TRADING VISION PROJECT - TVP

DD130 - DETAILED DESIGN

Version: 1.2

Status: Draft

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	

References

Reference	Title	Author	Version

Table of contents

1 Introduction 3
1.1 Purpose 3
1.2 Scope 3

2 Preparation guidelines 3
2.1 Selecting a modeling tool 4
2.2 Selecting models for detailed design 4
2.3 Analysis model 5
2.3.1 Use case realizations 6
2.3.2 Analysis class design 6
2.4 Design model (the actual class design) 7
2.4.1 Use case realization 8
2.4.2 Class design 8
2.5 Implementation model 9
2.6 Format 9
2.7 Review recommended 9

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 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.

3.1.1 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 interaction diagrams that illustrate how the objects interact in order to implement the use case. Entity-Control-Boundary Pattern identifies the elements for a scenario of system behavior; you can align each participating element with one of three key perspectives: Entity, Control, or Boundary. Although specifics of languages, frameworks, and heuristics of quality design will drive the final design, a first cut that covers required system behavior can always be assembled with elements of these three perspectives.

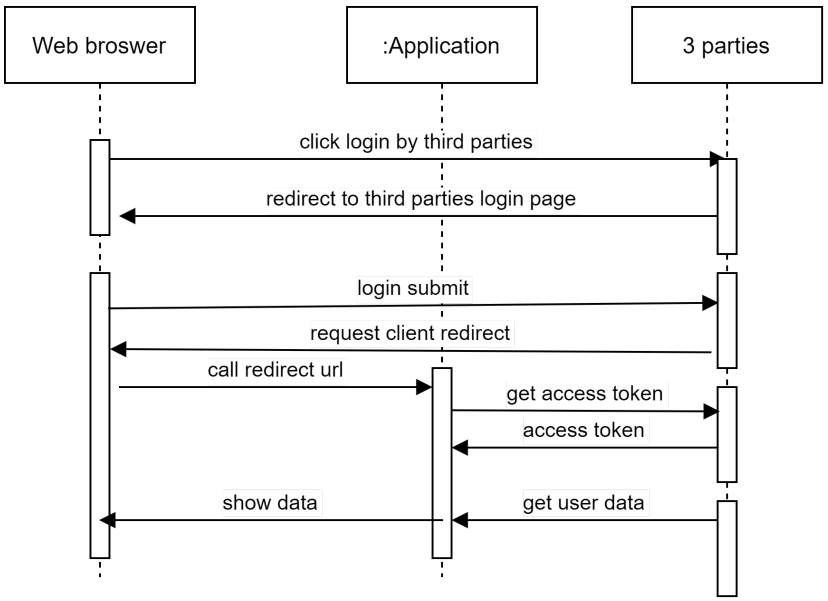


Image 1: Login

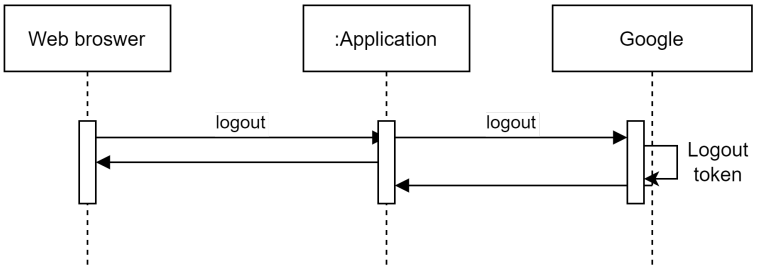
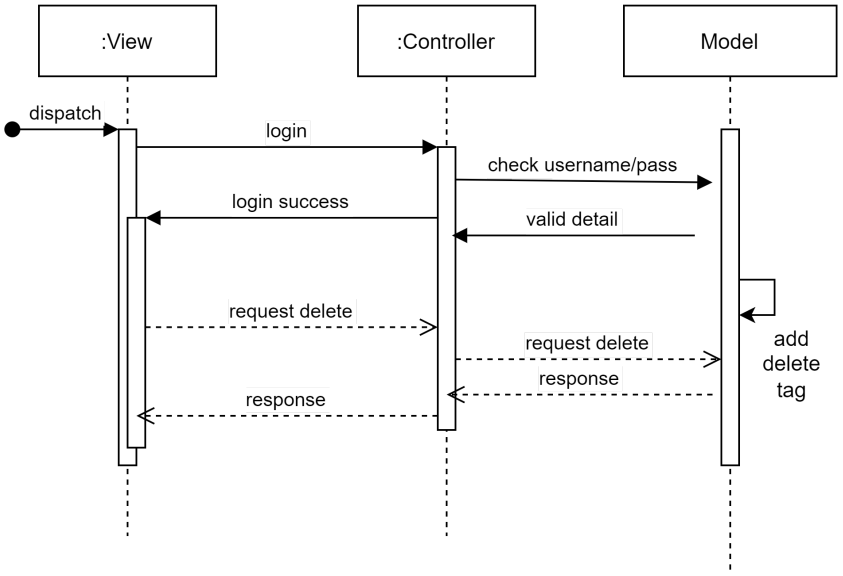


