My Development Experience building the exercise.

I built this system using Spring Boot and React because I needed somethin that could handle both simple and complex position management tasks. Based and the requirements of the exercise and after working on similar projects, I knew Spring Boot would give me good security features and make database work easier. For the frontend, React was my choice because I'm comfortable with it and Material-UI saved me a lot of time on design. I kept the frontend and backend separate mainly because it makes development easier for me, so I can work on one part without worrying abot breaking the other.

While building this, I ran into some challenges. The biggest one was dealing with lots of position data, so the system was getting slow when loading all positions at once. That's why I added pagination, which helped a lot. For security, I went with API key authentication, like it was ask in the features of the exercise, and because it was the quickest way to get things secure. I know it's not perfect, but it works for now

Making these decisions wasn't always easy. Using H2 database (like it was suggested) for development makes everything super quick to set up, but sometimes things work differently when switching to MySQL in production , I learned that the hard way. Setting up CORS took some time too. At first, I had the frontend and backend on the same port, but separating them made development much smoother, even though it meant extra security setup. Using Material-UI was great for getting things done quickly, but now I'm a bit locked into their design style.

Looking forward, there's quite a bit the project could be improved. The API key authentication works, but I think is better using JWT tokens, it would be better for handling user roles and permissions. It is also need to add more tests; right now, I only have basic ones. The position list could use some optimization too, especially when dealing with lots of data. These aren't urgent problems, but these are some suggestions which are well know as standars.

I tried to make sure the system can handle growth by making some early decisions about scalability. The pagination helps with large datasets, and keeping the frontend and backend separate means I can upgrade one without touching the other. I also tried to keep the code modular so I can add new features without having to rewrite everything. It's not perfect, but I think it meets the requirements of the exercise.