StockStockrs

Design Document
17th September 2016

Atul Aneja

Jalaleldeen Aref

Tarang Khanna

Wyatt Larkey

Joel Van Auken

Team 7

Index

PURPOSE

Most people view stock market trading as only available to those who are already wealthy, or those who devote hours upon hours of time into perfecting the art of trading. Many people today find themselves intimidated by higher risk investment practices like stock trading. Some of these people do not have the time to devote to researching into companies, others simply do not trust stock brokers, but the bottom line is that for many people stock market trading seems out of their grasp. Many current stock trading applications do not cater to the beginning investor, and continue to only cater to those already experienced in trading.

What we wish to accomplish with Stock Stockr is a stock information application for the average person interested in stock trading. We will provide a program that uses machine learning algorithms to analyze stock prices and other market data to give users an idea where markets are heading. This will allow more people to confidently begin investing in the stock market. We also plan on allowing users to learn about the metrics and jargon associated with stock trading to better familiarize users with common stock trading terms. This way, we hope to be transparent and explain to users at a high level how stock prices can be predicted. Lastly, we would also like to implement stock trading tools like virtual currency trading to also help users hone their trading skills. The bottom line is to provide the most helpful and user friendly stock trading application available.

Functional requirements

1. Users can create a personalized environment to view stock information. As a user, I would like to…
    a. Create an account, that will serve as a home screen when opening the application.
    b. Customize my account to display stock information that I as a user am interested in.
    c. View Predicted stock data visually or numerically.
    d. Add new stocks to my watchlist.
    e. Receive notifications about stocks by email.
    f. Test their trading skills with a fake, in-application currency.
    g. Adjust the scope of predictions (short or long term investing goals).
    h. Request new stocks to be predicted by selecting from a list of available stocks.

2. Users can browse stock data they are interested in and add it to their account. As a user I would like to…

    a. Browse trained stock data on the application.
    b. Access helpful explanations for data metrics and other educational information.

c. Contact developers, other users, and publicly make posts.

Non functional requirements

1. Client Requirements
   a. The application should be able to be used on a web browser.
   b. The application should also be able to be used on Android or iOS by use of a hosted application or through a web browser.
2. Server Requirements
   a. Our Server should be able to store and receive data from our database tables, and prioritize speed or security independently for each table.
   b. Our server should be able to communicate directly with our machine learning processor and receive trained data from it.
   c. Our server should be able to communicate with an API in order to get new raw stock data for the machine learning algorithm to train.
3. Design Requirements
   a. Our backend and frontend should communicate using JSON structures.
   b. Our server should update the front end on user requests and also every so often specified by a developer.
4. Performance Requirements
   a. The application should be responsive to all user input, and handle error cases and time outs gracefully.
   b. The applications stock data should be updated every minute, or possibly faster if possible.
   c. The application should scale to include many more stock market data sets without having to change any server or front end code.
5. Appearance Requirements
   a. The UI must be responsive and usable on all devices.
   b. The interface must be intuitive and all basic navigation between pages should be available at any point on the screen.

6. Security Requirements
    a. User data such as usernames, passwords, email addresses, and other personal data should be securely stored and encrypted to protect users.
    b. Stock market data and faux currency will also be encrypted, but will be done differently as the performance requirements differ from user data.

DESIGN OUTLINE

Our application uses a Client-Server model that takes advantage of API and machine learning/processing. The server will access multiple database tables. One will be for user data that keeps personal information as secure as possible. The second, will be used for storing trained stock market data that will need to be accessed frequently, so we will prioritize speed. We will use a powerful machine learning algorithm to train data sets, and use an API to gather low-latency stock market data. All information will be displayed in a responsive html page or mobile OS client.

1. Web Client

a. As a user, the application will be available via a website, using standard HTML/CSS.

b. The user will send/receive data to our server using JSON structures. The page will update after a user request and also every minute regardless of user input.

c. The UI will be done with Bootstrap so that the display is responsive to all screen sizes and resolutions.

2. Android Client

a. This version of the application will be hosted on the Android client. It will be independant from our web application but will access the same server and databases.

b. The client will receive JSON structures from the server and present the data in the form of an Android application.

c. The UI will be tailored for mobile and structurally different from the web application.

3. iOS Client

a. This version of the application will be hosted on the iOS client. It will be independant from our web application but will access the same server and databases.

b. The client will receive JSON structures from the server and present the data in the form of an iOS application.

c. The UI will be tailored for mobile and structurally different from the web application.

4. API Server

a. The API server will provide an easy way for the client interface to request and send information to/from the database and ML processes.

b. The API server will utilize a RESTfull web service structure in order to be easily consumed.

c. The API server will efficiently query the database to retrieve the requested information with little overhead.

d. The API server will return a response conforming to a contract with the client as an HTTP response.

