Hala Alhamdani

Lilly Technical Challenge Documentation Template

Approach

I approached this challenge **step by step**, starting with understanding the task and make understand the objectives well and know what file I can edit or no and then setting up my environment. I installed Python and Git, verified the installations, and cloned the repository from GitHub.

Then, I ran the setup script to start the backend server and checked if everything was running properly.

Next, I focused on fetching data from the backend and displaying it on the frontend. I tested API endpoints using Postman to understand the data structure and handled missing or invalid data to prevent crashes.

Once the core functionality was in place, I made UI improvements to enhance the design and usability of the site.

Finally, I tried the optional objective to challenge myself and see if I could do it.

I referred to the provided documentation, GitHub guides, and some online resources when needed **like w3schools**.

Objectives – Innovative Solutions

I focused on making the site easy to use and worked on improving the user experience. One thing I'm proud of is how I handled missing or invalid data to prevent crashes.

I also worked on making the data input more user-friendly multiple times. I tested different ways to improve the form so users could easily add and send data to the backend.

At first, I displayed the data in a list and tried different ways. In the end, I decided to use a table because it looked more organized and was easier to read.

The design by adjusting the font, colours, and overall layout to make the site look better and more user-friendly. I wanted to make sure the information was clear and easy to understand.

To ensure accurate input, I added validation to prevent users from entering a string in the price field or leaving it empty. This helps keep the data clean and avoids errors in the system.

Problems Faced

While working on the **first objective**, I had an issue where the data was not fetching, even though I had set everything up correctly. The system wasn't showing any errors, but it also wasn't trying to call the **/medicines endpoint**. I realized the problem was with the link—it was incorrect. After fixing it, the data was fetched successfully.

To achieve the **third objective**, I used the existing backend file to store and fetch user input. However, the instructions stated that this was one of the files I was not allowed to modify. Although I didn't actually modify it—I only used it to display new data on the frontend—I was still concerned about following the rules correctly.

Because of this, I decided to create a separate backend file called **user_data.json.** This allowed me to store and fetch user input without affecting the existing backend files.

When working on the **optional objective**, the average price was not showing at first due to an incorrect link. Later, when it did show, it wasn't updating when a new medicine was added. To fix this, I made sure the average only displayed when the user clicked a button, and I updated it every time a new medicine was added. Debugging helped me find and fix these issues.

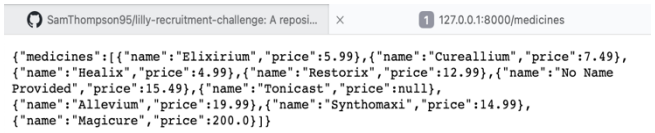
Evaluation

I really enjoyed this challenge! It was fun, and I learned a lot about maintaining a good user experience while designing and solving the task. Some parts went smoothly, while others took more time, especially understanding how the backend and frontend worked together.

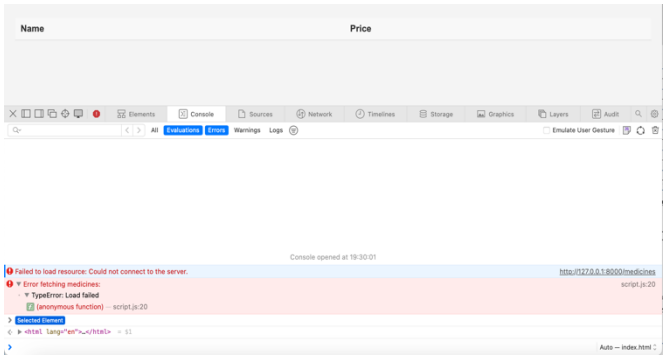
If I had more time, I would focus more on improving the design and adding extra features. Like: search bar to help users quickly find a specific medicine and a way for users to filter medicines by type or category.

I wouldn't do anything differently, as I think I took the right steps in the correct order, keeping in mind the goal of designing a user-friendly site. However, checking out different medicine trackers could have given me more ideas, and planning more before starting could have helped. Even though I spent time understanding everything first, there's still room for improvement. Overall, I'm happy that I met the objectives and created a user-friendly design.

Hala Alhamdani



This is after I cloned and set up the environment. I ran both the backend and frontend to see what they looked like.



I started by creating the table structure, but the data wasn't fetched because the link was wrong. The correct link is fetch('http://127.0.0.1:8000/medicines'), but I accidentally wrote double 8s instead.

