

**Garden App**

**Report 1 Final**

**CSCI 441**

**GitHub URL:** [**https://github.com/Kirapants07/CSCI441\_VA\_Group1\_Spring2023\_GardenApp**](https://github.com/Kirapants07/CSCI441_VA_Group1_Spring2023_GardenApp)

**February 23, 2023**

**Group Kira: Daniel Dietrich, Gavin Cyr, Kira Luethe, Richard Williams, Todd Wood**

# Individual Contributions Breakdown

## Responsibility Matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Task** | **Daniel** | **Gavin** | **Kira** | **Richard** | **Todd** |
| Work Assignment, Glossary of Terms | 20% | 20% | 20% | 20% | 20% |
| Functional Requirement Specification, Use Cases (3a,3b,3c.i 3c.ii, 3c.iii, 3c.iv, 3d) | 100% |  |  |  |  |
| Business Goals Diagram, System Architecture (5e, 5f) and Project Size Estimation (6) |  | 100% |  |  |  |
| User Interface Graphics, User Interface Specification (4a, 4b) and Team Leader |  |  | 100% |  |  |
| Decomposition into Sub-Problems, Database Development, References |  |  |  | 100% |  |
| Customer Problem Statement, Plan of Work, System Architecture (5a,b,c) |  |  |  |  | 100% |

# Table of Contents

[Individual Contributions Breakdown 2](#_Toc127988641)

[Responsibility Matrix 2](#_Toc127988642)

[Table of Contents 3](#_Toc127988643)

[Work Assignment 5](#_Toc127988644)

[a. Team Profile (Members & Strengths) 5](#_Toc127988645)

[b. Sub-Teams 5](#_Toc127988646)

[Part 1: 6](#_Toc127988647)

[1. Customer Problem Statement 6](#_Toc127988648)

[a. Problem Statement 6](#_Toc127988649)

[b. Decomposition into Sub-Problems 9](#_Toc127988650)

[c. Glossary of Terms 9](#_Toc127988651)

[2. Goals, Requirements, and Analysis 10](#_Toc127988652)

[a. Business Goals 10](#_Toc127988653)

[b. Enumerated Functional Requirements 11](#_Toc127988654)

[c. Enumerated Nonfunctional Requirements 11](#_Toc127988655)

[d. User Interface Requirements 11](#_Toc127988656)

[I. User Interface Requirements Diagrams 13](#_Toc127988657)

[Part 2: 18](#_Toc127988658)

[3. Use Cases 18](#_Toc127988659)

[a. Stakeholders 18](#_Toc127988660)

[b. Actors and Goals 18](#_Toc127988661)

[c. Use Cases 18](#_Toc127988662)

[i. Casual Description 18](#_Toc127988663)

[ii. Use Case Diagram 19](#_Toc127988664)

[iii. Traceability Matrix 20](#_Toc127988665)

[iv. Fully-Dressed Description 21](#_Toc127988666)

[d. System Sequence Diagrams 27](#_Toc127988667)

[4. User Interface Specification 36](#_Toc127988668)

[a. Preliminary Design 36](#_Toc127988669)

[UC-1: Create a User Account 36](#_Toc127988670)

[UC-3: Filter by Zip Code 38](#_Toc127988671)

[UC-4: Search by Plant 40](#_Toc127988672)

[UC-5: Enter Planting Date 42](#_Toc127988673)

[b. User Effort Estimation 44](#_Toc127988674)

[UC-1: Create a User Account 44](#_Toc127988675)

[UC-3: Filter by Zip Code 44](#_Toc127988676)

[UC-4: Search by Plant 45](#_Toc127988677)

[UC-5: Enter Planting Date 45](#_Toc127988678)

[5. System Architecture 45](#_Toc127988679)

[a. Identifying Subsystems 45](#_Toc127988680)

[b. Architecture Styles 47](#_Toc127988681)

[c. Mapping Subsystems to Hardware 47](#_Toc127988682)

[d. Connectors and Network Protocols 48](#_Toc127988683)

[e. Global Control Flow 48](#_Toc127988684)

[i. Execution Orderness 48](#_Toc127988685)

[ii. Time Dependency 48](#_Toc127988686)

[f. Hardware Requirements 48](#_Toc127988687)

[6. Project size Estimation Based on use case points 49](#_Toc127988688)

[7. Plan of work 51](#_Toc127988689)

[a. Gantt Chart 51](#_Toc127988690)

[b. Product Ownership and Breakdown of Responsibilities 51](#_Toc127988691)

[8. References 52](#_Toc127988692)

[9. Summary of Changes 53](#_Toc127988693)

# Work Assignment

### Team Profile (Members & Strengths)

Daniel Dietrich has 3.5 years of collegiate education in the field of computer science. His programming experience includes PHP, SQL, JavaScript, HTML, CSS, Java, and C++. Past projects include construction of Device Inventory forms and an API using PHP, JavaScript, SQL and HTML.

Gavin Cyr has 3 years of coding experience in multiple coding languages through classes for a bachelor’s degree in computer science. He has coded multiple different projects as a student using the languages HTML, JavaScript, CSS, PHP, SQL, Java, and C++.

Kira Luethe has some experience coding in several languages, including JavaScript, PHP, Java, C++, CSS, SQL, and Python. She has built several web apps as a student, and she especially enjoys working on the graphic interface design of projects.

Richard Williams has 12 years of experience at a medical software company as a front-end development manager. Have 2 years of experience in designing and deploying a custom API for internal use. Bringing skills in HTML, CSS, JavaScript, PHP, MySQL, C#, AWS deployment, Java, and Agile Development Lifecycle.

Todd Wood has over 14 years' experience in management. Including over 5 years at the Director level. Additionally, he has nearly completed his BS in Computer Science from Fort Hays State University. Todd’s skills include programming and problem solving in C++, C, Java, Python, JavaScript, HTML and SQL. He has built several applications including an author’s webpage and a text-based memory game. Todd’s strengths are data calculation and manipulation.

### Sub-Teams

Work on this project has been divided into two teams, one for the front-end development of the website, and one for the back-end development of the API, database, and user authentication. The team lead, Kira Luethe, will organize communication between group members and the instructor, and ensure timely completion of project stages.

The first team, Daniel Dietrich and Richard Williams, will build the backend components, including the API, database design and implementation, and user authentication.

The second team, Gavin Cyr, Kira Luethe, and Todd Wood, will design and implement the website. This encompasses user interface and graphic design, and features, which include planting zone selection, and date trackers.

# Part 1:

## Customer Problem Statement

### Problem Statement

Gardening is a universal way to grow your own food, and live healthier and more sustainably. For many it is a hobby, and for some it is a way to grow much needed food. Regardless of the reason behind it, many Americans have some form of garden, whether it be a small box on a porch or a large field in the backyard.

Growing a garden can be a particularly challenging proposition. It is easy to throw seeds in the soil and hope they come up, but planning a garden and tracking the plants is a more challenging prospect. Many people are busy and do not have time to document everything they do, or research what type of seeds a person should use for their region or figure out when they should plant those seeds. After planting, people often do not know when they can expect those plants to start to grow, or when they will be ready to harvest.

People try various methods to keep track of the garden. They will write it all on a piece of paper, or they will place markers in the garden. Some will just not bother and will hope for the best. None of these solutions give the gardener the tools to plant the best possible garden. A better solution would be an application that can help the average gardener research their plants, plan their garden and gain information without a significant amount of writing, internet sleuthing and data collection.

How does the average person determine what seeds they will plant? Many times, the person will go to the store and look through racks of seeds and try to figure out what might work. They then plant them in hopes that the plants will work well. Or they go to the local nursery hoping to find something that will work well in their region. Ideally an application would allow the person to sit down and enter their zip code and from their computer or mobile device browser be able to search or look through a list of plants that work well in their region.

But the person does not want a general database they have to refer to repeatedly. Nobody wants to have to re-look up their plants or their garden over and over. Instead, an application should allow the person to create an account and save the gardeners' plant choices so that they can easily retrieve the information about those plants by simply logging back in whenever they desire.

What about after the person goes to the store and finds the seeds or plants, they want to put in the ground? When do they put them in the ground? Do they plant them right away or do they wait? The package often tells them when it is best to plant, but that is often a challenge for the average gardener to keep track of that information. The application should have information about plants already loaded and allow the gardener to see when the plants they have chosen are best planted. It should tell them the time of year and roughly when the best time is to put them in the ground.

Once the gardener plants their garden they often forget what they did and when. The gardeners will find themselves going out periodically to check and see if their plants have started to grow. But when can that person determine that this seedling failed? Has the seed not germinated? Is it not growing? For the average gardener this is a guessing game. The application should allow the gardener to enter a date for planting. The application should store the planting dates and save them until the gardener decides that they no longer need that information and chooses to delete it.

Additionally, the application should be able to do the rough estimates and tell the gardener when to expect germination or when they can expect to see the newly growing plants. This way the gardener can determine when something has gone wrong and may choose to replant or choose a different planting.

The application should also allow the gardener to see roughly when the plantings will have grown to full maturity. While growing times are not an exact science the application should at least give the gardener a generalized window of when they can expect to reap the rewards of their hard work.

What about next time? When the garden has run its course and it is time to put up the hoe for the winter, will the plantings that the gardener just used be available or will they have to start over next year? The application should give the gardener the options to start over or to save their plantings to be able to use them again next year. That way if the garden worked well and they enjoyed their crops the gardener can simply log in the next season and see what they did before and do it again.   
 But the gardener does not want to spend time working on a database or dealing with a lot of text documents. That will be as much work as just doing it on paper. The application should sport a graphical user interface that is easy to use and understand.

It should include an intuitive user login screen so that the gardener can quickly get to their account. The menu should be simple with only relevant options. Once logged in the interface should be simple with easy use areas for researching their plants and adding them to their own list. The plant list should be instantly accessible, with planting dates, germination dates, and harvest dates listed in an easy to read and understandable format. The gardener should not have to dig to find all their information. The application should populate the gardener’s active list on log in so that they do not have to dig around to find their current information.

Future iterations of the application interface should include graphics of the plants, and easy to use charts and graphs including US zones, and growth charts.

There should be ongoing improvements in this application so that there is additional functionality in the future. For example, the application should eventually let the gardener know which plantings complement other plantings, also known as companion planting. For example, what types of tomatoes grow best with other types of tomatoes? The functionality should extend to making suggestions for the gardener to consider.

Eventually the application should migrate to a handheld device or cell phone app. This will allow the gardener to have the information in their pocket and be able to add or remove things in real time while they are building their garden.

In the future the application should include a garden planner graphical garden that will allow the user to not only pick their plantings, but also decide where in the garden they should go. This will allow the gardener to see their crops, the layout, and where they placed each planting.

