TRADING VISION PROJECT - TVP

DD130 - DETAILED DESIGN

Version:	1.3
Status:	Approved
Approver:	Ngô Thái Bình Nguyễn Thị Diễm Trang
Author:	Nguyễn Bảo Nguyên Quách Hoàng Minh Ngô Gia Hân Nguyễn Vũ Anh Thư

netcompany

© Copyright 2020 Netcompany. All rights reserved.

Document history

Version	Date	Author	Status	Remarks
1.0	08/02/2022	Nguyễn Bảo Nguyên	Draft	
1.1	28/04/2022	Nguyễn Vũ Anh Thư Quách Hoàng Minh	Draft	
1.2	27/05/2022	Nguyễn Vũ Anh Thư	Draft	
1.3	29/5/2022	Nguyễn Vũ Anh Thư	Draft	
1.3	29/5/2022	Nguyễn Bảo Nguyên	Approved	Reviewed and reformat

References

Reference	Title	Author	Version

Table of contents

1 Introduction

1.1 Purpose

The purpose of DD130 - Detailed Design is to develop the implementation perspective from *O0500 - Software Architecture* and expand upon descriptions of components, classes, attributes, methods and relations.

The developers and architects who are to understand the design and implement or review the solution are the target group. The customer's technical architect may also be interested in reviewing the document for quality assurance of the solution. However, the document is intended to be an internal document which does not have to be approved by the customer.

It is also a prerequisite for giving an exact build estimate. The estimate should always be validated after the detailed design is done.

1.2 Scope

Although the classes are related to their position in the overall software architecture, the architecture should not be described in this document. Instead, please see *O0500 - Software Architecture* so that the architecture does not have to be maintained in two locations. It is preferred to link to relevant paragraphs in O0500 if the architecture needs to be addressed.

The deliverable is a snapshot of the detailed design at the point of transition to the Build phase and in principle will require no subsequent maintenance. However, it may be possible to update DD130 regularly through the different phases of the project, as it may be a requirement from the customer.

2 Preparation guidelines

In practice, detailed design/class design is prepared as increasingly detailed use case realizations, identifying the components, subcomponents, classes and methods of the system. This process is inspired by RUP (see [RUP-A] and [RUP-D] for a detailed analysis of the corresponding RUP phases).

Detailed design can be prepared for example using a dedicated UML tool. The deliverable can then either comprise the model itself, or it will be possible to extract the main content for the deliverable from the model in the form of a Word/RTF file. The UML model is typically begun at the start of the design phase, and content for a number of detailed design end products is extracted from here.

Irrespective of whether the deliverable is documented in a dedicated tool or in Word, it should include an introductory reading guide which also describes general design choices such as the tool used, the modeling of specific general aspects, the use of particular patterns or restrictions in terms of platform or standard packages. But the general architecture should not be described but instead referred to.

2.1 Selecting a modeling tool

We use *draw.io* for drawing diagrams. *draw.io* is free online diagram software. You can use it as a flowchart maker, network diagram software, to create UML online, as an ER diagram tool, to design database schema, to build BPMN online, as a circuit diagram maker, and more. draw.io can import. vsdx, Gliffy and Lucidchart files.

Besides, we also use *Lucidchart*. *Lucidchart* is a web-based proprietary platform that allows users to collaborate on drawing, revising and sharing charts and diagrams.

3 Solution

3.1 Solution for prediction

Taking everything into consideration, we saw from the Journal Articles of various authors that our problem could be solved by one of three algorithms. These algorithms are Long short-term memory (LSTM), Support Vector Machine (SVM) or Linear Regression. After evaluating the performance of the three algorithms on the Vietnamese stock market, we decided to choose the LSTM algorithm for our project. First, LSTM has the minimum MAPE value when compared with 2 other algorithms. Moreover, for linear regression, it is proved that it is only effective for a certain period of time in the old regimes. As the regime shift happens in the financial industry, the model became less effective.) In addition, LSTM is well-known for its application in time-series forecasting due to its ability to have several lags of unknown duration between important events in a time series.

In order to evaluate the algorithms' performance, we will use mean absolute percentage error (MAPE), which is a method of evaluation for LSTM model, stated in. MAPE means the percentage of difference between predicted values and actual values. For example, MAPE value of 11.5% means that the average difference between the predicted value and the actual value is 11.5%.

