FLASK APP

This document is about the web app use cases and architecture. At the end of the document, you should find a plan table. Response to me if you find any problems.

# THE INITIAL IDEA

1. Past conversation (can skip)

We have talked about the output expectation of the project which is a website provides users with history stock prices, news data to make a buy decision. This decision is backed with our ML models, users are prompted with a range of choices {Absolutely don’t buy, don’t buy, hold, buy, absolutely buy}. We also have discussed about the real-time characteristic of the system, that is, it needs to provide and process real-time data so that the decisions are reflected to the up-to-date market.

Those are the two main problems that we have spent lots of time discussing. We approach to the first problem using ML models which are a sentiment model and a price model. Furthermore, we also suggested a third model combine the two models to give the buy decision. The second problem was less discussed; therefore; I believe it’s time to begin implement the system. For building a real-time system, we’ll look into this in details later.

1. Let’s discuss (can skip)

Time is scary concept, so the best way is to tackle it asap. This document is mainly about building the system and I will mainly focus on the system from now. However, I still working on the sentiment model. I planned to finish a prototype in less than 3 weeks. It will be a lot of works but I’m positive about it. Next, let’s actually look into the idea.

1. The idea

So the web will be a decision orienting for buying stock, it will give a perspective of a certain stock of interest based on history prices and news. So the first question comes to my mind is why a person would want to use the system. Imagine we are new traders with no idea what stock to buy so how can we help ourselves. To answer this question, I put myself into a housewife situation trying to buy a new phone. There are many choices to choose, difference prices and brands and the right time to buy. To make the best decision, I can read reviews in the comment section to have good sense of what I going to buy. Next, I can look at the history prices and guess at what month should I buy it to gain advantage from discount. Those are the core ideas I’ll be applying to build the website.

1. User requirements

From the user standpoint, the system should help me with these things:

* Suggest me a buy decision based on history prices and news.
* Show me the data that it used to make decision.
* Remember my interest and notify when something good/bad happen.

For future development:

* Analyze user financial status and use it to make a suggestion.
* Build a plan based on user financial status.

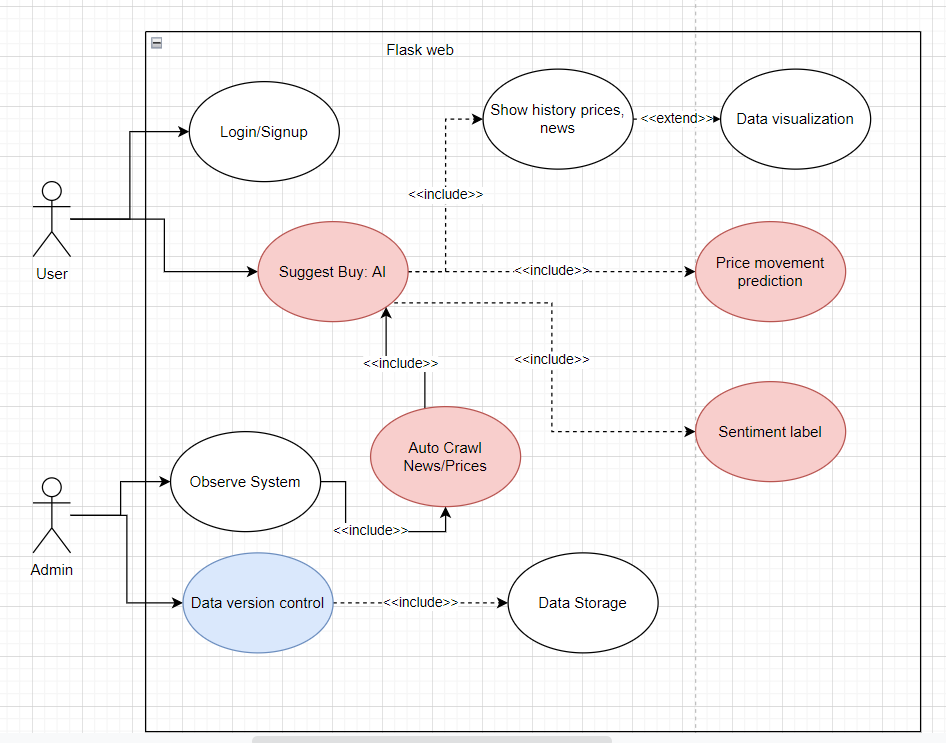
We won’t look at the things in the future for now.

1. System requirements

To meet user requirements, the system should be able to do these things:

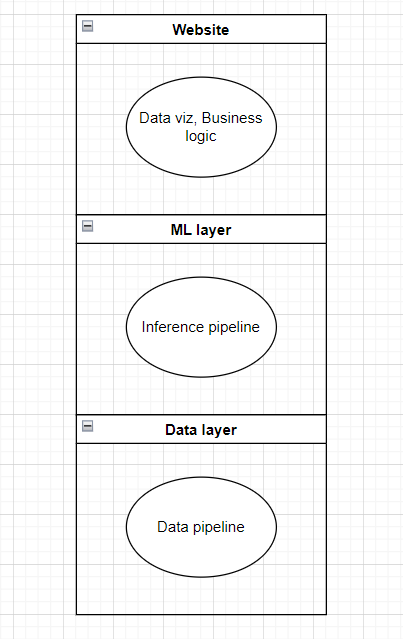
* Have storage for individual user data, and authorization for user data and system data. The data should also be version controlled.
* Have a proper way to show data to users. It should not show the whole thing to user as this can break the system.
* Have data visualization for big data.
* The system should store user contacts to notify user when needed.
* The user can register events for notifications.
* Allow user to view other stocks in same section: Not for now.

1. Use cases



The image above describes the whole use cases of the system. The red ones are the core functions and hard to implement as them need ML models. The blue ones are the available functions.

1. System architecture



The system consists of 3 layers and it follows the n-tier architecture. The reason is for fast development, future evolution and easy to debug. Please note, each layer can be a functional website so the project can be a combination of 3 different web services. A detailed version is available but I won’t show it for now.

1. The plan

Any project need a plan and I hope that the prototype can be done in less than 3 weeks.

|  |  |  |  |
| --- | --- | --- | --- |
| Group | Task | Layer | Duration |
| 1 | A simple web with CURD | Website | 2 days |
| Data pipeline | Data Layer | 1 days |
| 2 | Interact to model | Website | 3 days |
| Inference pipeline | ML Layer | 3 days |
| ML flows inspect | ML Layer | 3 days |
| 3 | Auto crawl data | Data Layer | 4 days |
| Auto update data | ML Layer | 4 days |
| DVC version control | Data Layer | 4 days |