Plantalouge

Software Design Document

Name (s):
Faisal Albellaihi
Zach Payne
Flannery Collins

Software Engineering CPSC 430
Section 02

Date: (03/29/2016)

TABLE OF CONTENTS

1. INTRODUCTION

1 This section provides an overview and description of this entire document. Separate subsections regarding purpose, scope and terminology are provided. This software document provides a documentation which will be used to assist the development of the Plantalouge system by providing the details of development process. Also, within this Software Design Document are graphical representations and diagrams of the software design

1.1 Purpose

The purpose of this Software Design Document is to provide a detailed description of the design of the Plantalouge software and system. It will allow for the software developers, client, and any third parties to process to continue with an understanding of what is to be built, and how it is expected to be built. This document provides the information that is necessary to describe all of the details for Plantalouge to be built.

1.2 Scope

The Plantalouge is a web application that helps Dr. Wynn and UMW biology department to keep track of the plants that are being donated by volunteers who donate their plants. The plants will be catalogued and organized in a relational database schema. Based on this data, Dr. Wynn will have the ability to create and manage tasks for a plant and assign them to students or volunteers in a work schedule. Based off of this info and worker completion feedback, the system will generate updates of student's completed tasks. This system will provide the admin user with the ability to print a label containing the plant's information.

1.3 Overview

- The remainder of this document will consist of 6 sections
- **Section 2**
  - System overview - will provide an overview of the Plantalouge system
- **Section 3**
  - Describes the implementation architecture of the system, including its architectural design rationale, decomposition diagram, and diagram description, and the rationale for why the architecture is being developed this way.
- **Section 4**
  - Describes the data design, including a data description of how the data structure of the system will be developed, along with the entities that the system will include. The ERD diagram will be located here to illustrate the data design.
- **Section 5**

- Describes the component design, where we will take a closer look at what each component does in the software in a clearer, more systematic way.

- **Section 6**
  - Describes the human interface design, and will contain details about the user interface and how the software will look and act with the use
- **Section 7**
  - The Requirements matrix. Here will be a quick reference guide to use cases in the System specification document to system components that fulfill those use cases.

1.4 Reference Material

- System Requirement Specification Document for Plantalouge
- Use Case Diagrams for Plantalouge
- ERD Diagram of Plantalouge Database

1.5 Definitions and Acronyms

Term	Definition
Database	A collection of data that is organized and held in a computer and can be accessed in different ways.
Web Application	Web application is an application that can be stored in a server and is delivered over the internet and an internet browser
Student	The person interacting with the system with a limited privileges level
Donator	The person who donated a plant and is interacting with system this user has the minimum privileges level
Admin	The person interacting with the system who has the maximum privileges level
SQL	Structured Query Language. A common programming language for accessing and managing a relational database.
	.

Figure 1- Definitions table

2. System Overview

2 Dr. Wynn, a Biology professor here at the University of Mary Washington, is in charge of running the plant lab, as well as managing the work schedules for all the student workers who take care of the plants in the previously mentioned lab. She wants a software that can keep a database of information for each and every plant that is being kept in the lab. However, Dr. Wynn does not take care of all of these plants by herself. She is in charge of the many students and volunteers who work under her, and take care of the plants according to her instruction. The software will also be used for Dr. Wynn to create and assign the tasks and work schedules for each of the workers that she is in charge of. The software will also be used to pull all of the information about a certain plant from the database, put it on a label, and print the label, so that Dr. Wynn can put the printed label onto its respective plant

Figures below are the use case diagrams provided to show all the uses of the system

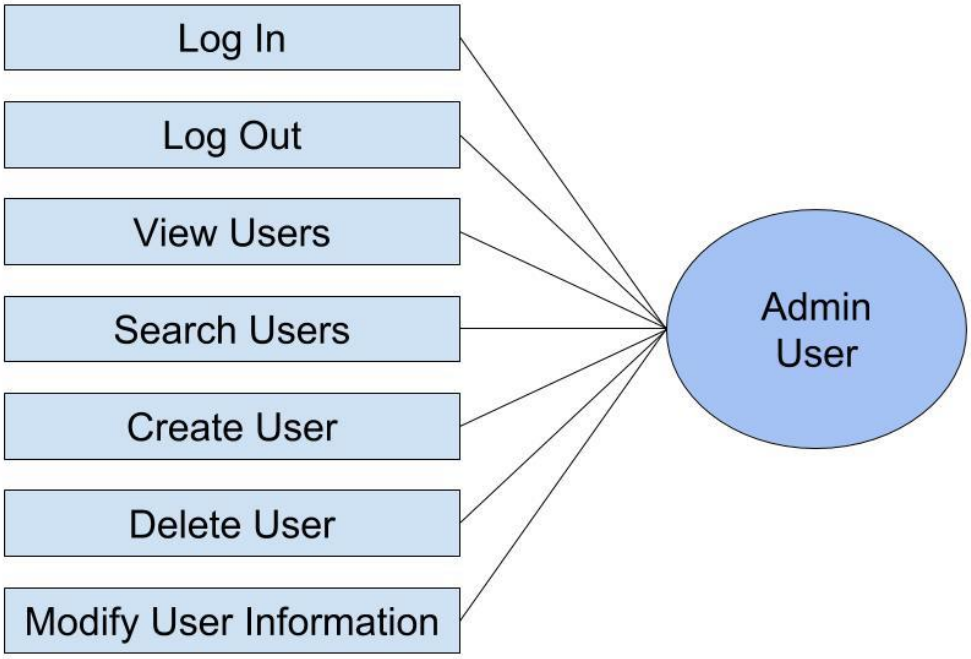

Figure 2- This use case shows admin actor accessing users and user profile. In this use case admin create, search, remove, view, and modify users.