3.2 Design model (the actual class design)

The Design model is mandatory and extensive and should be aligned with the project's technical architecture. Like the Analysis model, the Design model is created by identifying the components, subcomponents, classes and methods of the system by means of use case realizations or user stories. The classes of the Design model are directly equivalent to classes/components to be implemented in the final system. Typically, all significant classes are identified as Design classes in the model prior to design work, while less significant classes/components are added on an ad hoc basis during the Build phase. It is not always relevant to update the detailed design during the build phase. Often the detailed design is just used as the starting point for the implementation.

3.2.1 Sequence diagram

As for the Analysis model, use case realizations are implemented by means of descriptive text and often a UML class diagram that illustrates the links between the Design classes included in the use case realizations. One or more UML

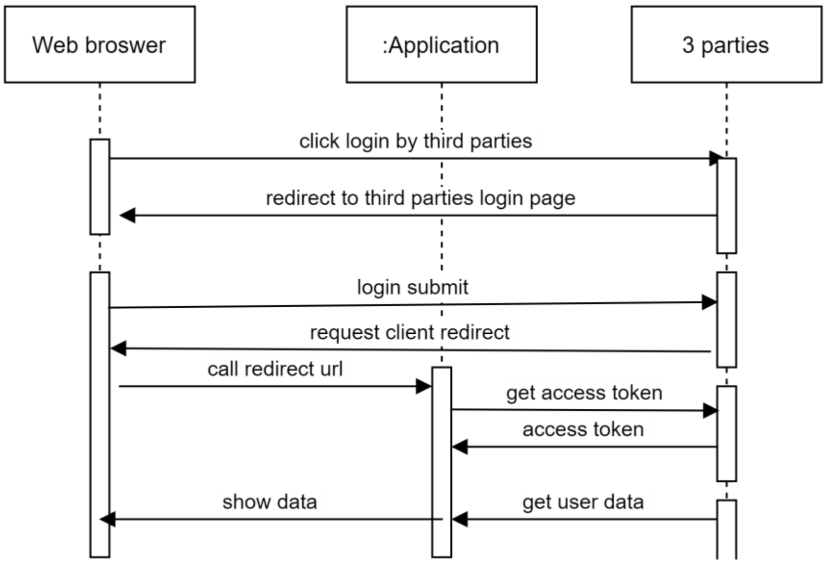

Figure 1: Login Diagram

The login diagram show the flow of login function in our website. User will click login and the request will redirect to third parties and it will pop up the login page of third party for user fill in login form. After user has fill username and password and submit the form it will request user redirect and web browser will call redirect URL from our website. After that, website request to get access token from 3 third party and it get user data to application and show the data for the web browser.

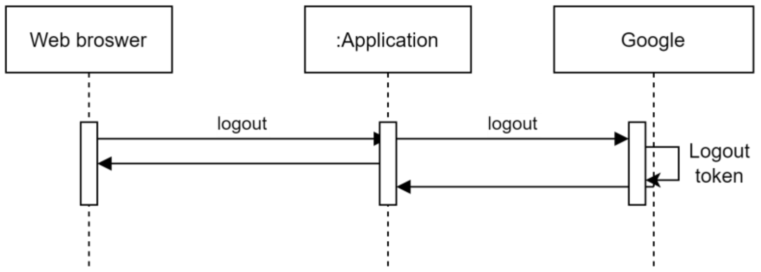

Figure 2: Logout

Logout function working when browser sent request to application and application give the request next to Google. After Google check token. It will sent the response to application and browser.

