Team 6

By Harriet Godward, Harlie Scarratt,
Lesley Lopez, Oviyah Ravikumar, and Tess Afansyeva

CFG Software Degree, Spring 2023

Introduction

As plants are increasingly becoming a focal point for our homes, it's becoming clear that the hectic nature of modern life makes it difficult for many of us to keep plants alive. We enjoy bringing nature indoors; however, we are not equipped with the knowledge to care for them properly. There are benefits to having plants in your home, such as improved air quality and a sense of serenity (Mayers, 2018). This study also showed that the main reason for plants dying is over or under watering. This leaves us with the problem of how to keep the benefits and lovely décor when the plants always seem to be on their last legs. By providing users with the ability to log the last time the plant was watered and sending out reminders of the next watering date via email, "Thirst Trap" makes keeping plants alive and well a breeze.

The application features Perenual API, which has a large database of plants and their required watering frequency. This API allows for ambiguous search, both for lay and scientific names and accommodates users that do not remember the exact type of their plant. Together with other plant features, the watering frequency is stored in the MySQL database, which is then linked with the backend code and the frontend website. The application stores the date it was last watered in the database, it then calculates the date the plant needs to be next watered and sends the email notification. This feature removes the possibility of users swiping the reminder away and forgetting about it, as is easily done with mobile notifications. In the future this feature will have a calendar integrated. There is also a ChatGPT-based API tool which provides fun and useful facts about the plant of interest, with the aim of helping the user create the perfect environment for her green friend and give it the best opportunity to flourish. In addition, having a user interface for plant information on the website allows for easy interaction.

Here, we report the successful completion of the stated objectives in the timeline of the project as outlined in the previously submitted project plan. This report goes over the basis for our project, the challenges solved, and the tools used. It also provides an overview of a high-level timeline, architecture, and testing strategy of the code, as well as outlines the necessary Python libraries for the successful deployment on a non-native machine. In addition, it described the split of the tasks and the work done by each member of the project. Finally, it presents our ideas on the further improvements of this application. If extended with a user-friendly mobile or desktop front end, the code written in the duration of this project has significant potential for commercial deployment.

Specification and design

Perenual API

Early in the project design, we successfully overcame the challenge of selecting the appropriate API to retrieve information on plant care. We looked at a range including Plant.id, Trefle.io, Garden.org and, Houseplants by Mnai. The Perenual.com API met all our requirements as it didn't require an image of the plant, it provided the watering frequency in an easy-to-follow format, and it had a comprehensive library of plants.

Flask-based website

We created a website to improve the user interface. This allowed us to expand our tool kit to include Flask and HTML. The website for the project is laid out with an initial welcome where

you have the option to look up watering frequency for your plant without signing in on the right side of the page. You input the plant name and then are either taken to a page with a list of close matches or straight to the plant information. If you decide to not create an account, no data is stored, and you will not get notifications. You also have the option to log in or sign up. If you decide to sign up, our intention was that you would be taken to a welcome page where you can find a guide on how to use the website and how to use the search tool to find your plant. Once you have chosen your plant you are taken to another page with the watering frequency displayed. The latter will be displayed within the pycharm terminal due to time limitations of the project.

We also added a manual to guide users through the application.

Back-end logic

Our minimal viable product is the app which asks a user for a plant name, and then gives reminders on when to water, with the user being able to tick off if a task is completed. To complete the functional application in the desired time frame, we decided to limit our range of notifications to email-based, however, the user of the Observer pattern in the design of the app would allow low time-consuming expansion toward other notifications, such as Android and OS push notifications. This module integrates API to allow the code to access the plant watering frequency, and an SQL database to allow for storage of the user and plant data.

Implementation and execution

Initially to decide roles we used the SWOT analysis and tried as much as possible to link people's strengths and their tasks. We stayed flexible and worked together through pair/ group coding and allowing collaboration on code through GitHub.

Throughout the project, we each had numerous roles:

Harlie's first focus was finding an appropriate API and setting up the API call, ensuring that the data we were using aligned correctly. She explored various routes of obtaining relevant name data and researched further into 'fuzzywuzzy' to ensure we found accurate matches. After this, she then helped Harriet and Lesley to problem-solve with the issues the group was facing with regard to the calendar and to-do list code. She then created the website using Flask and helped design the website alongside Harriet. Harlie worked on figuring out how to display the final project either through flask entirely or explored creating an iframe to display our program. Thanks to Harlie we also have a cool name, background and logo for the project.

Tess created the custom GPT-based API to which the application sends requests for information about the provided plant species. She then wrote the main.py, reworked the logic for the calculation of the next watering date, and user login modules, and reformatted the remaining modules with the exceptions of Harlie's API to follow OOP practices and to play together nicely. Whilst writing logic for the calculation of the next watering date she also wrote the tests, however, those tests were obsolete at the later stages of the development and were later taken over by Oviyah. She also edited the final report.

Lesley created the database and wrote the code for the email notification system which is an integral part of the project. She also helped to solve the issues with the notification and watering code. Lesley also worked on assisting in ensuring Harlie's flask design matched the code that was being developed.

Harriet worked on the code for the calendar and task list using the data from the API, with this information she then coded the logic for the next watering date. She stylised the website and

helped to create the user profile and website architecture. She also wrote the project proposal and report and was influential in keeping the group moving forward.

Oviyah worked on testing the database, the API, and the logic for the calculation of the next watering date. She also created the aesthetic design of the powerpoint to be used at a later stage as well as a rough script and format for the presentation.