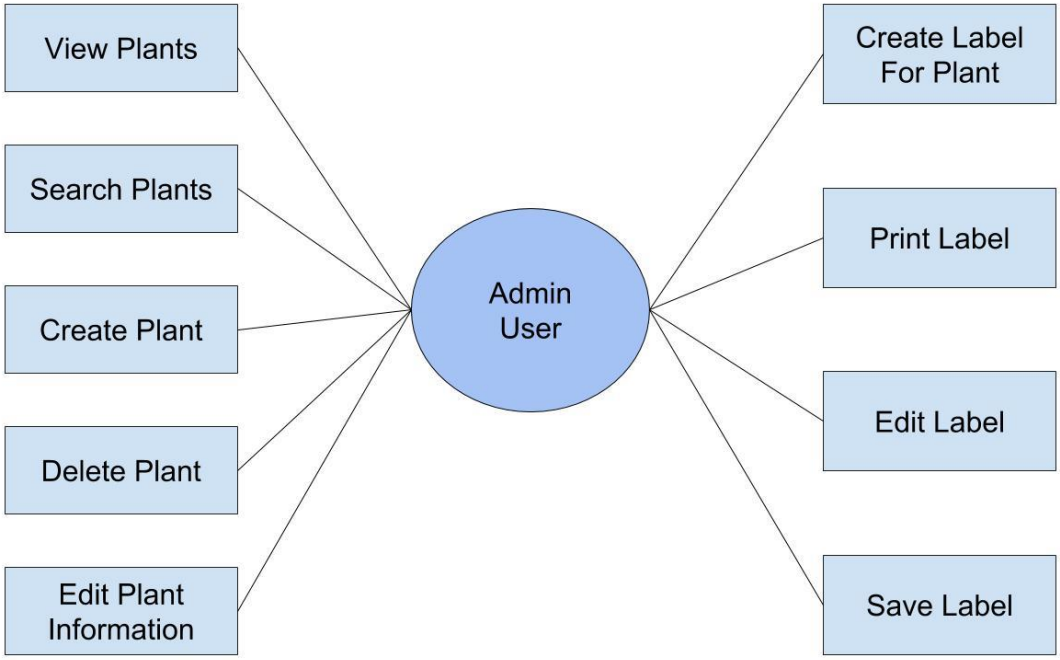

Figure 3- Manage Plants use case, Actor in this use case is the admin who can view, a plant and search a plant. This use case shows what admin can do to plants, like deleting a plant, adding a plant, modifying a plant, or creating a label for a plant.

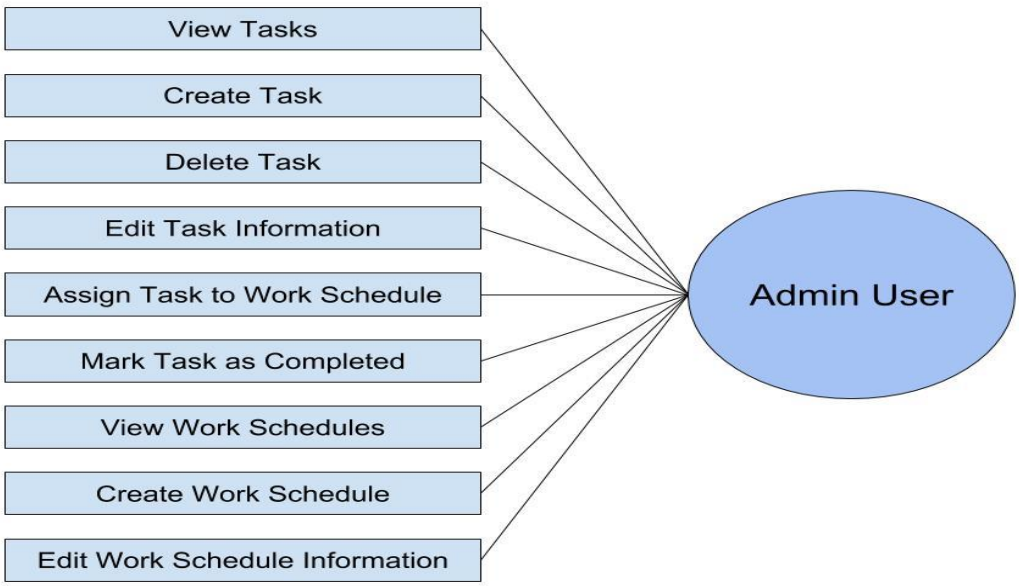

Figure 1- Manage Tasks and Work Schedules use case diagram. The actor is the Admin, who creates tasks for plants, and assigns them to a student's work schedule for them to complete

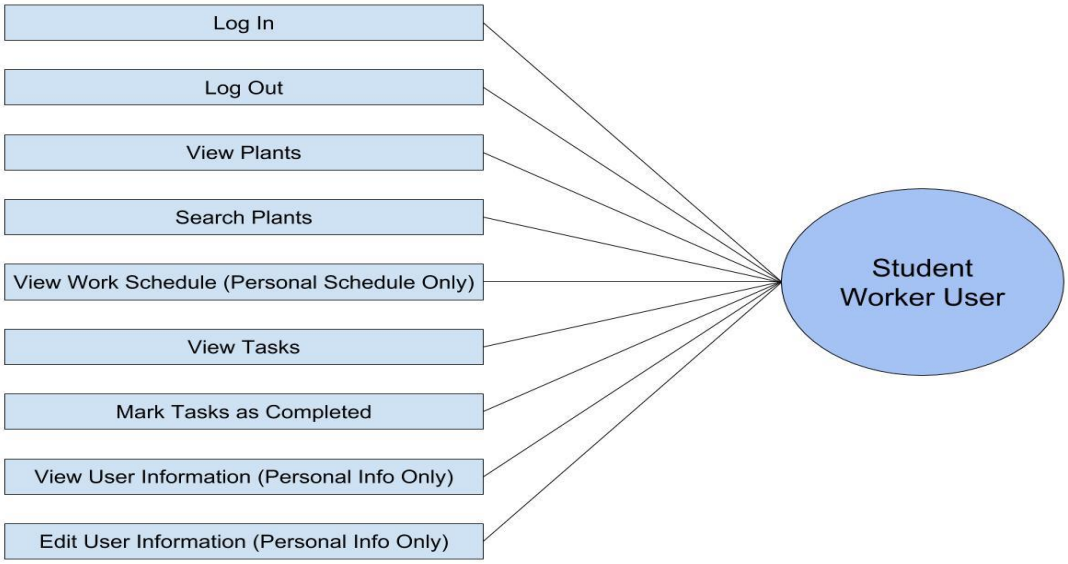

Figure 5- Worker use case Diagram. This diagram illustrates the functionality and use cases available to the worker user.

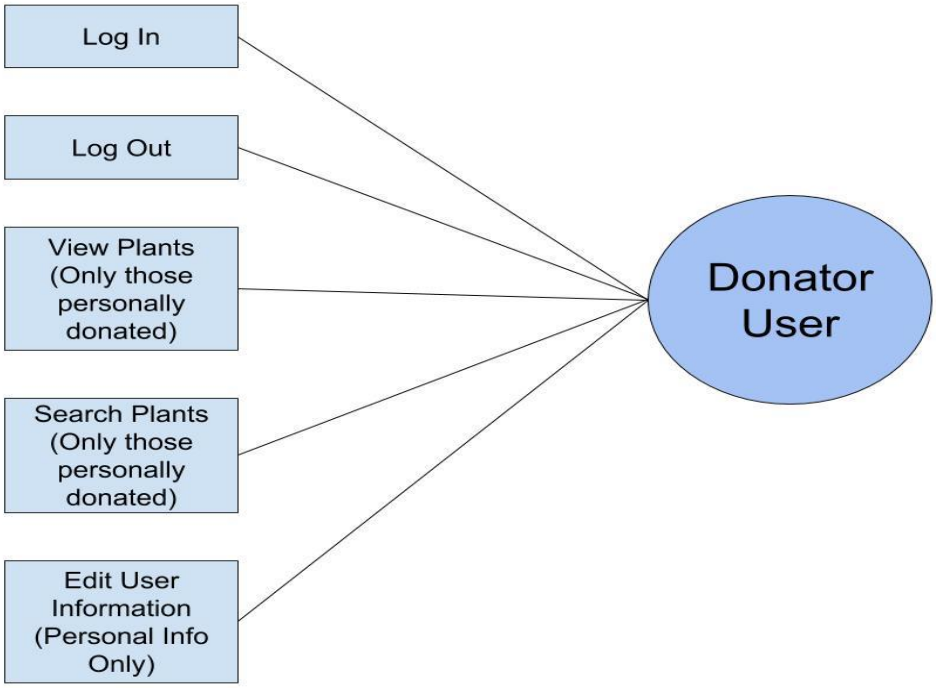

Figure 6- Donator use case. This diagram demonstrates the use cases available to the Donator User.

