features are core to the apps functionality and allows us to interact with many parts of the frameworks and services we are using.

7.2.2 Work Done

This iteration was a slow start. Most of S1 for a total of 5 points in iteration 1. Plants can be scanned and basic identifying information will be displayed to the user about what plant it is. The setup of the app was a large learning process, we do not have mobile development experience and have not used React, Expo, or Supabase before. Researching these different tools and going through examples took up lots of time in this iteration. We are still trying to figure out authenticating through Supabase as well, that has not been straightforward. I (Brenden) was the only one who completed work this iteration, Nishant is out of the country and I have not received word from him.

7.2.3 Testing Coverage

Figure 6 shows the testing coverage for iteration 1. Part of learning how to use Supabase was learning how to integrate tests with it. We only ended up creating one server-sided function and that function does not have a test currently, so we have no testing coverage. This coverage is not good enough, but is not as bad as it seems. Most of the work done in this iteration has been front end work and backend setup through the Supabase portal. The one function that needs to be tested is an API call and some basic text reformatting. The coverage report attached shows several front end files, we have not completely figured out configuring the test suite to pick up relevant files. Coverage will be increased quickly as the suite is correctly configured and the test for the function is implemented. Future commits will include tests since the testing will be set up.



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	0	0	0	0	
PlantWatch	0	0	0	0	
expo-env.d.ts	0	0	0	0	
PlantWatch/.expo/types	0	0	0	0	
router.d.ts	0	0	0	0	
PlantWatch/constants	0	100	100	0	
theme.ts	0	100	100	0	6-30
PlantWatch/hooks	0	0	0	0	
scanPlant.ts	0	0	0	0	1-24
use-color-scheme.ts	0	100	0	0	1
use-color-scheme.web.ts	0	0	0	0	1-20
use-theme-color.ts	0	0	0	0	6-19
PlantWatch/tests/utils	0	0	100	0	
supabase-client.ts	0	0	100	0	1-7

```
Test Suites: 1 failed, 1 total
Tests:       0 total
Snapshots:   0 total
Time:        7.047 s
Ran all test suites matching ./tests.
```

Figure 6: Iteration 1 test coverage report

7.2.4 Retroespective & Reflection

Brenden
The major challenge I had in this iteration was the unfamiliarity I had with the tools we decided to use. Supabase is a backend replacement and takes a good amount of reading/learning to understand. I have not worked with mobile before, and learning how to use React Native and Expo was also a long experience. I've gotten more comfortable with all the components of the project, but I did not give myself enough time to apply the concepts. Next iteration, I will begin working right away and steadily make progress towards completing stories. I also found that watching videos on how other set up their Supabase projects has been helpful, so I will do that prior to jumping into coding more often especially in this second iteration. Another thing I fell short on was commenting, so I will make sure to do that along the way this time as well. It will especially be helpful for understanding what I am doing, and cluing my partner into what I have done. I have learned way too much to detail here this iteration, so I will just summarize it as basics of React Native, Expo, and Supabase and usage of API keys.

7.3 Iteration/Sprint 2

7.3.1 Planning

[Which stories did you plan for this iteration/sprint. Add the total points for this plan. You can also explain the reason behind your planning, and what major feature(s) your team is focusing on delivering by completing these stories. You may use a table for a summary display of the planning, but elaborate in text more detail in your focus and feature plan.]

7.3.2 Work Done

[Which stories did you complete in this iteration/sprint. Which ones did you partially complete? Who worked on which story? You may elaborate in paragraph(s) to add more detail about the work done.]

7.3.3 Testing Coverage

[Testing is very important. Show your coverage here. Is this coverage good enough? Explain why you think so. Is it not good enough? Explain a plan to increase the coverage. You may also elaborate on why some artifacts do not undergo much testing. If the testing changed from the last iteration, explain the reasons.]

7.3.4 Retroespective & Reflection

[What were the pitfalls, challenges, and issues you had in this iteration? How can you address them to improve the process in the next iteration? Did anything not go according to plan? Why so and how to avoid the same mistake? Write a personal reflection on what you learned in this iteration (even if a small technical thing like Database storage).]