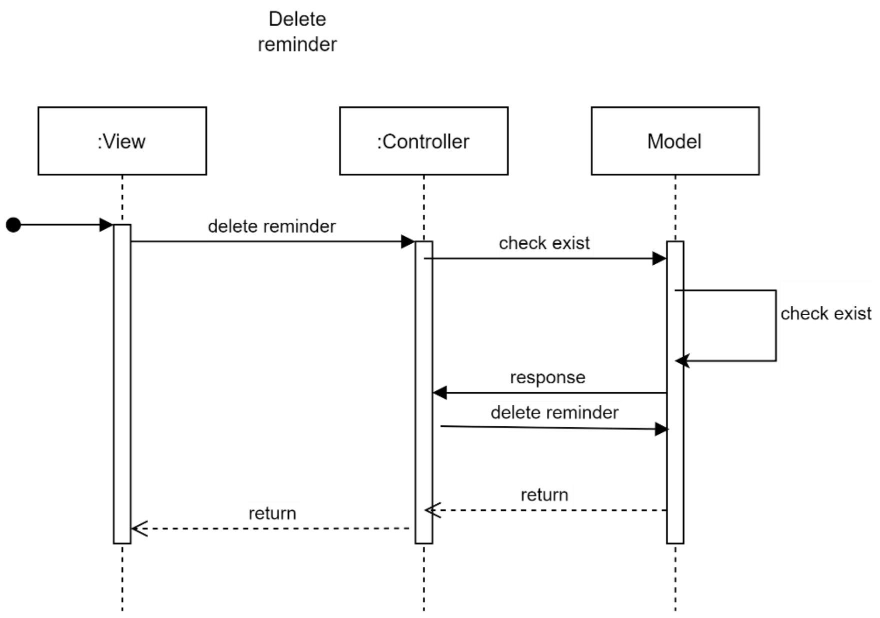

Figure 3: Delete Reminder

The delete reminder working when the view sent request to controller and controller send the request next to model to it check exist. After the check exist success. Model will send response to controller then controller could require model delete reminder. After the process success, the result will return from model through controller and view

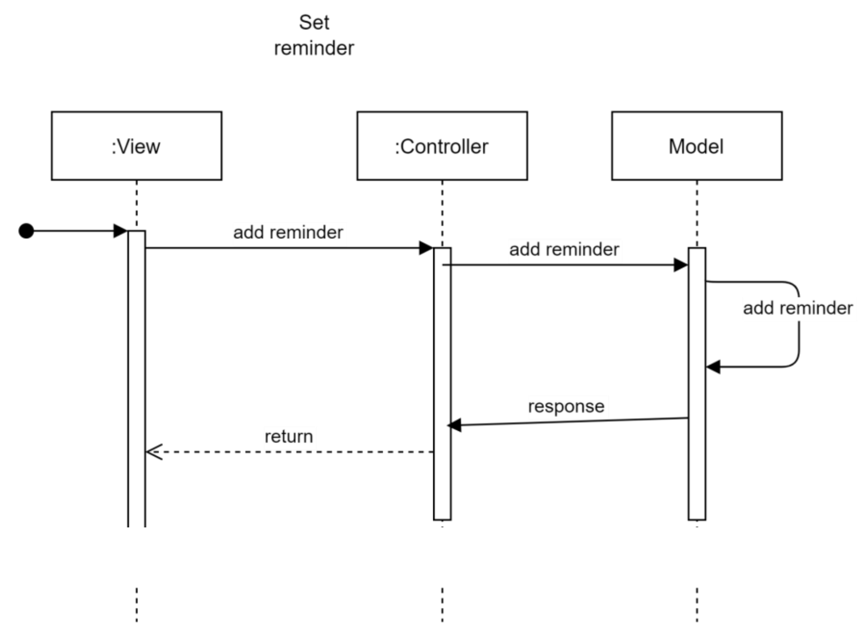

Figure 4: Set Reminder

The set reminder begin when view send request to controller and controller send it next to model to do it. After success, model will send response to controller and go next to view

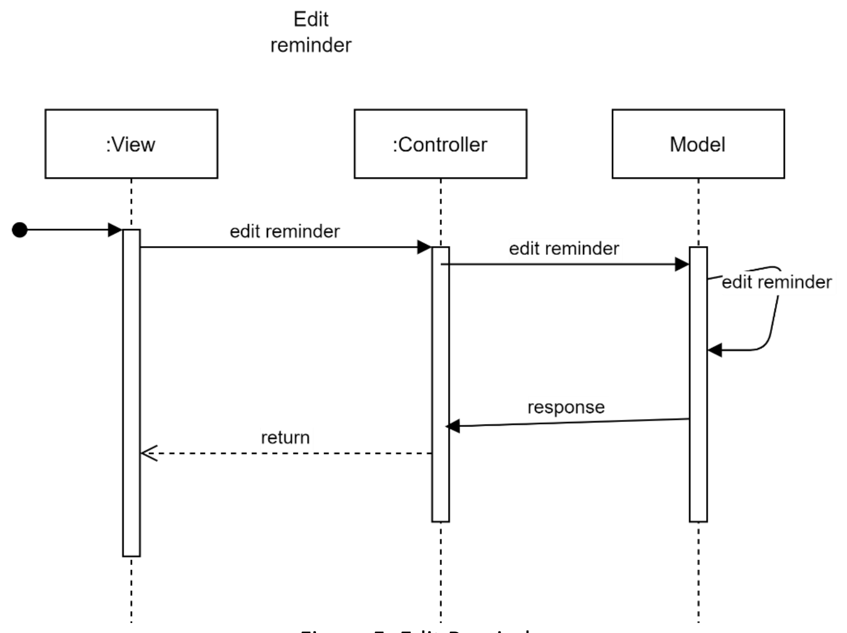

Figure 5: Edit Reminder

The edit *reminder begin when view send request to controller and controller send it next to model to do it. After success, model will send response to controller and go next to view*