2.1 Technologies Used

The system will be developed for an internet browser, specifically Mozilla Firefox. The required technologies will include a computer or mobile device with a steady internet connection, and a web browser installed on the previously mentioned computer.

3. System Architecture

3.1 Architectural Design

Figure 1 shows the structure of the database. Plants table information about each plant in the system. It is linked to Tasks table, because each task is created specifically for a plant. Therefore, there needs to be a link between those two tables. There also is a link between the Plants table and the Users table, because each plant is linked to the Donator user who it belongs to. The Plants table is also linked to the Labels table, because the Admin can create a label for a plant that can be printed out.

The Users table contains the information about each user that will be in the system. All three user types will be contained in this table, and are given a privilege of 1, 2, or 3, which corresponds to Admin, Student Worker, or Donator. The Users table is linked to the Work Schedules table. This link exists because each Student Worker user will be given a work schedule by an Admin.

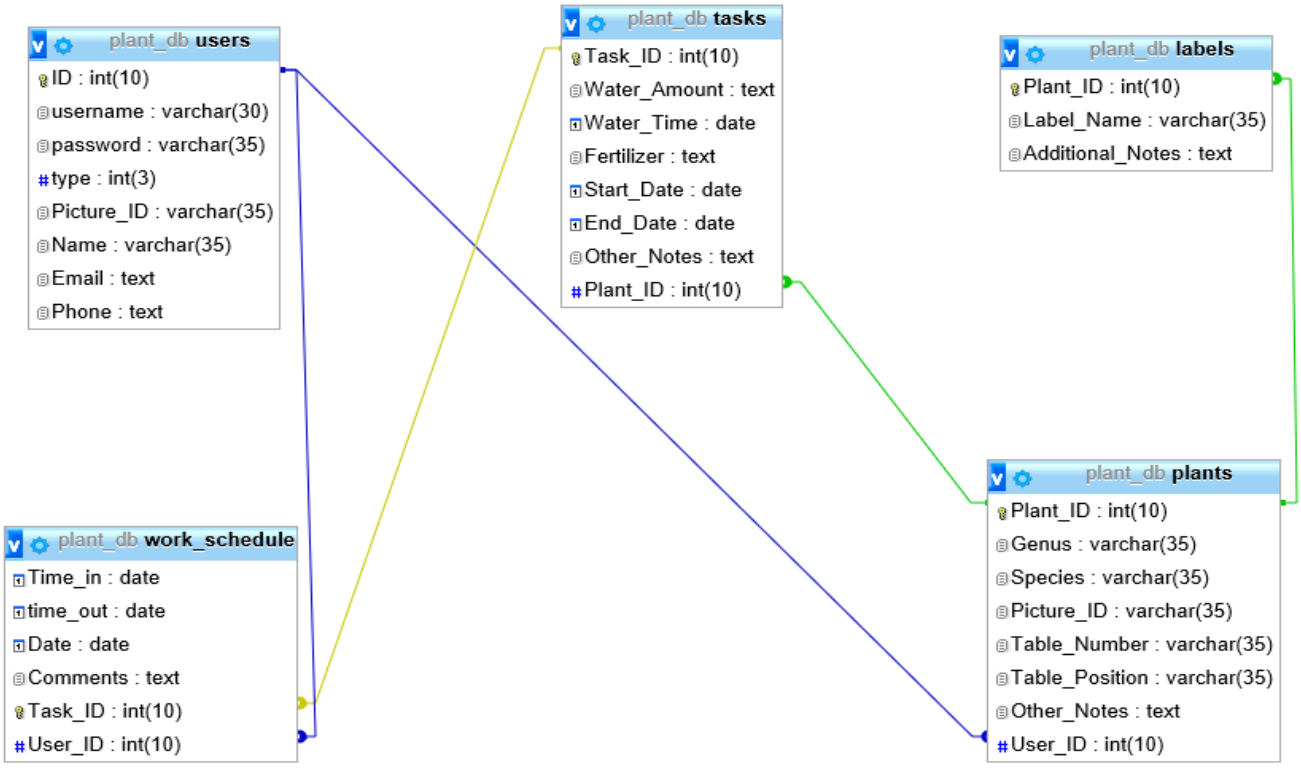

Figure 7- Entity Relationship Diagram

In Figure 7 the Work Schedules table contains the information about a student's work schedule, and is linked to the Tasks table. The Tasks table contains the information about each task, which is linked to the plant that the task has been created for. The Work Schedules table and the Tasks tables are linked because a single task, or multiple tasks, can be assigned to a work schedule. The Work Schedule is then assigned to a Student Worker.

