Use Case Descriptions:

Table 1. Use Case Descriptions

Use Case: Select Crop to get Notifications From
Primary Actor: Gardener
The goal in Context: Allow the gardener to select which crops they want to receive notifications for so they can better focus on the crops they believe they would need notifications for and allow them to filter out notifications for crops they do not need.
Preconditions: The gardener has added a crop to their garden.
Trigger: The gardener selects to see notifications for the selected crop.
Scenario:

1. The view sends a request to the controller to save this crop to the user's notification list.
2. The controller sends a request to the model to save the crop of the user's notification list database.
3. The model saves the crop to the user's notification list database.
4. The model sends a success verification to the controller and sends the watering frequency value to the controller.
5. The controller will calculate when the next day the crop should be watered and send it to the model.
6. The model saves this value into the crop database.
7. The model sends a success verification to the controller that this value was saved.
8. The controller sends a success verification to the model.
9. The view updates/refreshes the page and shows the "plant was successfully updated" message.

Exceptions:

1. *The user no longer wants to see notifications for this crop.* The same scenario will occur, except the action of the model will be to remove the crop rather than add the crop to the notification list database.
2. *The user deletes this crop from their garden.* The crop will also be removed from this notification list database.

Open Issues:

1. Maybe the user should also be able to select everything in their garden rather than select each crop individually. This would give the user more convenience, but this may have to be a stretch goal since I want to focus on getting the key features of this project working first.

Use Case: View the Frequency of How Often a Crop Should be Water and When to Water it Next
Primary Actor: Gardener
The goal in Context: Allows the gardener to see how often a crop they have planted should be watered so they have this information for reference. The gardener does not need to be receiving notifications for this crop to view this information. However, if the gardener does have notifications enabled, they will also be able to see when they are expected to water their crops again.
Preconditions: The gardener has added a crop to their garden.
Trigger: The gardener is viewing their crop's page.
Scenario:

1. The view sends a request to the controller for the crop's data
2. The controller sends a request to the view for the crop's data
3. The model retrieves the watering frequency data from the crop database and the date of when to water it next
4. The model sends the data to the controller
5. The controller sends data to the model
6. The view displays and formats the data

Exceptions:

1. *The crop has notifications disabled.* The value of the "when to water it next" date will be returned as 'unknown' if the crop has notifications disabled. The view will not display anything for the "when to water it next" data.

Open Issues:

1. How will I be able to edit the UI that is currently implemented for a crop's page and allow for the information I want to display to easily integrate into the page?
2. A feature that I would like to implement is to make the watering frequency more predictive. In being predictive, it would use information like weather, sunlight, location, etc. to determine how often the crop should be watered and if it already has been watered (ex. It rained today). However, due to the number of components and features in this feature, I will have to keep this feature as a stretch goal.

Use Case: View Watering Mail Alerts
Primary Actor: Gardener
The goal in Context: Allows gardeners to see daily notifications that let them know which plants they need to water. Under the current growstuff implementation, the system will do a daily check to see if it has to send out any plant notifications, this system architecture will be the baseline to determine when the system sends out this mailing notification.
Preconditions: The user has a crop that needs to be watered and they selected to see notifications for the crop.

Trigger: The system performs its daily notification check (the "send_planting_reminder" task is called).

Scenario:

1. The task sends a request to the notifier class to see an email to the user about their current crops that need to be watered.
2. The notifier identifies all the crops the user has that need to be watered.
3. The notifier sends out an email to the user that lists out what crops need to be watered and what garden they are from.

Exceptions:

1. *Email is not found.* When using the notification system, there will be a message that pops-up that lets the user know that they should verify their account's email if they want to use this feature.
2. *The user currently does not have any crops selected to receive notifications for.* Before sending out the daily email, the system will check to see if the user has any crops selected before doing the notified class processes.
3. *The user currently does not have any crops that need to be watered.* Before sending out the daily email, the system will check to see if it could find any crops that need to be watered, and if it could not, then it will not mail the notification.

Open Issues:

1. How do I make this notification system informative, but, at the same time, do not spam/annoy the user with notifications?
2. Maybe the users should be able to customize what time and how often they receive their watering notifications. But if I were to attempt to implement this, would I have enough time to do so? Maybe this may be better off as a stretch goal feature?

Use Case: Check off Which Crops Were Watered Today

Primary Actor: Gardener

The goal in Context: Allows the gardener to select which crops they have watered today. This will prevent the gardener from receiving notifications for their crop until it is the next time to water them.

Preconditions: The user has a crop that needs to be watered and they selected to see notifications for the crop.

Trigger: The gardener has accessed their watering checkoff list page.

Scenario:

1. The gardener checks off a crop they watered.
2. The view will send a request to the controller to notify the database that this crop has been watered.
3. The controller sends a request to the model to get the crop's watering frequency data.

4. The model sends the watering frequency value to the controller.
5. The controller will calculate when the next day the crop should be watered and send it to the model.
6. The model saves this value into the crop database.
7. The model sends a success verification to the controller that this value was saved.
8. The controller sends a success verification to the model.
9. The view will display a success message to the user to indicate that their checkoff was registered.

Exceptions:

1. *The gardener wants to undo a checkoff.* When a gardener checks off a task, it overwrites the next day the gardener should water the plant. So when the gardener unchecks the task, it will restore the next day the gardener should water the plant to the value it was previously.
2. *The gardener has no plants registered*. Instead of displaying a checklist, the page will display a message letting the gardener know they have no plants currently registered.

Open Issues:

1. This will require me to make a new page for the site that allows the user to check off what crops they watered. How will I be able to implement this page and match the growstuff architecture? And, how will I make the UI of this page so that it will be easy and convenient for the user?
2. A feature that I would like to implement is to make the watering frequency more predictive. In being predictive, it would use information like weather, sunlight, location, etc. to determine how often the crop should be watered and if it already has been watered (ex. It rained today). However, due to the number of components and features in this feature, I will have to keep this feature as a stretch goal.