Medicine Name	Price
Elixirium	$5.99
Cureallium	$7.49
Healix	$4.99
Restorix	$12.99
No Name Provided	$15.49
Tonicast	Price Unavailable
Allevium	$19.99
Synthomaxi	$14.99
Magicure	$200.00

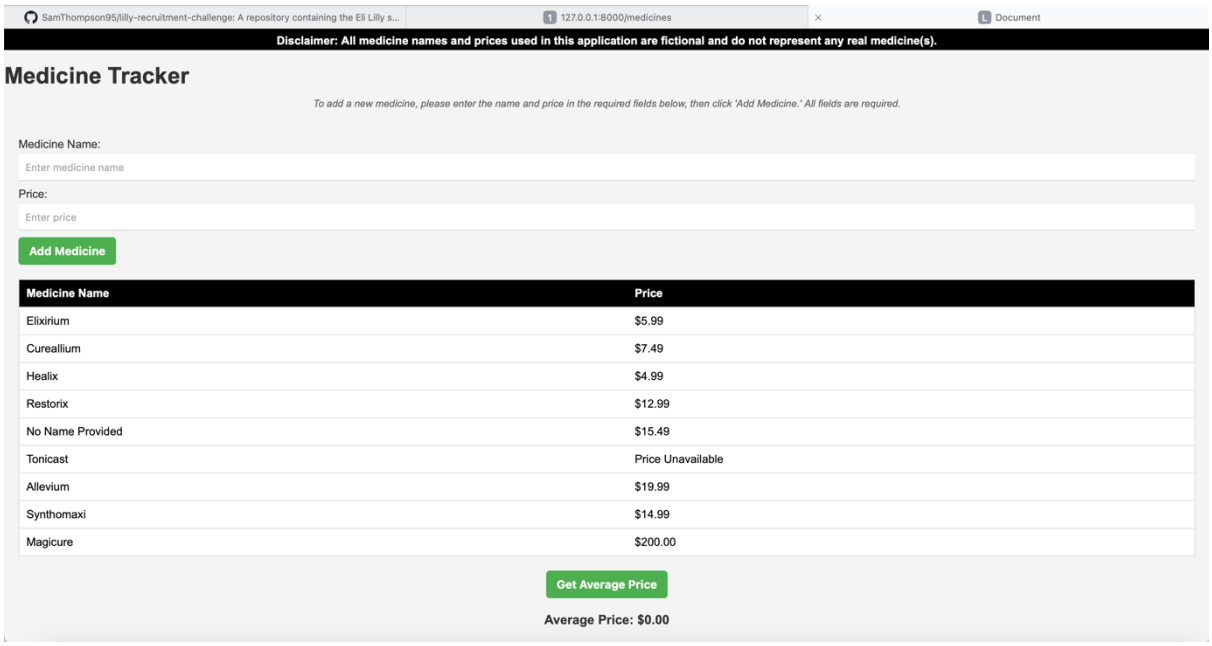
Medicine Name	Price
Elixirium	$5.99
Cureallium	$7.49
Healix	$4.99
Restorix	$12.99
No Name Provided	$15.49
Tonicast	Price Unavailable
Allevium	$19.99
Synthomaxi	$14.99
Magicure	$200.00