3.2 Decomposition Description

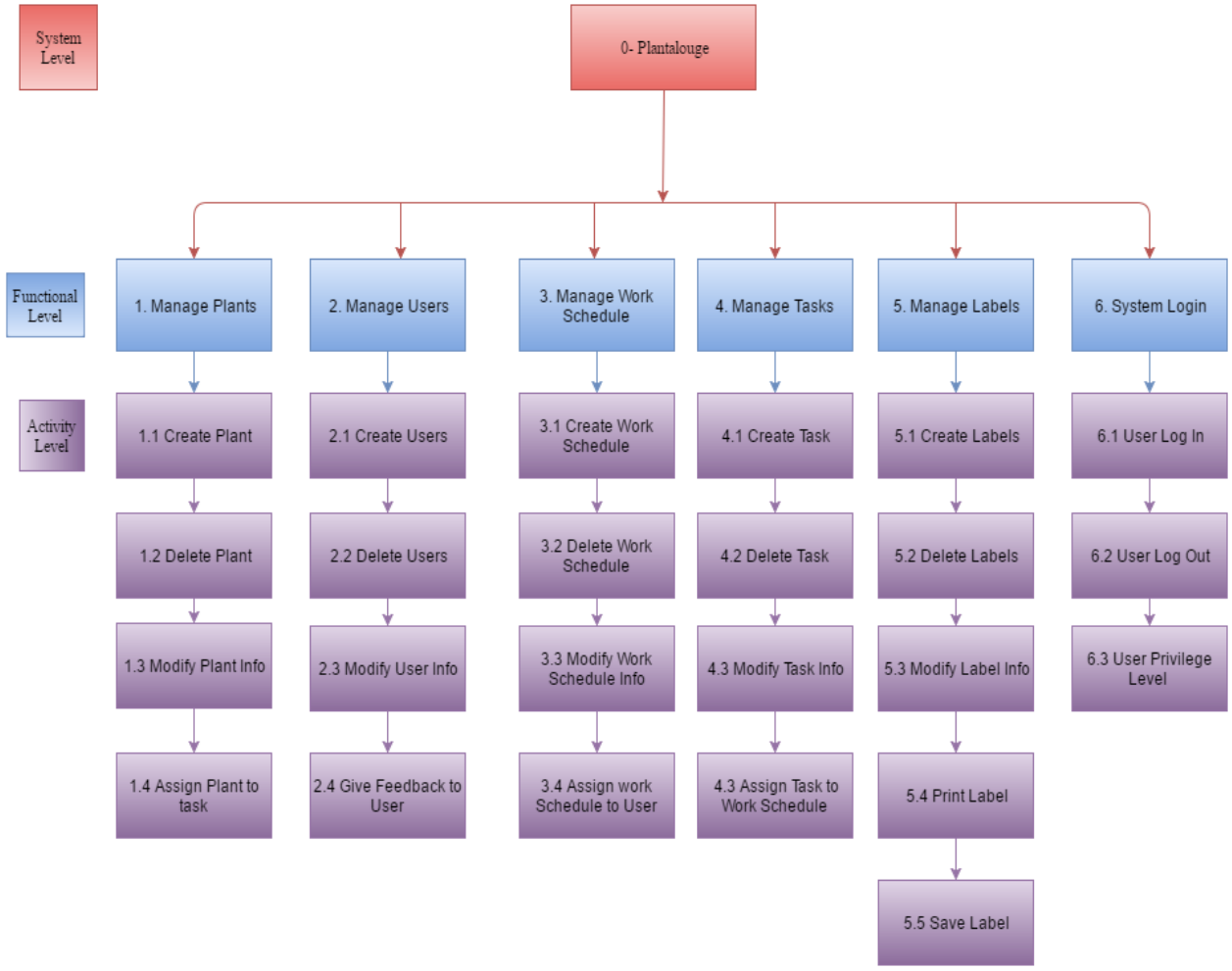

Figure 8- Decomposition Description

3.3 Design Rationale

Since the software's main function is to store information, creating a database (as shown in section 3.1) was a logical decision. Plantalouge is used to store information about plants and the people who take care of them. Therefore, we chose to create separate tables for each type of information that it is storing. This was the logical design, since you can create links between tables, so that specific entries can have relationships with entries in the other tables. This architecture was agreed on by all three members of our development team, and was an easy decision. There were no alternative designs, since this design was perfect for what the software needs to accomplish

4. DATA DESIGN

4.1 Data Description

The software will support SQL and will have a set of tables that will store information about plants, users, and tasks. Each user has a level of privilege, the admin is the highest and the rest have a limited privileges. Only the admin has the ability to edit and delete information from the database.

4.2 Data Dictionary

The following section defines the entities from the diagram in section 3.1. The entity is in bold, and the data type is in parenthesis.

- **Labels** (Table) - contains the information about labels that are to be printed, and the information about a specific plant that the label was created for; labels table contains the following entities:
    - **Additional Notes** (String) - Any additional notes that the user wants to include on the label
    - **Label Name** (String) - the name that the user gives the label when they save it

- **Plants** (Table) - contains the information about plants that are created in the database; plants table contains the following entities:
    - **Genus** (varchar) - the scientific genus that the plant belongs to
    - **Other Notes** (text) - any additional information that they user wants to include about the plant
    - **Picture ID** (varchar) - the system generated identification number that will be uniquely assigned to the photo uploaded for the plant
    - **Plant ID** (int) - the system generated identification number that will be uniquely assigned to each plant upon creation
    - **Species** (varchar) - the scientific species that the plant belongs to
    - **Table Location** (varchar) - the specific location of the plant on the table that it is located on inside of the biology lab; each table in the lab will be broken into numbered sections
    - **Table Number** (varchar) - the table number that the plant is located on inside of the biology lab

- **Users** (Table) - contains the information about users that are created in the database; users table contains the following entities:
    - **Email** (text) - the user's email address

  - **ID** (int) - the system generated identification number that will be uniquely assigned to each user upon creation
  - **Name** (varchar) - the user's actual name, for example "John Smith"
  - **Password** (varchar) - the password that the user will use to log in to the system
  - **Phone** (varchar) - the user's phone number, in case they need to be called
  - **Picture ID** (varchar) - the system generated identification number that will be uniquely assigned to the photo uploaded for the user's profile
  - **Type** (int) - the user's privilege type; can either be 0, 1, or 2, corresponding to either Admin, Student Worker, or Donator
  - **Username** (varchar) - the user's username that they will use to log in to the system

- **Tasks** (Table) - contains the information about tasks that are created in the database; tasks are the care instructions of how to take care of a specific plant; the tasks table includes the following entities:
  - **End Date** (date) - the date that the task is to be completed by
  - **Fertilizer** (text) - the amount of fertilizer that the specific plant requires
  - **Other Notes** (text) - any additional information that the user wants to include about the task
  - **Start Date** (date) - the date that the task is to be started on
  - **Task ID** (int) - the system generated identification number that will be uniquely assigned to the task upon creation
  - **Water Amount** (text) - the amount of water that the task's corresponding plant requires
  - **Water Time** (text) - the frequency of watering that the task's corresponding plant requires

- **Work Schedules** (Table) - contains the information about work schedules that are created in the database; they are assigned to student workers and are made up of the following entities, as well as tasks that are linked to specific work schedules:
  - **Comments** (text) - additional information that does not pertain to one of the other entities that the admin wants to include on the work schedule
  - **Date** (date) - the date that the work schedule was assigned to the student worker
  - **Time In** (date) - the time that the student worker is expected to start work
  - **Time Out** (date) - the time that the student worker gets off of work