Finally, the application should eventually move to include offline functionality. In the ideal world the gardener should not need to be connected to the internet to see their plantings and the information regarding those plantings. Allowing local storage of the user information will improve the ability of the gardener to have the info on hand even when there is no internet in the garden.

### Decomposition into Sub-Problems

The larger problem was first broken down into its two pieces, the website and the API. The website features are further broken down by their importance and the API into two parts, the public data domain and the private user data / user access authorization.

### Glossary of Terms

**Application** – Website built to display information to and be interacted with by the end user.

**Companion planting** – growing plants together where one or both plants benefit the other in ways that include pest control, pollination, attracting beneficial insects, or creating other favorable conditions.

**Database** – Online repository of information

**Gardener** – User of the application

**Germination** – Point in time from initial planting of seed to the time it sprouts from the seed.

**Germination date** – Time from initial sowing of seed to germination.

**GUI (Graphic User Interface)** - The web page, or app page that users will use in order to interact with the software.

**Harvest dates** – The date at which a plant is fully developed and is ready to be harvested.

**Local Storage** – Data storage located physically on the device in question.

**Mobile friendly** - an application that can be used across devices, such as a computer, tablet, and cell phone, without losing functionality.

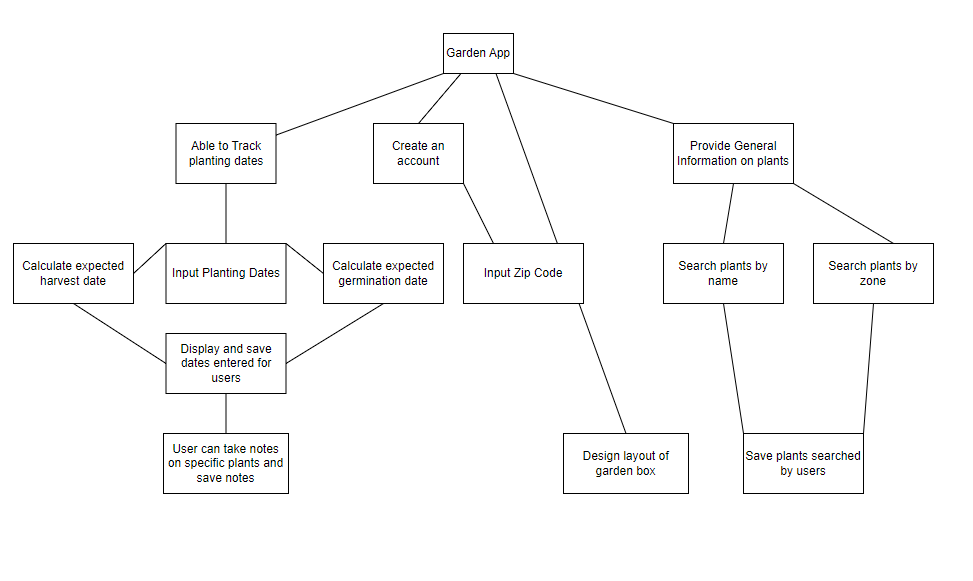
**Planting date** – Date of initial sow of seed into the ground or source of dirt.

**Seedling** – A seed after planting but before germination.

**USDA Plant Hardiness zones** – A collection of geographic areas separated by average annual minimal temperature.

## Goals, Requirements, and Analysis

### Business Goals



### Enumerated Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Identifier:** | **Priority:** | **Requirement:** |
| REQ-1 | 5 | The user can create an account. |
| REQ-2 | 1 | The application shall allow user to take notes on their searches for personal garden. |
| REQ-3 | 5 | The application should allow users to input a zip code. |
| REQ-4 | 5 | The user can search for a plant by name. |
| REQ-5 | 3 | When selecting a plant by name, the application displays plant information. |
| REQ-6 | 4 | The application shall filter plant by zone. |
| REQ-7 | 5 | The application should display results based on filtering criteria. |
| REQ-8 | 5 | The user can input the date of planting. |
| REQ-9 | 3 | The application shall calculate approximate germination date. |
| REQ-10 | 4 | The application shall calculate approximate harvest date. |
| REQ-11 | 4 | The application should display calculated dates. |
| REQ-12 | 5 | The application should save user’s entered planting dates. |
| REQ-13 | 5 | The application should display all plants for a given user. |
| REQ-14 | 1 | The application shall allow users to design layout of garden box. |

### Enumerated Nonfunctional Requirements

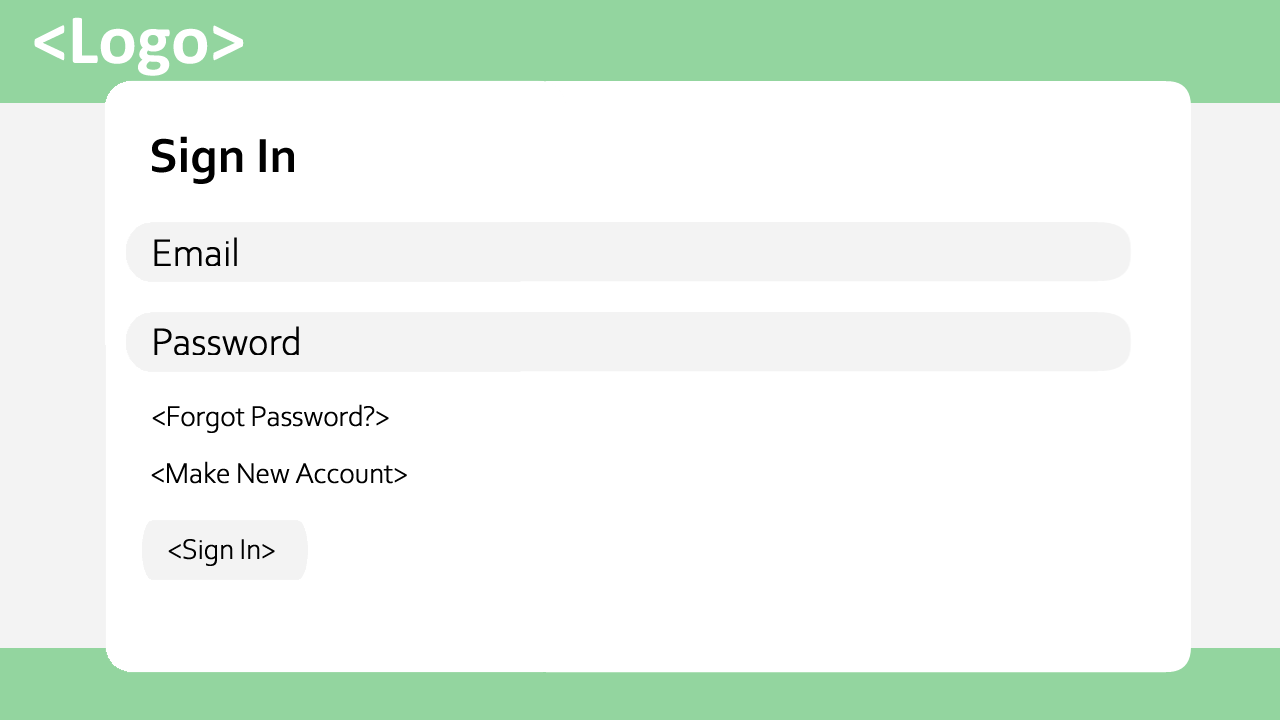
|  |  |  |
| --- | --- | --- |
| **Identifier:** | **Priority:** | **Requirement:** |
| REQ-15 | 5 | The application should require online connectivity. |
| REQ-16 | 2 | The application shall display relevant image of plant. |
| REQ-17 | 1 | The application shall store a local cache of user data. |
| REQ-18 | 3 | The application shall use Browser assisted location detection. |
| REQ-19 | 5 | The application shall reliably store user data with a loss of user data less than 10%. |
| REQ-20 | 4 | The application shall be easy for users to use. |

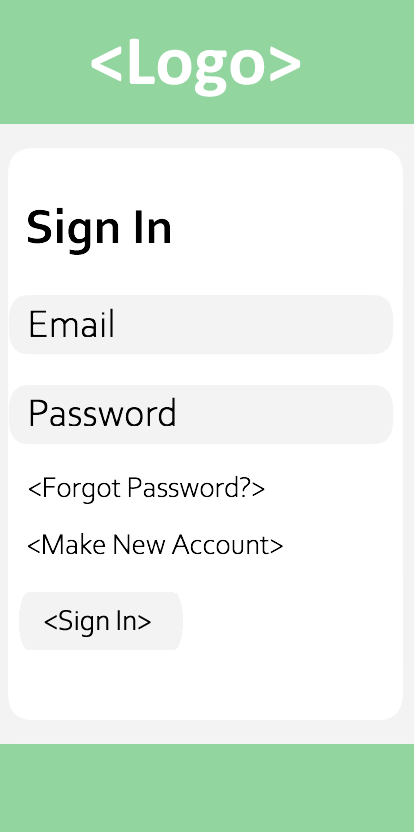
### User Interface Requirements

|  |  |  |
| --- | --- | --- |
| **Identifier:** | **Priority:** | **Requirement:** |
| REQ-21 | 5 | The application should require the user to enter a valid email and password. |
| REQ-22 | 5 | If the user forgets the account password, the applications UI should show the reset password features in the login page. |
| REQ-23 | 4 | The application shall require the user to first log in using appropriate login credentials to edit user information. |
| REQ-24 | 5 | If the user does not log in to a valid account, the application should allow the user to access basic search features that do not require saved user information. |
| REQ-25 | 2 | The application shall allow users to disable their account. |

