Stock Market Simulator Application

Analysis and Design Document

Student: Barabás Hunor

Group: 30432

Revision History

Date	Version	Description	Author
<04/04/18>	<0.1>	First Iteration	Barabas Hunor
<21/05/18>	<0.2>	Second Iteration	Barabas Hunor

Table of Contents

I. Project Specification

The objective of this project is to develop an application that can help simulate day-trading and investing in the stock market. The application will serve as an educational/practice tool for those who seek to enter the market, but have not enough experience to trade, relatively speaking, safely.

The user will have the options to open wallets, which come with a default amount of money on them. They can then spend this money to buy and sell stocks, which have their price updated each day. Both the price data and the money used is artificial. The price data used will be historical, meaning that if you want to buy a Bitcoin today, you might very well be spending only a couple hundred dollars for it in the application. This is to keep the original curvature of the price fluctuation. Of course, you can cheat and look the actual data up online, but then you defeat the purpose of the application, which is to experiment and learn.

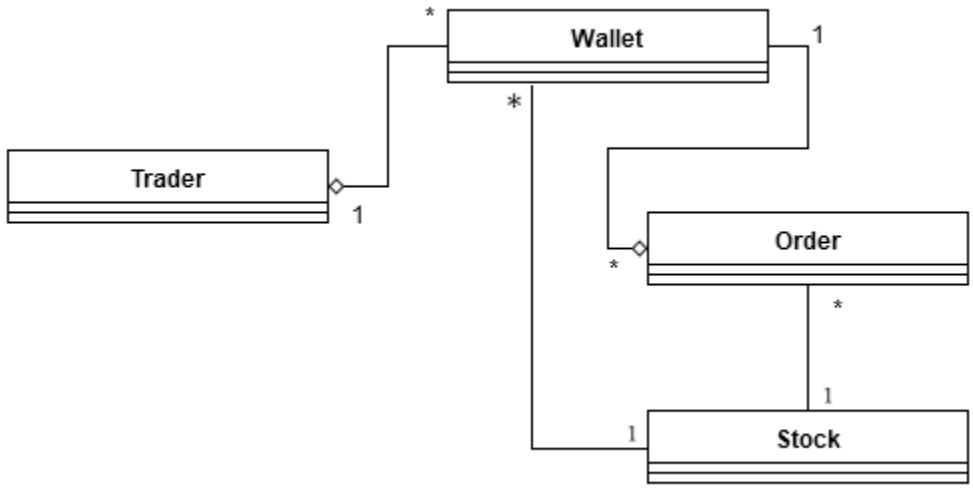
II. Elaboration – Iteration 1.1

Domain Model

The domain is somewhat similar to Assignment 1 in that it contains the models required to differentiate users: I'll have a Login model with information such as username, password and role. I'll have a Trader Model with basic user information.

Each trader can open several wallets, which all come with a predefined balance of USD. They can then make several orders for various stocks.

We have a model for the Stock itself, with its symbol and description. Finally, we have StockPrice, which has price data for each day for every stock.

