REFLECTIONS

A short report to reflect upon my progress

ABSTRACT

A short reflection, summery of implementation and explanation. With pictures included.

Ahmed Mursal, 406465151

AWT (Advanced Web Technologies)

Contents

Reflection

I started off this project with an idea which was to create a calendar website that stores my events into a database and is able to display said events back to the user in real time, a to do list page which a user could add items to and section off to a list which consists of complete, in progress, haven't started, which I successfully created along with a weather prediction page which would call back an API to grab the weather data.

I successfully made the project before the due date and attempted to deploy it through Heroku however this is where I struggled the most since I had no experience with deployment what's so ever, I knew python, Html, CSS and JavaScript which helped but I didn't know flask I didn't know about routes however this module has helped expand my horizons in ways which I couldn't have expected I personally learned so much in so little time which was by far my favourite part.

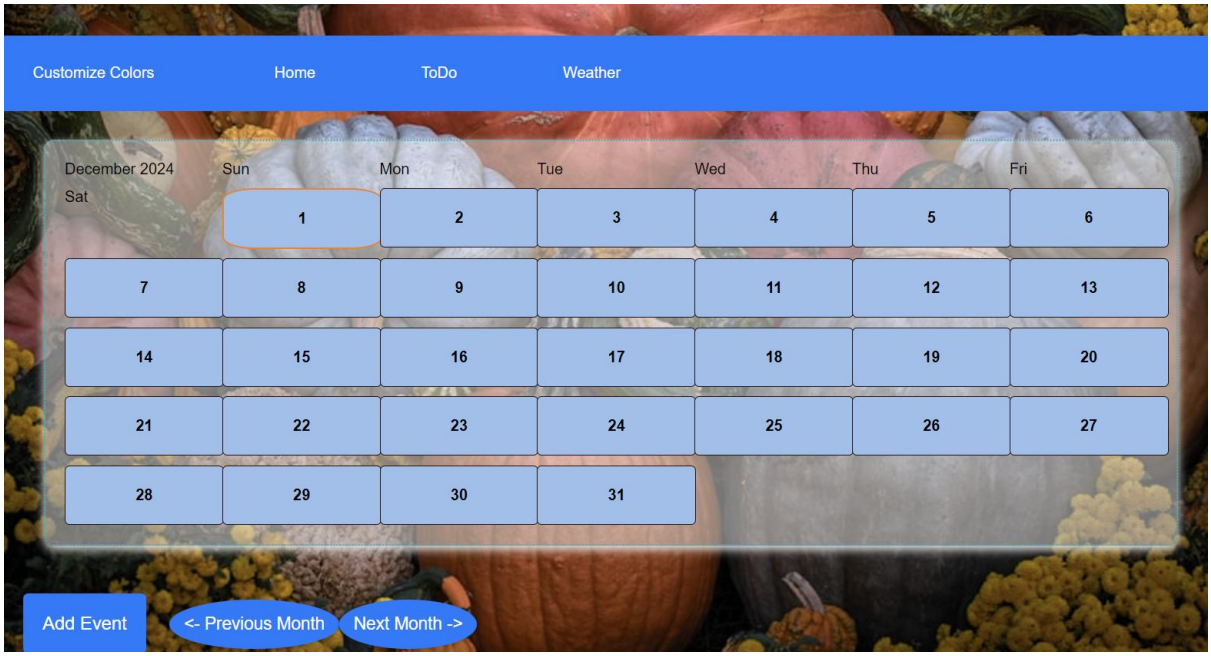
Website Structure

The structure of the website includes:

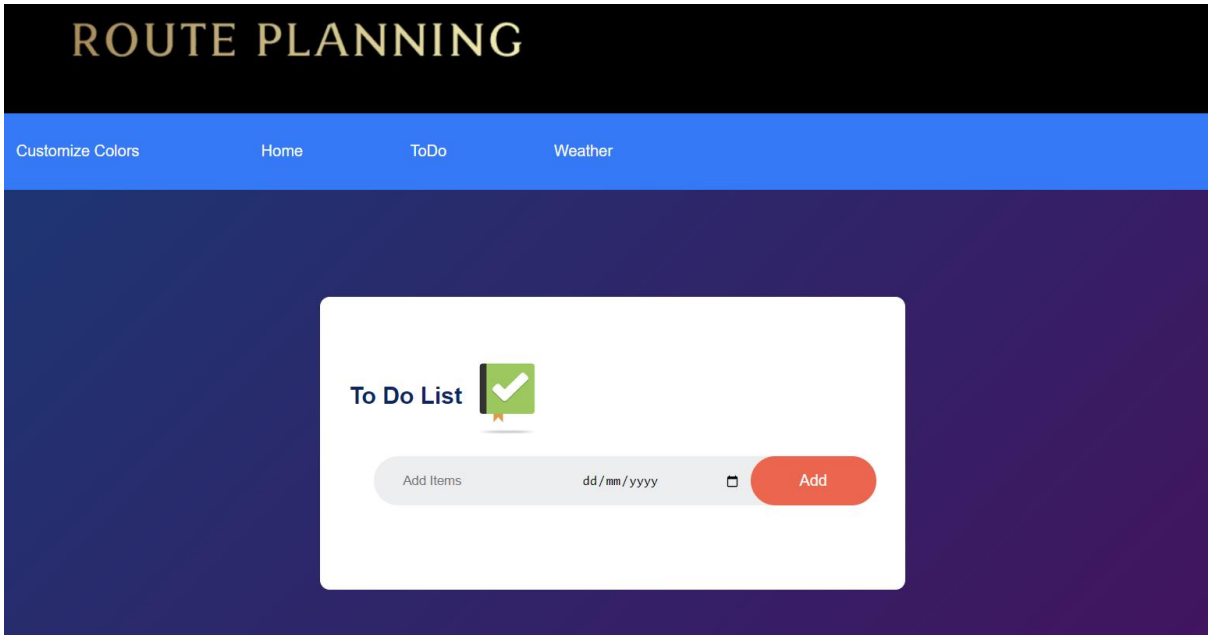
- A splash screen which has a nice logo I named route planning because initially I wanted to create a copy of google maps but with a twist which was granting the user to create their own routes without having the point A to B destination idea the point of that app was to help instructors and new drivers get used to the different roads by allocating the paths which had less or more roundabouts, less traffic lights or traffic in general, more or less dual carriageways, however I had come to the conclusion that the project might have been a tad too much so I changed it.
  The splash screen automatically directs the user to the next page after moving their mouse for a second I learned how to make it from my first website project 2 years ago back in HNC however back then I didn't know how to make it change pages without having to click a button in-order to go to another page.

- 

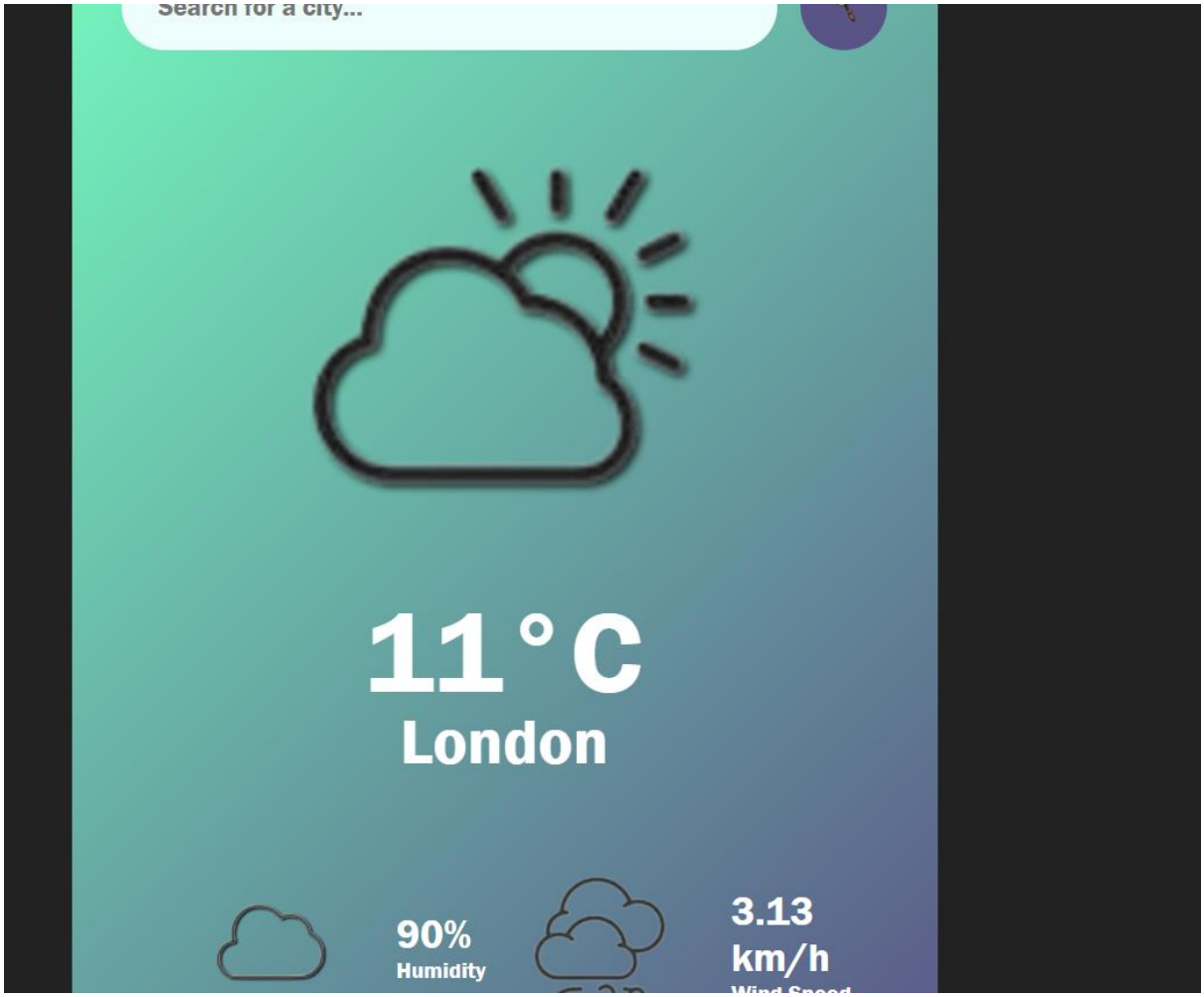
- My second page or route is the home page which consists of the calendar it's a transparent calendar with a random image generator API for the background along side a bar to change the background, text, navigation bar colour and more my initial idea was to learn how to use cookies to allow me to store the users preferences for the colour change however I couldn't manage to accomplish it.