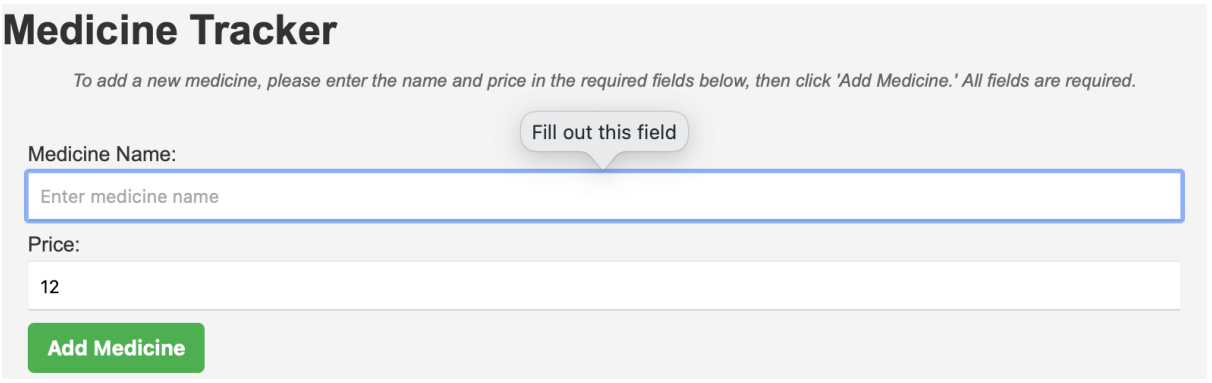
Get Average Price

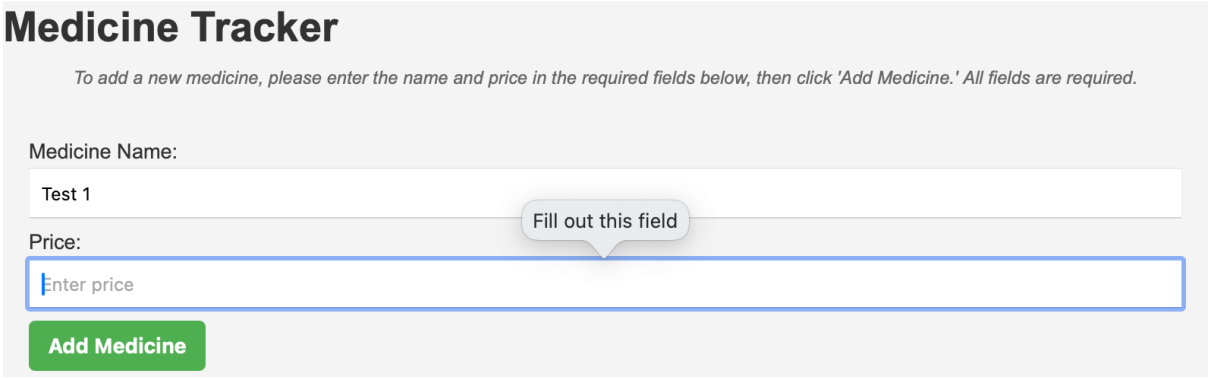
Error calculating average price.

Here is some of my first design, which had too much white space. The other screenshot shows how the button for the average price looked, but it wasn't working.