7.4 Iteration/Sprint 3

7.4.1 Planning

[Which stories did you plan for this iteration/sprint. Add the total points for this plan. You can also explain the reason behind your planning, and what major feature(s) your team is focusing on delivering by completing these stories. You may use a table for a summary display of the planning, but elaborate in text more detail in your focus and feature plan.]

7.4.2 Work Done

[Which stories did you complete in this iteration/sprint. Which ones did you partially complete? Who worked on which story? You may elaborate in paragraph(s) to add more detail about the work done.]

7.4.3 Testing Coverage

[Testing is very important. Show your coverage here. Is this coverage good enough? Explain why you think so. Is it not good enough? Explain a plan to increase the coverage. You may also elaborate on why some artifacts do not undergo much testing. If the testing changed from the last iteration, explain the reasons.]

7.4.4 Retroespective & Reflection

[What were the pitfalls, challenges, and issues you had in this iteration? How can you address them to improve the process in the next iteration? Did anything not go according to plan? Why so and how to avoid the same mistake? Write a personal reflection on what you learned in this iteration (even if a small technical thing like Database storage).]

7.5 Iteration/Sprint 4

[CS496 has 5 sprints. CS482 only has only 3 sprints (remove Iterations 4 and 5 from this doc if you are writing a doc for 482]

7.5.1 Planning

[Which stories did you plan for this iteration/sprint. Add the total points for this plan. You can also explain the reason behind your planning, and what major feature(s) your team is focusing on delivering by completing these stories. You may use a table for a summary display of the planning, but elaborate in text more detail in your focus and feature plan.]

7.5.2 Work Done

[Which stories did you complete in this iteration/sprint. Which ones did you partially complete? Who worked on which story? You may elaborate in paragraph(s) to add more detail about the work done.]

7.5.3 Testing Coverage

[Testing is very important. Show your coverage here. Is this coverage good enough? Explain why you think so. Is it not good enough? Explain a plan to increase the coverage. You may also elaborate on why some artifacts do not undergo much testing. If the testing changed from the last iteration, explain the reasons.]

7.5.4 Retroespective & Reflection

[What were the pitfalls, challenges, and issues you had in this iteration? How can you address them to improve the process in the next iteration? Did anything not go according to plan? Why so and how to avoid the same mistake? Write a personal reflection on what you learned in this iteration (even if a small technical thing like Database storage).]

7.6 Iteration/Sprint 5

7.6.1 Planning

[Which stories did you plan for this iteration/sprint. Add the total points for this plan. You can also explain the reason behind your planning, and what major feature(s) your team is focusing on delivering by completing these stories. You may use a table for a summary display of the planning, but elaborate in text more detail in your focus and feature plan.]

7.6.2 Work Done

[Which stories did you complete in this iteration/sprint. Which ones did you partially complete? Who worked on which story? You may elaborate in paragraph(s) to add more detail about the work done.]

7.6.3 Testing Coverage

[Testing is very important. Show your coverage here. Is this coverage good enough? Explain why you think so. Is it not good enough? Explain a plan to increase the coverage. You may also elaborate on why some artifacts do not undergo much testing. If the testing changed from the last iteration, explain the reasons.]

7.6.4 Retroespective & Reflection

[What were the pitfalls, challenges, and issues you had in this iteration? How can you address them to improve the process in the next iteration? Did anything not go according to plan? Why so and how to avoid the same mistake? Write a personal reflection on what you learned in this iteration (even if a small technical thing like Database storage).]

8 Final Remarks

8.1 Overall Progress

[Have you completed everything? If so, present evidence on how you brought value to your client, and the overall client satisfaction. Otherwise, estimate how much progress you done and how long it would take to finish this project. Be concrete about your progress, you know how many story points your software is, how many points you completed (this shows your progress). You also how many points your team delivers at each iteration, therefore you can estimate how many more iterations it would take to finish the leftover points (show the math).]

8.2 Project Reflection

[Your personal reflection on the project. What lessons did you learned. What would you have done differently? How can you do better work in future projects? You may write this as a team or per person (or both — if all your iterations were team reflections, then it would be better to write individual reflections here)]

Appendix

[Appendix section if needed]