Project MidPoint Check In

For this project, I am attempting to create an online calendar/scheduler which would ultimately be used in a medical office. As background, my sister is an office manager at a pain clinic and is responsible for distributing each employee’s weekly schedule. Her boss is impatient with technology and has historically used paper only to prepare schedules. If each employee’s schedule was predictable and static, this wouldn’t really be an issue. But as employees request time off, all edits are made with ink. The more changes that are made, the more indecipherable the calendar becomes. While numerous online tools exist that would allow one to plot people’s schedules, my sister’s boss wants something more automatic. She wants the scheduler to consider specific rules that are unique to her office and her employees. For example, she wants the tool to recognize that, if Dr. Jones is unable to work his regular shift, Dr. Smith will be the one to take his place. If Dr. Smith is unavailable, then the scheduler will place Dr. Stevens on the calendar.

Implementing this type of application using Flask does not require a dataset, per se, but it would require establishing a SQL database. The tables in this database would store data such as the doctor’s name, the days they normally work, the office location they are in (there are two separate offices), who would be the first one to cover for them if they are unavailable, etc. My Python code would contain a series of if statements based on actions the user of the calendar performed. For example, if the user indicated that Dr. Jones was not available on a certain day, my code would query the database to determine who would be scheduled in their place.

So far, I have made little progress with regards to this task. I have spent most of my time thus far figuring out how to deploy a web application using flask. Once I was able to actually launch the application, I had to delve into the Calendar and Datetime libraries to determine how to access data that would reflect the current month (April 2020). Since these are brand new libraries for me, it took a significant amount of time to sift through all of the online documentation to find examples that were relevant. As of now, all I have to show for this is a web application that displays a calendar of the current month. The site is completely static and isn’t interactive at all. The next page will display the code I have so far in both my app.py file and the html template for my home page.

Based on how slow my progress has been thus far, I am quite concerned that this project won’t be feasible. The biggest obstacle I still have to overcome is making my page interactive by adding links that alter what’s displayed on the page. For example, I have to figure out a way to allow the user to indicate that Dr. Jones won’t be able to work on Tuesday, April 28th. Then, I have to figure out how to relay that information back to my Python file to tell it to “Run a query to determine who covers for Dr. Jones” and update the information on my site accordingly. Given the timeframe for this project, I’m thinking it’d be best to pivot to a project that focuses more on Python rather than web-development. I haven’t had a chance to dive in to other ideas too much, but as a preliminary idea, I’m thinking of taking various CSV files that look at different data points by ZIP Code (for example, cost of living, health care costs, demographics, etc…), opening and reading those file, storing the data in a SQL database, then running code to produce a heatmap or graph. That way, I’m only preoccupied with displaying data in my Flask application rather than gathering input from the user (which seems far more complicated than I originally thought). I’d love to discuss this with a TA further to get a sense of what my options are at this point.

Code So Far:

App.py:

from flask import Flask, render\_template,   
import calendar  
import datetime  
  
  
# Configure application  
app = Flask(\_\_name\_\_)  
  
# Ensure templates are auto-reloaded  
app.config["TEMPLATES\_AUTO\_RELOAD"] = True  
  
@app.route("/", methods=["GET"])  
def home():  
  
 now = datetime.datetime.now()  
  
 month = now.strftime("%B")  
 month\_as\_int = int(now.strftime("%m"))  
 year = int(now.strftime("%Y"))  
  
 text\_cal = calendar.HTMLCalendar(calendar.SUNDAY)  
  
 html = text\_cal.formatmonth(year,month\_as\_int)  
  
 html = html.replace('month', 'month table table-bordered')  
 html = html.replace('th class="', 'th class="thead-dark ')  
  
  
  
 return render\_template('home.html', month = month, year=year, html=html)

home.html

<!DOCTYPE html>  
  
<html lang="en">  
 <head>  
  
 <!-- Required meta tags -->  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">  
  
 <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">  
 <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" >  
 <link href="/static/styles.css" rel="stylesheet">  
  
 <title>Scheduler</title>  
 </head>  
 <body>  
<div>  
 <h2>{{ month }} {{ year }}</h2>  
 <table class="table table-bordered">  
 {{ html|safe }}  
</div>  
 </body>  
</html>