5. Database

a. We will use multiple database tables for our application.

b. One will handle user data and will be prioritized for security. This way a user's email, password, and other sensitive information is safe.

c. The second will store trained stock market data and will prioritize speed so that the app has as low latency as possible.

6. Machine Learning Module

a. We will use multiple machine learning algorithms on different data sets to provide different predictions

b. Using adjusted close to predict adjusted open stock price

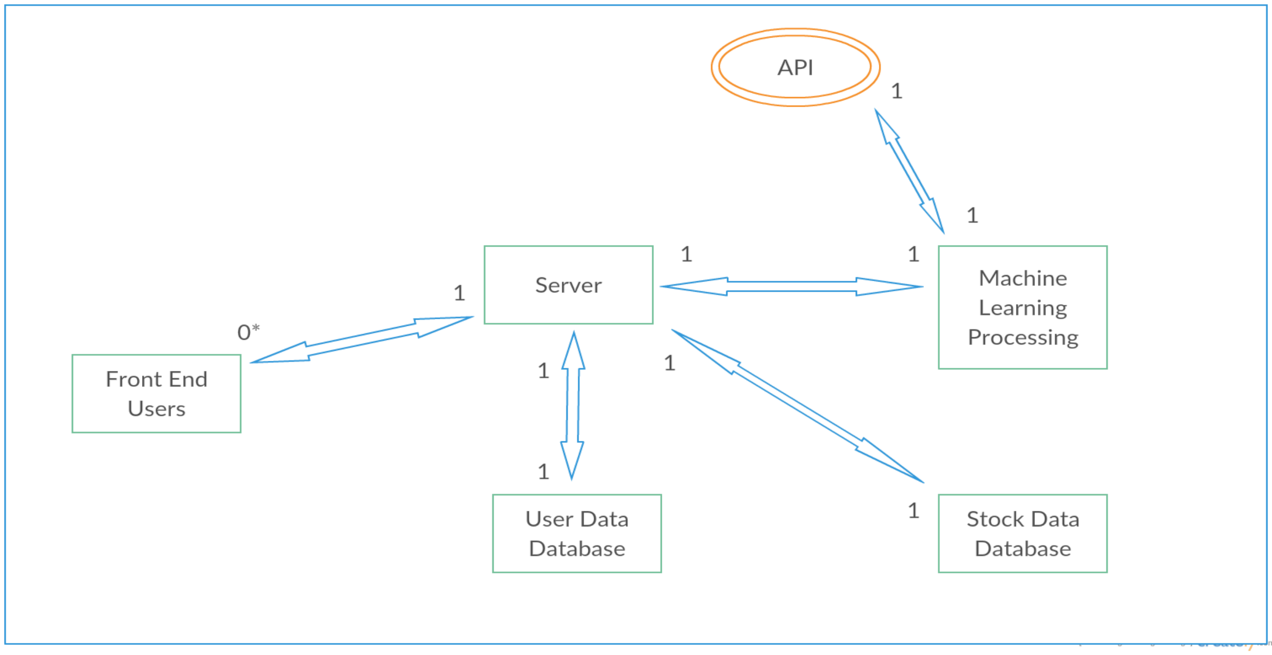
c. Using time series to forecast stocks

d. Using data from twitter and analyzing to provide qualitative predictions and alerts