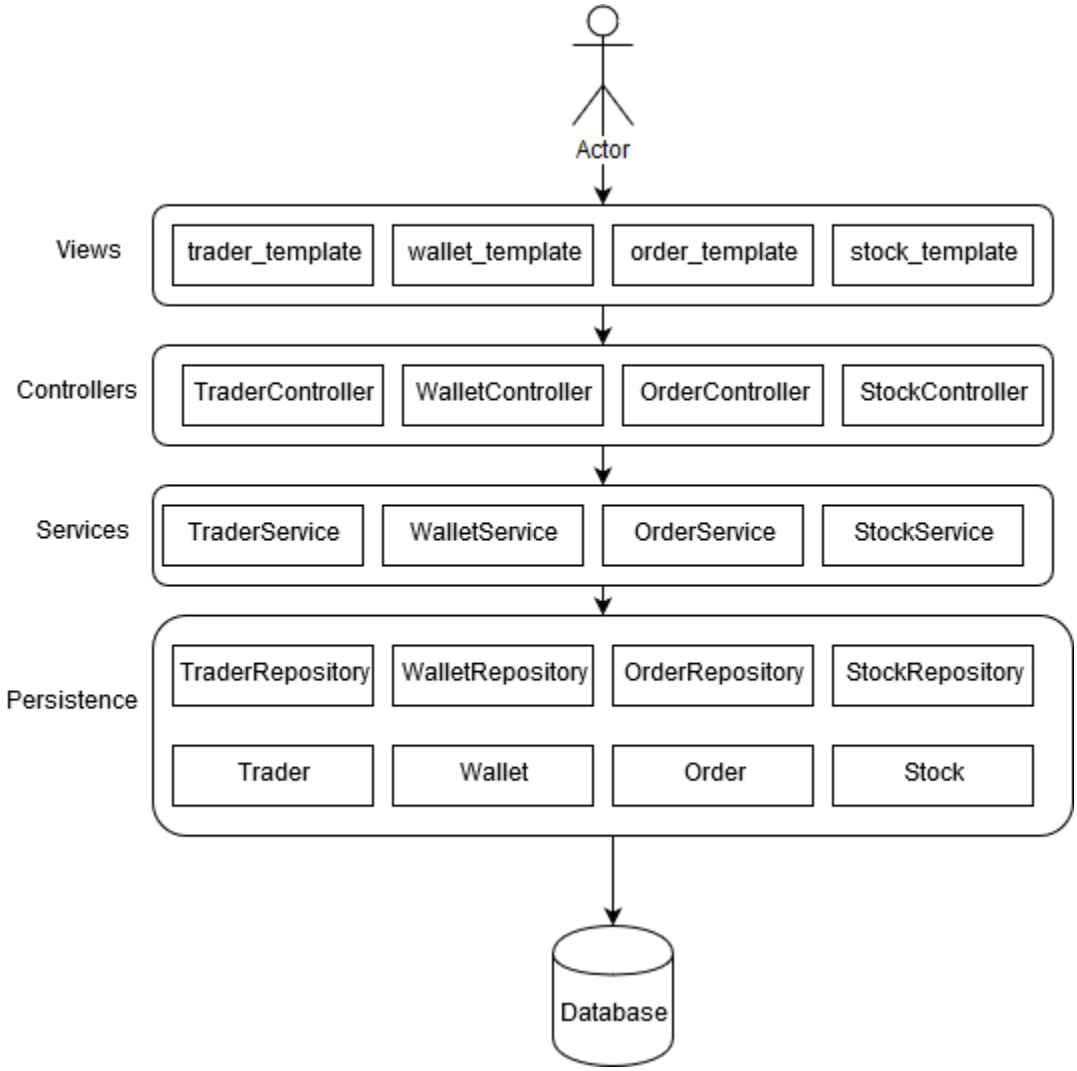


Version: <1.0>

Date: <04/04/18>
<Project – Barabas Hunor>

Architectural Design

Conceptual Architecture



For the purpose of this project I will use the Layer Architectural Pattern. If I were to use real-time data this would not be enough in itself, as I would need a mechanism to update my database every second. Thankfully, due to the nature of a simulator, I can use past prices of certain stocks which I will store in a database. The application will access, process and show that data to the user through a traditional 3 layer architecture:

Persistence

Pretty self-explanatory, this layer will have the responsibility of accessing data in our database. That can be anything ranging from prices for a certain stock and certain date, user wallet data, login information etc. This layer will mostly contain the above mentioned models and data access objects for each and every one of them.

Services

Input validation and any business logic will be solved here so to make sure that the data going through fulfills our requirements.

Presentation

The user interface will be built in this layer. Users will be able to buy/sell stocks with imaginary money. They will have the option to follow price charts and export reports which detail their balance changes over the past x days.

There are two components making up this layer. The Controllers, and the views, which are html templates built with Thymeleaf.

MVC

To complement the Layered Architecture I will use the Model-View-Controller design pattern. This is composed of the 3 parts that give its name:

Model – Lowest level, responsible for data maintenance.

