**Scenario**

Our school has a year-long stock investment project, which every student who enrolled in economic courses must attend. Justin, a friend of mine, is aiming to earn a higher amount of profit so that he can earn his pocket money for purchasing games from this project. However, it seems hard for him to decide to sell or buy stocks just with a single graph on websites such as yahoo finance. So, I offered him a plausible possibility by developing an application that would allow Justin to track his stock records and offer him suggestions of when to sell or buy based on statistical algorithms.

Justin agreed with this idea, which we spend another 30 minutes on WeChat messages and phone calls to make up with a blueprint. Both as students who want to minimize the time spent on projects, we agreed that information must be simple but direct. I suggested that the account should be secured well enough so that only the owner can change the purchase list on the application. Justin suggested adding visual representation such as bar graphs to show individual’s performance on the project.

We then discussed the format of this application, which we agreed to let the application be a simulation rather than a well-restricting time product. This is due to the consideration of students might not track their stocks on time while time lag exists when students communicate to school to purchase the stock.

**Rationale**

The Java Spring boot was chosen to develop the frontend of the application. Spring boot is a light and open-source framework that allows building high level Java applications. Unlike using programming language like C++ has a self-memory management system while comparing to Python has better performance speed.

Within Java programming language’s other similar frameworks such as spring, spring boot is friendly to develop within my need. More specifically, it provides auto-configuration and contains servers which spring does have and furthermore simplifies the time-consuming development which includes features that we don’t need.

Backend, however, uses python is used in this application. Even though statical algorithms are involved, languages like R are not used in this application due to the complexity of data not that huge and complex that need an actual statical analysis. In addition, one feature of this application is scrapping from databases to get the stock data that we need, which Python has better communication with those database APIs.

**Success Criteria**

1. User can successfully create and log-in to an account
2. Menu page is shown with simple data performance compared to last day
3. Stock can be searched by stock code
4. Stock can be searched by name
5. Stock can be added with correct input
6. Errors should be informed by wrong input (integers/string/etc.)
7. User can be able to select date before the log-in date to track history record or add stock on specific date
8. User should be informed if they can purchase the stock by the remaining money
9. User can view their own stock list and view on individual stock
10. User should be able to delete individual stock on their list
11. Numeral analysis is included to show daily/monthly performance
12. A bar/line graph is included to show daily/monthly performance
13. An excel spreadsheet can be executed to show the stock price history