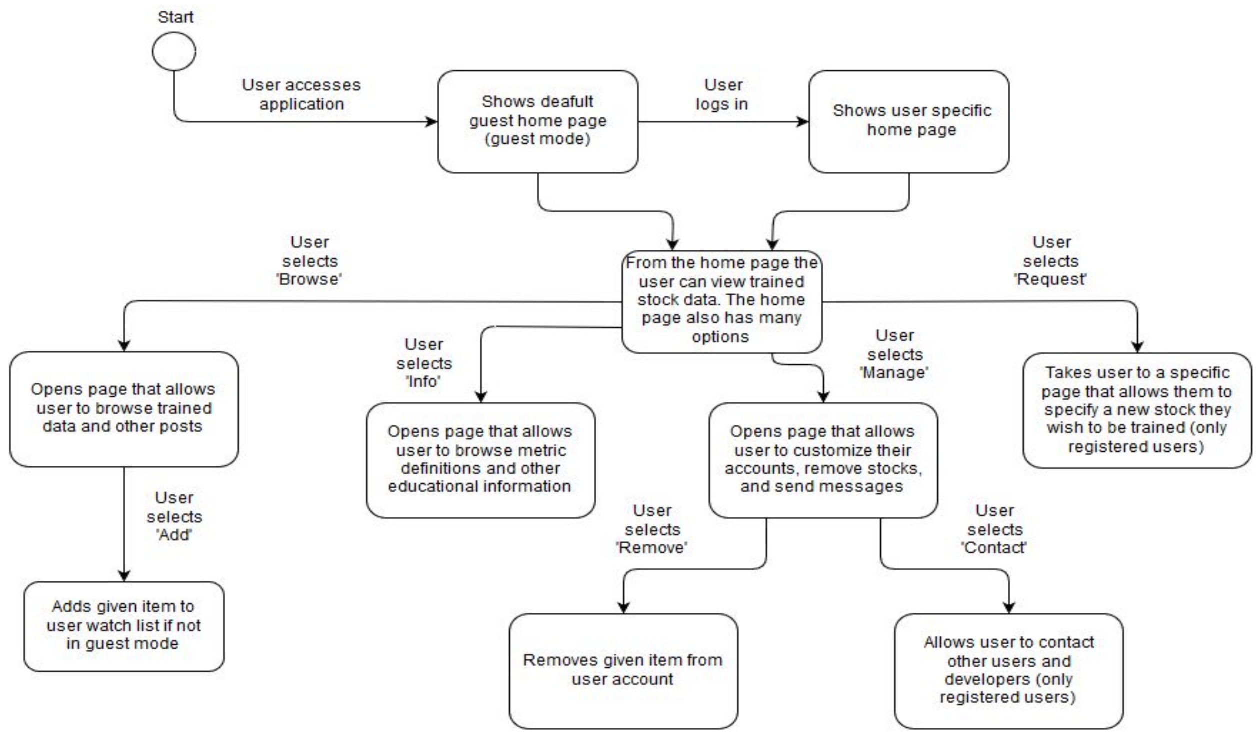
e. Will be written in Python

Overview of System

The application will be a Web Client to Python REST API Server architecture. This will allow many concurrent users to access the application as well as an easy interface with our Python machine learning algorithm. The server will be the central linking structure in the system. It will have a one-to-many relationship with clients while maintaining a one-to-one relationship with other systems, including the mySQL database tables, machine learning processing, and the stock information API. Clients will request information that will be sent to the server in a JSON structure. Then, the Server can access the databases, retrieve data, and request training based on that request from a client. After the server has completed a request, it sends information back to the client using a JSON structure.



State Diagram


Start
User accesses application
Shows deafult guest home page (guest mode)
User logs in
Shows user specific home page
User selects 'Browse'
From the home page the user can view trained stock data. The home page also has many options
User selects 'Request'
User selects 'Info'
User selects 'Manage'
Opens page that allows user to browse trained data and other posts
Opens page that allows user to browse metric definitions and other educational information
Opens page that allows user to customize their accounts, remove stocks, and send messages
Takes user to a specific page that allows them to specify a new stock they wish to be trained (only registered users)
User selects 'Add'
User selects 'Remove'
User selects 'Contact'
Adds given item to user watch list if not in guest mode
Removes given item from user account
Allows user to contact other users and developers (only registered users)

DESIGN ISSUES

Functional Issues:

Issue: Method of user login/registration

- Option 1: Social Media (facebook, G+)
- Option 2: User made username/password combination
- **Option 3: Combination of both**

The ability to log in and sign up with social media is essential, it makes signing up painless for the user. But some users do not have social media profiles or prefer not to link it to an unknown application, therefore we also need a plain sign up using a username/password combination.

Issue: How do users request new stocks to be trained

- **Option 1: Choose from a list of possible options**
- Option 2: A request box in the app to put in any name
- Option 3: Email us

Choosing from a list of possible options sounds the best because at times users would want some random stock to be trained for which we or our any of our online databases might not have any past stock metric data. If no data for a stock is available we won't be able to train our algorithm and wouldn't be able to produce any predictions. Emailing us would again come with the same disadvantage, we can't accept any random stock names. Therefore, a list of stocks to pick from for which we or our databases have enough data to generate predictions.

Issue: How does the user access the application on mobile?

- Option 1: Applications hosted on the platform (iOS, Android)
- Option 2: Responsive web application that can be easily viewed on mobile
- **Option 3: Do both Option 1 and Option 2.**

A big part of the application is having it work for anyone, regardless of platform and technical knowledge. Many potential users have access to only one platform to access applications and we want to have anyone be able to use the application. By having the application run on all devices, we can expand our potential users and reach a larger group of people.

Non-Functional Issues:

Issue: What type of machine learning backend should we use?

- **Option 1: Python**
- Option 2: R
- Option 3: C

Python has better libraries for machine learning than C or R such as Sklearn, which provides off the shelf algorithms such as SVM, KNN. Doing machine learning in a low level language such as C will be very time consuming and distract us from higher level thinking required for machine learning, C also does not have many good machine learning libraries. R is a good option too and has a lot of statistical libraries but our team has more experience in Python. Python is easier for us to use since our server will be in Python too. The prediction part of our machine learning will be most used, and prediction takes very less time so using C to optimize this is not needed.

Issue: What database software should we use ?

