Beaver Botanica: The OSU Plant Map

GitHub Repository and Trello

Github
Trello

Communication Channel and Rules

Communications Channels:

- MS Teams
- Text group chat

Rules:

- MS Teams will be used for non-imperative communication. Non-imperative communication includes questions, discussions, or helpful status updates regarding the group project.
- All members must check MS Teams once a day and respond within a day of a message being sent.
- A texting group chat will be used for conveying time-sensitive information. If a problem has to be addressed within a day then it is considered time-sensitive.
- All member must be able to receive and respond to a text message within 4 hours of it being sent.

Product Description

Abstract

The campus of OSU is home to a wide diversity of plants contributed by talented botanists over the years. It is hard for new botany students to know where all these plants are. They are left with little option save for asking for help from professors or exploring the campus themselves. Our goal is to help these botany students by providing them an online map that can tell them where plants lie on OSU's campus. This project is a plant identification app for the Oregon State University campus. Users will take pictures of plants around campus, identify them, and upload them to a website that will show a map of campus with the locations of each identified plant. There will be layers to the map for each season, so users can see what plants are in bloom at any given time. Each plant, when clicked on, will show details of the plant that the user provides.

Goal

Our goal for this software before the end of the term is to create a map of plants on campus and allow for users to upload pictures and descriptions of plants to OSU Plant Map's database. Our end goal is to allow any student at OSU the ability to look-up and identify plants around the OSU campus.

Current Practice

Plants are everywhere in Oregon. In this state alone, forests cover 30.5 million acres, or almost half the state. With how important plants are in our day-to-day lives, these resources below are some of the only ways we can learn about our local flora:

- The botany department currently has a tree inventory map, but it is specific to trees and is simply an inventory system.
- The website Pl@ntNet has a global plant map, but it only shows broad regional areas of where plants grow.
- The website OregonFlora has a Oregon plant map, but it can be hard to search through and is unhelpful to non-botanists.

It's because of this that we want to be able to create something that is both in-depth and local to what we live with here at OSU. An application like this can increase awareness and education about local flora while keeping it open for anyone to use regardless of their background in botany.

Novelty

The novelty of this project is that it will be a social media plant identification app specifically for the Oregon State University campus. Making our app restricted to the OSU campus will allow us to make the plant map more "zoomed in" on an area. This is novel because most maps of plants that are out there are catalogs of plants in a region, which don't allow users to find more specific locations of plants. Our app will be a user driven plant identification system that will allow users to identify and map plants on campus themselves, as well as drive social interaction based around the plants.

Effects

The target audience of Plant Map will be botany students and other such interested parties at OSU. With this software they will be able to easily look up and find plants around the OSU campus. This will save them time looking around to identify a plant and it will also help them study various plant types at OSU. Additionally, this software will help entertain visitors and regular students of OSU by giving them a tool to find cool plants around campus that they may not have otherwise known existed.

Technical Approach

The idea is to use a SQL database that holds each plant identified by the users. The website will display a map and overlay each plant's location using the Google Maps API and querying the database through code. We are still deciding on how exactly we will build the website.

Risks

One of the largests risks with developing OSU Plant Map software is in making sure the system is scalable. We, as a 7 person team, will not be able to map out all the plants spanning the whole OSU campus. Our solution to this will be allowing users of OSU Plant Map to upload pictures and information to the database.

With the large amount of plants and people we will need to make sure our software can be scaled up to allow for more information. This will be difficult as we will have to think ahead and design our system carefully. To mitigate this risk, our team will research early on how to make our system scalable so that we always code with the expectation of scalability.

Major Features

- Movable map interface that displayes plant locations
- Seasonal layers on the map
- Plant descriptions and details
- Social media features such as commenting and liking plants

Stretch Goals

- User profiles and leaderboards
- Plant identification AI integration

Requirements

Functional Requirements

Searching for a Plant - Adison

Actors: A OSU Student
Goal: To search for a plant
Triggers: The user can recognize the plant based on the image displayed after filtering the results.
Preconditions: the user searches for the plant by sorting through media results.
Step: The user sorts the media by taxonomic classification.
Postconditions: The plant they are looking for will be displayed with a detailed description of what it is.
Extensions: The user is unable to find the plant they are looking for, but is able to change that by registering it into the software.
Exceptions: The plant does not show up because it has yet to be catalogued.