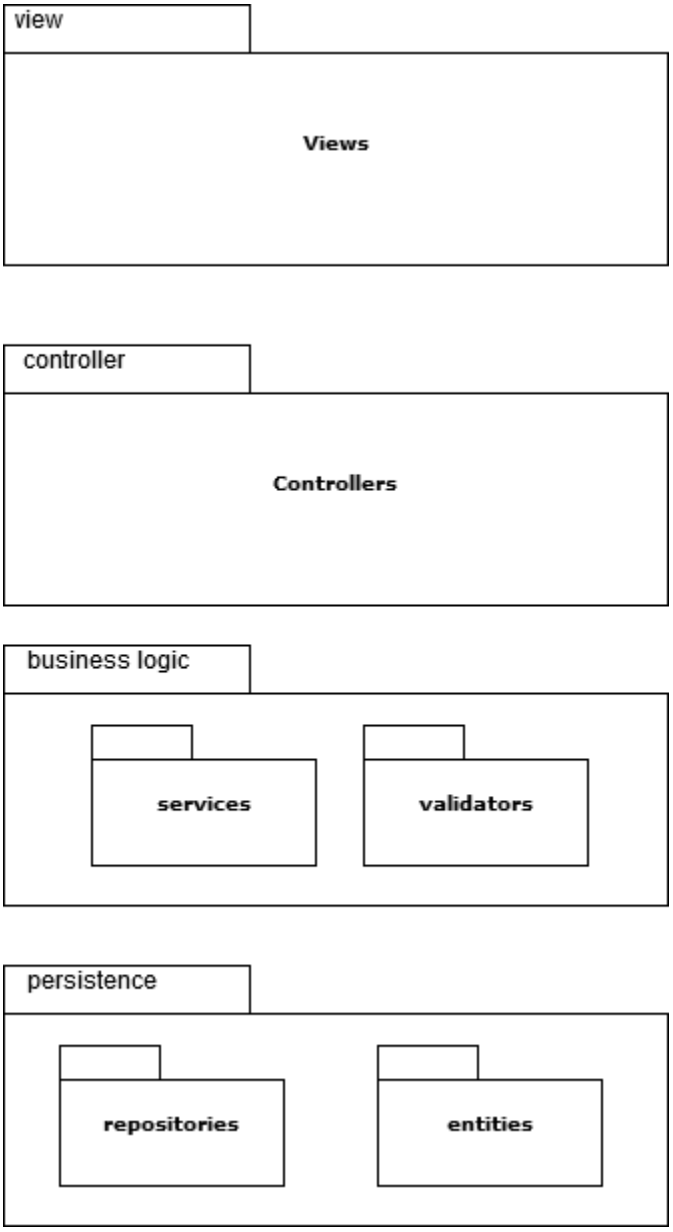
View – Responsible for displaying data on a user interface

Controller – Controls interactions of the above two.