- Option 1: Oracle
- Option 2: MongoDB
- **Option 3: MySQL**

After thoroughly researching each database software, we came to the conclusion that MySQL is the best-fit for our purposes because it is efficient, inexpensive, secure, reliable, and manages memory very well. Unlike Oracle, MySQL was designed to have fewer features which allows it to run very fast and efficiently. Since we do not need all of the extraneous features included in Oracle, MySQL will be more than enough. Also, Oracle is not a free software as opposed to MySQL. Finally, MongoDB and other noSQL database software are not the best option for our project because at Purdue we have been learning MySQL at Purdue and our team has more experience with MySQL.

Issue: What framework should we use?

- **Option 1: Flask**
- Option 2: Pyramid
- Option 3: Django

Flask being recent has a couple of advantage over the other two options, Django and Pyramid. First of all, it has been developed without all the extra features which aren't used for small projects like ours. Flask being lite makes it faster compared to Django and Pyramid. We also have past experience with Flask which makes it a better choice. Flask is comparatively easy to implement and the api generation is a lot better compared to the others.

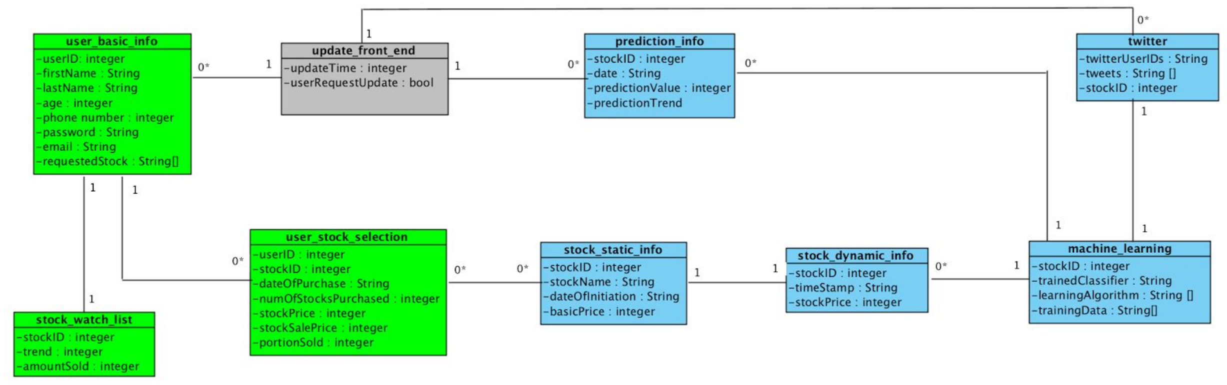
Issue: How to provide users with the real-time stock information?

- **Option 1: Xignite API**
- Option 2: Getting data directly from stock exchange departments like NYSE
- Option 3: Bloomberg Terminal

Contacting the NYSE directly for every stocks real time data that we have in our app would be very expensive. Also having the data available in an API makes it easier to integrate into our system. Xignite is a free service with an easy to use API. Bloomberg Terminal is good too but it is paid which makes Xignite a better option. By lowering costs of obtaining data we can do more data training and provide better predictions on the application.

DESIGN DETAILS

Class Level Diagram



Class Level Description

1. User Basic Info:

- Stores basic information about users
- Contains user id, user firstname, user lastname, phone number, age, password, email, and list of requested stocks

2. Stock Static Info:

- Stores basic information about stocks that will not change over time (static).
- Contains stock id, company name, number of stocks, date of initiation, and basic price.

3. Stock Dynamic Info:

- Stores stock prices progressively over time
- Time series data about various stock prices over time. Contains stock id, timestamp, and stock price

4. User Stock selection Profiles:

- Stores past and current user stock selections over time
- Contains user id, stock id, data of purchase, number of stocks purchases, price of sale, and portion sold

5. Prediction info:

- Contains predictions about future trends of stocks and their prices, e.g., stock id, date, prediction value, prediction trend, etc.

6. Stock Watch list:

- The stocks that the current user is watching/tracking and has not bought
- Contains e.g., stock id, trend, amount sold, etc.

7. Twitter:

- Stores real-time news data for stocks from various credible twitter accounts
- Contains stock id, twitter user ids, tweets