- 

- My third page is a to do list which also stores the users input into a database and shows it back to the user with the due date and grants the user the ability to check the items as done, undone or remove them.

- 

- My last page was a weather prediction, it uses an API from open weather map to allow the user to see what the temperature, wind speed and humidity is like based on cities the default city is London.

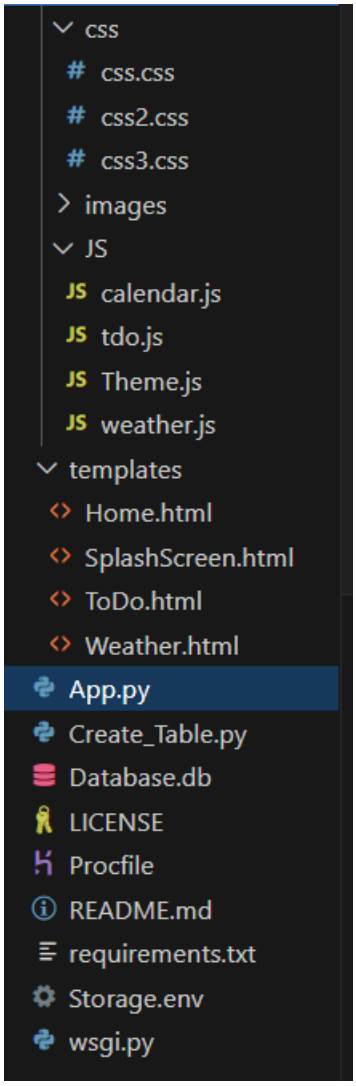
- 

Method of implementation

Steps of implementation:

1. I started off by creating the GitHub repository.
2. I then proceeded to add my templates or static files such as the html, CSS and JavaScript and linked them all together.
3. I proceeded by creating the app.py and added the routes or at least this is how I would do it moving on because in all honesty my implementation was somewhat messy but it's a part of the learning process, after creating the route and templates using jinja.
4. I started working on creating the basic structure of each page by first adding in the tags, IDs and Classes.
5. After which I added the CSS to style them and JS to allow them to function my first item on the list was creating the calendar I used some inspiration from code pen and YouTube tutorials to teach me how to make the calendar instead of making it statically and suppurating the days and months individually I made a Java script function called create calendar which allowed me to create daydivs monthdivs and yeardivs they all worked as containers which joined onto one another to create the basic calendar.
6. I then created a form to allow me to take in the events and connected the database.
7. But before step 5 I added in the elements that'd allow me to change the pages colours based on users preferences thinking that I could add in the cookies at the end of the project if time allows it.