Image 2: Logout



Set
reminder

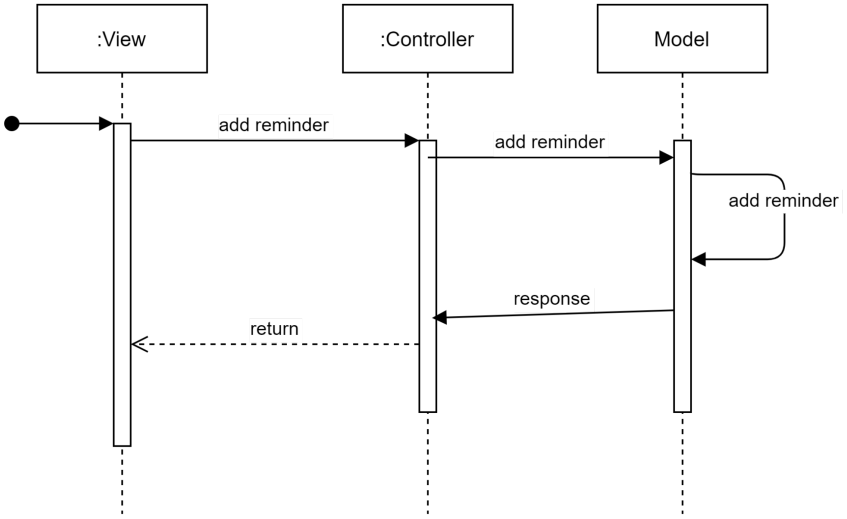


Image 3:Delete Account

Image 4:Set Reminder

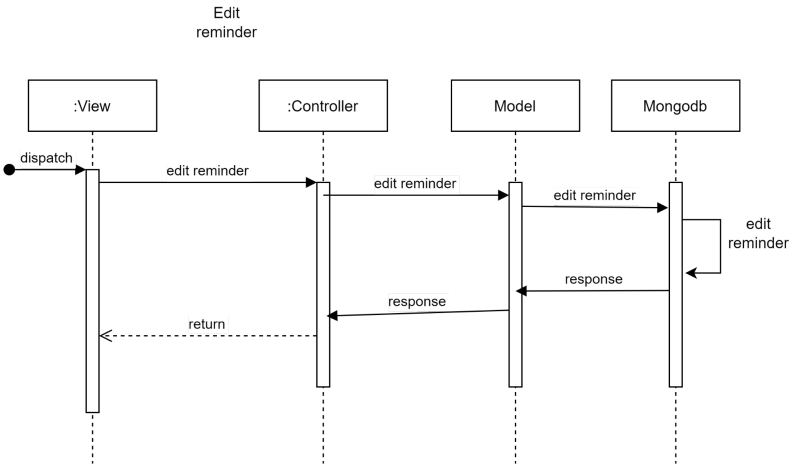


Image 5: Edit Reminder