Adding a Known Plant to the Database - Kathryn

Actors: A OSU student or faculty member.
Goal: To add a plant instance to the database. The plant type is already in the database.
Trigger: The user fills out all plant information and clicks the “submit” button.
Preconditions: the user searches for the plant by sorting through media results.
Steps:

- The user begins to add the plant, through clicking on an “Add plant” button.
- The user inputs the common name and scientific name of the plant.
- The user is prompted to take a picture of the plant.
- The user is asked to give a location on a map where this plant was found.

Postconditions: The plant instance is added to the database with a label of “unverified” or a similar marking to avoid mislabeling plants. The map is updated to add the plant instance.
Extensions: The user could decline to take a picture of the plant, which could still go though as a successfully added plant instance. The user might also stop in the middle of adding a plant, in which case the software would either save the progress or discard it.
Exceptions: The user may not find the plant they’re looking for, in which case they would be directed to a form requesting the addition of a new plant.

Commenting on a Post - Luke

Actors: A OSU Student
Triggers: The user fills out a comment box with text (maybe with images?) and clicks submit.
Preconditions: The user is logged in to the website and on the map page. At least one post has been made.
Steps:

- The user clicks on the desired plant post on the map
- The system displays the plant details page

- The user clicks on the comment button
- The user enters text into the comment box
- The user clicks the submit button
- The system refreshes the page and displays the comment at the end of the comment chain for all users.

Postconditions: The comment is now displayed for all users on the post.
Extensions:

- The user can cancel their comment by clicking on the cancel button or by navigating away from the page.
- The user times out
- The user saves the comment as a draft
- The user is an administrator and pins the comment

Exceptions:

- The text contains invalid characters
- The text contains a spam link
- No text is provided
- An image is the wrong format
- The post is deleted before the comment is posted

Verifying User-Added Plant Entries - William

Actors: Beaver Botanica moderator
Trigger: The user clicks on the verify button on a plant post.
Preconditions: A plant entry with the unverified property is in the database.
Steps:

- The user logs in using their moderator account.
- The user clicks on one of the plants on the map.
- The user can scroll through the description of the plant to check its attributes.
- If the plant is unverified the user clicks on a verify button to verify it.

Postconditions: The plant entry that had the unverified property gets the verified property.
Extensions: In the case that a moderator clicks on a plant entry and does not want to verify it, they can remove the plant entry by clicking on the delete post button. This will remove the plant entry from the database and map.
Exceptions: In the event that clicking on a verify button does not add the verified property to a plant, the user will receive a message notifying them that the plant was unsuccessfully verified. If a plant was already verified then clicking on the verify button will not change the property of the plant, and the user will receive a message notifying them the plant was already verified.

Editing a Plant’s Details - Anshu

Actors: OSU Student or faculty member. Goal: To edit the details of the plant entry in the database. Trigger: On the plant's detail page, the user selects the "Edit Plant" button.
Preconditions: The user might be the original submitter and has the authority to change the plant data. Steps:

- The user accesses the information page of the plant.
- The user selects the "Edit Plant" button.
- The system shows a form with the current plant information already filled in (e.g., common name, scientific name, description, location, image).
- The user changes the desired fields, like adding a new image or changing the plant's name and description.
- User clicks on the "save change" button to confirm the edits.

Postconditions: New plant details have been added to the data. The plant details page reflects the uploaded information.
Extensions: By using the "cancel" option, the user can edit, remove the modifications, and return to the plant's information page. If a user uploads an image in a format that isn't supported, the system will prompt them to upload an image in a format that is accepted such as JPEG or PNG. Exceptions: When a user leaves a mandatory field empty, the system indicates an error and asks them to fill it up. When the user enters an incorrect value, such as a special character in the plant name, then the system verifies the information and displays an error notification.

