# Lilly Technical Challenge Documentation Template

## Approach

Throughout most of the challenge, took a step-by-step approach - tackling the objectives in order. I did, however, leave the front-end styling to the very end (after the optional objective), as I felt it was more crucial that I get the functionality right first and foremost.

I began by launching the backend and using Postman to send GET requests, ensuring everything was working properly before implementing the objectives.

As I haven’t utilized JavaScript in a while, I went back and forth between the code, YouTube, and various online resources to familiarize myself with its fetch API, which was crucial to completing the objectives in the challenge.

## Objectives - Innovative Solutions

* Implemented data checks and validation while fetching and displaying medicine data to avoid issues in case of missing data.
* Used a grid to display the medicine list in a visually appealing manner.
* Added a new FastAPI endpoint to get the average medicine price.
* Implemented data validation in the new medicine form to ensure invalid price data wasn’t sent to the backend.

I did go back and re-write the Javascript code a few times once the key objectives were finished, but this was mostly to make it look cleaner, not to adjust the functionality.

## Problems Faced

One issue that took a fair bit of my time during this challenge was posting new medicine data onto the backend after entering it through a form on the web page - the medicine list wouldn’t update. I used the browser’s console to troubleshoot and found that it was throwing error 422, which I then started to look up. After some research, I found that the likely cause was the wrong content-type in the fetch request.

I found after further research that ‘x-www-form-urlencoded’ is the general format for sending data via HTML forms, whereas I was using ‘application/json’ as my Content-Type in the POST request header. Replacing ‘application/json’ with ‘application/ x-www-form-urlencoded’ and encoding the data in the body using ‘new URLSearchParams’ resolved the issue. Solution source (YouTube)

## Evaluation

I felt good about the challenge, as it helped me brush up on key web development concepts. Creating a function in the Python backend went better than the other tasks for me, as I’m actively using Python in some of my current modules at uni.

I did go over the recommended amount of time, as I had to get a handle on some Javascript concepts before being able to tackle the tasks head-on, particularly the use of the fetch API.

If I were to do this again given more time, I’d likely implement the delete and update functions that are available in the backend, ensuring out-of-stock medicines can be removed, and any price changes can be updated.

I would also like to refine the front-end a bit more. For instance, an approach where the medicine list, average price, and new medicine entry could all be viewed by pressing separate buttons on the home page, rather than being displayed altogether on page load. There’s also an issue where the average price doesn’t load sometimes – I would have liked to resolve this as well.