8. Machine learning:

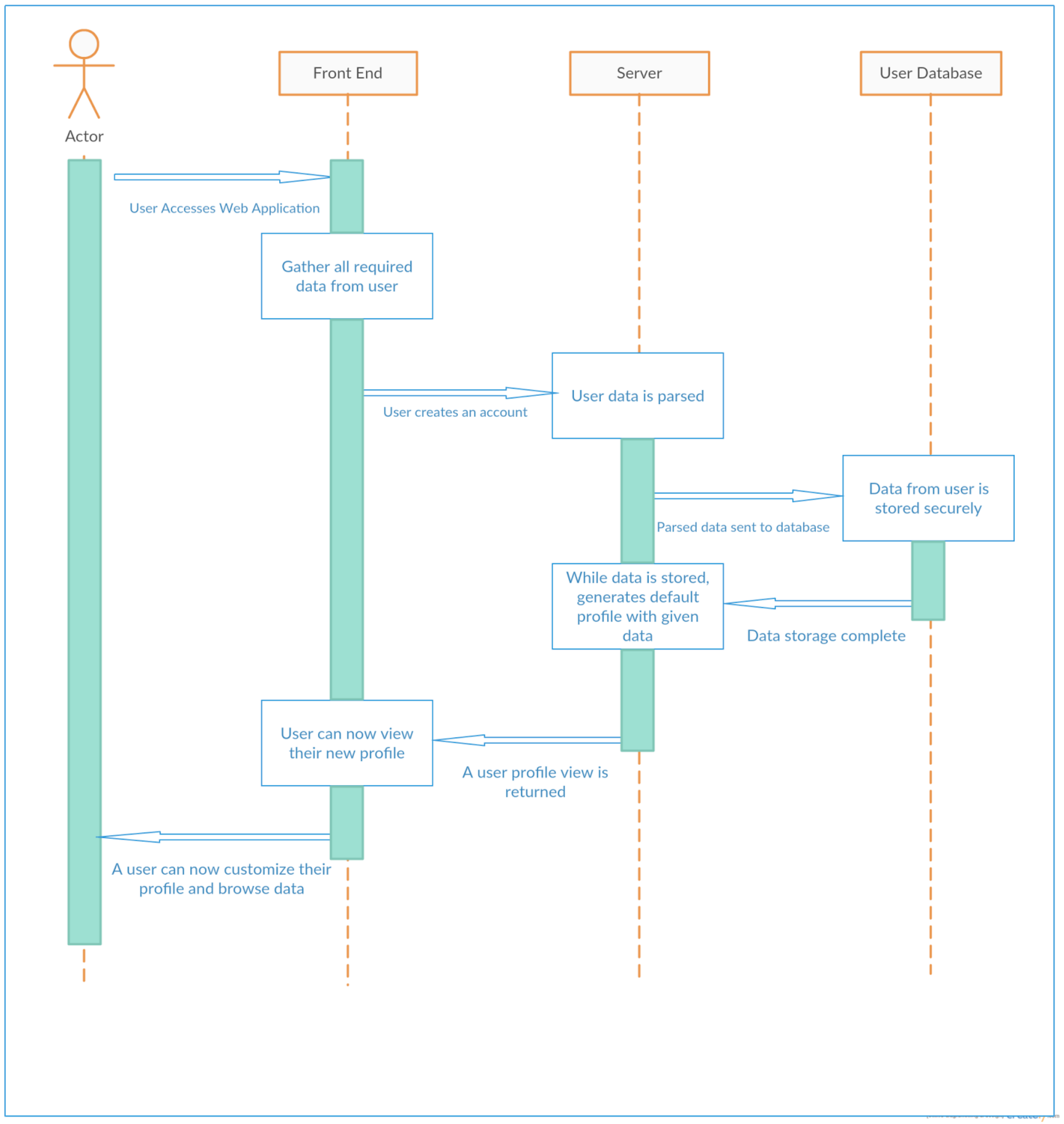
- Acquires data from twitter class or external api and trains it.
- Contains the different machine learning algorithms and training data.
- Makes prediction given stock id
- Contains training data, trained classifier, learning algorithms