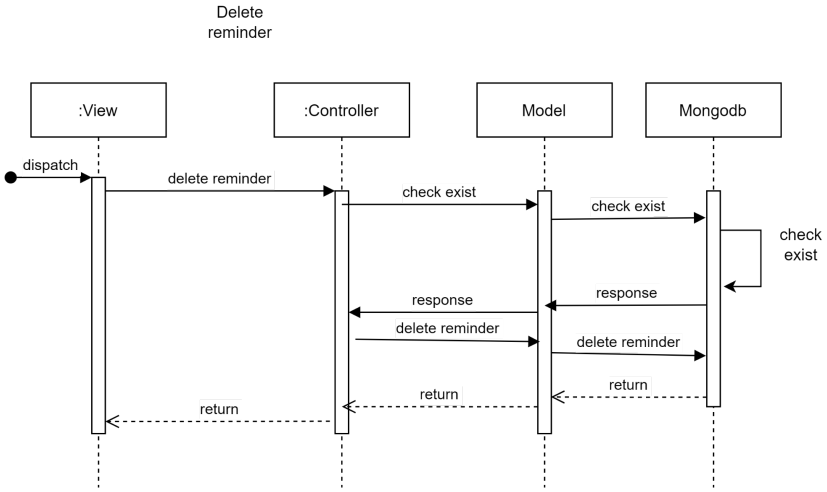


Image 6: Delete Reminder

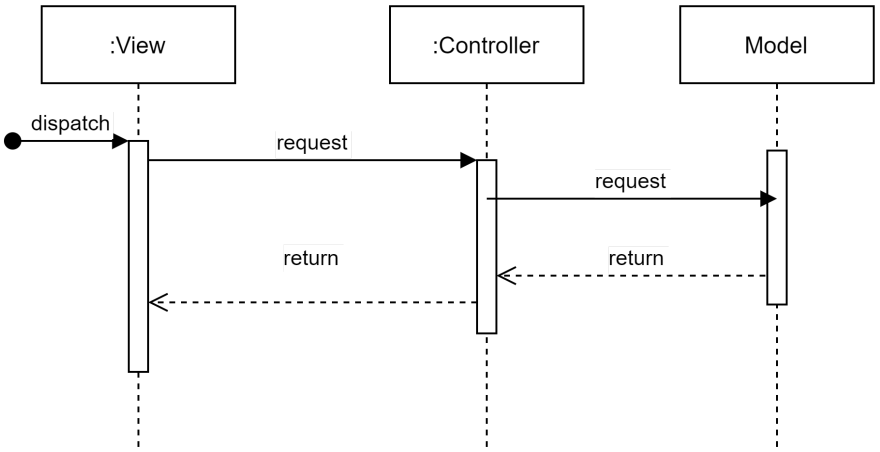


Image 7: View chart

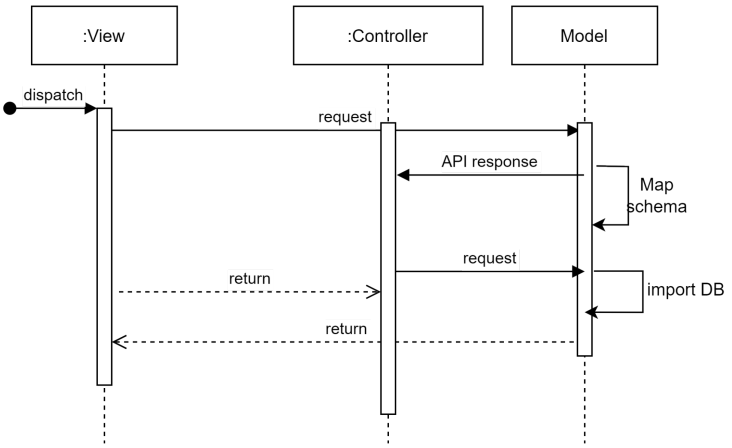


Image 8: Fetch data