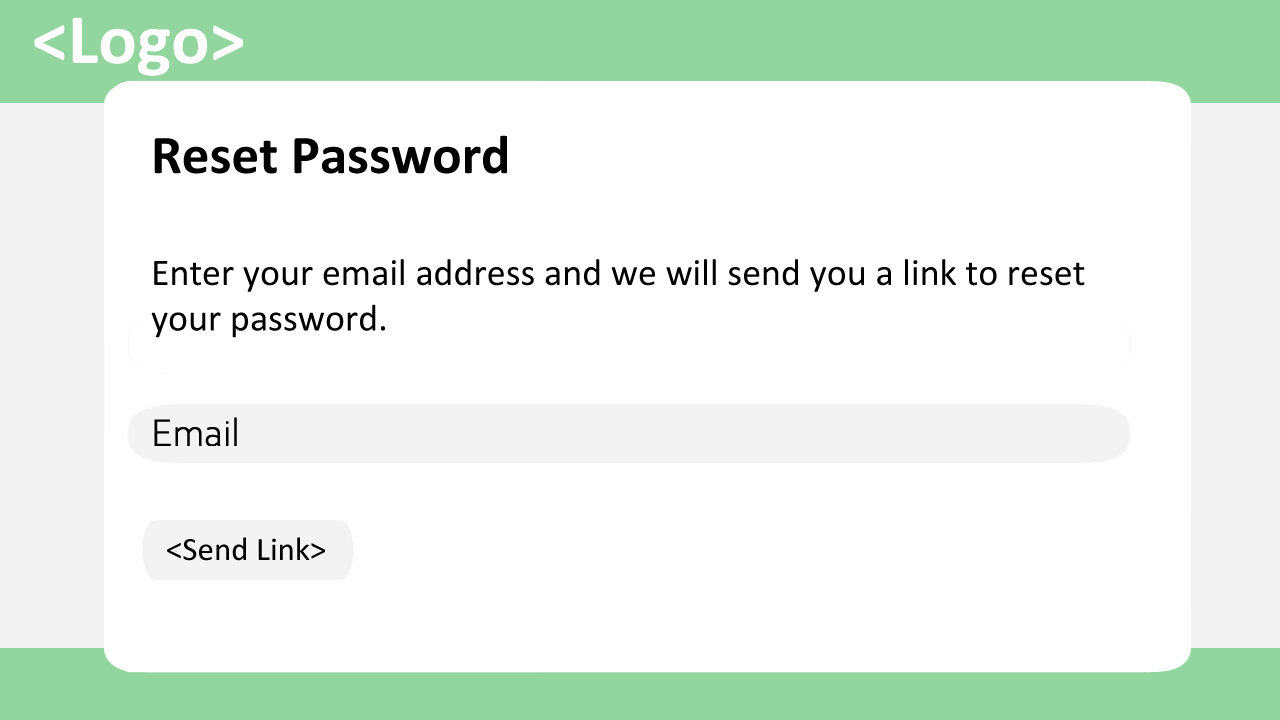
#### User Interface Requirements Diagrams

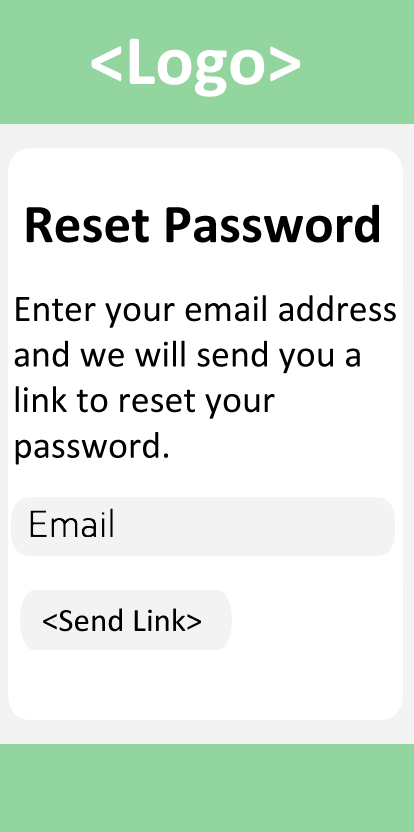
REQ-21



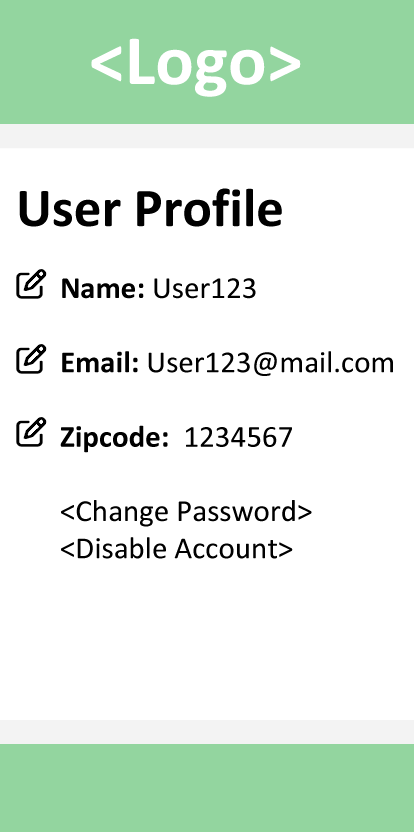
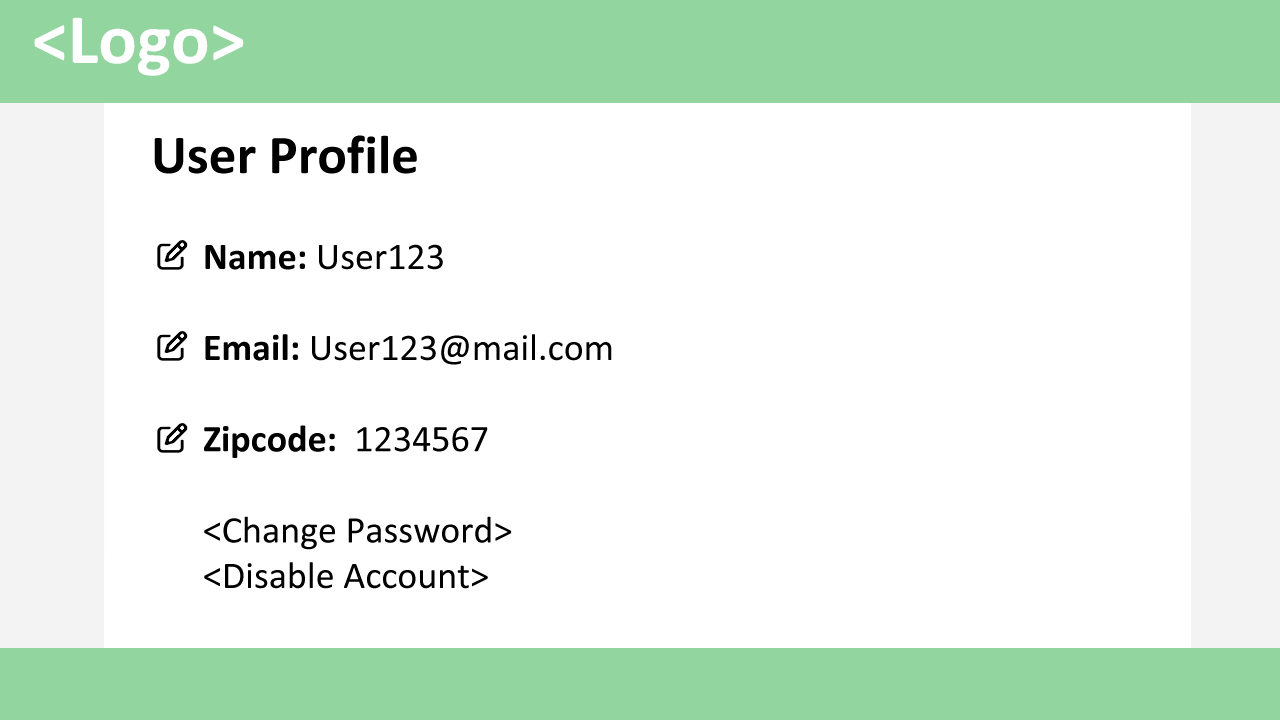


REQ-22

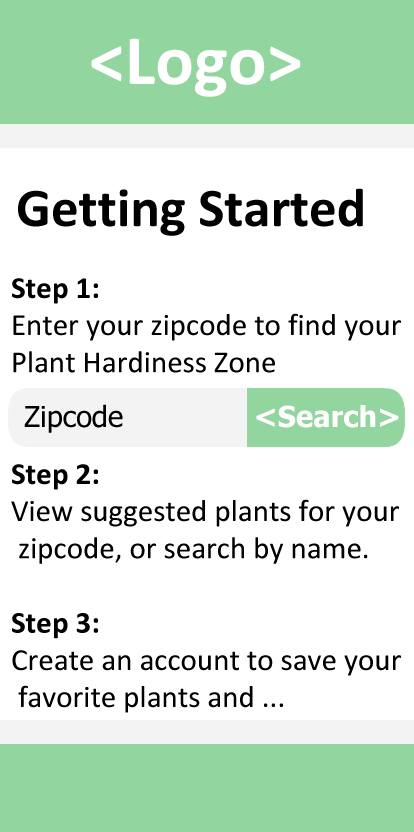
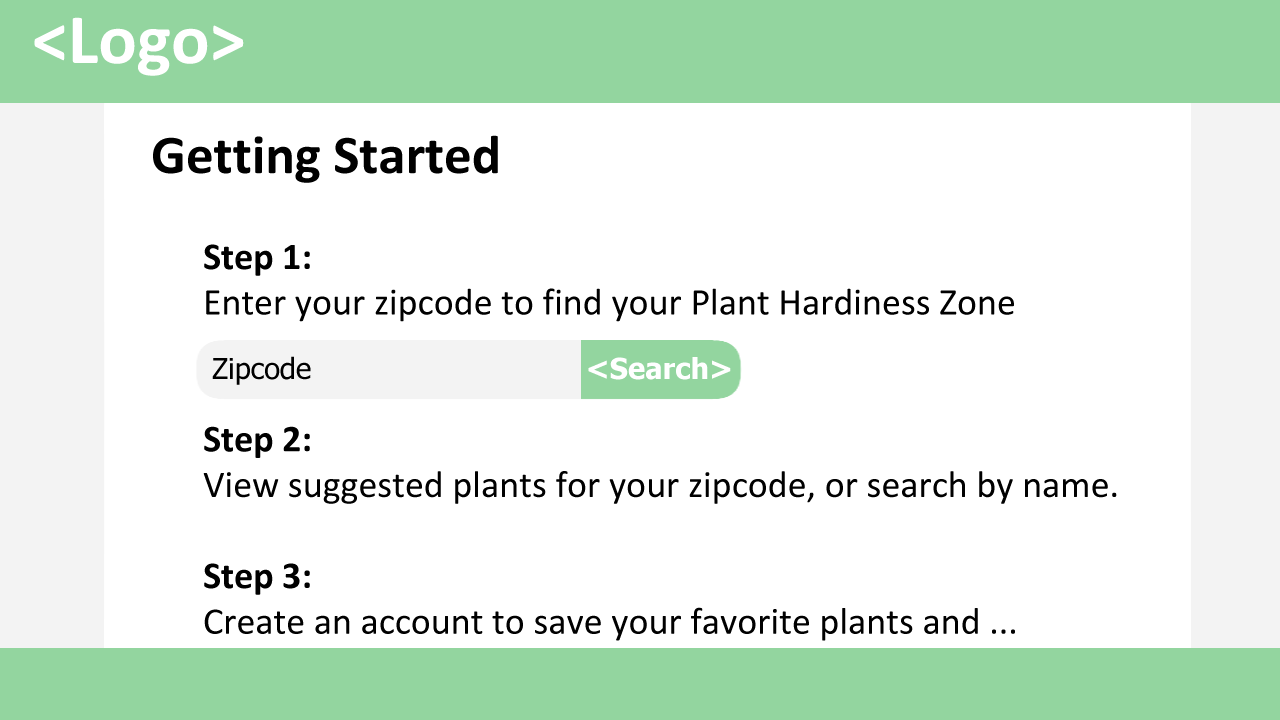