5. COMPONENT DESIGN

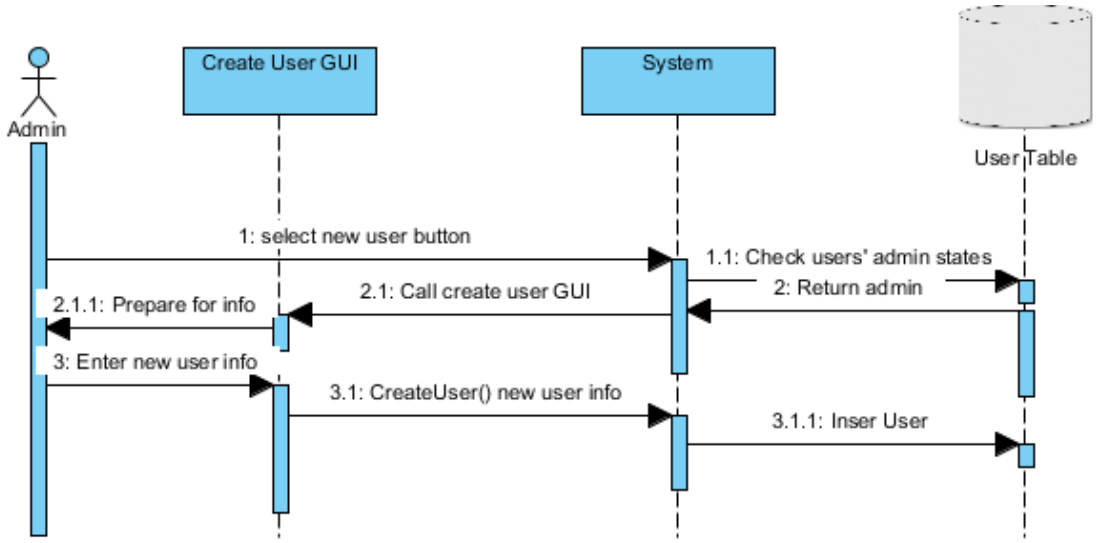

Figure 9- Create Users Sequence Diagram

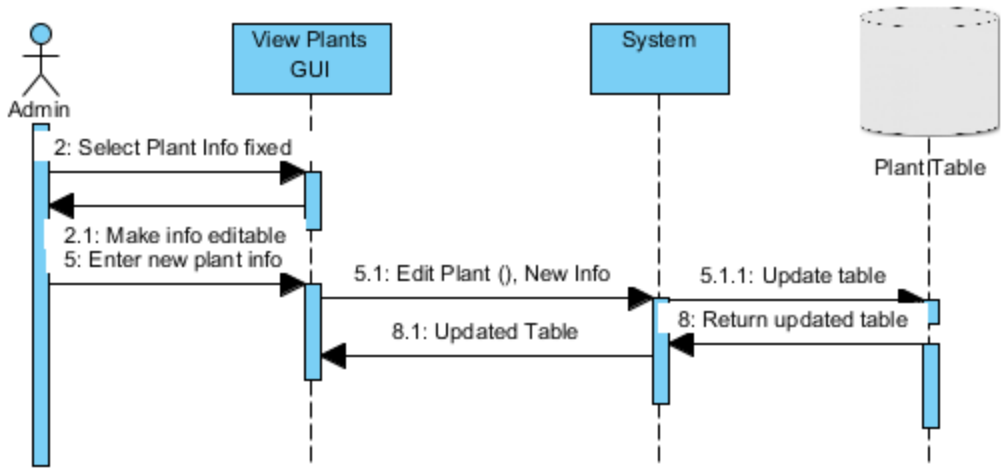

Figure 10- Edit Plants Sequence Diagram

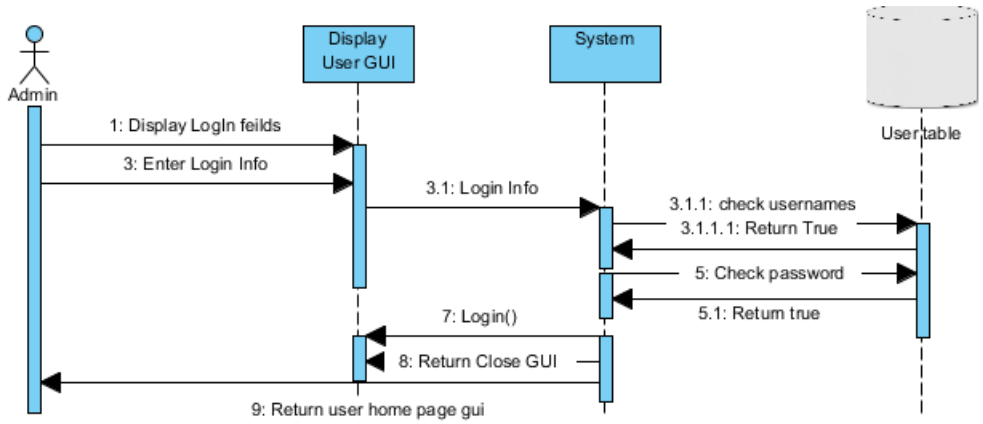

Figure 11- Log in Sequence Diagram

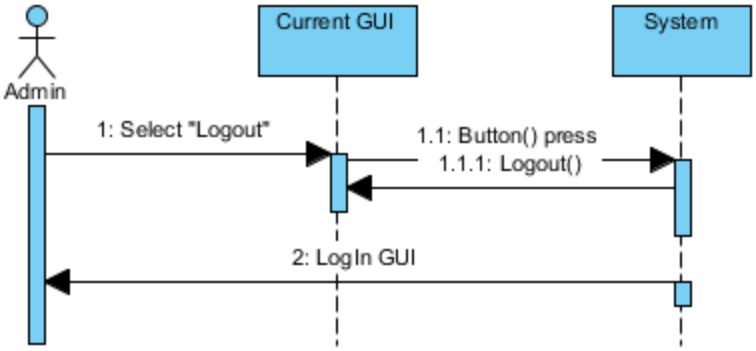

Figure 12- Logout Sequence Diagram

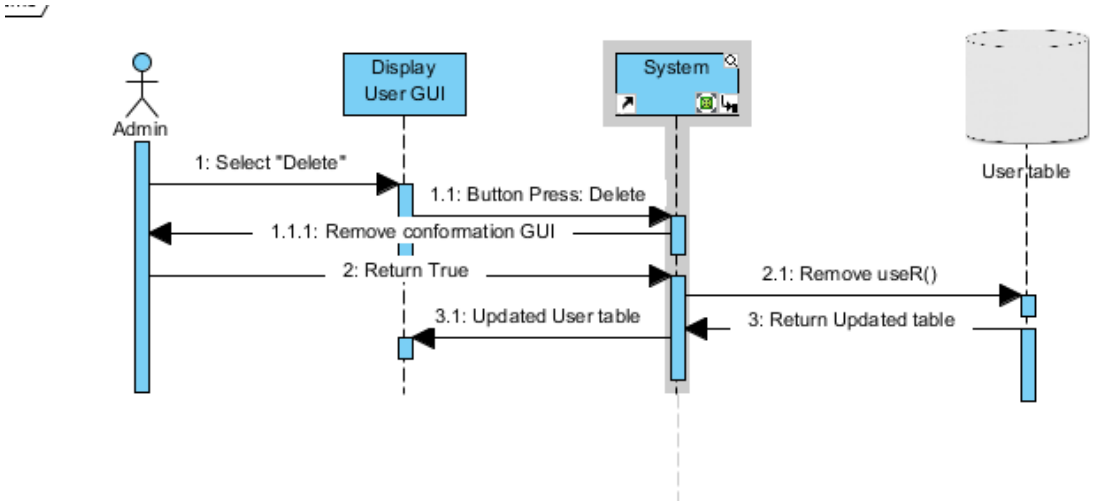

Figure 13- Remove User Sequence Diagram

Class name	Controller	Class's local variables to be used in method
removeTask(ID)	A method that allows admin to delete tasks from table	

