Recitation Number	:	3038-101	**Team Name**	:	Team Bonsai
Team Number	:	1	**Application Name**	:	Bonsai Buddy

Team Members

- Leo Huettel
- Luke Bakken
- Brandon Glandt
- Hassan Alsahli
- Madison Rivas

Application Description

Bonsai Buddy is a web application used to monitor the health of a bonsai tree. Bonsai Buddy is the perfect tool for bonsai lovers and plant lovers who don't have the time, or frequently forget to water and properly maintain their trees. The app will allow users to remotely water their tree, set a timer schedule, or allow the application to water the plant automatically based on sensor input. This can be optimized to save water or promote growth.

Sensors will include : photo-resistivity, soil-moisture, temperature, humidity.

Functions will include: camera, pump, lights.

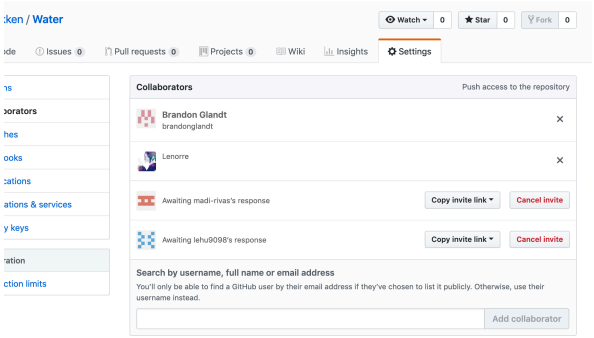
Our Vision Statement

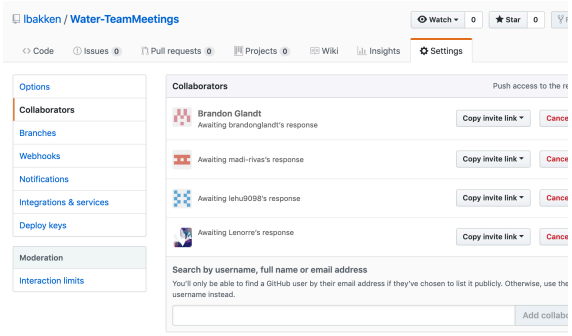
Bonsai care and management requires daily attention. This application is designed to remove some of that burden and allow bonsai owners to spend more time enjoying their plant and less time worrying. The website will allow an owner to remotely monitor the condition of their bonsai while they are away for vacation or work.

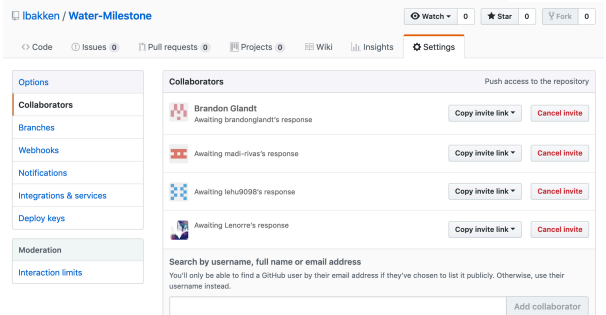
Version Control

Team Bonsai Buddy will be using GitHub for version control.

- Project Code / Components : https://github.com/lbakken/Water
- Milestone Submissions : https://github.com/lbakken/Water-Milestone
- Team Meeting Logs : https://github.com/lbakken/Water-TeamMeetings







Development Method

Agile Development

Group members will work on stories during two week sprints. The stories will be broken down into atomic tasks, using the following framework:

- As a ___ (User, Developer, Project Manager, etc.)
- I want ___ (An exception to be thrown)
- So that ___ (It is easier to locate errors)

Stories will be broken down into fundamental, atomic tasks which cover a very specific, single use case.

Example:

- As a user,
- I want to login from a dropdown menu
- So that I do not need to wait for a new page to load

This story tasks the developer to move the login widget into a dropdown menu, rather than forcing a redirect to a separate login page. The time taken to finish stories will be minimal. The story will only cover one atomic idea.

All stories will be decided by the Project Manager, Luke Bakken, before the start of each sprint, and managed at the week mark to decide if new stories are necessary and if some stories should be removed.

The first sprint will begin Wednesday, February 27th. The first sprint will focus on initializing the application framework.

The strategy used to maintain the codebase follows these rules:

- **NO** direct commits to Master branch
- Sprints will have their own "development" branch, which branches off of Master
- Members will branch off of development for their stories
- Story branches are then merged back into development
- Development is then merged into Master at the end of the sprint

Details of this strategy are listed in the README on https://github.com/lbakken/Water

Communication Plan

- General Communication : Slack Channel
- Collaboration Tool : GitHub Pull Requests
- Task Manager : Trello Board

Slack will act as the main channel for communication between group members. Questions about meeting times, story clarification, collaboration between group members, and all other networking communication will be handled within Slack.

GitHub pull requests will be used to finalize code changes for the project. Each pull request must be approved by the project manager, Luke Bakken, before it can be merged into master. This is further outlined in the Development Method section above.

Trello will be used to manage the stories for each sprint. The project manager, Luke Bakken, will outline all stories necessary for each sprint before the sprint begins. Group members will then move each story, one at a time, from "to do" → "doing" → "done"

Proposed Architecture Plan

Languages to be used :

- NodeJS
- HTML
- Python
- CSS
- C
- PostgreSQL

Express.js : Express is a NodeJS framework
Heroku hosting : PostgreSQL hosting
Raspberry Pi 3 + Arduino Nano : Sensor and Relay Control

Express will act as the connection between the frontend and backend of the application. The frontend will be constructed using HTML, CSS, and NodeJS. The NodeJS will also act as a router between pages and interact with the Heroku backend servers and the Raspberry Pi. The Raspberry Pi will be running a Python script to collect sensor data, normalize it, then post it to the Heroku PostgreSQL database. The Raspberry Pi will also be listening for commands to water the plant or turn on the lights. The Arduino will be used to collect analog sensor input, serialize it, and send it to the Raspberry Pi via serial connection to be normalized and sent to the Heroku PostgreSQL database.

Meeting Plan

Wednesdays, 4:00 pm → 5:00 pm in CSEL

This time will be for group members to discuss upcoming stories and choose their workload for the upcoming week. Group members will be available to discuss problems with their code and get in person help completing stories.