TechBasics II - Theoretical Part: Report

PlantyPoppins

Name: Isabella Niese
Contact Information: 3048187, isabella.niese@stud.leuphana.de
Seminar: TechBasics II, Winter term 2023/24
Date of Submission: June 30, 2024

Introduction

PlantyPoppins is an application, which is designed to bring a delightful and enganing experience to plant care, inspired by the world of Mary Poppins. The project aims to provide plant enthusiasts with a comprehensive and enjoyable platform for plant care. It is designed to educate users, help them connect with their (real life) plants, and inspire them to confidently expand their plant collection.

1. The Idea

Even though many apps revolve around plant care, PlantyPoppins stands out as a fun and educational companion for plant lovers. By incorporating elements like gamification (e.g. fun music when watering a plant) and personalization (e.g. a test that guides users to a new plant), the app enhances the user experience, making plant care more engaging and rewarding.

PlantyPoppins addresses the desire for a straightforward, user-friendly tool that makes plant care more accessible and enjoyable. While other plant care apps can be overwhelming with their extensive and complex features and interfaces, PlantyPoppins provides essential functionalities (watering, health checks, personalized plant recommendations, and shopping lists) in a simple interface. This makes the app perfect for beginners and casual plant owners who need a simple introduction to plant care without being intimidated by advanced features. It seeks to foster a deeper connection between users and their plants by encouraging them to maintain a healthy and fulfilling plant care routine.

Through the integration of the Mary Poppins theme, the app reduces the pressure to be the perfect gardener. Unlike other plant care apps that focus on the technical aspects of plant care, PlantyPoppins brings a sense of lightness and fun to the experience. Mary Poppins, known for bringing joy to everyday tasks, serves as a perfect metaphor for making plant care an enjoyable activity. This approach not only makes the app memorable but also adds a storytelling element that engages users with their plants on a non-stressful level.

2. Methodology

In the TechBasics I project, the aim was to build an interface that provides an overview of potential but yet unrealized functions. The idea included features such as a plant identifier, a test to find the best plant match, a dedicated page for the user's plants, a plant community, and a profile with user settings and preferences.

When starting the TechBasics II project, I decided to simplify the concept and focus on functions that were possible to implement. I retained the homepage idea but reduced the functions to "Plants," "Test," and "Shopping List". The primary reason behind this decision was to build a "MVP" containing simple and enjoyable features (e.g. I cannot build a plant identifier, so I dropped that idea).

3. Limitations

The app is kept very simple and has its limitations. As previously stated, it was most important for me to include features that are not too complex but doable to code and, most importantly, that work.

3.1 Plant Page

Initially, I had the idea that users could change the pictures and names of the plants, as well as add new plants to the page. However, I found myself losing focus on the main app and the variety of features while spending too much time trying to create lists with possible names. Eventually, I decided to keep the page rather simple to ensure it worked correctly. Therefore, in the current version, users can imagine they have up to two plants at home, named Ernie and Bert. While watering the plant and listening to music, users are supposed to water their real life plants. If the app were further developed, it would be great if Ernie and Bert would show symptoms in the app such as dryness or being overwatered. Even if users don't own plants, Ernie and Bert are perfect for those more interested in virtual plants than real ones! I chose the names Ernie and Bert because they are characters from another creative series, "Sesame Street," also, Bert fits into the "Mary Poppins" theme as it is the name of the chimney sweeper.

3.2 Test Page

One could argue that more questions are needed to find out which plant fits the user best. Still, the app is just an MVP, and the result can be seen as an inspiration for users on their way to finding the perfect plant.

3.3 Shopping List

The shopping list could include more items to select, but it is perfect for all plant beginners who may not know what to buy on a simple plant-shopping tour. Like the test page, users can be inspired by the suggested items. If this MVP were developed with more time and coding knowledge, it would be helpful to integrate the possibility of downloading the shopping list and adding personal items. For now, I included a picture of Bert, reminding users to take a screenshot of the shopping list.

4. References Overview for PlantyPoppins Code

The PlantyPoppins project is built upon various coding principles and sources. In the following paragraphs, a structured overview of the references used in the app is listed.

4.1 Connection between app.py and extras.py

The basic idea of having an MVP which includes the main code and a “helpers.py” file was covered in TechBasics II. The helper’s file of Planty Poppins is called “extras.py”, which can be found on GitHub.