Overall, the project progressed at a suitable pace, aligning with our initial goals. Whenever necessary, we made decisions as a team to adapt and make changes. Initially, we devoted time to brainstorming and gathering resources for each project, leading us to choose the plant watering app. Our initial tasks included calling the API, retrieving data, and setting up the database. Simultaneously, we also implemented the AI fun fact code, recognizing that not all team members needed to be involved in the aforementioned tasks. Subsequently, we developed the notification function. Throughout the process, we ensured that all sections were integrated smoothly and could be easily called and tested. Refactoring was facilitated by our collective understanding of each other's code and our ability to modify it to ensure the project worked cohesively.

We had to review our idea of having a fully operational webpage instead of using the terminal. This was due to us coming to the end of the project and we were unsure if we could finish the merge of the code and website in time for the submission.

With the time constraints of the project and the flask code taking longer for us to work on it wasn’t possible for us to be able to fully join the flask app to the database code, this has meant that our project is very close to being a fully connected website, API, backend code and database but it is not there yet. We still wanted to show the work we had done on the website as none of us had worked in flask before and so the produced code is something we are still very happy with. The program is in two parts. One part displays an example of how the website would work using an accurate plant search (without login), and a login page. Once ‘submit’ is clicked after typing values into either the ‘create an account’ or ‘log in’ areas, the page redirects to a ‘page stop’ which will tell you to return to the pycharm terminal to see the whole of our program.

Implementation of the Flask webpage was a big challenge during the project process. Working with a new language of code with HTML and CSS allowed us to stretch ourselves and our knowledge. It was also the first major project for most of us so pulling all of the code together in one document to call all of the separate pages of code was a task that we hadn’t faced before and so it took organisation of our work and the team to get right.

For the most part, we managed to avoid having to wait for certain pieces of code by working in tandem for most of the project and then piecing it all together at the end. Whilst the joining of the code took quite a while it was worth this extra work as it may not have been possible to get as much of the code completed if we had been waiting for other tasks to be completed.

The required libraries that need to be imported for the code to run successfully are Fuzzywuzzy (fuzz), Datetime (timedelta, datetime), Requests, Flask (Flask, render_template, request), Mysql.connector, Steamship (steamship), Ssl, Smtplib, Unittest (Mock, Magicmock, Patch), Time, Abc (ABC, abstract_method), Threading (Threat), Faker (Faker) and Images from the original report.

Testing and evaluation

Our testing was done as a mix of tandem testing, whilst we coded and testing the code after it was completed. This was due to the personal preference of the person coding the unit tests.

We found that both ways worked well, and it was a team effort to check the tests and to try and break the code. This allowed us to have a wider range of tests that hopefully encompassed the majority of issues that could arise.

The user testing was focused on the user input with regards to their username and email address, this is thanks to the plant search simply returning no matches if the user input into the plant search bar was incorrect.

The functionality testing was where we had to focus our efforts. We needed to ensure that the database, API, watering logic and website could not be broken easily due to an unforeseen error.

There were many limitations throughout the project which if mitigated would have resulted in a more elaborate and professional outcome. The main issue was finding an API that was mostly free and broad enough for it to be useful. The API we used was one of the more extensive examples we found, however there are still limitations to this, as there are more plants we could have accessed if we had paid for the data. The data we extracted from the API wasn't very extensive and it would have been beneficial to have an API which provided us with more relevant information.

The website could be developed further, yet as mentioned previously, we have not worked with flask or HTML before and so we were limited by having no previous knowledge of the language. We have endeavoured to make a website that is also mobile friendly by ensuring the page format does not change when resized, however we would need to do further testing on this.

With the project only being four weeks long it would have been interesting to see how our work could have developed over a longer period of time. There were many limitations to the time we spent together such as some of the team having full time jobs, others being in full time education and us all having other time commitments. This meant it was sometimes difficult to organise group meetings and have time to do the project work.

Future directions

In our future work, we are primarily focused on developing an engaging website that facilitates user interaction, such as leaving comments and engaging in discussions. Additionally, we intended to incorporate both word and image search functionalities, but due to time constraints, we were unable to find an API that supported both features simultaneously. Currently, our website is in its early stages and not yet ready for consumer use. In order to bring it up to standard, we need to consider several additional aspects. Firstly, we must ensure that the user interface remains intuitive and user-friendly. Secondly, we need to evaluate the responsiveness of the website across different devices, particularly mobile phones, to address any potential issues. Furthermore, incorporating options for multiple languages and a text-to-speech feature for visually impaired users would be beneficial. Lastly, we must prioritise the protection of customer data and ensure proper security measures are in place.

To enhance the plant care experience, we envisioned allowing users to provide more detailed information about each plant. This additional information would enable us to calculate the overall health of the plants and provide tailored solutions to specific problems. However, deciding which features to prioritise and which to postpone for future development proved challenging, as there are numerous and varied options for improving the app.

Conclusion

The project aim was to provide an uncomplicated way to keep on top of watering your plants and educate people about the plants they have in their home. The project we have created solves this problem with a simple and easy to understand concept with email notifications, whilst also providing users with the option to see the information without signing up for emails.

We successfully completed our desired aims initially set out in the project proposal. These include providing notifications for users alongside fun facts. We believe the website we have created, and the project is a great example of our strengths and how we have really pushed ourselves during the last four weeks. We would like to continue working on the idea and add all the extra features we initially came up with. We have really enjoyed the process and have worked well as a team and learnt a lot from each other and the project. It is not often you are in a group where you get to work alongside people you genuinely like and work well with. We have been incredibly lucky to have such a wonderful team and to have worked on a great project.

Bibliography

Mayers, K. (2023, April 28). *Houseplant Statistics in 2023 (incl. Covid & Millennials)*. Retrieved from GARDEN PALS: https://gardenpals.com/houseplant-statistics/