Navigating to a Plant's Location - Finlay

Actors: A user of the Beaver Botanica app.
Goal: The user should be able to find the locations of a certain plant on the OSU campus and get directions to the plant's location.
Triggers: The user is viewing a specific plant after searching and then presses the button that opens the map view. The user then selects a location and chooses to get directions to it.
Preconditions: At least one plant is in the database and that plant has at least one location placed on the map.
Steps:

- Selects a plant after entering a search query
- Open the map view for that plant to view all locations of the plant
- Select one location marker and click the "Get Directions" button
- The application will then display directions to reach the selected location

Postconditions: The application displays a map view with any locations for the plant displayed as markers.
Extensions: The user might not choose to receive directions to the plant's location, instead navigating by themselves.
Exceptions: A plant might not have a location associated with it, in which case the map would not display any locations and the user would not be able to get directions. The mapping service we use might also be down, which would cause no map to be displayed to the user, meaning they couldn't get directions.

Filtering the Map - Jake

Actors: An OSU Student
Goal: To view the locations of all plants in a specific Genus that are in season
Triggers: The user clicks the filter button
Preconditions: The user is on the Map page and has a specific Genus in mind they want to see locations for plants that are in season.
Postconditions: The map shows the locations of all plants that match the given filter.
Steps:

- The user clicks the filter button
- The system displays the filter sidebar (this will hide the add or Info sidebars)
- The user either selects the desired Genus from the Genus dropdown or starts typing in the dropdown to find it faster (for got what this input type is actually called)
    - The system should then fill higher level taxonomy filters
- The user checks the box in season
- The System Filters the map based on the set filters

Extensions/variations:

- The user changes their mind and wants to clear the filters this can be done with the clear all button
- The user decides they actually wanted a different genus in the same family
    - Now they just select the genus dropdown and it should only show genus in that family

    - Should also clear all taxonomy filters below genus.
- The user decides they actually wanted to see the whole Family
    - The user clicks the clear button for Genus

Exceptions:

- The user can’t find the desidered genus in the drop down
    - This should mean that there are no plants in that genus in the database.
    - Maybe suggest Advanced Search as the result might be missfiled.
- There is nothing on the map
    - This means that no plants of that genus are in season.
    - Should display a message that no plants are in season for a given filter and recommend turning off the in season filter.

Non-Functional Requirements

Dark Mode and High-Contrast Mode

Both modes improve user accessibility by providing alternative color schemes, such as dark mode for low-light environments and high-contrast mode for users with visual impairments or color blindness. They must load without impacting software speed, and may automatically adjust based on the user’s device settings. The user must be able to activate dark or high-contrast mode in no more than four clicks. The colors used in high-contrast mode must be tested to comply with industry standards.

Expandability

The software should be able to store and display all plants on the OSU campus (2000+) and also allow them to be searched. With the software allowing more plants to be added, it enables the users to add plants that aren't already in the system so they can all be catalogued.

Incorrect/Malicious Entry Prevention

The software should prevent users from creating incorrect or malicious entries for plants and/or their locations. It should be built in a way that totally prevents these entries being displayed as correct information or removes it from the database and software quickly.

External Requirements

Error-Free

The software needs to be able to prevent errors and crashes from occurring. This can be done by being able to look out for common system errors such as no results popping up when a user searches for something, invalid inputs, and failure to save things. Certain types of input such as images with an unknown format or non-valid characters will be sanitized or fully blocked to prevent errors.

Installable

The website will be accessible to a range of devices, allowing more users to access the service. A mobile version of the website is absolutely necessary since users will be walking around with it on their phones. The URL will be publicly available.

Buildable

The software should have up-to-date documentation that allows other developers to set it up, allowing the project to extend to other college campuses or locations. The software should also allow developers to customize their version of the application by doing things like using a different map, different plants, or any other changes to features.

Scope

This software should be well-designed and have a robust number of bug-free features, as listed in the functional and non-functional section of the Project Proposal document.

Timeline

Week 3 - Planning and Foundation

Goals: Finalize requirements, set up basic infrastructure, and assign roles.
Tasks:

- Finalize the app's functional requirements.
- Break down tasks into use cases.
- Assign roles to team members (e.g., developer, UI/UX designer, documentation).

Backend Team:

- Design a basic database schema to store plant information and user data. Measureable Goal: Have plant table, and user tables.
- Set up the development environment (e.g., version control, frameworks, initial database setup). Measureable Goal: Have all things needed installed and folder structor created

Front End Team:

- Begin wireframing the user interface (UI). Measureable Goal: Have all pages designed so can get feedback start of next week.