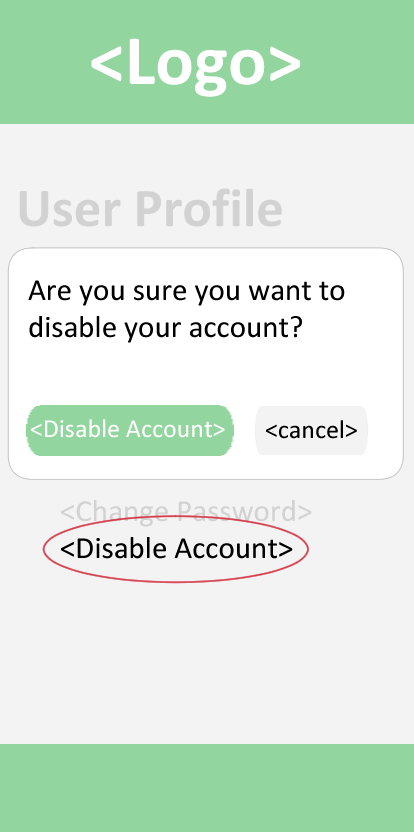
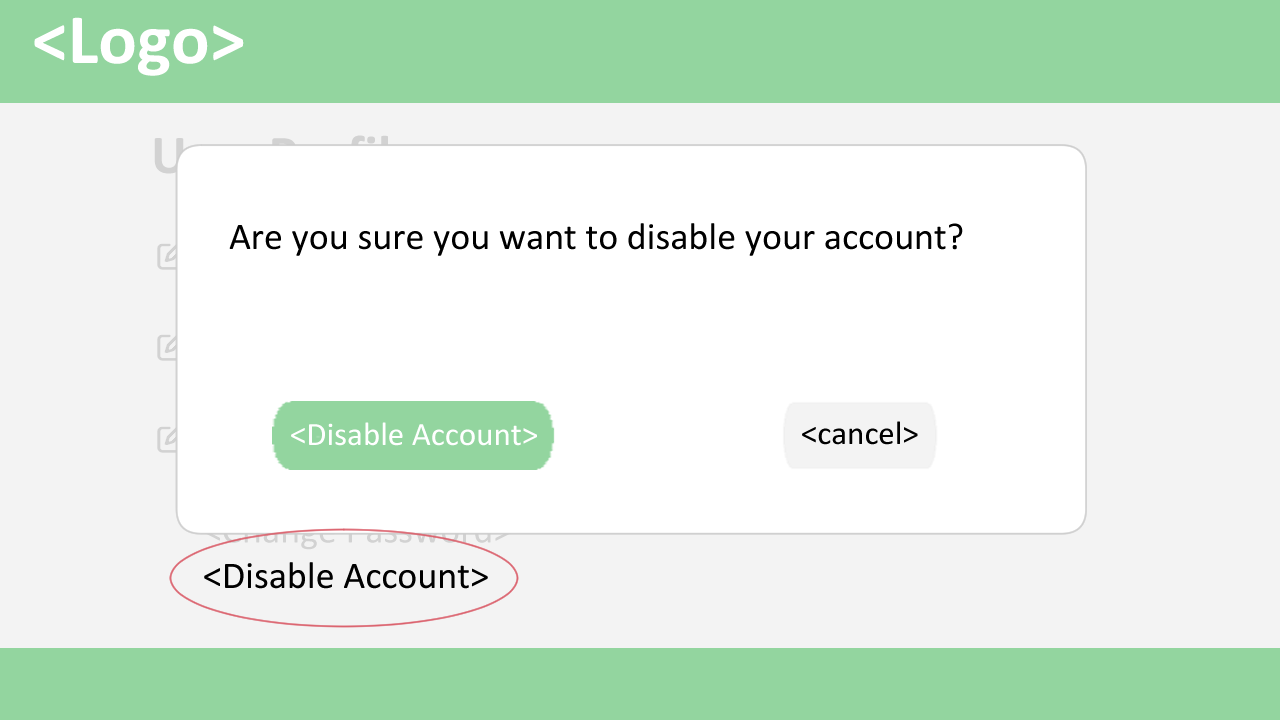
REQ-23



REQ-24



REQ-25



# Part 2:

## Use Cases

### ***a. Stakeholders***

**Development Team** – The interest of the Development Team in this project is primarily focused on the software engineering process and on the performance and functionality of the end project.

**Course Instructor** - The interest of the Course Instructor in this project is primarily focused on the collaboration and development process followed by the Development Team. The Course Instructor also has interest in the end result.

**End Users** – The interest of the End Users in this project is primarily focused on the functionality and ease-of-use of the final product.

### ***Actors and Goals***

**User:**

|  |  |  |
| --- | --- | --- |
| Actor | Goal | Use Case Name |
| User | To create a new user account. | Create User Account (UC-1) |
| User | To create and save note to personal garden. | Take Note (UC-2) |
| User | To filter plant data by Hardiness Zone using Zip Code. | Filter by Zip Code (UC-3) |
| User | To search for planting information on a specific plant. | Search by Plant (UC-4) |
| User | To enter and save a planting date, calculate a germination date and calculate a harvest date. | Enter Planting Date (UC-5) |
| User | To design and save a graphic of their personal Garden Box. | Design Garden Box (UC-6) |
| User | To log in to the system and access their stored data. | Log In (UC-7) |
| User | To reset the password associated with their user account. | Password Reset (UC-8) |
| User | To update the stored user information. | Update User Information (UC-9) |
| User | To disable their user account and remove their saved data from the database. | Disable User Account (UC-10) |

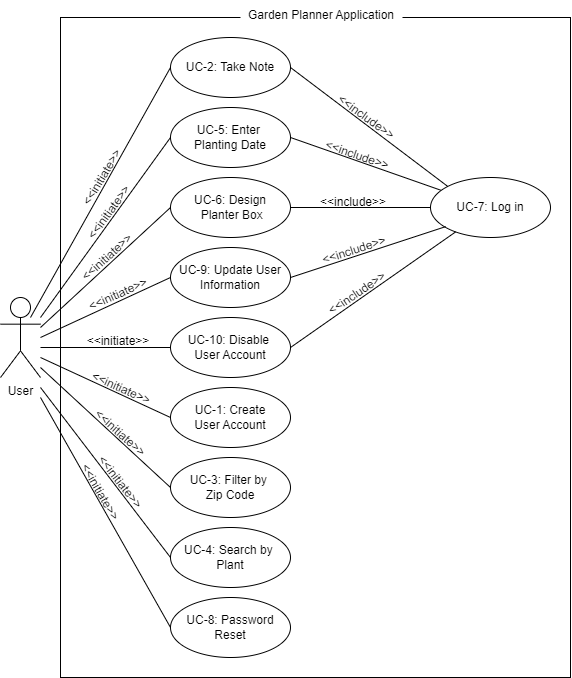
**Database** – Primarily a participating actor that acts as storage for data used and modified by the System and by User.

### ***Use Cases***

#### Casual Description

|  |  |  |
| --- | --- | --- |
| Use Case | Description | Requirements |
| UC-1: Create User Account | Allows users to create an account, setting a valid username and password combination. | REQ-1, REQ-15 |
| UC-2: Take Note | Allows user to take a note on their personal garden. | REQ-2 |
| UC-3: Filter by Zip Code | Allows user to filter plants by zone by entering a valid Zip Code. | REQ-3, REQ-6, REQ-7, REQ-17 |
| UC-4: Search by Plant | Allows user to search for information on a plant by entering a valid plant name. | REQ-4, REQ-5, REQ-7, REQ-16 |
| UC-5: Enter Planting Date | Allows user to enter planting date to be stored in the application database. Entering a date for a given plant calculates the approximate germination and harvest date and display them for the user. | REQ-8, REQ-9, REQ-10, REQ-11, REQ-12, REQ-13, REQ-17, REQ-19 |
| UC-6: Design Garden Box | Allows user to design the layout of a garden box. | REQ-14 |
| UC-7: Log In | Allows user to sign in to account using valid credentials. | REQ-21 |
| UC-8: Password Reset | Allows user to reset the password for their account. | REQ-22, REQ-20 |
| UC-9: Update User Information | Allows user to update user information once they have signed in using valid credentials. | REQ-21, REQ-23 |
| UC-10: Disable User Account | Allows user to disable their account, removing all their stored data from the database. | REQ-25 |

#### Use Case Diagram



#### iii. Traceability Matrix

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **REQ ID** | **PW** | UC-1 | UC-2 | UC-3 | UC-4 | UC-5 | UC-6 | UC-7 | UC-8 | UC-9 | UC-10 |
| REQ-1 | 5 | X |  |  |  |  |  |  |  |  |  |
| REQ-2 | 1 |  | X |  |  |  |  |  |  |  |  |
| REQ-3 | 5 |  |  | X |  |  |  |  |  |  |  |
| REQ-4 | 5 |  |  |  | X |  |  |  |  |  |  |
| REQ-5 | 3 |  |  |  | X |  |  |  |  |  |  |
| REQ-6 | 4 |  |  | X |  |  |  |  |  |  |  |
| REQ-7 | 5 |  |  | X | X |  |  |  |  |  |  |
| REQ-8 | 5 |  |  |  |  | X |  |  |  |  |  |
| REQ-9 | 3 |  |  |  |  | X |  |  |  |  |  |
| REQ-10 | 4 |  |  |  |  | X |  |  |  |  |  |
| REQ-11 | 4 |  |  |  |  | X |  |  |  |  |  |
| REQ-12 | 5 |  |  |  |  | X |  |  |  |  |  |
| REQ-13 | 5 |  |  |  |  | X |  |  |  |  |  |
| REQ-14 | 1 |  |  |  |  |  | X |  |  |  |  |
| REQ-15 | 5 | X |  |  |  |  |  |  |  |  |  |
| REQ-16 | 2 |  |  |  |  |  |  |  |  |  |  |
| REQ-17 | 1 |  |  | X | X | X |  |  |  |  |  |
| REQ-18 | 3 |  |  |  |  |  |  |  |  |  |  |
| REQ-19 | 5 |  |  |  |  | X |  |  |  |  |  |
| REQ-20 | 4 |  |  |  |  |  |  |  | X |  |  |
| REQ-21 | 5 |  |  |  |  |  |  | X |  | X |  |
| REQ-22 | 5 |  |  |  |  |  |  |  | X |  |  |
| REQ-23 | 4 |  |  |  |  |  |  |  |  | X |  |
| REQ-24 | 5 |  |  |  |  |  |  |  |  |  |  |
| REQ-25 | 2 |  |  |  |  |  |  |  |  |  | X |
| **MAX PW** | | 5 | 1 | 5 | 5 | 5 | 1 | 5 | 5 | 5 | 2 |
| **TOTAL PW** | | 10 | 1 | 15 | 14 | 32 | 1 | 5 | 9 | 9 | 2 |