9. Update Front-end:

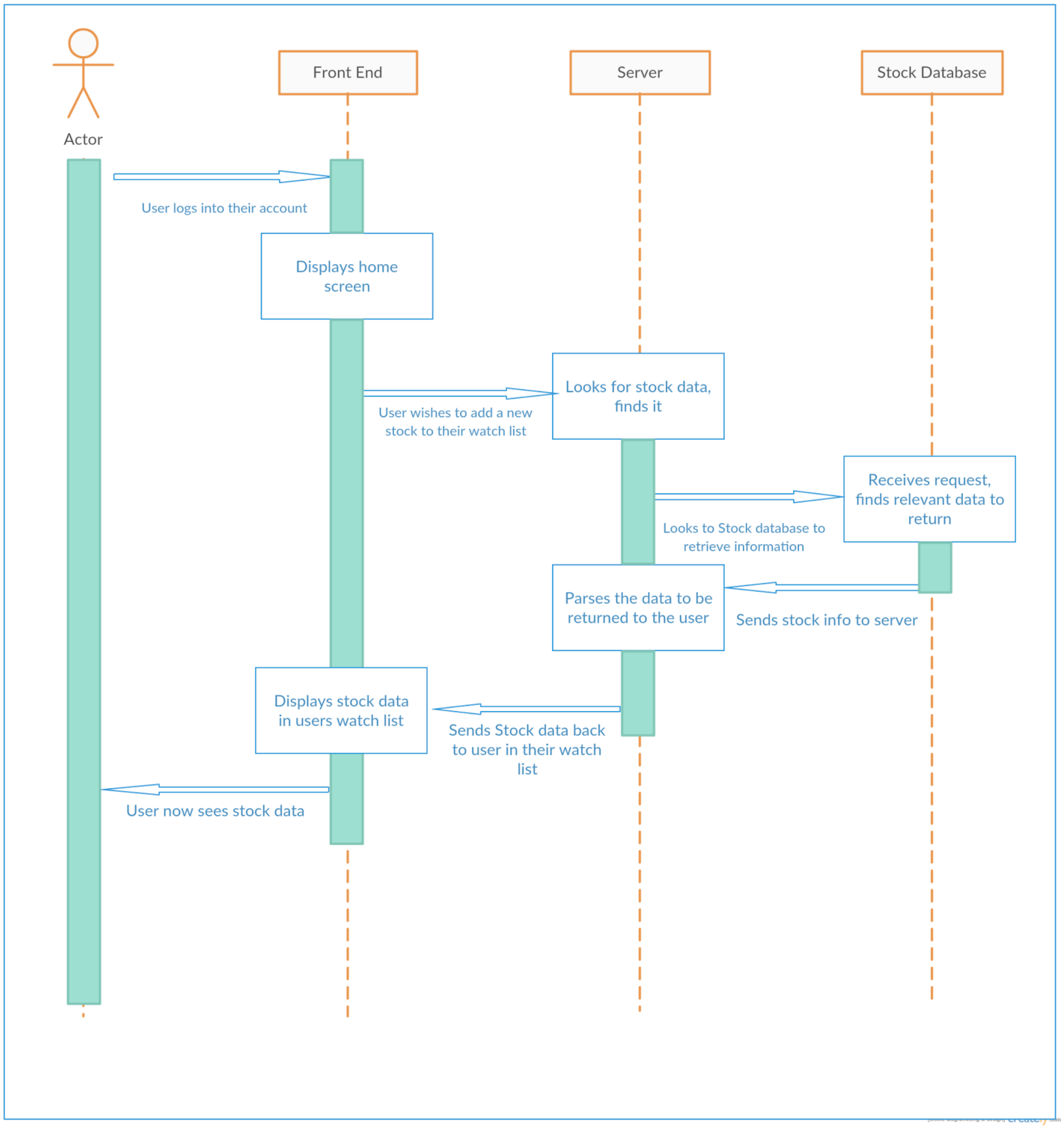
- Used to continuously update the front-end of the applications
- Contains update time and a user request flag

Sequence Diagrams

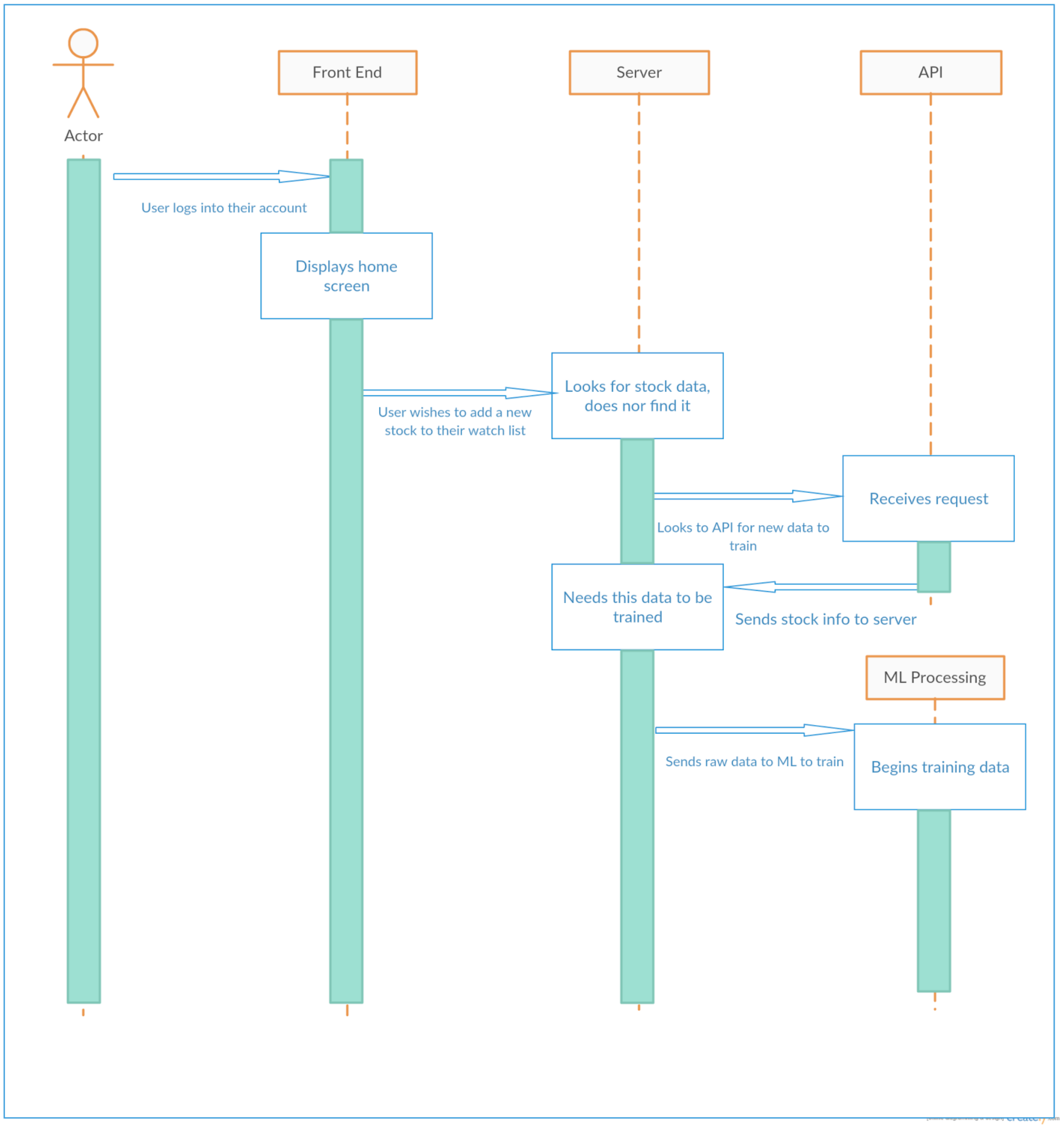
How a user creates an account:


Actor
Front End
Server
User Database
User Accesses Web Application
Gather all required data from user
User creates an account
User data is parsed
Parsed data sent to database
Data from user is stored securely
Data storage complete
While data is stored, generates default profile with given data
A user profile view is returned
User can now view their new profile
A user can now customize their profile and browse data

How a user looks at new stock data:

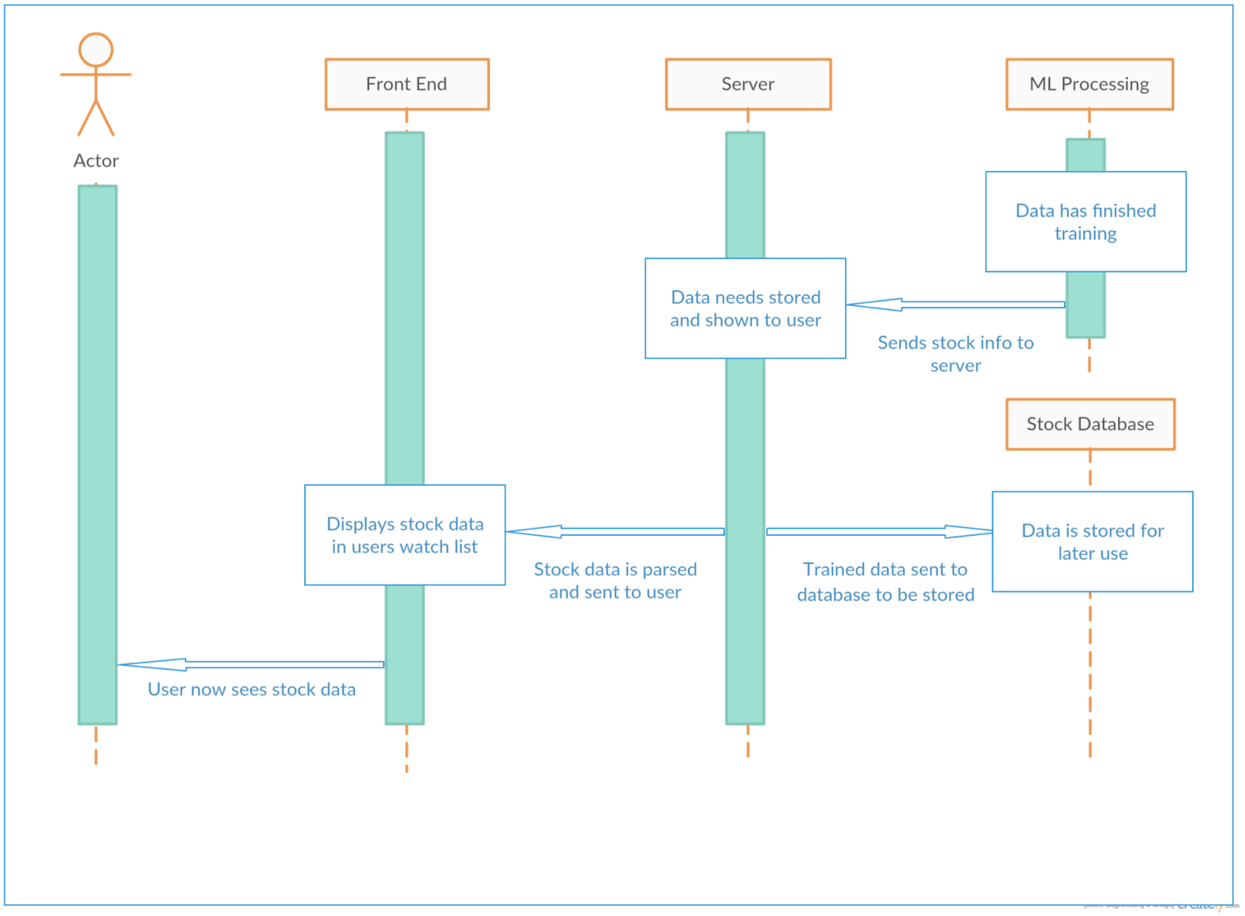

Actor
Front End
Server
Stock Database
User logs into their account
Displays home screen
User wishes to add a new stock to their watch list
Looks for stock data, finds it
Looks to Stock database to retrieve information
Receives request, finds relevant data to return
Sends stock info to server
Parses the data to be returned to the user
Sends Stock data back to user in their watch list