Interaction Team (Front end to Back End Interaction Team):

- Help out the other teams.

Deliverables:

- Project Requirements Elicitation.
- Development environment ready.
- Wireframe drafts for key screens.

Week 4 - Map and Database

Goals: Develop the interactive map and basic database functionality.
Tasks:

Backend Team:

- Create the database structure for plant information, including taxonomic classification, images, and locations.

Front End Team:

- Continue refining wireframes based on feedback.
- Implement a basic campus map.
- Enable users to view plants pinned on the map.

Interaction Team:

- Enable users to view plants pinned on the map.
- Integrate the database with the map to display plant data dynamically.

Deliverables:

- Project Architecture and Design Specifications.
- Functional campus map with placeholders for plants.
- Database connected to the map.

Week 5 - Plant Identification and Upload

Goals: Implement plant identification prompts and the information and file upload features.
Tasks:

Backend Team:

- Allow for the database to store photos

Front End Team:

- Design the UI for adding plants.
- Allow for a user to upload a photo.
- Allow someone to identify a plant based on the photo.
- Allow a user to add information to an identified plant.
- Add prompts to help users identify plants.

Interaction Team:

- Allow for photos to be uploaded.
- Allow for plant data to be added and deleted.
- Seamless updates to the system when a plant is uploaded, updated, or deleted.

Deliverables:

- Updated Database.
- Functional plant upload and identification system.

Week 6 - User Authentication and Basic Social Features

Goals: Implement social features and user authentication for the software.
Tasks:

Backend Team:

- make sure user table works
- add likes table.

Front End Team:

- make sign in form
- make account creation form
- make logged in indicators
- Implement social features such as liking an identification.

Interaction Team:

- Allow a user to sign in and out of the system.
- Handle account creation.

Deliverables:

- Functional authentication system.
- Functional social features.

Week 7 - Advanced Map Features and Seasonal Layers

Goals: Add and improve map features and create seasonal layers.
Tasks:

Backend Team:

- make sure database can hadle filtering and search.

Front End Team:

- Build Map filter form.
- Build Advanced search page including sorting plants by name, location, or type (e.g. trees, flowers)

Interaction Team:

- Impliment Map filters api calls

- Implement advanced search functionality.

Test Team:

- Test the map with fake plant and seasonal data.

Deliverables:

- Project Implementation.
- Map filter functionality
- Search functionality.

Week 8 - Testing and Refinement

Goals: Make sure all core features work as expected, and refine them based on user feedback.
Tasks:

Backend Team:

- Fix bugs and usability issues found from tests.
- Make sure the database can handle edge cases (e.g. duplicate entries, incomplete data?).

Front End Team:

- Fix bugs and usability issues found from tests.
- Improve the UI/UX design based on feedback.

Interaction Team:

- Fix bugs and usability issues found from tests.
- Make sure the database can handle edge cases (e.g. duplicate entries, incomplete data?).

Test Team:

- Conduct user testing with students, faculty, and/or others.

Deliverables:

- Project Testing and Beta Release.
- Refined app with fewer bugs and a smoother interface.
- Feedback report from user testing.

Week 9 - Finalizing Additional Features

Goals: Complete non-functional features and continue refining the software.
Tasks:

Backend Team:

- Implement the non-functional features.

Front End Team:

- Continue to improve the UI/UX based on feedback and integrate the non-functional features.

Interaction Team:

- Implement the non-functional features.

Test Team:

- Test that the non-functional features don't interfere with the core features of the software.

Deliverables:

- Completed non-functional features in the software.
- Fully-functional app ready for final testing.

Week 10 - Final Testing and Launch

Goals: Test the app and prepare for the official launch.
Tasks:

Backend Team:

- Fix any remaining bugs and performance issues.

Front End Team:

- Fix any remaining bugs and performance issues.

Interaction Team:

- Fix any remaining bugs and performance issues.

Test Team:

Documentation Team:

- Finalize Documentaion

All:

- Prepare our final project presentation/demo.
- Deploy the app for public use.

Deliverables:

- Project Final Release.
- Fully tested and deployed app.
- Final presentation/demo.