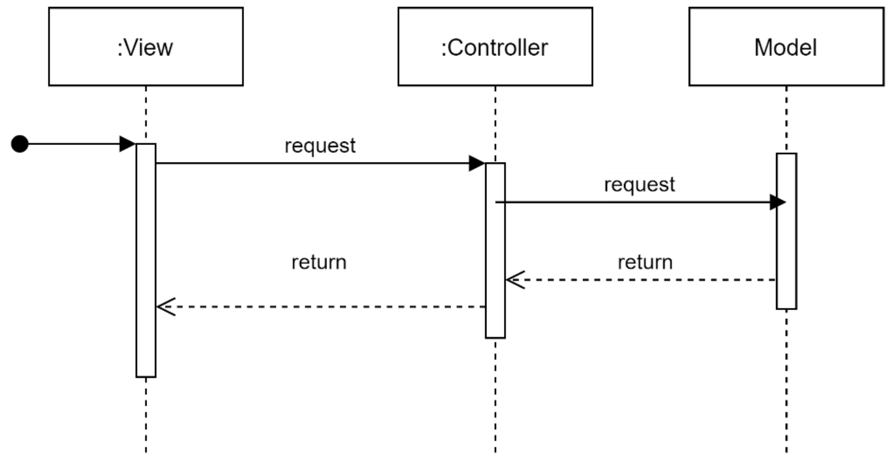

Figure 6: View Chart

The view chart *begin when view send request to controller and controller send it next to model to do it. After success, model will send response to controller and go next to view*

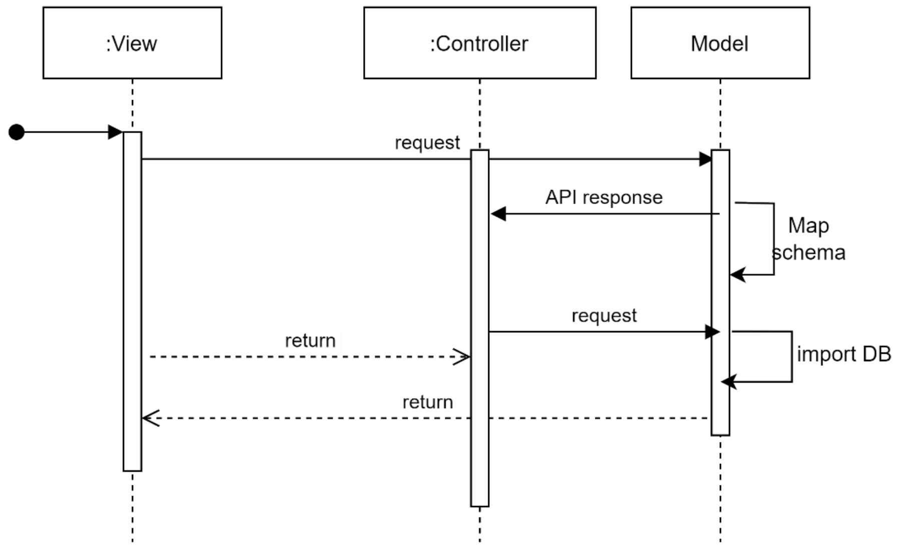

Figure 7: Fetch data

The fetch data begin when view sent request to controller and it send next to model. Model will map schema and response API for will, then controller send the next request for model import data. After model done, it will sned response for controller and view.