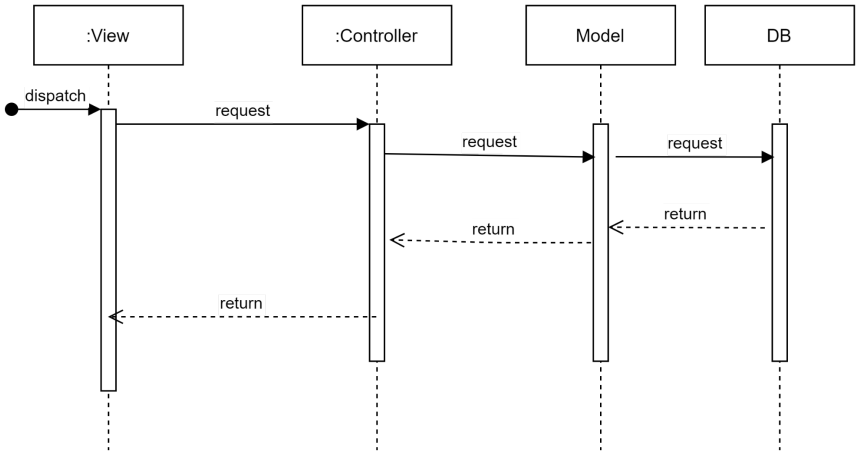


Image 9: View predict trend

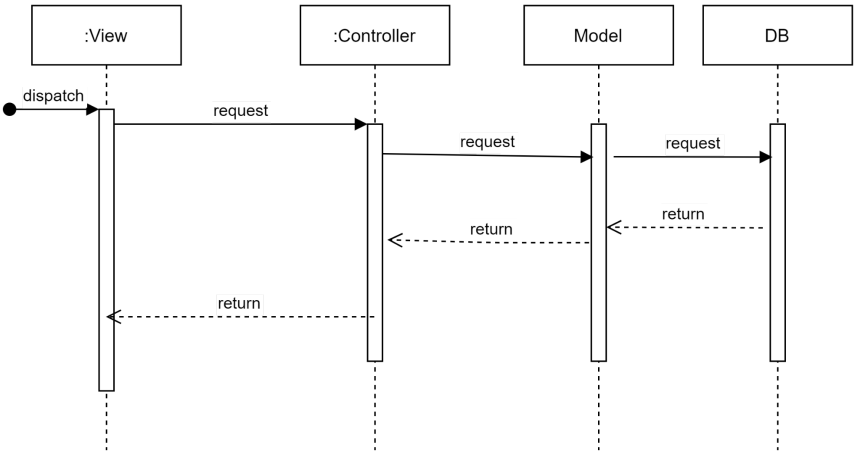


Image 10: Compare Stock

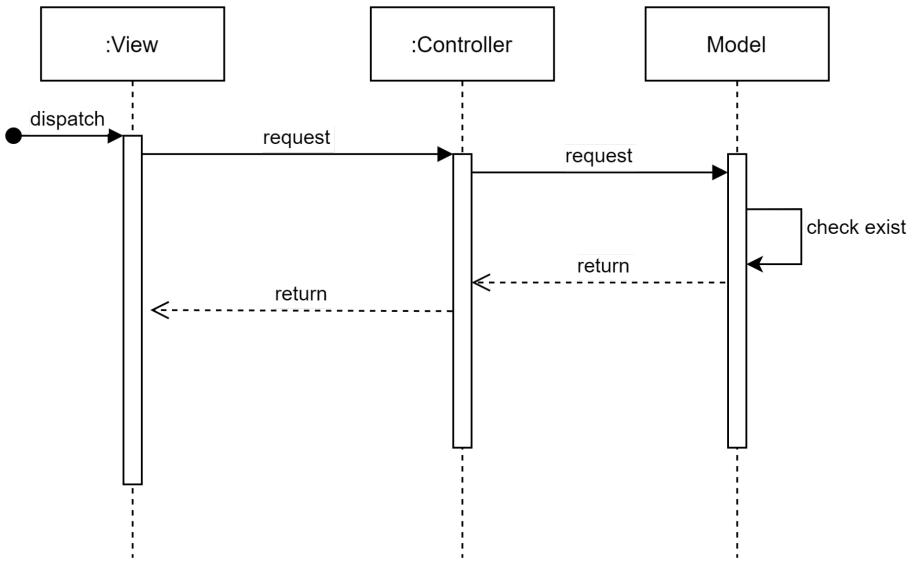


Image 11: Search ticket