Team Process

Members

Adison Daggett - Front End Designer and Developer

The role of the front-end designer has two parts: designing the front end and implementing it. This role allows for consistent design and implementation of an agreed-upon front end for the software.

Kathryn Butler - Technical Documentation and Developer

Technical documentation is an incredibly important role in software engineering. Our team needs this designated role to clarify the process for adding and verifying plants, detail the system's database integration, and outline user features such as the search and map functionality. Kathryn is suited for this role due to her strong communication skills developed from her leadership roles in film production and peer tutoring. She also structures documentation effectively, making it easy for others to navigate and use as a reference.

Luke Scovel - Team Coordinator and Developer

For a team to function, it has to be a collaborative effort with everyone on the same track. To disseminate important information and reiterate what tasks need to be done without multiple voices clamoring over each other, a coordinator is needed. More developers are needed to bounce ideas off of each other and do the grunt work of coding.

William Brennan - Developer

A developer is someone who designs and writes the code to be implemented in the software. This role is needed for this project because we need someone to design and program the database and logic controller for the software. William Brennan is suited for this role because he is a computer science student and he has taken courses in web development.

Anshu Avinash - Front-End Designer

By bridging the gap between functionality and user experience, a front-end designer makes sure that the user interface's implementation and design meet the needs of both users and business objectives. Also, it streamlines the handoff between design and coding, improving productivity for the entire team.

Finlay Curtiss - Developer and Technical Documentation

Multiple developers are needed to ensure that all features are implemented properly and in a reasonable time frame. Finlay has experience with developing and can contribute their knowledge and perspective to the technical implementation of the software. Technical documentation is also necessary, for the development team, users, and any developers who wish to work on this software in the future. Finlay has experience with technical writing from their classes and job.

Jake Thompson - Developer

We need many devolopers due to the timeline of the Project and the scope of what is needed to be implimented. Without developers nothing gets made. Jake has experience with developing database design and other skills needed for this project.

Software Toolset

We will be building Beaver Botanica as a web application hosted on OSU's engr server. We will use the following frontend and backend software to achieve this goal.

Development Tools:

- MS Teams - Communication and file sharing.
- Trello - Task tracking and assignment.
- Git/GitHub - Version control and project management.

Frontend:

- HTML - Will be used for all the text on the website. We need this to have plant names and descriptions.
- CSS - Will be used for styling the website. We need this to make the website have a consistent theme.
- JavaScript - Will be used for making the website interactable. We need this to be able to click on plants to learn more about them.

Backend:

- SQLite or MySQL - Will be used for creating the database. We need this to have a database full of plant names and their properties.
- Node JS - Will be used for managing the database and handling requests. We need this to process plant entries and add them to the database.

Risks

- **Scalability**: One of the largest risks with developing Beaver Botanica is in making sure the system is scalable. We, as a 7 person team, will not be able to map out all the plants spanning the whole OSU campus. Our solution to this will be allowing users of OSU Plant Map to upload pictures and information to the database.

- **Data Accuracy**: Ensuring the accuracy of plant data uploaded by users is a significant risk. Incorrect or misleading information could reduce the app's reliability and usefulness. To mitigate this, we will implement a verification process for user-submitted data.

- **User Engagement**: Another risk is the potential lack of user engagement. If users do not participate in uploading and verifying plant data, the app will not reach its full potential. We will address this by incorporating peer-to-peer interaction elements to encourage use.

- **Technical Challenges**: Developing this application involves overcoming various challenges such as integrating the Google Maps API, ensuring cross-platform compatibility, and maintaining a responsive design. To mitigate this, we will allocate time for testing and debugging and seek assistance if needed.

External Feedback

We will be getting external feedback in week 6. All of the core features will be done by then so testers will be able to give useful feedback while still leaving us time to refine our product. We have dedicated week 7 to testing and refinement based on feedback. We will get this feedback by giving a fellow student in CS 362 our software. We will later interview them by asking them the following questions.

- What aspects of Beaver Botanica did you like or not like? Why?
- What would you like to see added or removed from Beaver Botanica?
- If there was one change you could make to Beaver Botanica, what would it be?
- Did you find Beaver Botanica useful and will you use Beaver Botanica in the future?