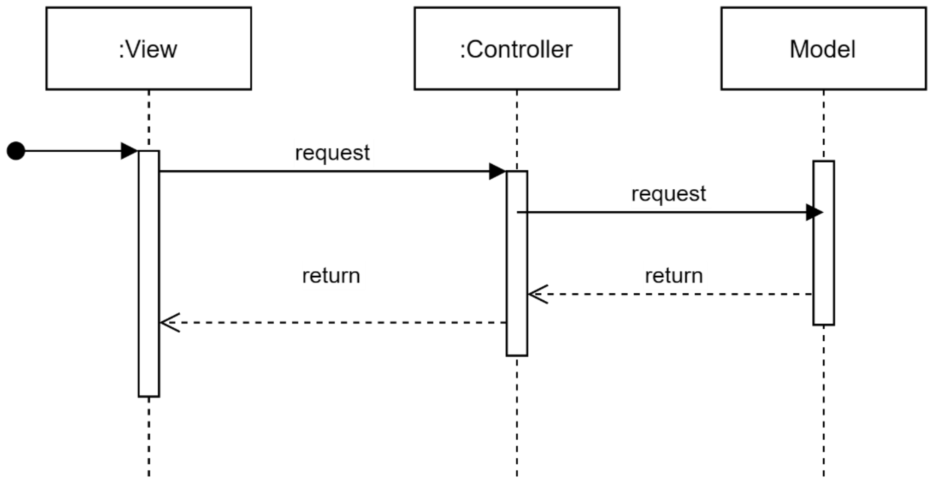

Figure 8: View Predicted Price

The view predict price *begin when view send request to controller and controller send it next to model to do it. After success, model will send response to controller and go next to view*

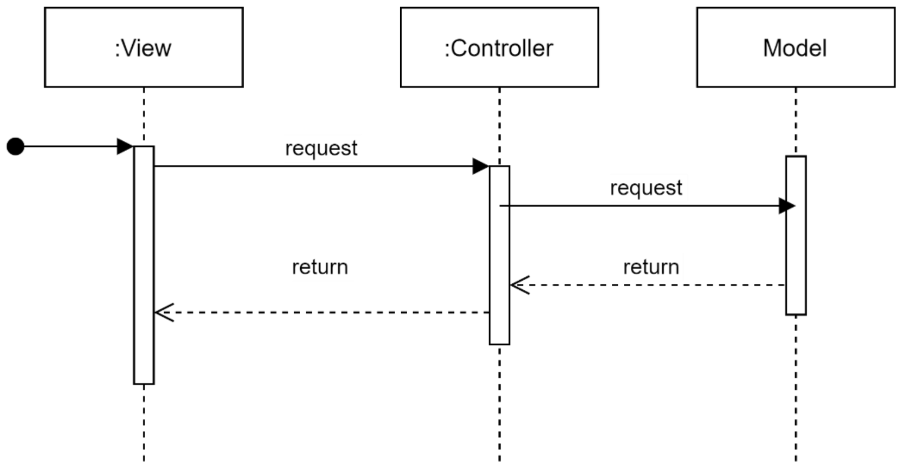

Figure 9: Compare Stock

The compare stock *begin when view send request to controller and controller send it next to model to do it. After success, model will send response to controller and go next to view*

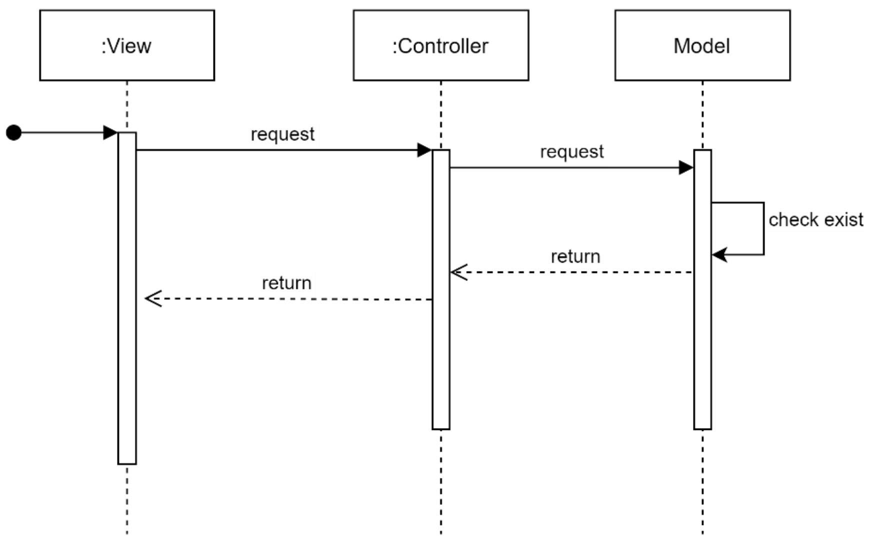

Figure 10: Search Ticker

The search ticket *begin when view send request to controller and controller send it next to model to do it. After success, model will send response to controller and go next to view*