8. I spent most of the time debugging the calendar and the background image which is why I sectioned my code off into multiple CSS and JavaScript files, to allow me to manage it with ease.



9. After ensuring that everything worked, I ran prettier which is a code clean up extension to make the code look more presentable and understandable for anyone else reading it.
10. I went back and commented what I had missed in order to clarify what I made.

Additional technology integrated

Additional Extensions.

- Prettier which is an extension on vs code used to format and clean up the code making it more visually appealing.
- Error lens which is another extension however it allows me the developer to see exactly what line the error is on and what the possible cause could be it is very helpful yet sometimes distracting tool
- Live Server which is yet another extension that allows me to see what my static no flask application looks like on the click of the ctrl s button without having me reload the page over and over.

- Pylance which is an extension that works alongside Python in Visual Studio Code to provide performant language support., Using Pyright, Pylance has the ability to supercharge your Python IntelliSense experience with rich type information, helping you write better code faster.

Additional Libraries.

- Sqlite3 which allows me to create a local dataset on my machine and lets the app run locally allowing me to test the database logic and schema, hosting services don't like SQLite they prefer something like PostgreSQL and am not yet sure as to how to change over to Heroku because of that.

- Flask, which is the lightweight library, used to allow for routing between the pages and it relies on python.

- Requests which allow you the developer to send HTTP/1.1 requests easily it removes the need to manually add query strings to your URLs, or to form-encode your PUT & POST data — but nowadays, just use the Json method which I took advantage of and used plenty on my JS files

- Python-dotenv allows me to create configuration files called. .env and call them to store important information such as my API keys or secret keys, these files are environmental files meaning that when pushing items to GitHub the .env files should not be pushed along side them however since this is a learning experience, I pushed them too.
- I also initially added flask WTF for form handling however I later on removed the form element from my html files and exchanged them with an onclick event listener and handler

In the deployment stage I added these libraries:

- Psycopg2 which allows me to create a PostgreSQL database to migrate from my local machine to a hosting service and since I couldn't figure out how to use SSL, I attempted to learn how to host through Heroku.
- Another library I used was the Gunicorn responsible for creating dynos to give specific commands to my hosting service based on what information I have In my Procfile and wsgi file

Time Constraint

Due to the time constraint and my inexperience, I couldn't manage to include cookies or add some features that make my application smoother features such as API generated background images on all pages, more transparent containers, elements reload so that everything works smoothly, and hosting the actual app. However, in the given amount of time I managed to learn how to use flask, improved my understanding of JavaScript and CSS, learned about .env, learned about hosting services and improved my problem-solving skills.

Ideas for the future

Features I would like to Add.

- Transparent effect on all containers and buttons
- API generated background image for all pages
- A list or container to store the images statically or in storage as in cache so that am not calling the image change function too much making it time out.
- Cookies to save the users preferences for the theme and inputs.
- A page responsible for grabbing images based on users input to allow the user to make the images into their wall papers my idea is to work with windows granting the app permission to change the users wallpapers dynamically based on season as well or their own input which should be stored in preferences.
- A feature I would like to be enhanced is a current bug I have which is the page needs to be refreshed for the items added onto the to do list to appear.
- A last feature I would like to add is making the app deployable which also maintaining its ability to operate locally as it has been.

Deployment

I jest you not by far the most challenging part of this entire project was and still remains deployment I have watched plenty of tutorials on YouTube and Udemy to try and understand why my website doesn't want to work on Heroku, I have come to the conclusion that the issue could be one of the many I shall list down below or possibly a combination of all of them.

1. No database URL found

I set up an entire branch on GitHub dedicated towards deployment and gave an if statement saying if d burl doesn't equal that of a PostgreSQL database, then use the existing local SQLite database which is for testing and future development purposes, but it may be the cause of the error that or I haven't specified the URL the way Heroku want it on the Storage.env file.

2. Heroku may not be connected to GitHub properly when I was on stack overflow, I found a command git remote -v which allows me to see what all of my remote branches or connections are.

My output was different to what I had seen my Heroku remotes were missing so I added them in manually using another command git remote set-url heroku <new-remote-url> to no avail the next error was there is no main file or an access point.