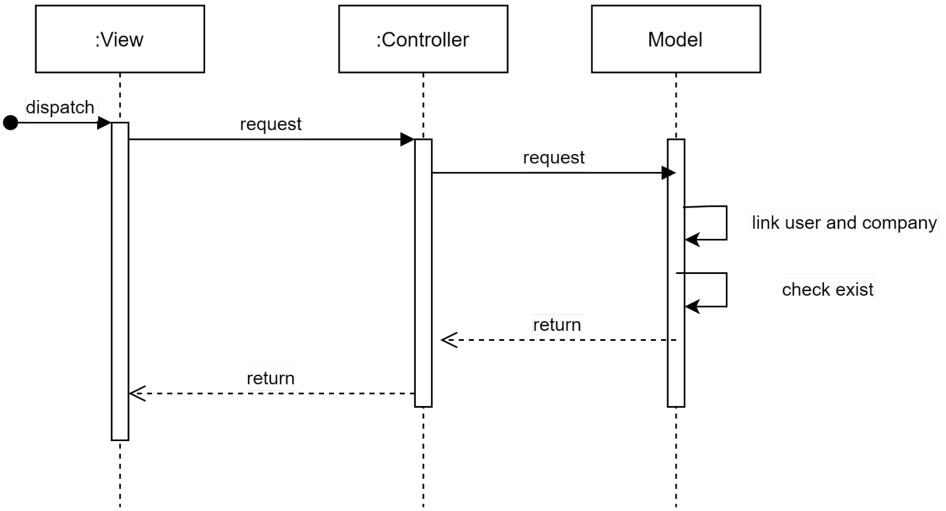


Image 12: Get favorite

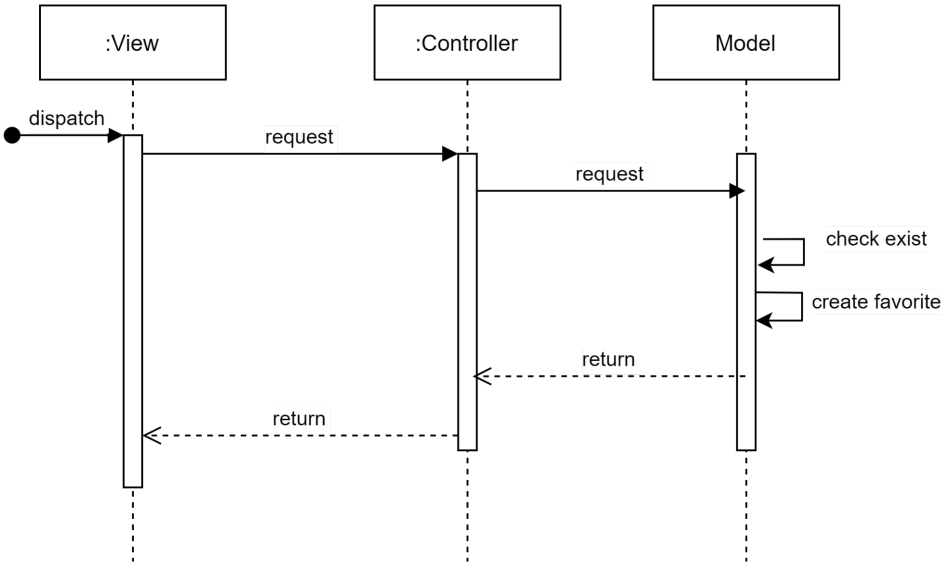


Image 13: Create favorite

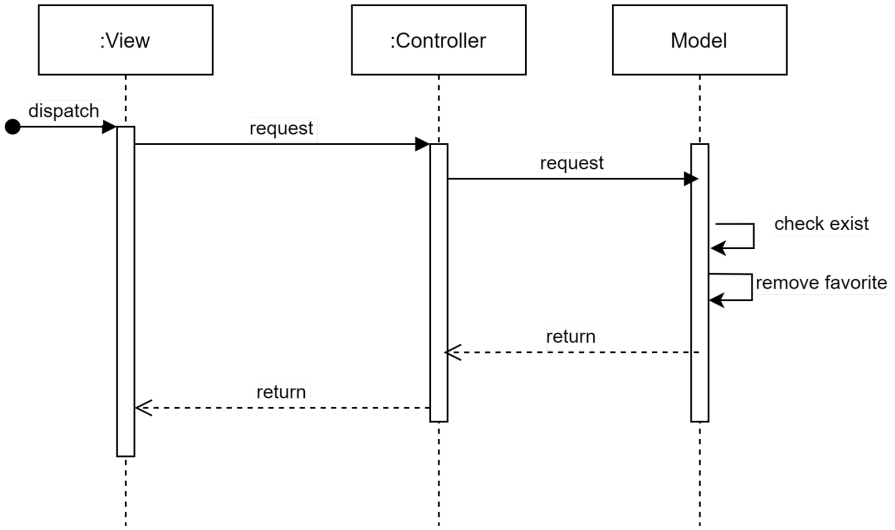


Image 14: Delete favorite