	Void RemoveTask(){
	// remove task from database using task controller class using passed id
	//remove task from tasklist
	}	

DeleteUser(ID)	Method to allow admin to delete users from database	

	RemoveUser(int ID){
	//remove user from database using users controller class using passed id
	// remove user from userslist	

	}	
createUser()	A method allow admin to create new user	***String NewUsername;*** ***String NewPassword;*** ***int NewType;*** ***String New Email;*** ***String NameOfUser;*** ***String PictureID;*** ***String PhoneNumber;***

	Void createUser(){
	//create new object of users (newUser)
	// call set function to set "newuser" username
	//call set function to set password of newuser password
	//call set function to set newuser type	

editPlant(Plant p)	A method allow admin to edit plants available in the database	***PlantsController pcontroller***

	Public void editPlant(Plant p){
	pcontroller.edit(p)
	
	}	

editUser(User user)	A method allow admin to edit users available in the database	

	Public void editUsers(User u){
	UsersController.edit(u)
	}
		UsersController uController

Logout	A method that logs users out and destroy session	

	Public String logout(){
	//set username to null
	//set password to null
	//set type to null
	//Destroy session
		String username;
		String password;
		int type
		List<Users> usersList;
		List<Plants> plantList;
		List<WorkSchedule> workList;

	Return login page }	***List<Tasks> tasksList;*** ***List<Labels> labelList;*** ***Users Controller uController*** ***TasksController tController*** ***WorkScheduleController wController*** ***LabelsController lController***
Login()	A method to log user in and determine the type of user whether the user is admin, student, donator. And retrieves all users, plants, tasks, work schedule from database.	

	Public String login(){
	// create new user object (user)
	//User = user controller login(username, password, type)
	//Check if user == null return null
	Else {
	//get entities from user controller class and fill plant list
	//get entities from plant controller class and fill plantlist
	//get entities from tasks controller class and fill tasks list
	//get entities from tasks controller class and fill tasks list
	
	if user type ==0 :
	// return user to admin control panel page
	Else if user type == 1:
	// return user to student control panel page
	Else if user type == 2:
	// return user to donatorhome page
	Else:
	// return to admin control panel	

Table 14- Pseudocode

6. HUMAN INTERFACE DESIGN

6.1 Overview of User Interface

The Plantalouge system, to a user can be divided into 3 experiences. The Admin, the Worker, and the Basic user.

The Admin user can use the Plantalouge website to manage, and keep track of the collection of plants under UMW care, volunteer workers and their tasks and schedules, and labels for each of the plants that can be created and printed with ease. The Admin should be able to log in and have instant access to tasks completed and yet to be completed. From this homepage with task updates, the admin can view the databases of plants, workers, and work schedules with only a click of a button in the main menu. This system should allow for new elements to be added, existing elements to be modified, searched and deleted. The Admin can also easily access the data of a plant element and with the click of a button, have a label formatted and printed for any plant in the system.

The Worker user can use the Plantalouge website to keep track of the tasks assigned to them through a work schedule. The system will have tools to collect the user's work hours, and details of the work done, while providing all the data needed to carry out the tasks the user can do. The worker should be able to log in and have instant access to tasks completed and yet to be completed assigned to them as well as open tasks to pick up.. From this homepage with task updates, the worker can view their profile, and plants under their care. Their profile can be modified but their plants cannot be altered by the user.
The Basic user can use the Plantalouge website to keep track of the plant they donated to keep tabs on the progress and condition of their plant. The Basic user can do very little within the site, as specified by the client, with only the ability to view their plant and profile.

6.2 Screen Images

Figures provided in this section are from Plantalouge website.

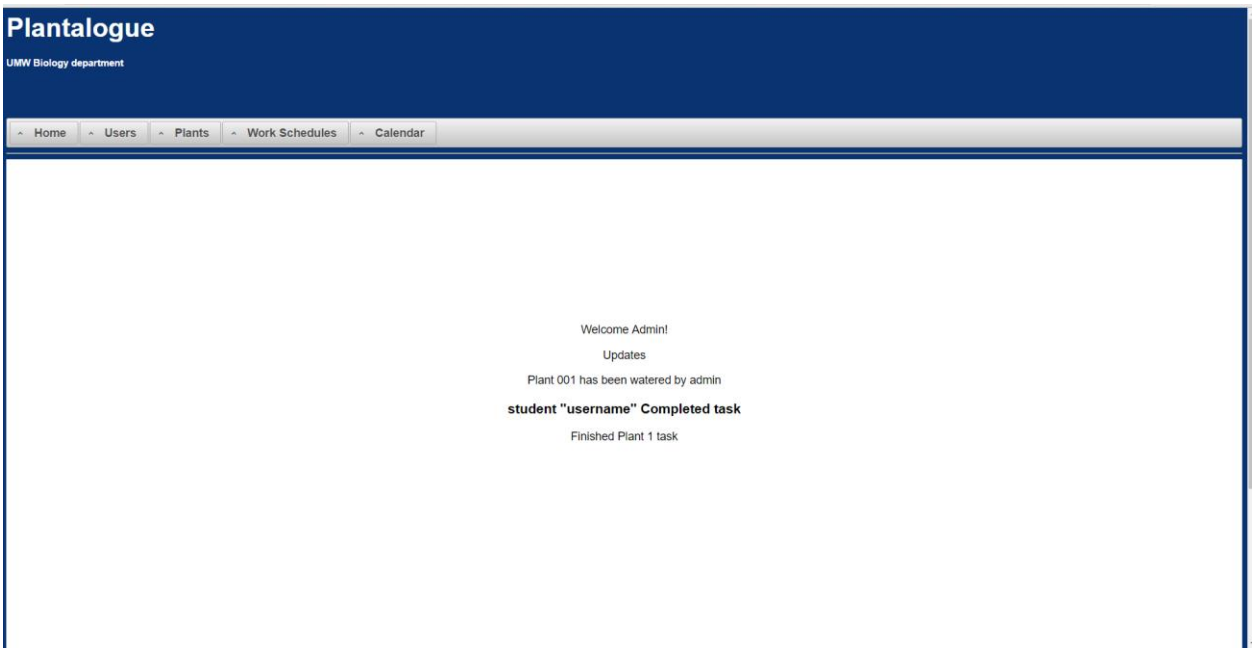

Figure 15- Admin welcome Page

Figure 15 is pictured the home screen of the Plantalouge website. Specifically the home screen of the admin user which displays updates and notifications of tasks completed and what jobs need to still be done. The home screen will be similar in look and design of the other 2 user types, with the exception that it would display updates pertinent to the user in question [Workers see tasks assigned and open tasks, Basic users see their plant(s)