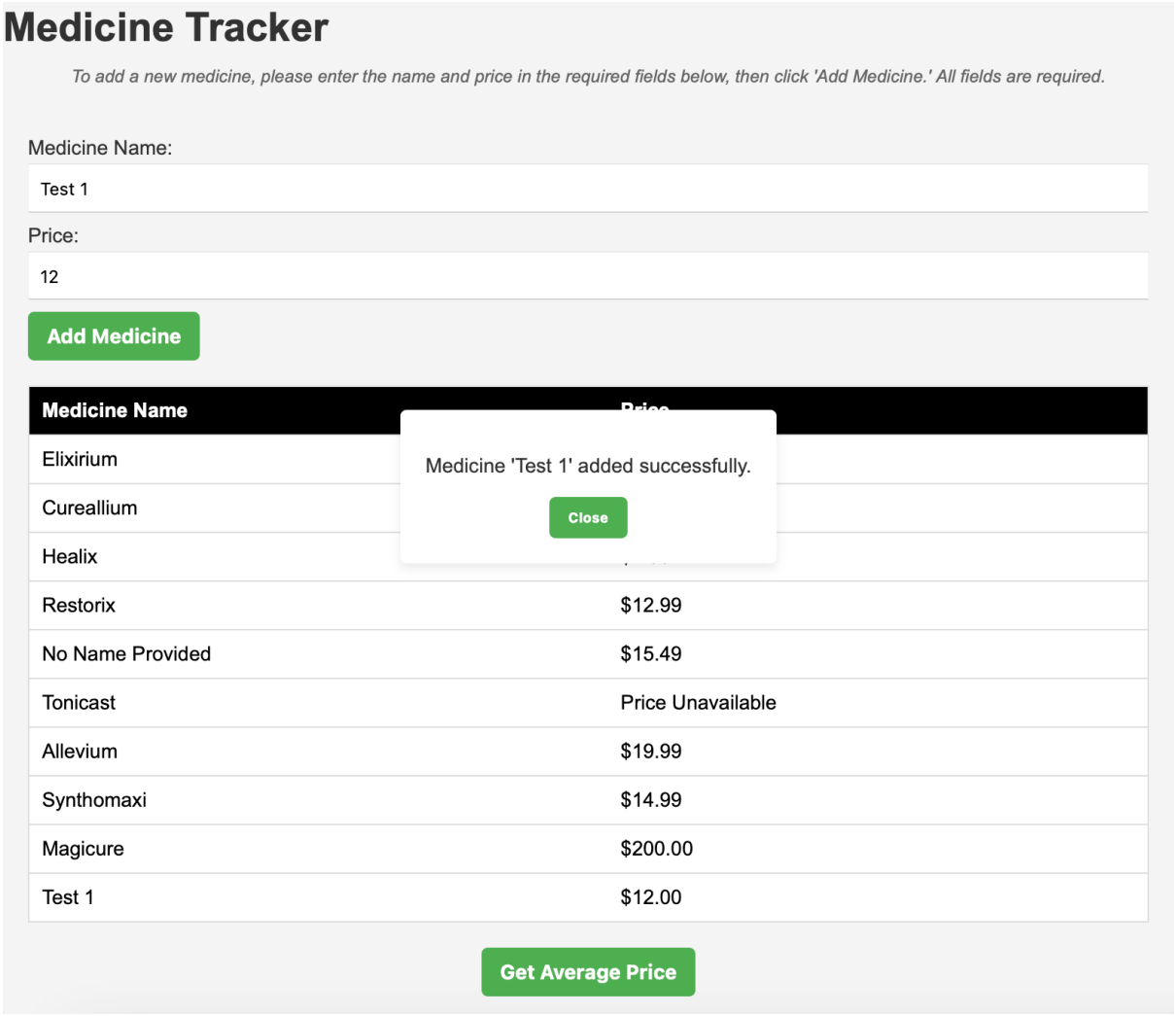
Overall Result with validation

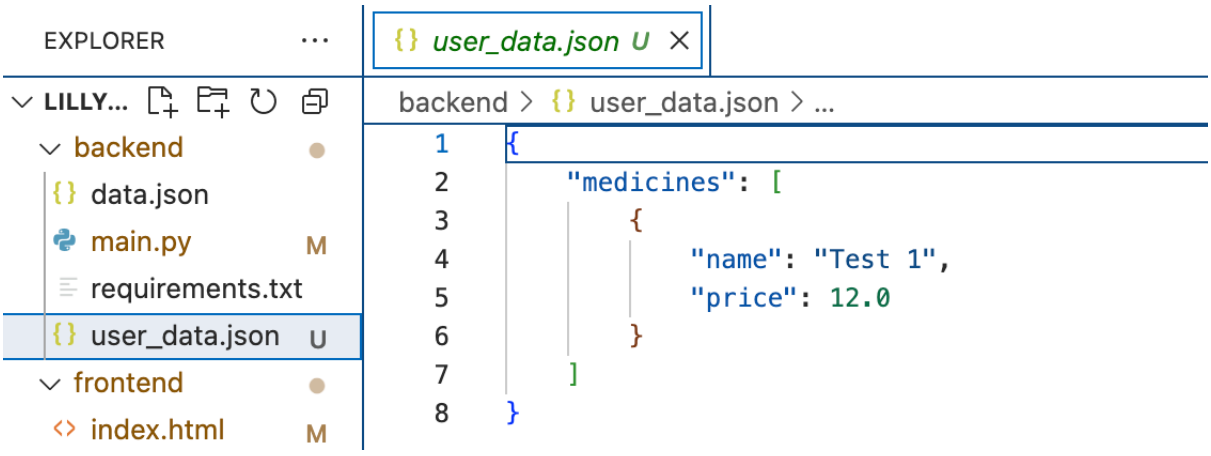






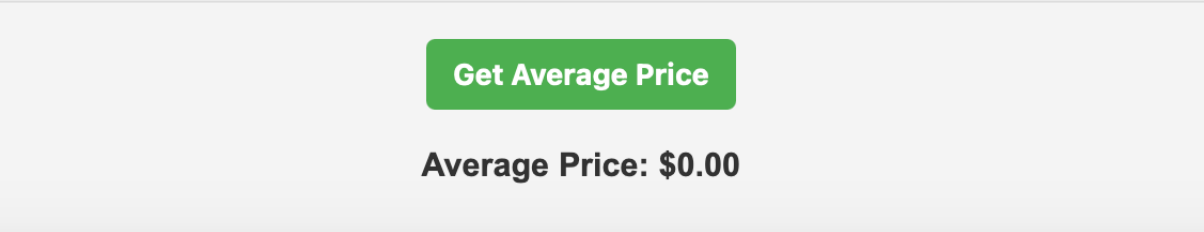
I added validation to ensure that neither the medicine name nor the price can be left empty. As I believe this will ensure that the data is complete and accurate.

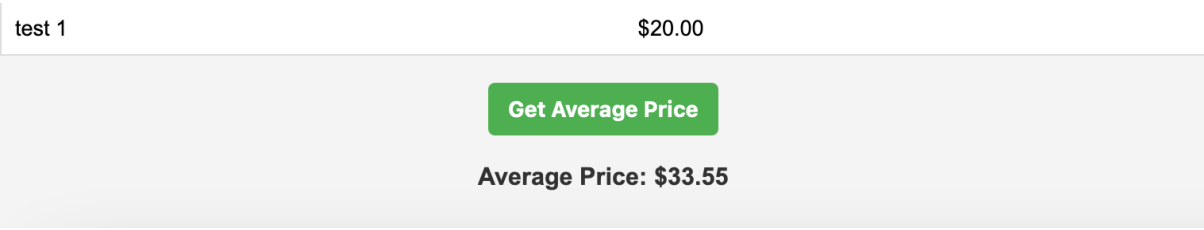




I added a new medicine that goes to the backend I created and shows a pop-up message. The reason why I added a pop-up message because it lets the user know that the medicine was added

successfully. This gives them clear feedback and ensures they feel confident that their action worked as expected.





Here's how the average price works:

It starts at zero. When the user clicks the button -> it calculates the average of the existing data, excluding any null values -> When a new medicine is added -> the user clicks the button again -> and the average price is updated with the new data.