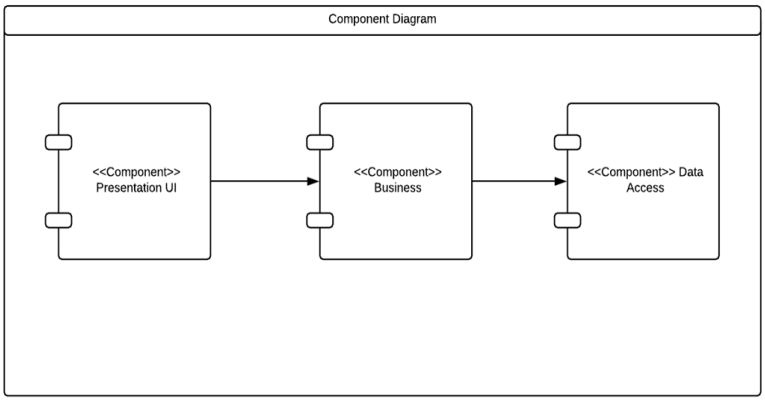
Version: <1.0>

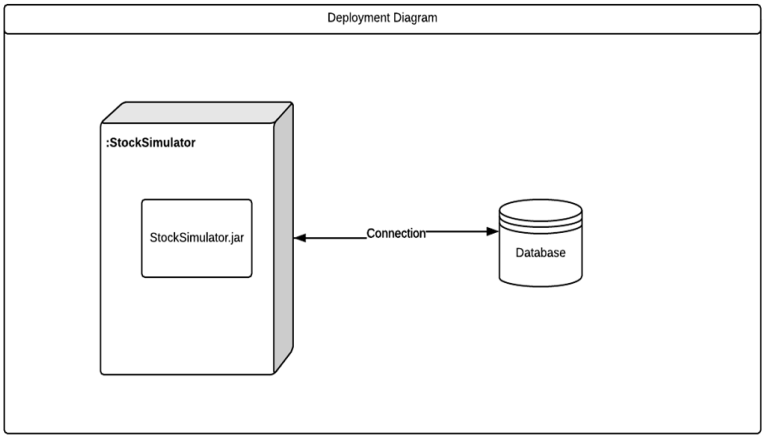
Date: <04/04/18>
<Project – Barabas Hunor>

Package Design



Component and Deployment Diagrams


Component Diagram
<<Component>>
Presentation UI
<<Component>>
Business
<<Component>> Data
Access