3.1.2 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 also 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.

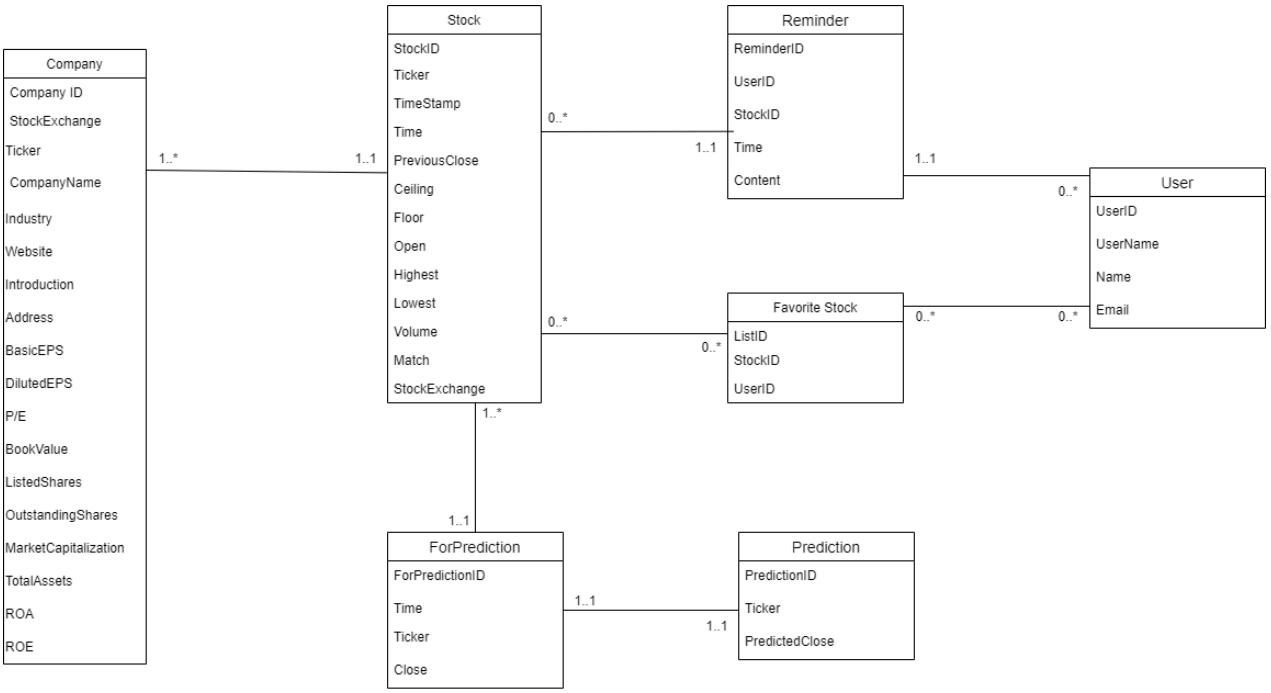


Image 15: Class Diagram