#### iv. Fully-Dressed Description

|  |  |
| --- | --- |
| **Use Case UC-1:** | **Create User Account** |
| **Related Requirements:** | REQ-1, REQ-15 |
| **Initiating Actor:** | User |
| **Actor’s Goal:** | To create a user account with |
| **Participating Actors:** | Database |
| **Preconditions:** | User has accessed the Sign In page for the website via a device with internet connectivity, such as a computer or smartphone. |
| **Postconditions:** | New user account is created and stored in the database. |
| **Flow of Events for Main Success Scenario:** | |
| ® | 1. **User** selects the on-screen “Make an Account” option |
| ¬ | 1. **System** displays a page on which the user can enter the email, password, name and zip code that they would like to use for the account. |
| ® | 1. **User** enters email, password, name and zip code in the appropriate fields and selects the on-screen “Create” option. |
| ¬ | 1. **System** (a) verifies in Database that the entered zip code is valid; (b) saves the new account data in **Database** and signals account creation. |
| **Flow of Events for Extensions (Alternate Scenarios):**  4a. System is unable to filter by appropriate Hardiness Zone. | |
| ¬ | 1. **System** displays page with message to **User** stating that the system was unable to filter data and to ensure that the entered zip code is valid. |

|  |  |
| --- | --- |
| **Use Case UC-2:** | **Take Note** |
| **Related Requirements:** | REQ-2 |
| **Initiating Actor:** | User |
| **Actor’s Goal:** | To make a note on their specific garden on their user account. |
| **Participating Actors:** | Database |
| **Preconditions:** | User is currently logged in and is currently on the page showing their saved garden data. |
| **Postconditions:** | User’s custom note is saved to the database and displayed on their garden information page. |
| **Flow of Events for Main Success Scenario:** | |
| ® | 1. **User** selects enters desired information into on-screen text entry field and selects the “Add Note” option. |
| ¬ | 1. **System** stores the new note in **Database** and updates the garden information page to include the newly added note. |

|  |  |
| --- | --- |
| **Use Case UC-3:** | **Filter by Zip Code** |
| **Related Requirements:** | REQ-3, REQ-6, REQ-7, REQ-17 |
| **Initiating Actor:** | User |
| **Actor’s Goal:** | To filter the plant data by Hardiness Zone by entering a valid US Zip Code. |
| **Participating Actors:** | Database |
| **Preconditions:** | User has accessed the Plant data search page using a device with internet connectivity, such as a computer or smartphone. |
| **Postconditions:** | The data page is updated with data filtered by the valid zip code entered by the user. |
| **Flow of Events for Main Success Scenario:** | |
| ® | 1. **User** enters a valid zip code in the appropriate text entry field and selects the “Filter” option. |
| ¬ | 1. **System** (a) determines the appropriate Hardiness Zone based on the entered zip code; (b) displays an updated data search page with information filtered by the appropriate hardiness zone. |
| **Flow of Events for Extensions (Alternate Scenarios):** | |
| 2a. System is unable to filter by appropriate Hardiness Zone. | |
| ¬ | 1. **System** displays page with message to **User** stating that the system was unable to filter data and to ensure that the entered zip code is valid. |

|  |  |
| --- | --- |
| **Use Case UC-4:** | **Search by Plant** |
| **Related Requirements:** | REQ-4, REQ-5, REQ-7, REQ-16 |
| **Initiating Actor:** | User |
| **Actor’s Goal:** | To search for data on a specific plant. |
| **Participating Actors:** | Database |
| **Preconditions:** | User has accessed the Plant data search page using a device with internet connectivity, such as a computer or smartphone. |
| **Postconditions:** | The data page is updated with data on the plant specified in the search criteria. |
| **Flow of Events for Main Success Scenario:** | |
| ® | 1. **User** enters a valid plant name in the search field and selects the on-screen “Search” option. |
| ¬ | 1. **System** (a) searches **Database** based on the entered plant name; (b) displays an updated data search page with information on the specified plant. |
| **Flow of Events for Extensions (Alternate Scenarios):** | |
| 2a. System is unable to filter by submitted plant name. | |
| ¬ | 1. **System** displays page with message to **User** stating that the system was unable to filter data and to ensure that the entered plant name is valid. |

|  |  |
| --- | --- |
| **Use Case UC-5:** | **Enter Planting Date** |
| **Related Requirements:** | REQ-8, REQ-9, REQ-10, REQ-11, REQ-12, REQ-13, REQ-17, REQ-19 |
| **Initiating Actor:** | User |
| **Actor’s Goal:** | To enter the planting date for a specified plant. |
| **Participating Actors:** | Database |
| **Preconditions:** | * User has logged in with valid user credentials. * User has accessed the Garden Management page using a device with internet connectivity, such as a computer or smartphone. * User has entered a valid zip code and a Hardiness Zone has been associated with the User’s account. |
| **Postconditions:** | Planting date, Germination date and Harvest date for the specified plant have been saved to the database for the user’s account and are displayed on the Garden Management page. |
| **Flow of Events for Main Success Scenario:** | |
| ® | 1. **User** enters a valid plant name in the Plant Name text entry field and a valid date in the Planting Date text entry field and selects the “Add Plant” option. |
| ¬ | 1. **System** (a) searches the database for information based on the entered plant name; (b) calculates the Germination date and Harvest date based on data for specified plant; (c) saves the Plant Name, Planting Date, Germination date and Harvest Date for the entry to **Database** for the user account; (d) displays the Garden Management page with all added plants for the current user along with accompanying dates. |
| **Flow of Events for Extensions (Alternate Scenarios):** | |
| 2a. System is unable retrieve data based on entered Plant Name. | |
| ¬ | 1. **System** displays Garden Management page with message to **User** stating that the system was unable to create entry and to ensure that the entered plant name is valid, followed by the other contents of the page. |

|  |  |
| --- | --- |
| **Use Case UC-6:** | **Design Garden Box** |
| **Related Requirements:** | REQ-14 |
| **Initiating Actor:** | User |
| **Actor’s Goal:** | To create a visual representation of a Garden Box. |
| **Participating Actors:** | Database |
| **Preconditions:** | * User has logged in with valid user credentials. * User has accessed the Garden Management page using a device with internet connectivity, such as a computer or smartphone. |
| **Postconditions:** | A graphic displaying the Garden Box design has been stored to the database for the user account and is displayed when the user accesses the Garden Management page. |
| **Flow of Events for Main Success Scenario:** | |
| ® | 1. **User** selects the “Design Garden Box” option on the Garden Management Page |
| ¬ | 1. **System** displays the interface used to design the Garden box Graphic. |
| ® | 1. **User** designs Garden Box using interface and selects Save Design option. |
| ¬ | 1. **System** (a) saves the graphic in **Database** for the user data; updates the Garden Management page to display the graphic. |

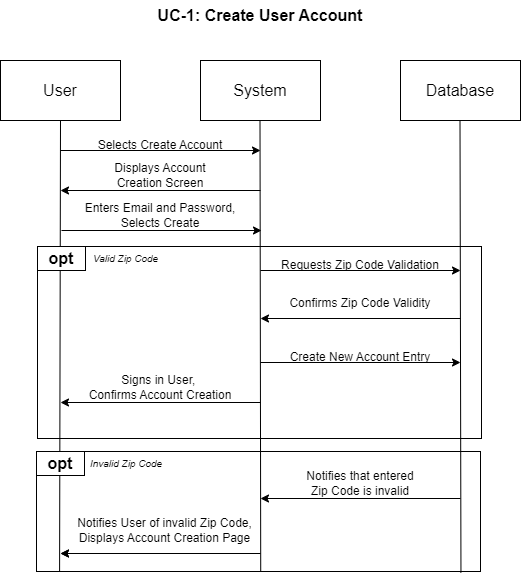
|  |  |
| --- | --- |
| **Use Case UC-7:** | **Log In** |
| **Related Requirements:** | REQ-21 |
| **Initiating Actor:** | User |
| **Actor’s Goal:** | To log in to system using valid user credentials. |
| **Participating Actors:** | Database |
| **Preconditions:** | * User has accessed the Main page using a device with internet connectivity, such as a computer or smartphone. * User has previously created an account using an email and password combination. |
| **Postconditions:** | User is signed into system and the main page is updated and the user’s stored information is accessible via the Garden Management Page. |
| **Flow of Events for Main Success Scenario:** | |
| ® | 1. **User** selects the on-screen “Sign In” option on the main page. |
| ¬ | 1. **System** displays the Sign In page with an Email text entry field, a password text entry field, a “Forgot Password” hyperlink, a “Make New Account” hyperlink and a “Sign In” button. |
| ® | 1. **User** enters a valid email and password into the appropriate fields and selects the “Sign In” button |
| ¬ | 1. **System** (a) validates user credentials from **Database** of user accounts; (b) displays the Garden Management page to **User**. |
| **Flow of Events for Extensions (Alternate Scenarios):** | |
| 4a. Invalid Email/Password entered – Unable to validate User. | |
| ¬ | 1. **System** displays the Log In page with message to **User** stating that an invalid email or password was entered, followed by the Log In form. |