4.2 Clearing widgets (def clear_page), creating a homepage and a back button

The function “clear_page(root)” (extras.py, ll. 7-10) was covered in class, where we created the “clear_widgets” function in the “helpers.py” file. This included creating a homepage (app.py, ll. 18-27) and a back button (e.g. app.py, ll. 91-97)

4.3 The Test (def test)

Regarding the plant test issue (in comparison to the TechBasics I submission), I sought help in the DigiLab. Together with mentors, we targeted the problem by inserting several “print” functions in the test to identify where it stopped working. We found out that the function that cleared the widgets also erased the selected options. To ensure the program remembers both choices and predicts the best option, choices were included in the lambda function (e.g., app.py, l. 197, 204) and the result definition (app.py, l.208, 239).

4.4 The Checkboxes (def create_checkboxes)

The definitions “create_checkboxes” (app.py, ll. 489-536) and “show_selected_items” (app.py, ll. 537-589) could not be included in the “extras.py” file because the loop between importing them from another file would break the function of saving clicked items and showing them in a new definition. The core function of clicking on an item in the checkbox (app.py, ll. 522-528) is inspired by the website "Python Tutorial" (https://www.pythontutorial.net/tkinter/tkinter-checkbox/). The code had to be adjusted with the help of ChatGPT to link the checkbox with the list box, enabling the program to remember which items the user has selected.

4.5 The List Box (def show_selected_items)

I was inspired by the “Python Tutorial” Listbox and started with the copied code from the website (https://www.pythontutorial.net/tkinter/tkinter-listbox/). However, as stated in 4.5, the link between the checkboxes and the list box did not work smoothly, which is the reason ChatGPT was used to help adjust the code. Unfortunately, the code from “Python Tutorial” made it mandatory for the User to click on the Box before seeing the actual list. Therefore, the code in the “show_selected_items” definition (app.py, ll. 539, 563-564, and 572) was also adjusted with the help of ChatGPT.

4.6 The Message box (def health)

In the “extras.py” file, definitions for the health of the plants and the watering function are saved. The code for the message boxes (extras.py, ll. 20-42) is from "Python Tutorial" (https://www.pythontutorial.net/tkinter/tkinter-messagebox/). While programming, a warning appeared in the command line whenever a message box showed up (“*Warning: Expected min height of view: (<NSButton: 0x7fbbfd3847b0>) to be less than or equal to 30 but got a height of 32.000000. This error will be logged once per view in violation*”). This issue, discussed with DigiLab mentors, is specific to the Mac system and does not affect the functionality of the code.

4.7 Music and the Progress Bar (def water, def start_watering, def stop_progress)

When the user presses the "Water" button, music and a moving progress bar appear. The code for running music was covered in lecture 9, when adding Christmas music to a holiday card, and was copied and adjusted for this project (extras.py, ll. 83-85, l. 93). The code for the moving progress bar (extras.py, ll. 56-63, 65-71, 72-78, ll. 91-92) was inspired by "Python Tutorial" (https://www.pythontutorial.net/tkinter/tkinter-progressbar) and simplified for PlantyPoppins.

5. List of references

5.1 Images

Homepage.jpg, Plants.jpg and Test.jpg: https://www.canva.com Design: “Pastel Colorful Aesthetic Minimalist Elegant Bohemian Home Furniture Interior Design Basic Presentation” by AV Creatives

Test_mary.png: https://imgbin.com/png/ZU761n5U/illustrator-work-of-art-mary-poppins-png

Fiddlefig.jpeg: https://tierraplants.com/product/fiddle-leaf-fig/

Kentiapalm.jpeg: https://tierraplants.com/product/fiddle-leaf-fig/

Vietcoriander.jpeg: https://mygardenlife.com/plant-library/vietnamese-coriander-rau-ram-persicaria-odorata

Ginger_jpeg: https://tierraplants.com/product/fiddle-leaf-fig/

Mary_poppins_checkbox.png: https://www.deviantart.com/thatjoegunderson/art/Mary-Poppins-775071651

Dike-van-dyke.png: https://imgbin.com/png/46ndyiwq/dick-van-dyke-mary-poppins-bert-doll-mary-poppins-bert-doll-jane-banks-png

5.2 Music

Supercalifragilisticexpialidocious.mp3: https://archive.org/details/MaryPoppinsSoundtrack-ChildrensMusic/10+-+Supercalifragilisticexpialidocious.mp3