# Lilly Technical Challenge Documentation Template

## Approach

I started by looking at all the code to understand the challenge requirements and existing functionality. My initial focus was on the backend, where I used Postman to test the API endpoints. As someone with a Spring Boot background, I wasn’t very familiar with Fast API. To bridge this gap, I watched a YouTube video to quickly get up to speed (<https://youtu.be/SORiTsvnU28?si=mDiauxKL28zkfX4i>).

During my testing, I discovered that the POST endpoint required form data rather than JSON. Knowing this, I moved on to building the front end:

* **Data Fetching and Display:** I set up the initial space to fetch data and created a table structure to neatly display the medicine information.
* **Elements:** I added an “Add Medicine” button, which opens a modal form where users can input new medicine data. Each row in the table also includes update and delete buttons for easy management.
* **Styling and Layout:** After finishing the basic functionality, I focused on visual improvements. I refined the table for better clarity and alignment, applied background colours to the buttons, and ensured that the overall page was clean and easy to navigate.
* **User-Friendly and Accessible Design:** To make the interface more intuitive and accessible, I researched some web accessibility guidelines. I updated the button colours to align with widely used accessibility standards and added hover effects to signal interactivity to the user.

## Objectives - Innovative Solutions

While doing the challenge, I tried to implement some features:

* **Prioritising Accessibility:** I created a root element that has some colours known for their accessibility, so I can use that instead trying to copy paste the colour code everywhere.
* **Responsive Design:** I paid special attention to the layout; I tried make sure that the styling works well across different screen sizes. This guarantees a consistent experience whether on a desktop or mobile device.
* **API Integration:** The API integration initially was a little difficult, particularly in formatting the POST request correctly. Through multiple iterations and thorough testing with form data, I successfully fixed the front-end requests.
* **Usability:** Initially, the “Add Medicine” button was positioned at the bottom of the table. I realised that a long list of medicines could make it hard to access, I repositioned the button to the top of the page. Additionally, inspired by the Eli Lilly website, I added an up-arrow feature at the bottom so users can easily jump back to the top.
* **UI:** I am proud of the fact that I spent some time to think about how I can make the UI look good. So, I implemented a loading animation and designed an error message that appears in case of fetch errors.

## Problems Faced

I faced some unexpected challenges during the work:

* **API Integration:** Getting the API to work was one of the toughest issues I faced. My early attempts failed because I was using JSON for the POST request instead of form data as the body. After several times testing with Postman, I resolved the issue. Also, for the extra challenge, I spent 10 minutes figuring out why I was getting a 404 error, just for it to fix itself after a restart.
* **Error Handling:** I tried to do some error handling, so I performed some testing to catch and address potential errors.
* **Update Functionality:** I found issues with updating medicine entries. On inspecting the “data.json” file, I realized that there wasn’t a distinct identifier for each medicine other than the name. So, I thought that while the name should remain unchanged, only the pricing information could be updated. If an error occurs, the simply delete the entry and re-add it correctly.
* **Loading Screen:** I added loading screen animations to the web page, but I had to change up a lot of code to accommodate for the total utilisation of the row space inside the row which was added to the table only when an error occurred.

## Evaluation

I’m really pleased with the challenge outcome. I completed the core functionality in under 40 minutes, which allowed me to dedicate extra time to refining the styling, completing the extra challenge and creating this document. I believe I did a solid job balancing functionality with user-friendly and accessible design.

Looking ahead, if I had more time I would:

* Design better for the smaller screened devices.
* Research more into advanced user experience to improve accessibility.

Overall, this challenge has been a rewarding experience that allowed me to expand my skills, particularly in Fast API, while understanding the importance of clear, accessible, and user-centric design.