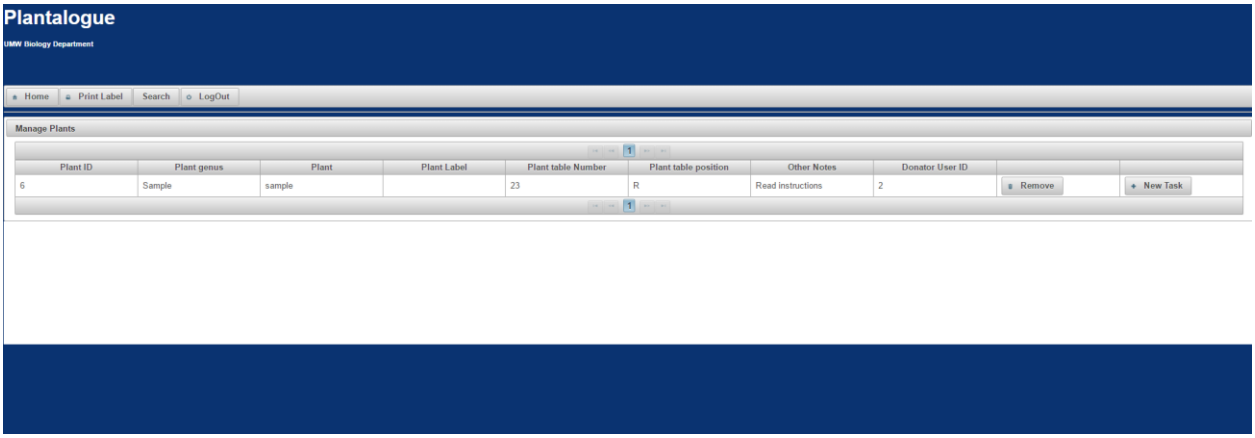

Figure 22- Manage Plants page

Figure 16 is the plant database display. This is where the admin can search for, and manage plants in the system by deleting, modifying, and adding new plants. The admin is the only

user with this access to the plant database so the other users either have very limited views of this database, or no ability to view the plants at all. This screen is almost identical to the admin's User database and Work schedule database with the same use and overall functionality. The only major difference is the contents displayed in the database screen.

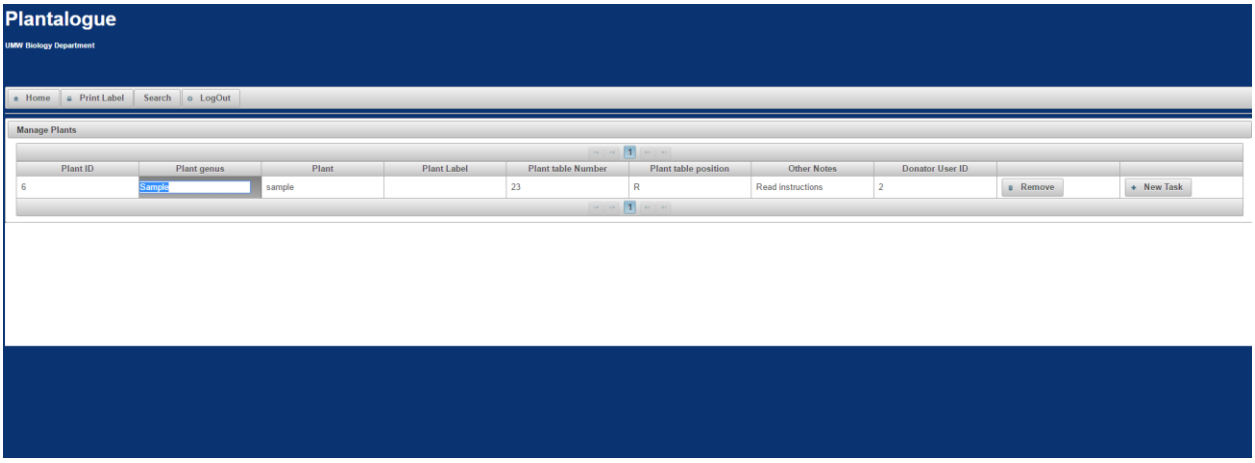

Figure 17- Editing Plant

Figure 17 shows how modification works in Plantalouge. The element that needs to be modified can be clicked and entering the changes in the text box. To finalize changes, the admin can hit the "ENTER" key or click anywhere else in the page.

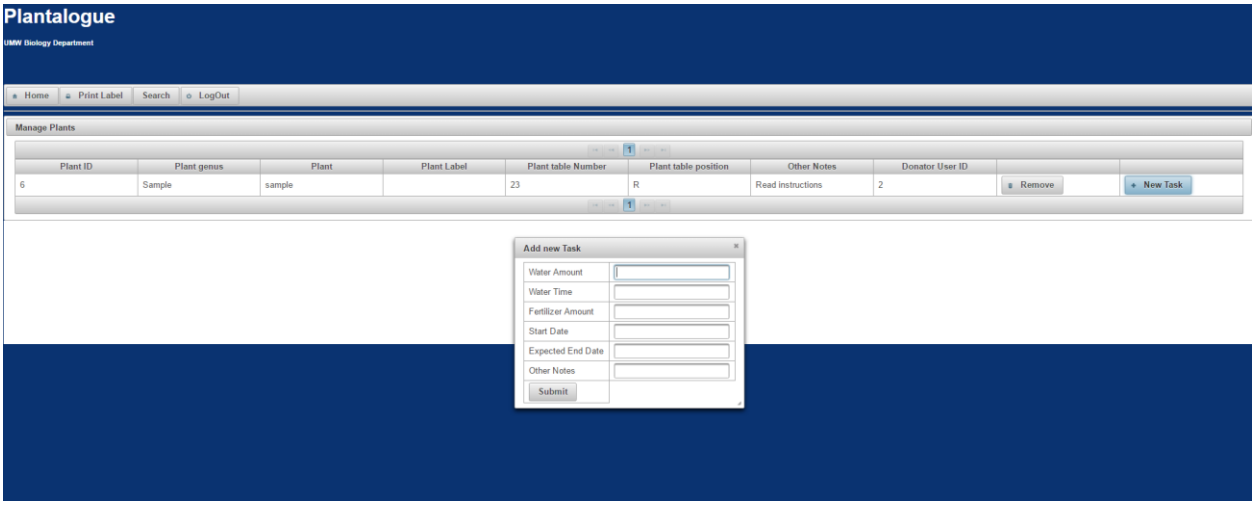

Figure 18 - Add new Task

Figure 18 Demonstrates how Task creation works in Plantalouge. The admin can select the New Task button by clicking it, and a popup menu shows up with the empty text field to be

filled. The admin can enter the task information in the text boxes and finalize the addition by hitting Submit.