3. No access point

```
4. if __name__ == '__main__':
5.     app.run(debug=True)
```

from what I have gathered so far this and the procfile are the access point so that error didn't make any sense to me however recently I have come across an interesting video that describes possible errors one could encounter and how to bypass them so I will allocate sometime before the deadline on the sixths and watch it.

Conclusion

In conclusion this was a very challenging, yet favourable project as stated above I learned so much in so little time the combination of different, yet similar modules really helped me improve my time management, communication and coding skills. I enjoyed every minute of this assignment except for the reports we all dislike them, but they are necessary to scope out the project and explain our ideas to others which is fine.

References.

Pretty Printed (2019). *Deploy a Flask App to Heroku With a Postgres Database*. [online] YouTube. Available at: https://www.youtube.com/watch?v=FKy21FnjKS0 [Accessed 1 Dec. 2024].

www.youtube.com. (n.d.). *Deploy Flask App With Database On Heroku For Webhosting - Flask Fridays #39*. [online] Available at: https://www.youtube.com/watch?v=SiCAIRc0pEI.

Haung Code (2024). *Create Login form with HTML / CSS* 🥳😎. [online] YouTube. Available at: https://www.youtube.com/watch?v=Rx4X8Y1tbDk [Accessed 1 Dec. 2024].

HaryPhamDev (2020). *Fix all errors with Heroku deploying failure - Top 5 common mistakes newbies always have*. [online] YouTube. Available at: https://www.youtube.com/watch?v=aOdGK589glo [Accessed 1 Dec. 2024].

www.youtube.com. (n.d.). *How To Make Weather App Using JavaScript Step By Step Explained*. [online] Available at: https://www.youtube.com/watch?v=MIYQR-Ybrn4 [Accessed 1 Aug. 2023].

www.youtube.com. (n.d.). *How to store data with Python and SQLite3*. [online] Available at: https://www.youtube.com/watch?v=RZI-v-Z1W4c.

GreatStack (2023). *How To Create To-Do List App Using HTML CSS And JavaScript | Task App In JavaScript*. [online] YouTube. Available at: https://www.youtube.com/watch?v=G0jO8kUrg-I [Accessed 19 Nov. 2024].

GeeksforGeeks. (2020). *Deploy Python Flask App on Heroku*. [online] Available at: https://www.geeksforgeeks.org/deploy-python-flask-app-on-heroku/.

Past Projects.

Codepen.io. (2017). *Event calendar*. [online] Available at: https://codepen.io/usamahamed/pen/ZvbGBg.

pythonbasics.org. (n.d.). *Flask SQLite database - Python Tutorial*. [online] Available at: https://pythonbasics.org/flask-sqlite/.

www.javatpoint.com. (n.d.). *Flask SQLite - Java point*. [online] Available at: https://www.javatpoint.com/flask-sqlite.

Udemy. (2024). *Online Courses - Learn Anything, On Your Schedule | Udemy*. [online] Available at: https://www.udemy.com/course/python-and-flask-bootcamp-create-websites-using-flask/learn/lecture/10533496#overview.

Polling, L. (2020). *Short Polling vs Long Polling vs Web Sockets - System Design*. [online] YouTube. Available at: https://youtu.be/ZBM28ZPlin8

www.youtube.com. (n.d.). *What is an API (in 5 minutes)*. [online] Available at: https://youtu.be/ByGJQzlzxQg

in (2023). *APIs Explained (in 4 Minutes)*. [online] YouTube. Available at: https://youtu.be/bxuYDT-BWaI

Web technologies: server side, by Simon Wells.

The flask project book, by Simon Wells.

Bishop, K. (2024). *A Step-by-Step Guide to Creating a Deployment Plan*. [online] www.fool.com. Available at: https://www.fool.com/the-ascent/small-business/project-management/deployment-plan/

Kumawat, A. (2023). *What is an API (Application Programming Interface)?* [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/what-is-an-api/

Rahman, H. (2024). *Long Polling vs WebSocket: Key Differences You Should Know*. [online] Api dog Blog. Available at: https://apidog.com/blog/long-polling-vs-websocket/

Google (n.d.). *Google Calendar API Overview*. [online] Google Developers. Available at: https://developers.google.com/calendar/api/guides/overview.

Google for Developers. (2024). *Google Tasks | Google for Developers*. [online] Available at: https://developers.google.com/tasks/

Weatherapi.com. (2021). *Free Weather API - WeatherAPI.com*. [online] Available at: https://www.weatherapi.com/.