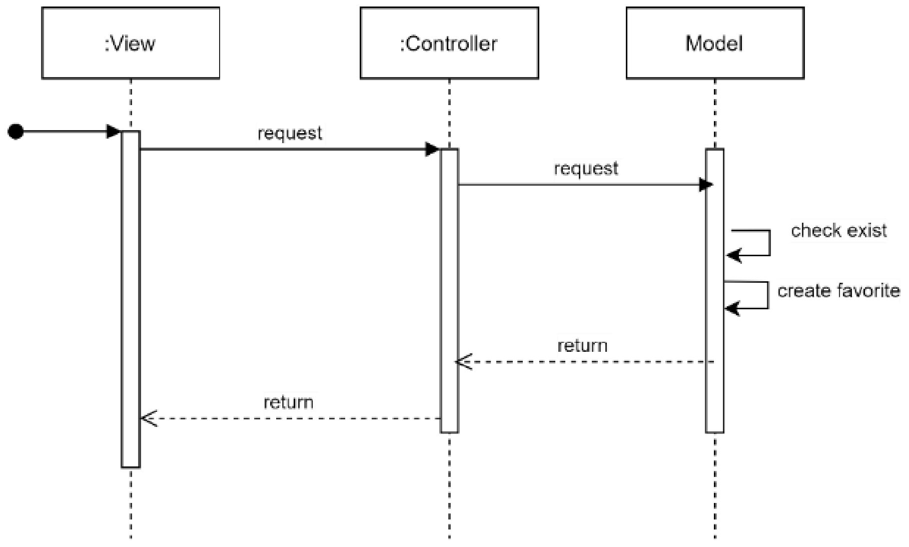

Figure 11: Add Favorites

The add favorite *begin when view send request to controller and controller send it next to model to check exist and create favorite . After success, model will send response to controller and go next to view*

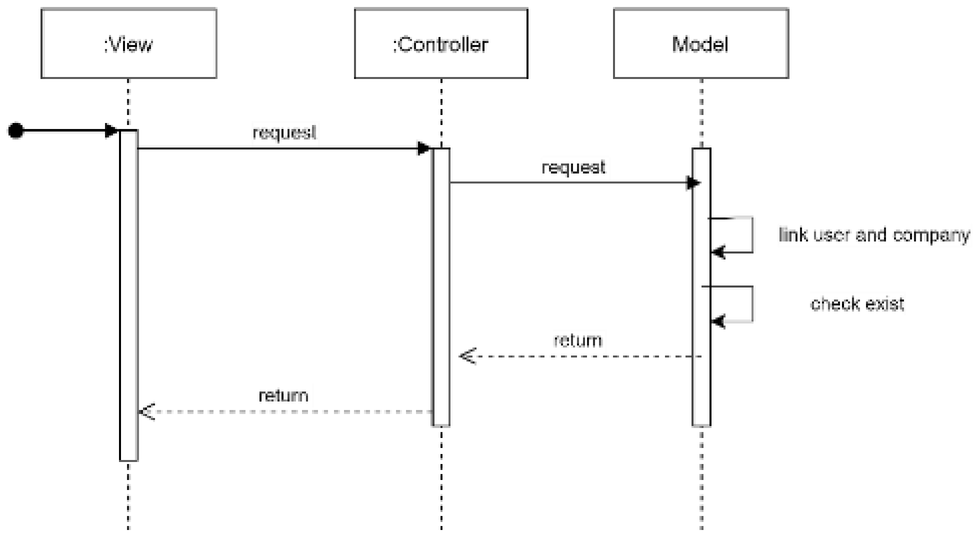

Figure 12: Get Favorites

The get favorite begin when view send request to controller and controller send it next to model to link user to company and check. After success, model will send response to controller and go next to view