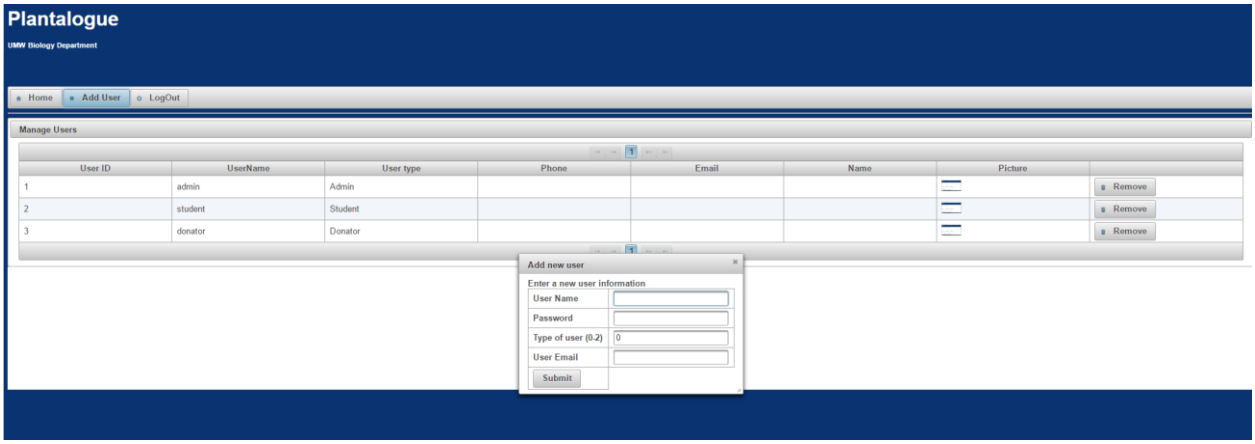

Figure 19- Add new User

Figure 19 Demonstrates how user creation works in Plantalogue. The admin can click the "Add User" button and a popup box appears with blank information fields that the admin can enter the required information about the user. Finalizing the addition is done by clicking the "Submit" button.

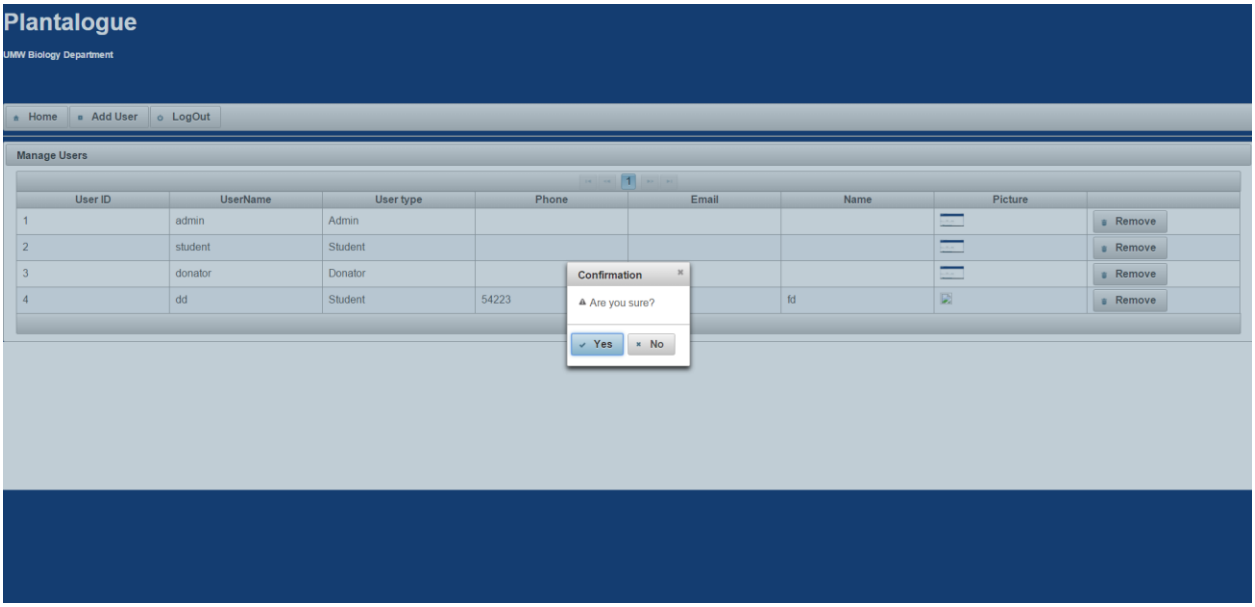

Figure 20- Remove

Figure 20 shows how deletion works in Plantalouge. The admin can click the "Remove" button to delete a specific element that can be a user, plant, or task. Then a confirmation dialogue appears to confirm the action that the admin is about to take, when the admin confirms deletion, the database will be updated and element will be deleted.

Figure 21- Calendar Page

Figure 21 functionality for this page allows admin and worker users to see a visual calendar that displays work schedules and upcoming tasks they have either created, or have been assigned. This page is fairly simple and straightforward. The only difference the admin and worker user's experience in this page is the content they are allowed, or not allowed to view.

6.3 Screen Objects and Actions

Menu Items: The Menu item are tabs, or links to different section of the website. These are ever present when navigation the website and they are the main way to traverse the website.

New Element Popups: These popups will appear when the admin chooses to create a new element. Required information must be filled before the popup will allow the admin to finalize the entry.

Confirmation Popups: These popups will appear when the admin selects irreversible options. The Admin just has to confirm to allow the change to occur

7. Requirements Matrix

Functional Requirement	System Component	SRS Doc Page #
3.3.1	Manage Plant Page	11
3.3.1	Manage Plant Page	12
3.3.3	Add Plant Page	13
3.3.4	Manage Plant Page	14
3.3.5	Manage Plant Page	15
3.3.6	Admin Home Page	16
3.3.7	User Profile Page	16
3.3.8	User Profile Page	17
3.3.9	Login Page	18
3.3.10	User Profile Page	20
3.3.11	User Home Page	21
3.3.12	Manage Task Page	21

3.3.13	Task Popup	23
3.3.14	Manage Task Page	24
3.3.15	Manage Task Page	25
3.3.16	Manage Task Page	26
3.3.17	Calendar Page	27
3.3.18	Admin Welcome Page	28
3.3.19	Create Label Popup	29
3.3.21	Create User Popup	31
3.3.22	Manage User Page	32
3.3.23	Manage User Page	33
3.3.24	Manage User Page	34

Figure 22- Requirement Matrix