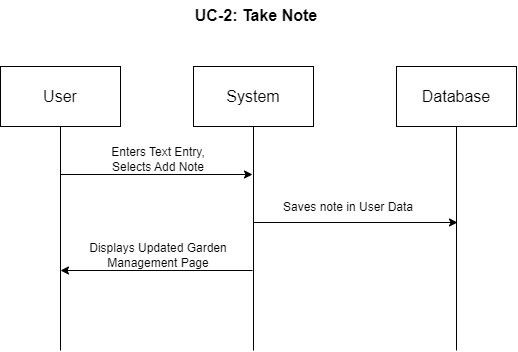
|  |  |
| --- | --- |
| **Use Case UC-8:** | **Password Reset** |
| **Related Requirements:** | REQ-22, REQ-20 |
| **Initiating Actor:** | User |
| **Actor’s Goal:** | To reset the password associated with their account. |
| **Participating Actors:** | Database |
| **Preconditions:** | * User has accessed the Sign In page using a device with internet connectivity, such as a computer or smartphone. * User has previously created an account using an email and password combination. |
| **Postconditions:** | The password associated with the given account has been changed and updated in the database. |
| **Flow of Events for Main Success Scenario:** | |
| ® | 1. **User** selects the on-screen “Forgot Password” option on the Sign In page. |
| ¬ | 1. **System** displays the Reset Password page with an Email text entry field and a “Send Link” button. |
| ® | 1. **User** enters a valid email associated with the account and selects the “Send Link” button |
| ¬ | 1. **System** (a) validates user email from **Database** of user accounts; (b) sends an email to the provided email address containing a link to reset the user password. |
| ® | 1. **User** (a) uses the link provided in the email to navigate to the Set Password page containing a New Password text entry field, a Confirm Password text entry field, and an “Update Password” button; (b) enters the new password into the New Password and Confirm Password fields and select the “Update Password” Button. |
| ¬ | 1. **System** updates the password associated with the user email in **Database** and displays the Sign In page. |
| **Flow of Events for Extensions (Alternate Scenarios):** | |
| 4a. Invalid Email – Unable to validate User. | |
| ¬ | 1. **System** displays the Reset Password page with message to **User** stating that an invalid email was entered, followed by the Reset Password form. |

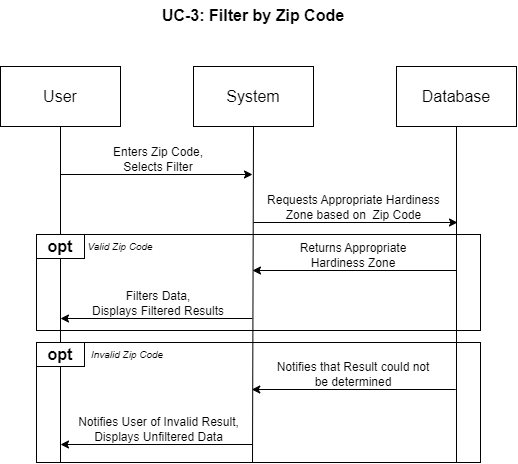
|  |  |
| --- | --- |
| **Use Case UC-9:** | **Update User Information** |
| **Related Requirements:** | REQ-21, REQ-23 |
| **Initiating Actor:** | User |
| **Actor’s Goal:** | To update user information such as Name, Email, Zip Code and Password. |
| **Participating Actors:** | Database |
| **Preconditions:** | * User has logged in with valid credentials. * User has accessed the User Profile page using a device with internet connectivity, such as a computer or smartphone. |
| **Postconditions:** | The edited user information field is updated on the User Profile page and in the database. |
| **Flow of Events for Main Success Scenario:** | |
| ® | 1. User selects edit icon for the information that they would like to update. |
| ¬ | 1. System displays a prompt box to user containing a text entry field for the updated information. |
| ® | 1. User enters the updated information into the text entry field and selects the “Update” option. |
| ¬ | 1. System updates the corresponding information in **Database** for the user to reflect the change, then displays and updated User Profile page to reflect the change made. |
| **Flow of Events for Extensions (Alternate Scenarios):** | |
| 3a. User selects the “Cancel” option. | |
| ¬ | 1. **System** does not change any user data and displays the User Profile page. |

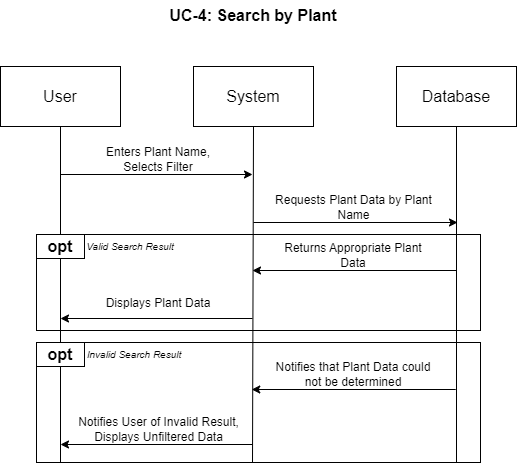
|  |  |
| --- | --- |
| **Use Case UC-10:** | **Disable User Account** |
| **Related Requirements:** | REQ-25 |
| **Initiating Actor:** | User |
| **Actor’s Goal:** | To disable the User’s account and remove all related data from database. |
| **Participating Actors:** | N/A |
| **Preconditions:** | * User has logged in with valid credentials. * User has accessed the User Profile page using a device with internet connectivity, such as a computer or smartphone. |
| **Postconditions:** | The User account credentials and all data related to the account have been removed from the database. |
| **Flow of Events for Main Success Scenario:** | |
| ® | 1. **User** selects the “Disable Account” option. |
| ¬ | 1. **System** displays a prompt to **User** to confirm if they would like to disable their account. |
| ® | 1. **User** selects the “Disable Account” option located in the prompt box. |
| ¬ | 1. **System** (a) deletes all of the store user data from **Database**; (b) displays the Sign In page |
| **Flow of Events for Extensions (Alternate Scenarios):** | |
| 3a. User selects the “Cancel” option in the prompt box. | |
| ¬ | 1. **System** does not change any user data and displays the User Profile page. |

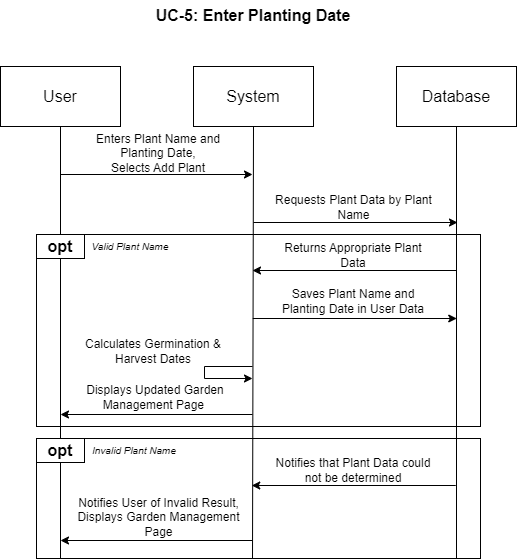
### ***System Sequence Diagrams***

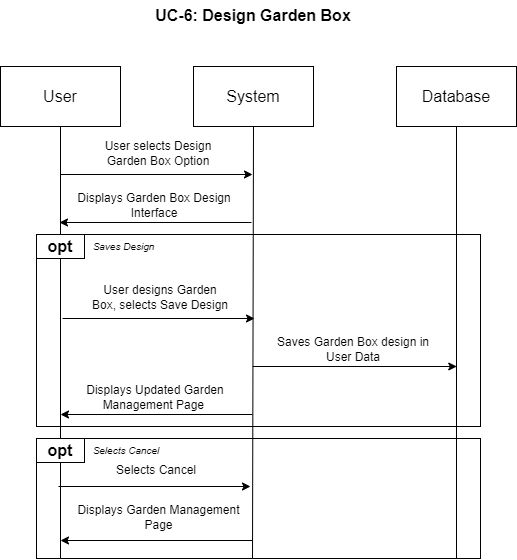


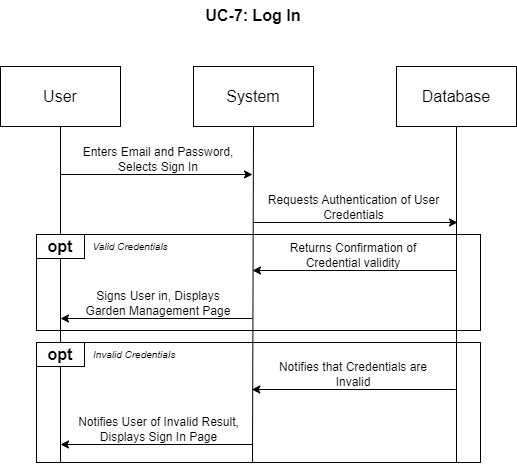


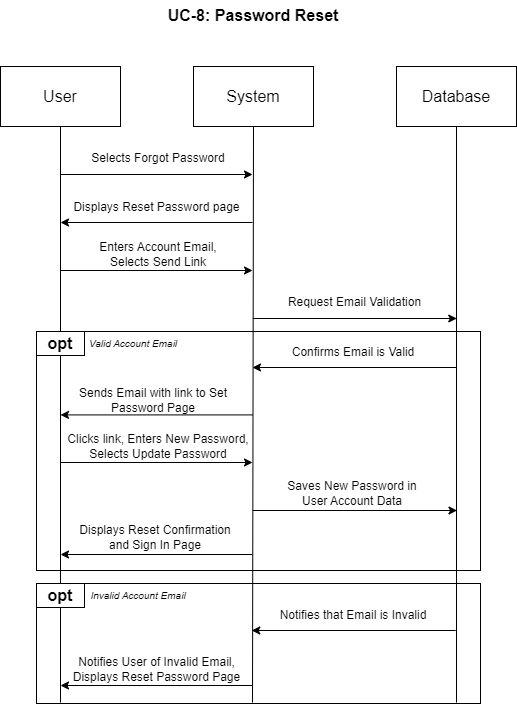


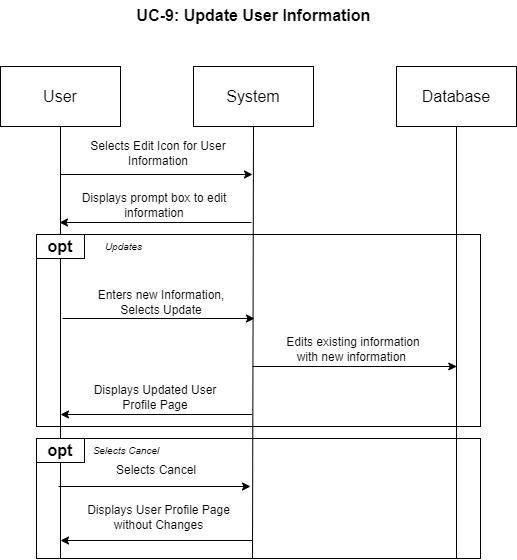


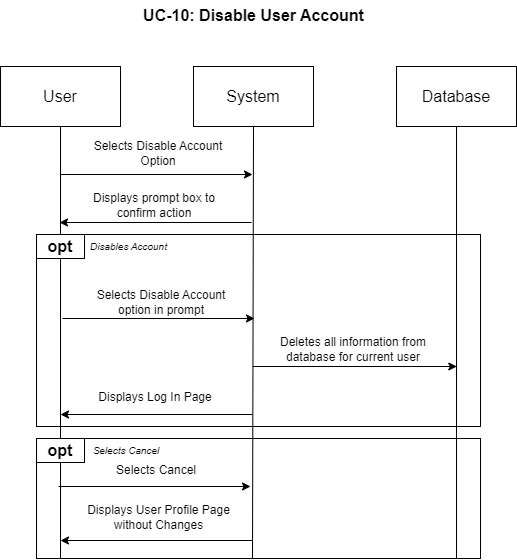












## User Interface Specification

### ***Preliminary Design***

#### UC-1: Create a User Account

Step 1: Select “Sign Up” from home page.

