# Lilly Technical Challenge Documentation Template

## Approach

I began by reading through all the provided files, such as *main.py, script.js,* and *index.html* to understand how all the files are connected. I also carefully read the accompanying instructions to understand the objectives of the challenge. Once I grasped that, I made sure my environment and the dependencies are set up properly. With the environment ready, I prioritised the functionality: first by fetching and displaying data in a table to make it more user-friendly while also making sure it can handle missing/invalid data, then implementing a form so the user has a user-friendly way to input data. As I code, I tested each part frequently, using console logs and alerts spot issues quickly. Whenever I encountered a gap in my knowledge or a small syntax confusion, I would use resources like W3Schools or StackOverflow. After the core functionality was in place, I spent additional time to refine the user interface and layout. This includes reorganising the form elements for a more intuitive layout and making sure table rows displayed consistently, even when some data were missing. I also tried to apply a cleaner, simpler aesthetic that loosely matched the Eli Lilly’s website to create familiarity for the users. By iterating and testing at every step, I resolved errors as they come up, ensuring the final solution was both functional and reasonably polished.

## Objectives - Innovative Solutions

While implementing each objective, I tried to ensure the user’s experience was smooth and intuitive. One approach I am particularly proud of is how I handled missing data. Rather than simply skipping over entries with empty names or null prices, I used fallback text like “No Medicine Name” or “No Price Data.” This allowed the application to remain stable and transparent, even when the data are not perfect. I also introduced a straightforward solution for adding new medicines. After a successful form submission, the application immediately re-fetched the latest data, so the user saw the updated table straight away, reducing confusion and providing quick feedback.

I did end up revisiting certain parts of the code multiple times, especially around form submission and displaying new data in the table. At first, I tried to add items directly to the DOM without re-fetching, but this could become a problem when data needed to be synced across the application. Switching to a re-fetch method also improved reliability and kept the code more organised. I am also quite pleased with how the styling turned out, I tried to loosely mirror the bold red and white palette of Lilly’s website to unify the design and give the UI a clean, professional feel. I also want to give a sense of familiarity for the user. Overall, throughout all the objectives I aim to keep it as simple as possible while also prioritizing the user’s experience.

## Problems Faced

One issue I encountered early on was the *start.ps1* script failing to install dependencies from *requirements.txt.* That was quite an easy fix, as I just had to run *pip install -r requirements.txt* manually, which resolved the missing packages problem. Another hiccup involved using the wrong endpoint to add new medicines. Initially, I posted data to */medicines* rather than the */create* endpoint, which caused my form submissions to appear successful but never actually update the database. Realising my error, I switched it to */create* to fix that issue.

I also spent some time debugging why newly added medicines did not show up in the table right after the submission. At first, I tried simply appending rows to the DOM, but this could cause a syncing issue in the future, if there are multiple users accessing the application. I solved it by re-fetching the entire medicines list post-submission to ensure the table matched the server’s actual state.

Throughout these challenges, I used console logs and sometimes Postman to verify requests and keep track of each fix. Taking notes at every step helped me remain organised and remember the solutions, ultimately making the development process smoother.

## Evaluation

I found this challenge quite enjoyable overall. I managed to finish the core functionality, such as fetching data, displaying it, and enabling new entries in about an hour, although I did spend extra time tidying up minor details and sorting out why newly submitted medicines were not appearing in the table. That slight detour, combined with adding some stylistic touches, stretched my total effort to roughly an hour and a half. Still, it was a fun project to work on, and I appreciate how it let me demonstrate both backend and frontend skills in one go.

If I had more time, I would definitely introduce a feature to prevent creating a medicine that already exists in the database, perhaps with a prompt asking if you’re sure you want a duplicate. I would also consider user role management, so some users could only view the medicines while others could add or remove them. In hindsight, these improvements would make the application more robust and user-friendly, especially if multiple people used it simultaneously. Overall, the experience was positive and I feel I have struck a decent balance between meeting the requirements and ensuring the user’s perspective remained front and centre.