# Lilly Technical Challenge Documentation

## Approach

I approached this challenge by first analysing the objectives set out in the README file of the git repository, to ensure I had a clear understanding of the requirements. I then broke down the objectives into smaller tasks and prioritised functionality before refining the user experience. I started by setting up my environment, making sure that all dependencies were installed and launched the provided start.bat file provided to start the backend.

After this my main focus was ensuring that the frontend could successfully communicate with the backend, retrieve data from the data.json file, and display it correctly. In order to do this, I used various HTTP methods such as GET to retrieve all of the items from the json file, POST to create medicines, PUT to update medicines and DELETE to remove a medicine. This allows resources to have CRUD operations (Create – Post, Read – Get, Update – Put, and Delete – Delete) Whenever I faced a challenge or obstacle, I debugged the issue using a variety of techniques such as print statements, try catch blocks and proper error handling, after this I would research the issue efficiently by searching the error exception that I had returned on places like StackOverflow.

After establishing a connection between the frontend and backend it was a lot easier to add the required functionality such as displaying elements, creating, updating, and deleting medicines. I completed these using forms for user input and based on this the user can modify the list of all the medicines. After this I added the additional objective of calculating an average price of all medicines. I did this by calculating the sum of all valid prices / the length of the array of medicines with a valid price rounded to 2 decimal places. After adding all the required functionality I completed some clean CSS styling, with little time left for this section I was still able to create a modern clean style with a grey and red colour scheme. Similar to the Eli Lilly colour scheme on the main website.

## Objectives - Innovative Solutions

While implementing the challenge objectives, I took extra measures to ensure efficiency and a smooth user experience. For example, when handling the objective of making sure missing/invalid data doesn’t cause crashes, I implemented default fallback values so if there were null entries into the data.json file “Name Unavailable” or “Price Unavailable” would be displayed for the user, this ensured that even if the data was incomplete, the user would still see a formatted list of all the data. I did end up revisiting some of my initial code as one innovative solution I found when facing the issue of sending data to the backend was, I initially sent data to the backend requiring manually inputting JSON, which was not optimal, instead I designed clean, structured forms with clear input fields and validation to prevent invalid entries. Furthermore, I implemented asynchronous fetch calls with await to ensure that data retrieval and updates happened smoothly without blocking the UI. I also modified my colour scheme in the css file later in order to match the theme with the Eli Lilly website, this makes the UI more familiar to the user and improves the user experience. I am particularly proud of how I handled missing data and optimised the UI to make the experience seamless for the user, these solutions made the application more robust and user-friendly.

## Problems Faced

One problem I faced during the challenge was completing the optional average price function in the backed, the problem being that the backend did not have a built-in way to calculate the average price, to solve this I created a new method in the main.py file which calculates the average price dynamically and returns it to the fronted.

In addition to this I faced some communication issues with the backend early on, as I could not seem to get to display all the items from the database, through debugging I found that the issue was due to some medicines in the database having missing names or prices which could cause errors and not display anything. In order to fix this, I added default fallback values to ensure that the frontend remained stable and readable.

Furthermore, creating a user-friendly UI was particularly challenging, especially with little remaining time as I spent most of my time on the functionality part of the app. However, I took some inspiration from the Lilly website and created a simple but effective UI which resembles the colour scheme of the Lilly website and provides a user-friendly layout with clear buttons and text fields meeting all the requirements of the challenge.

## Evaluation

Overall, this challenge was an engaging and insightful experience, that tested my ability to work with both frontend and backend systems. I enjoyed the structured problem-solving approach and the hands-on implementation of real-world software development practices. As for my strengths and what went well: I successfully established an effective communication between the frontend and backend, I ensured missing data did not break the application by implementing fallback values, I implemented user friendly forms for easy data entry, I added an average price calculation feature which was optional but a valuable addition, I created a simple yet effective UI, which aligns with the Lilly brand identity. And as for challenges and areas for improvement: I spent a lot of time focusing on functionality maybe next time I could make the UI more polished, and visually appealing, with animations and more user interactions. If I had more time, I would also write some unit tests for different scenarios to ensure that my solution is as robust as possible. In conclusion, I am happy with my implementation and the solutions I provided to make the application functional, resilient, and user-friendly.