Graphical user interface, text, application

Description automatically generated

Step 2: Type in Name, Email, Password, and Zip Code.

Graphical user interface

Description automatically generated with medium confidence

Step 3: Click “Sign Up”.

Graphical user interface, text, application, chat or text message

Description automatically generated

Step 4 (opt: zip code is valid): System displays Account creation Confirmation.

Graphical user interface, text, application

Description automatically generated

Step 4 (opt: invalid zip code): System warns user that zip code is invalid. Go back to Step 2.

Graphical user interface, text, application, chat or text message

Description automatically generated

#### UC-3: Filter by Zip Code

Step 1: User types zip code into zip code search box

Graphical user interface, text, application

Description automatically generated

Step 2: User Clicks Search button or presses “Enter”

Graphical user interface, text, application, email

Description automatically generated

Step 3 (opt: valid zip code): System displays Search results

Graphical user interface, text, application

Description automatically generated

Step 3 (opt: invalid zip code): System warns user that Zip Code is invalid. Go back to Step 1.

Graphical user interface, text, application, chat or text message

Description automatically generated

#### UC-4: Search by Plant

Step 1: User types query into Plant name search box.

Graphical user interface, text, application, chat or text message

Description automatically generated

Step 2: User Clicks Search button or presses “Enter”.

Graphical user interface, text, application, chat or text message

Description automatically generated

Step 3 (opt: query returns results): System displays Search results.

Graphical user interface, text, application

Description automatically generated

Step 3 (opt: query does not return results: System notifies User that no results were found.

Graphical user interface, text, application

Description automatically generated

#### UC-5: Enter Planting Date

Step 1: On a search result page, select “add” button next to desired plant.

Graphical user interface, text, application

Description automatically generated

Step 2: Enter correct planting date for selected plant.

Graphical user interface, application

Description automatically generated

Step 3: Save Planting Date

Graphical user interface, text, application, chat or text message

Description automatically generated

Step 4: New planting has been added to “Garden Management” Page

Graphical user interface, text, application, chat or text message

Description automatically generated

### ***User Effort Estimation***

#### UC-1: Create a User Account

1. NAVIGATION: total 2 clicks, as follows:
   1. Click “Sign Up” button

--- *after completing data entry as shown below ---*

* 1. Click “Sign Up” button to submit

1. DATA ENTRY: total 1 clicks and 17 keystrokes, as follows:
   1. Click cursor to “Name” text field
   2. Press any alphanumerical key(s)
   3. Press “Tab” key to move to next text field (“Email”)
   4. Press any alphanumerical key(s), followed by “@”, followed by any alphanumerical key(s), followed by “.com”
   5. Press “Tab” key to move to next text field (“Password”)
   6. Press any alphanumerical key(s)
   7. Press “Tab” key to move to next text field (“Zipcode”)
   8. Press any five numeric keys

#### UC-3: Filter by Zip Code

1. NAVIGATION: total 2 clicks, as follows:
   1. Click cursor to “Zipcode” search box

--- *after completing data entry as shown below ---*

* 1. Click “Search” button

1. DATA ENTRY: total 0 clicks and 5 keystrokes, as follows:
   1. Press any five numeric keys

#### UC-4: Search by Plant

1. NAVIGATION: total 2 clicks, as follows:
   1. Click cursor to “Plant Name” search box

--- *after completing data entry as shown below ---*

* 1. Click “Search” button

1. DATA ENTRY: total 0 clicks and 1 keystrokes, as follows:
   1. Press any alphanumerical key(s)

#### UC-5: Enter Planting Date

1. NAVIGATION: total \_ clicks, as follows:
   1. Click on “Add” button next to plant name on search results page

--- *after completing data entry as shown below ---*

* 1. Click “Add Planting” button

1. DATA ENTRY: total \_ clicks and \_\_ keystrokes, as follows:
   1. Click in “Planting date” field
   2. Type ten alphanumeric keys in the format: “mm/dd/yyyy”
   3. Click outside of Planting Date box to close popout calendar

## System Architecture

### Identifying Subsystems

Our application will consist of several subsystems:

**User Interface Layer:**

User Data Enter, update – This system will take in user data that allows them to create, delete or update information.

New Account Entry – This system will be used for creating a new account.

Plant Search, Display – This system will allow the user to search for and will be in charge of displaying the plant info.

User Plant add/ delete – This system will allow the user to add plants to their profile, and delete plants from their profile.

Zone system search / Display – This system will get allow the user to enter a zone and display zone information

**Application Layer:**

Data Validation – This system will validate the user's entered information is correct, and will make sure the data returned from the database is correct

Data Calculation – This system will perform the calculations needed for the users displayed information such as germination dates.

Data Verification – This system will verify the user data against the database.

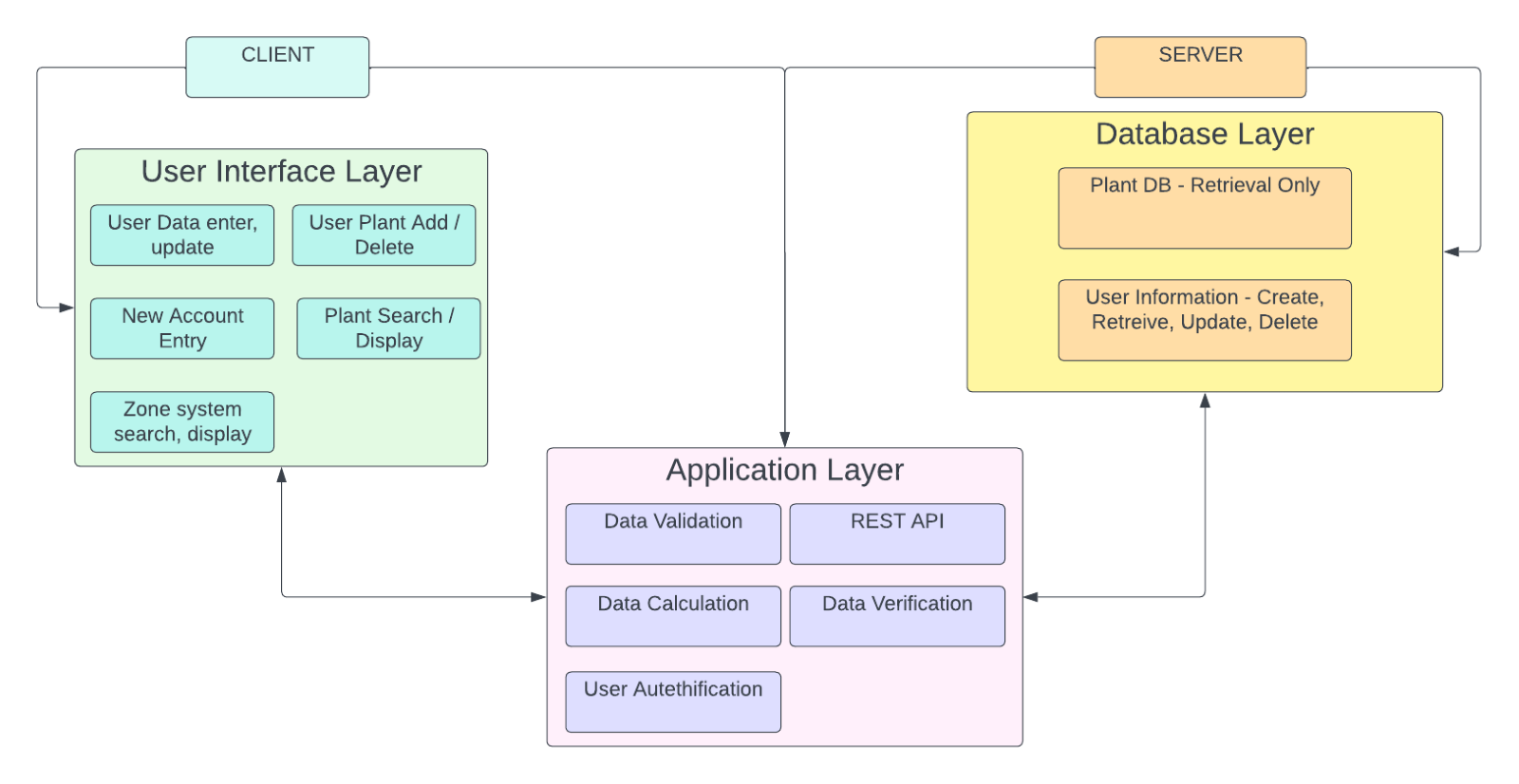
REST API – This system will communicate with requested information with the database, and receive the information from the database.

User Authentication – This will be responsible for password and user ID verification

**Database Layer:**

Plant DB – This database will contain all the searchable plant and zone information. It will be retrieve only.

User Information DB – This database will store all the user information. It will allow the creation of user accounts, the retrieval of user information, the updating of user accounts and the deletion of user accounts.



### Architecture Styles

For our application, we are using the layered architectural style along with client-server, separating the components into three main layers. We will have the user interface layer, the application layers, and the database layer.

The user interface layer will include all the user interface functionality. The layer will be responsible for collecting and displaying all information required to and from the user. It will allow the user to enter their login information, create an account, look at the list of plants, add plants to their planner, and view the data related to their plantings.

The application layer will be responsible for handling all data sending and requests using the Representational state transfer (REST) system. It will handle all requests to and from the database. The application will request user, or plant information through the REST API and the Database will send the required information back through the REST API. The application layer will be responsible for checking for appropriate data, authenticating the user info, and sending the appropriate data to the interface layer.

The database layer will handle all the data storage. We will follow the CRUD system for the database. The data backend will use MySQL along with PHP. There will be two separate storage systems. One will be static and will not allow user updates. It will be for retrieval only. This system will work whether the user has an account or not.

The other database system will store user data such as passwords, user chosen plant data, and user information and user history. The second system will allow for creation, updates and deletion of user information and will only be accessible with an account.

The main system will run in a client server pattern, with the client sending and receive the data while the server receives and processes the data. Multiple clients may connect to the server and use the database at once in a many to one relationship.

### ***Mapping Subsystems to Hardware***

The database system will be mapped server-side and will be hosted on Amazon Web Services and will require SQL and PHP support. Since AWS is scalable, we will use the minimum DB storage allowable, but with possibility of expanding further into the future. Both Databases will be stored in the same server space. However, the DB system can be hosted on any server that allows for SQL and PHP.