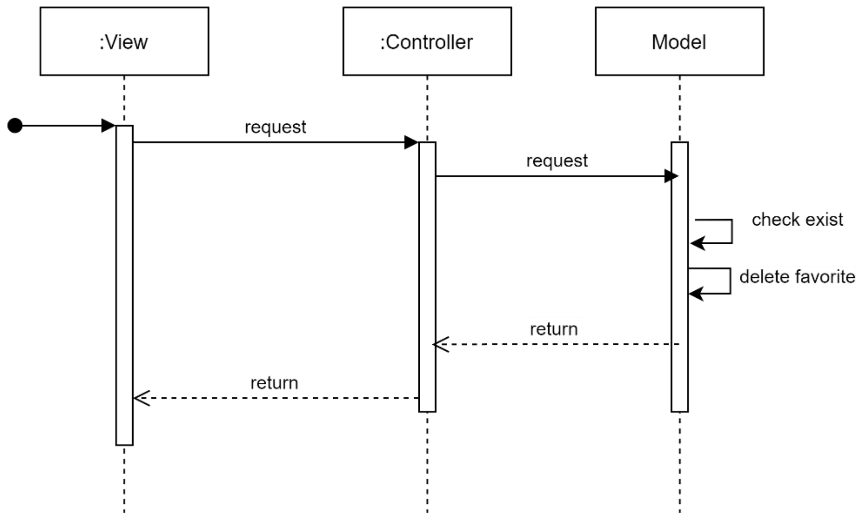

Figure 13: Delete Favorite

The delete favorite begin when view send request to controller and controller send it next to model to check exist and delete favorite . After success, model will send response to controller and go next to view

3.2.2 Class diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

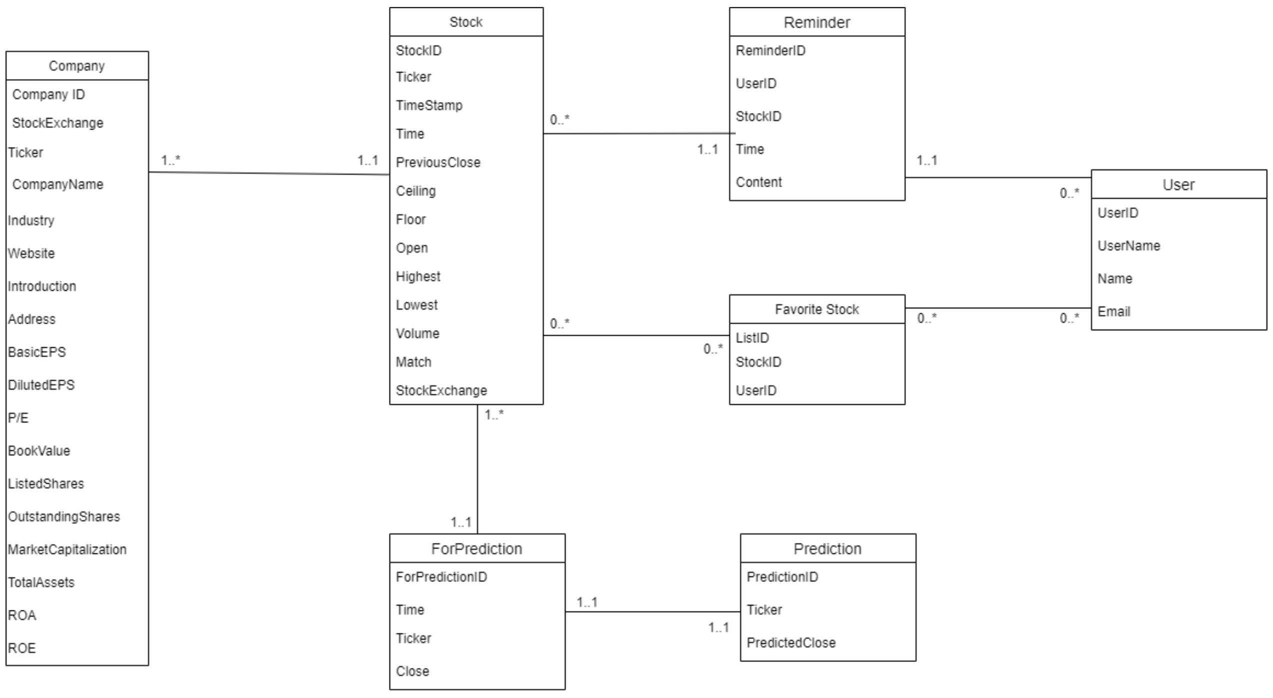

Figure 14: Class Diagram

4 Risk

After we investigated into login with Facebook function, we found that in order to be able to implement it, it's required to have an HTTPS domain. However, the domain name generated by Amazon EC2 is an HTTP domain. Therefore, it may be impossible to develop the function log in with a Facebook account.