Deployment Diagram
:StockSimulator
StockSimulator.jar
Connection
Database

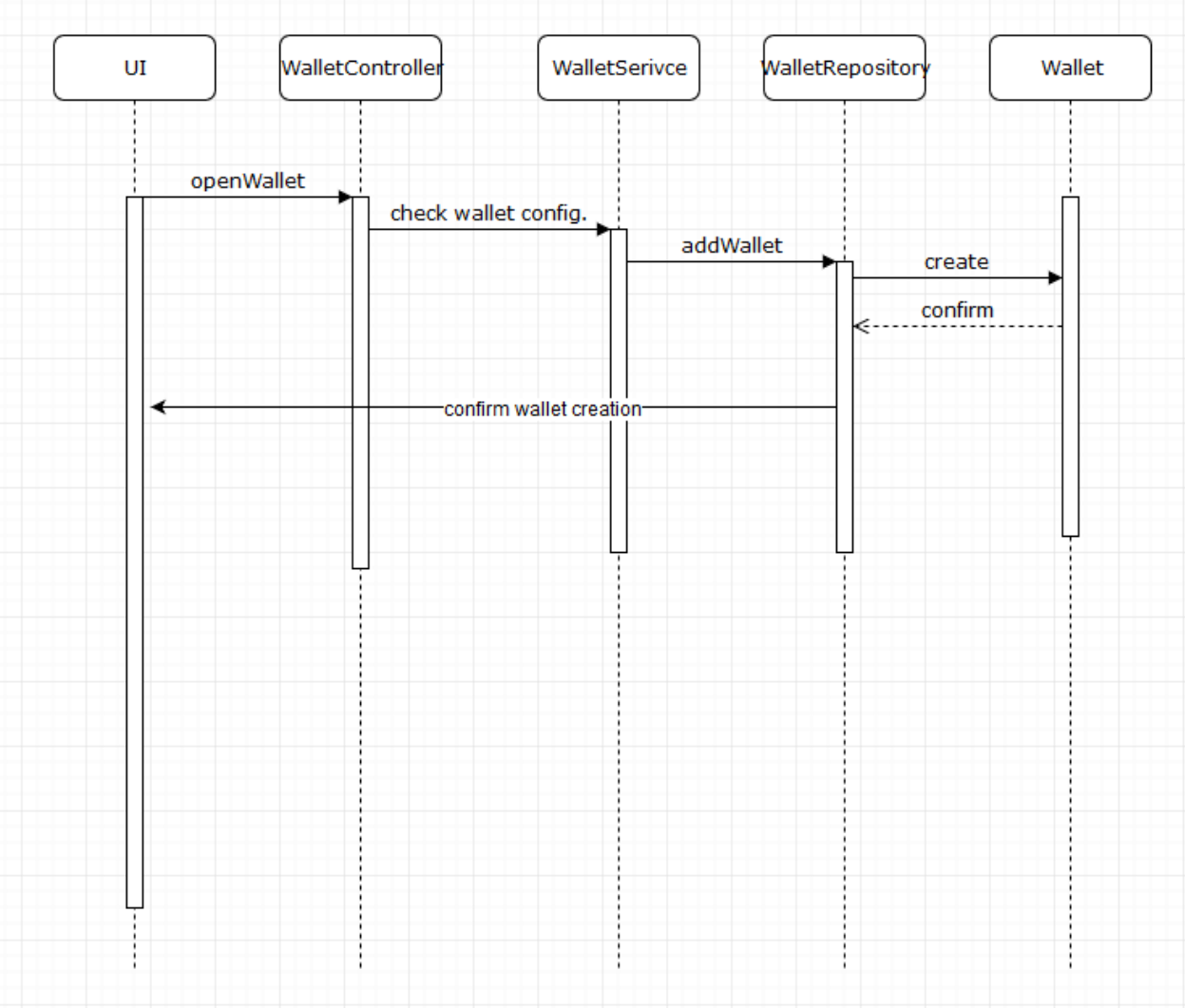
Version: <1.0>

Date: <04/04/18>
<Project – Barabas Hunor>

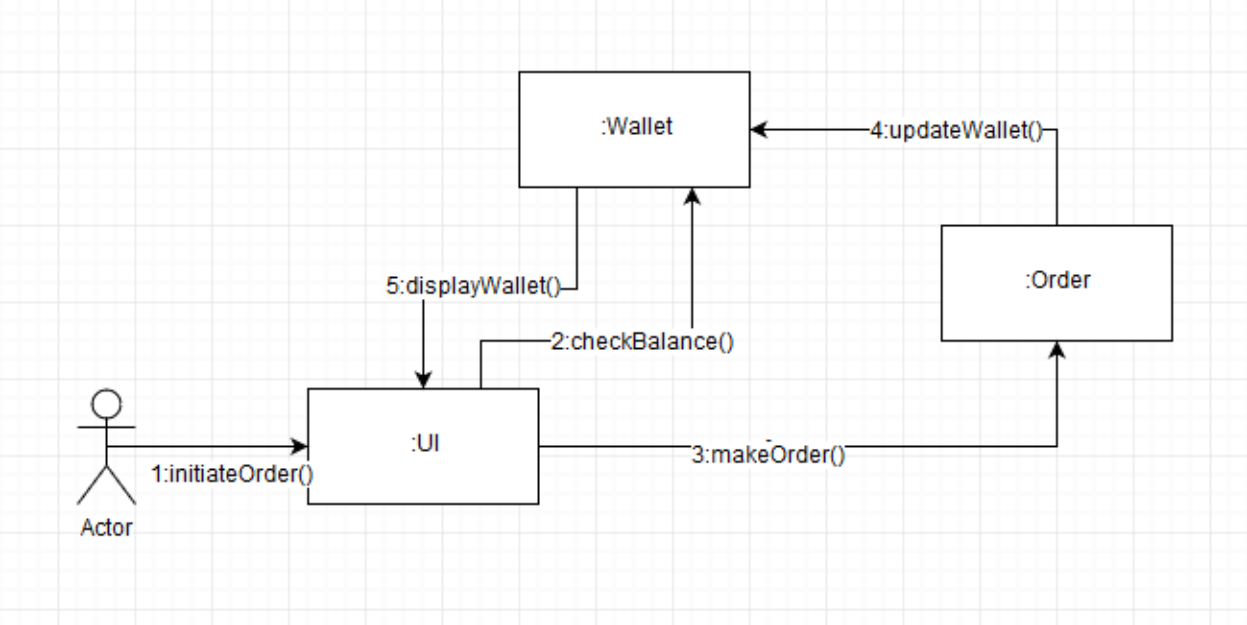
I. Elaboration – Iteration 1.2

1. Design Model

1.1 Dynamic Behavior



1.1.1 Opening a trading wallet – sequence diagram



1.1.2 Making an Order

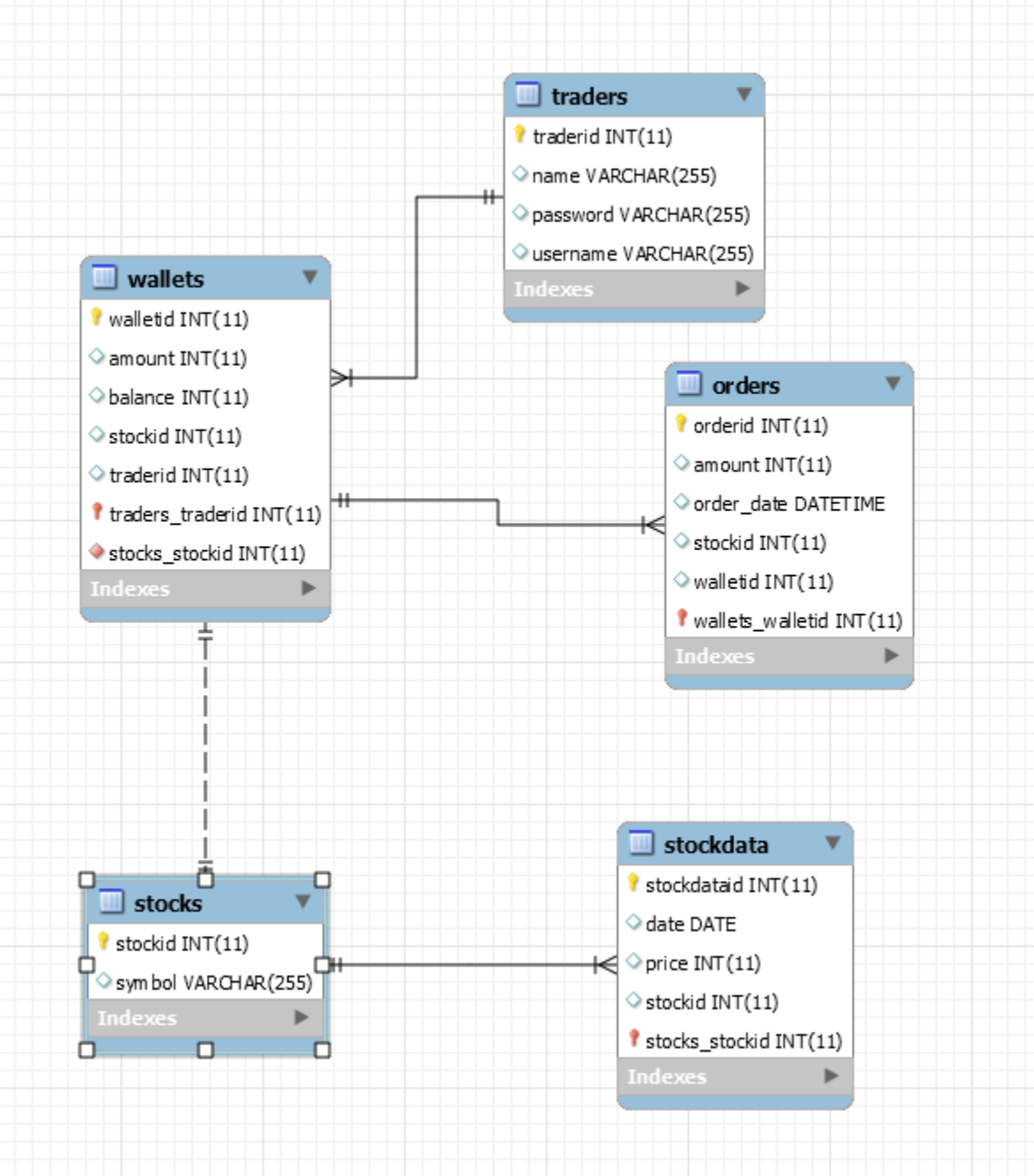
1.2 Class Design

My class design will reflect a traditional layered application's class design.

Version: <1.0>

Date: <04/04/18>
<Project – Barabas Hunor>

2. Data Model



Construction and Transition

1. System Testing

Unit and system testing will be done with mocking after the application is built.

2. Future improvements

Future improvements could include the implementation of real-time data. This could make the program much more useful as people could actually test themselves on not just observing price changes, but paying attention to the news regarding certain stocks. They could then trade accordingly. It would be more actual, but it would require some for m of collecting real-time data every day.

I. Bibliography

Spring Boot Reference Guide - https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/

Thymeleaf Documentation - https://www.thymeleaf.org/documentation.html