The application layer will be hosted partially server-side on the webserver and partially on the client-side. The REST API will be hosted on the webserver as will the user authentication system. Both will be written in PHP. The data verification, data validation and data calculation will be mapped to the client-side browser using JavaScript

The User interface will be mapped to the client-side browser and will be supported by all browsers, both desktop and mobile. It will use CSS along with HTML to create the interface. All services will require a data connection between the client and the server.

### *d.* ***Connectors and Network Protocols***

Not applicable.

### e. ***Global Control Flow***

#### Execution Orderness

This program runs in a linear execution style. The order of the events will differ depending on the user. First, the user can either create an account, sign in, or continue as a guest. If the user is creating an account, they will enter their information, create a valid login, and enter their zip code to save to their account. The options after this if they are only a guest is they can search plants by their name and see the information on the plants, filter plants by a zip code to see hardiness zones and information on the plants in those zones and enter planting dates for plants to see the expected germination and harvest dates. Users who are signed in can also save their planting dates, design a garden box, take notes on their personal garden already created, update user information, and disable user account.

#### Time Dependency

The program will not have any timers in the application and does not need any physical time down to the hour. It will use a calendar to select the date that it was planted and to calculate the germination dates and harvest dates. It will need to take in the current date to use when selecting the planting date.

### *f.* ***Hardware Requirements***

This application will require connection to either Wi-Fi or mobile data to run. This program will be able to be run on any device that can use a browser of some kind. It will require a display of some kind as well.

## Project size Estimation Based on use case points

|  |  |  |  |
| --- | --- | --- | --- |
| Actor name | Description of relevant characteristics | Complexity | Weight |
| User | User is interacting with the program via a graphical user interface. | Complex | 3 |
| Database | Database is another system interacting through a protocol. | Average | 2 |
| Unadjusted Actor Weight(UAW) | | | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| Use case | Description | Category | Weight |
| Create User Account (UC-1) | Complex user interface. 4 steps for the main success scenario. One participating actor (Database). | Average | 10 |
| Take Note (UC-2) | Moderate user interface. 2 steps for the main success scenario. One participating actor (Database). | Average | 10 |
| Filter by Zip Code (UC-3) | Complex user interface. 2 steps for the main success scenario. One participating actor (Database). | Average | 10 |
| Search by Plant (UC-4) | Complex user interface. 2 steps for the main success scenario. One participating actor (Database). | Average | 10 |
| Enter Planting Date (UC-5) | Simple user interface. 2 steps for the main success scenario. One participating actor (Database). | Simple | 5 |
| Design Garden Box (UC-6) | Complex user interface. 4 steps for the main success scenario. One participating actor (Database). | Average | 10 |
| Log In (UC-7) | Simple user interface. 4 steps for the main success scenario. One participating actor (Database). | Simple | 5 |
| Password Reset (UC-8) | Simple user interface. 6 steps for the main success scenario. One participating actor (Database). | Average | 10 |
| Update User Information (UC-9) | Complex user interface. 4 steps for the main success scenario. One participating actor (Database). | Average | 10 |
| Disable User Account (UC-10) | Complex user interface. 4 steps for the main success scenario. No participating actors. | Average | 10 |
| Unadjusted Use Case Weight (UUCW) | | | 90 |

Unadjusted Use Case Points (UUCP) = 95

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Technical Factor | Description | Weight | Perceived Complexity | Calculated Factor |
| T1 | Distributed, Web-based system | 2 | 3 | 6 |
| T2 | Users expect good performance but nothing exceptional | 1 | 3 | 3 |
| T3 | No exceptional demands for end-user efficiency | 1 | 3 | 3 |
| T4 | Internal processing is relatively simple | 1 | 1 | 1 |
| T5 | No requirement for reusability | 1 | 0 | 0 |
| T6 | Ease of install not required | 0.5 | 0 | 0 |
| T7 | Ease of use is very important | 0.5 | 5 | 2.5 |
| T8 | Minimal portability concerns | 2 | 2 | 4 |
| T9 | Easy to change minimally required | 1 | 1 | 1 |
| T10 | Concurrent use is required | 1 | 4 | 4 |
| T11 | Security is an average concern | 1 | 3 | 3 |
| T12 | No direct access to third parties | 1 | 0 | 0 |
| T13 | No unique training needs | 1 | 0 | 0 |
| Technical Factor Total: | | | | 27.5 |

Technical Complexity Factor (TCF) = 0.875

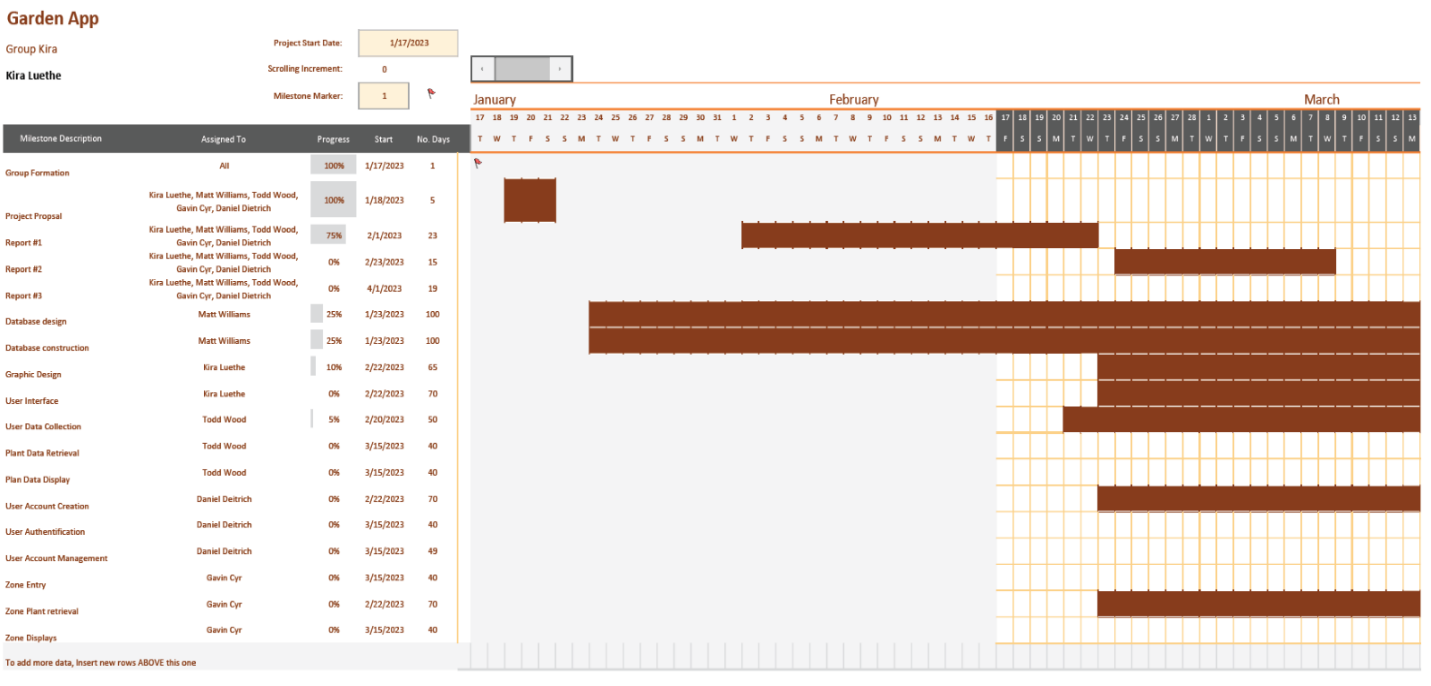
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Environmental Factor | Description | Weight | Perceived Impact | Caclulated Factor |
| E1 | Beginner familiarity with the UML-based development | 1.5 | 1 | 1.5 |
| E2 | Some familiarity with application problem | 0.5 | 2 | 1 |
| E3 | Some knowledge of object-oriented approach | 1 | 2 | 2 |
| E4 | Beginner lead analyst | 0.5 | 1 | 0.5 |
| E5 | Average motivated team members | 1 | 3 | 3 |
| E6 | Stable requirements expected | 2 | 3 | 6 |
| E7 | No part-time staff will be involved | -1 | 0 | 0 |
| E8 | Programming language of average difficulty will be used | -1 | 3 | -3 |
| Environmental Factor Total: | | | | 11 |

Environment Complexity Factor (ECF) = 1.07

Use Case Points (UCP) = 89

## Plan of work

### a. Gantt Chart



### b. Product Ownership and Breakdown of Responsibilities

The group began by determining the desired initial functionality for the project. The group planned the back-end database and user authentication requirements. Then, the group moved onto planning a basic website framework, user features, user data entry, user data display and retrieval features.

The group will then design and build the back-end database and user authentication modules including ERD diagrams and password authentication methods. Then rudimentary data entry features will be implemented for testing functionality.

The front-end team will build user data entry and retrieval functionality while coordinating with the backend for storage and retrieval. The feature will be tested using dummy data.

The front-end team will continue to build and test additional functionality, including germination and harvest date calculations and display, planting zones, and data retrieval tools. Each new feature will be tested using dummy data.

The page will continue to be built out for GUI features and style, making the application both functional and user friendly.

The project will be tested using real world data and each feature will be checked to ensure proper operation both with data storage and retrieval as well as display and user interface functions.

If time allows the group will expand functionality.

Product ownership shall be divided as follows.

Kira Luethe is the team lead, responsible for organizing communication between group members and the instructor, as well as ensuring timely completion of project stages.

Daniel Dietrich and Richard Williams shall implement the back-end features of the app. Richard has begun building a database and an API. Daniel will build the user authentication portion.

Gavin Cyr, Kira Luethe, and Todd Wood shall implement the front-end features. Gavin will create the planting zone selection feature. Kira will focus on the user interface and graphic design. Todd will build the planting, germination, and harvest dates tracker.

## References

**Fellow student report links:**

<https://www.ece.rutgers.edu/~marsic/books/SE/projects/Restaurant/2019-g13-report3.pdf>

<https://www.ece.rutgers.edu/~marsic/books/SE/projects/ParkingLot/2019f-g4-report3.pdf>

**API/Datasets Used to populate Database:**

<https://rapidapi.com/aptitudeapps/api/usda-plant-hardiness-zones/details>

<https://sage.nelson.wisc.edu/data-and-models/datasets/crop-calendar-dataset/>

## Summary of Changes

* Responsibility Matrix Updated
* Updated Cover Page
* Table of Contents Updated
* Use Cases, Fully-dressed description
  + UC-1: Added alternate use case
  + UC-3: Clarified description language
* Use Cases, System Sequence Diagrams
  + UC-5: Removed step that stored calculations in database
* Added additional References