Caroline Crooks
TP1 Design Proposal

Project Description

The name of the term project is “Your Calendar”. It will be a calendar application that will allow users to schedule both recurring and non-recurring events. It also has features to generate custom schedules based on data the user inputs.

Competitive Analysis

This calendar has the same basic functions as most calendar applications, but what makes it unique is its schedule generation tool. Two popular digital calendars are Google Calendar and the Apple Calendar. Both these calendars allow users to make “events” that have a title, time, description, and recurrence settings. These events are then displayed on a calendar with a view of the week. My calendar also will have this feature. An additional, unique feature that the Google Calendar and Apple Calendar do not have is the ability to schedule events for the user. “Your Calendar” will have a “flexible events” feature. A user can input something that they want to do for a certain amount of time each week, but it is something that the user does not have a specific time in mind for. An example of this would be wanting to read books for 5 hours each week. “Your Calendar” can schedule these types of events to fit in the user’s schedule. After every day, “Your Calendar” will ask users if they really completed these flexible events and their mood that day. The next time the user wants a schedule generated, the app will take the mood and the times the user actually completed the task into account.

Structural Plan

There are four main types of files: functionality files, drawing files, layout files, and data files. Functionality files contain functions that make computations or incorporate user input. An example of this type of file is the “date_functions” file, which has all the functions that pertain to making calculations about date and time. One function within this file is the function that finds the most recent date, given two days. Drawing files simply draw the components on the screen to make the UI, such as displaying the calendar. The layout files are files that calculate the placement of components on the screen. For example, the calendar layout file gives the coordinates of the components so that the calendar drawing file does not have to calculate this information itself. Finally, data files go in their own separate folder, the “data” folder. This data folder contains the information about actual events that the user inputs.

The app itself will begin in the “main.py” file, which will have the appStarted call and initialize any values. It will start on the “calendar” screen. Other screens include the “create event” screens” and “settings” screens, which the user can access with buttons on the calendar screen.

Algorithmic Plan

The trickiest part of the project is generating a new schedule based on user input. There are three types of events: strict non-recurring events, strict recurring events, and flexible events. First, I will organize the strict events in order by reading the csv file and then using the merge sort algorithm to organize them by date.

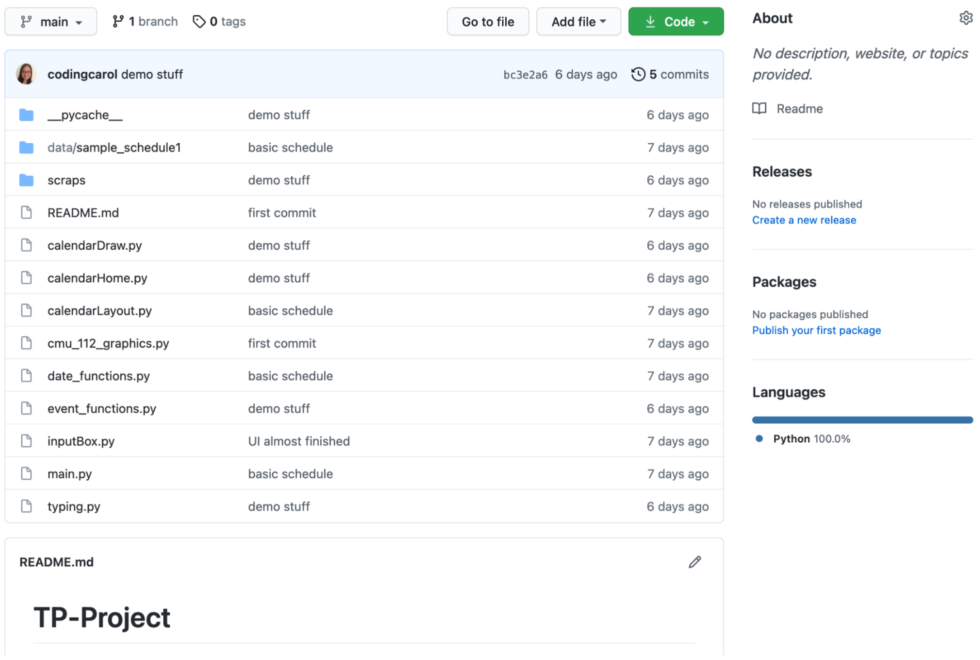
Then, I will use recursive backtracking with a priority factor to figure out how to integrate flexible events. Because I ask users to input information about the previous day, I will have a csv file of each flexible event, which includes information about times the user has missed the event, times the user has attended the event, and the mood for each time. I will make a new list for the event, listing each time from most ideal to least ideal. To make this list, I will iterate through the csv file of time data and give each time a score. Items that are more recent will be weighted more than items that are further back in time. Times with a higher score are more ideal for that event. A mood score will be generated a similar way. Then, I will have a list for each event with times that in order of most to least ideal. Most ideal times have the highest score. If two times have similar scores, the time with the higher mood score breaks the tie. After that, I can use recursive backtracking to place the flexible events around the strict events. It will work by attempting to place an event in a certain time slot and will backtrack if not all the events are able to be placed. The stop condition is if all the flexible events are placed.

Timeline Plan

April 20th - Basic Calendar UI, can schedule both recurring and non-recurring events
April 26th - The recursive backtracking piece is mostly finished, apart from minor bugs.
April 29th - Add more features to the UI, such as adding color to events and making the page that asks users for information about the previous day
May 1st - Touch ups/bug fixes finished
May 5th - Add extra features (not necessary ones, undecided yet)

Version Control Plan

I am using github to back up my code.



codingcarol demo stuff — bc3e2a6 6 days ago — 5 commits

__pycache__	demo stuff	6 days ago
data/sample_schedule1	basic schedule	7 days ago
scraps	demo stuff	6 days ago
README.md	first commit	7 days ago
calendarDraw.py	demo stuff	6 days ago
calendarHome.py	demo stuff	6 days ago
calendarLayout.py	basic schedule	7 days ago
cmu_112_graphics.py	first commit	7 days ago
date_functions.py	basic schedule	7 days ago
event_functions.py	demo stuff	6 days ago
inputBox.py	UI almost finished	7 days ago
main.py	basic schedule	7 days ago
typing.py	demo stuff	6 days ago

No description, website, or topics provided.

No releases published
Create a new release

No packages published
Publish your first package

● Python 100.0%

README.md

TP-Project

Module List

- Datetime, Calendar, OS
- Strings, random
- CSV
- Scikit-learn

TP2 Update

No changes.

TP3 Update

No Changes