Winter Semester 2023/24
Bachelor of Arts in Digital Media
Hsu-Tung Hsieh 3047206

TechBasics II Exam Report

Introduction

This App (GUI) is named "I HAVE A FEVER" and is built using Python with the TK, PIL, and datetime libraries. It aims to assist people when someone has a fever by allowing users to track body temperature and medicine intake. Additionally, it provides quick diagnoses, medical knowledge, and an export function for doctors' reference.

Instruction: How to Start the App?

Please refer to the README file.

About the App

1. Inspiration

   The inspiration for the app stemmed from a dinner night when a friend suddenly developed a high fever. Everyone present was unsure about the situation and whether we should contact a medical professional. In addition, much of the important information in existing apps like "Fever App" is presented in long-text format, which can be overwhelming to process.

2. Main Functional Pages

   Users first need to complete the registration process. Afterward, they are directed to the homepage, which features five buttons linked to different pages:

   - Temperature Page: Tracks and records body temperature.
   - Medicine Page: Displays dosage instructions in a tree-view table and tracks medication intake.

- Dr. Smart Page: Provides a quick analysis to determine if medical help is needed.
- Medical Knowledge Page: Contains infographics about fever and related topics.
- Export Function: Allows users to export the patient's basic information and data from the temperature and medication trackers.

3. Features

- User-Friendly Information Display: Unlike "Fever App," I HAVE A FEVER presents information in a more readable format. Different pages feature multiple layers with a moderate amount of text, enhancing readability. For instance, dosage instructions are neatly organized into tables for clarity. Additionally, the "Dr. Smart" feature condenses lengthy instructions on fever treatment from various articles into simple, straightforward recommendations.

- Active Warnings: The app provides active warnings for critical situations. For instance, if the patient is under 18, a warning pops up on the medicine page indicating that Aspirin should not be administered. This feature ensures important information is not overlooked.

4. Design

Video demonstration of the GUI design: https://youtu.be/-t-Q8Wt2KVs

- Registration Process: The registration involves multiple pages, each with a background image. The app's main character—an eggplant—appears frequently with variations to entertain users. Labels and entry boxes are positioned next to the eggplant using the "place" method for flexibility.

- Homepage: The center of the homepage features the eggplant. Five items—a medicine can, thermometer, stethoscope, book, and printer—are actually clickable buttons. The design tries to integrate the buttons into the theme, adding cuteness and consistency.

- Functional Pages: Widgets on the main functional pages are primarily positioned using the grid method. Pages with numerous elements or various types of content (like the medicine and medical knowledge pages) often use nested frames to maintain organization and layout consistency.

- General Design Principles: The app adopts a consistent theme color. Most labels, buttons, and frames follow predefined sizes or fonts stored in variables to ensure uniformity and harmony.

Challenges and Limitations

1. Organizing Different Layouts: Creating layouts with different types of frames and positioning methods can be time-consuming. Fixing small errors or inconsistencies in code syntax can take hours, especially since coding resources on the internet vary in reliability. Some functions, like "treeview," only support inserting text widgets, not other types. To achieve similar layouts, such as on the medicine knowledge page, I had to use buttons and various definitions, which complicated the code. Also, aligning many elements neatly on a page is challenging, often requiring nested frames to maintain a clean layout.

2. Moral Obligation: Developing a medical app comes with a moral obligation to ensure the accuracy of the information provided and to implement warnings to restrict certain users or provide extra care in specific situations. For example, cross-checking information from multiple sources on the internet and integrating them can be very time-consuming, especially when each article presents different data. This is why I included a long registration process: first, to enable the export function, and second, to incorporate additional warnings into the app.

Reference

1. References to other websites:

- "Grid" method:
  https://steam.oxxostudio.tw/category/python/tkinter/grid.html

- "Place" method:
  https://steam.oxxostudio.tw/category/python/tkinter/place.html

- "Sticky" command:
  https://steam.oxxostudio.tw/category/python/tkinter/grid.html

- "Anchor" command vs. "justify":
  https://stackoverflow.com/questions/37318060/how-to-justify-text-in-label-in-tkinter

- Image, ImageTk:
  https://steam.oxxostudio.tw/category/python/tkinter/photoimage.html

- Capitalize the first letter of each word (line 46):
  https://stackoverflow.com/questions/1549641/how-can-i-capitalize-the-first-letter-of-each-word-in-a-string

- Adjust the width and height of a canvas frame (line 330, 443, 755, 923):
  https://steam.oxxostudio.tw/category/python/tkinter/scrollbar.html

- Get current time (line 373, 373, 677, 678):
  https://www.toppr.com/guides/python-guide/tutorials/python-date-and-time/date-time/current-time/python-get-current-time/

- Fix a bug in the temperature list box (line 385~392):
  https://shengyu7697.github.io/python-tkinter-listbox/

- Treeview (line 540~627):
  https://pythonassets.com/posts/treeview-in-tk-tkinter/

- Add an absolute width to labels (line 1081~1114):
  https://stackoverflow.com/questions/74945714/python-tkinter-set-absolute-label-size

- Confirmation message (line 1117~1121):
  https://www.pythontutorial.net/tkinter/tkinter-askyesno/

- Export data to a text file (line 1122~1152):
  https://www.w3resource.com/python-exercises/tkinter/python-tkinter-dialogs-and-file-handling-exercise-7.php

2. Use of ChatGPT
   - Python Lists (codes related to "temperature_record_list" and "medicine_record_list")
   - Try and Except (line 116~126)
   - Dropdowns (line 207~209, 658~660, 962~964, 972~974)
   - Add scalable canvas frames (line 330~349, 443~462, 755~772, 923~941):
   - Adjust column width and height in treeview (line 508~522):
   - Fix typos in the treeview (line 540~627, no specific lines):
   - Correct syntax error of booleans (811~827)
   - Help check the coverage of all the conditions in Dr. Smart and help restructure the code (1018~1068)
   - Repetitive jobs (e.g. generate multiple buttons and labels, adjust layouts)

- Generate / polish texts used in the app, and fix grammar mistakes(e.g. terms and conditions)
- Fix grammar mistakes and improve the writing of this report and the README file