Displays stock data in users watch list
User now sees stock data

How a user requests a new dataset be trained:


Actor
Front End
Server
API
User logs into their account
Displays home screen
User wishes to add a new stock to their watch list
Looks for stock data, does nor find it
Looks to API for new data to train
Receives request
Needs this data to be trained
Sends stock info to server
ML Processing
Sends raw data to ML to train
Begins training data

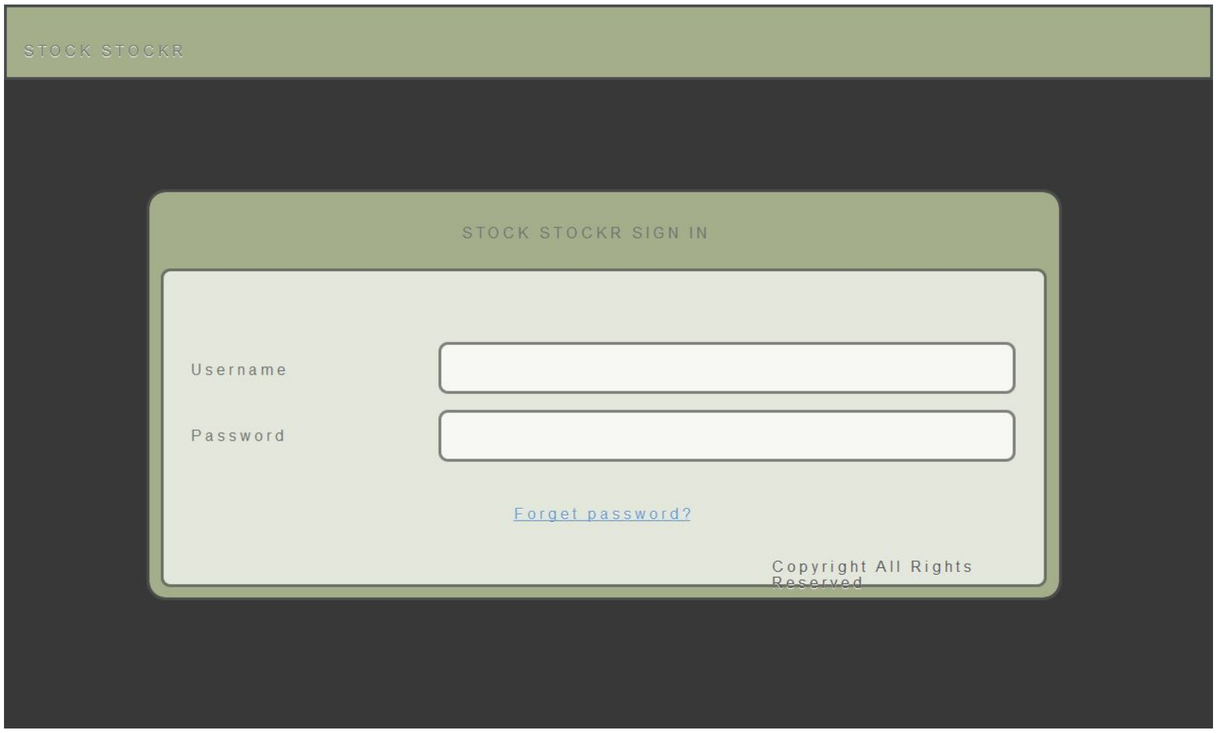
Continued...

Continued from above:


Actor
Front End
Server
ML Processing
Data has finished training
Data needs stored and shown to user
Sends stock info to server
Stock Database
Displays stock data in users watch list
Stock data is parsed and sent to user
Trained data sent to database to be stored
Data is stored for later use
User now sees stock data

UI Mockups

Web Application Login



STOCK STOCKR

STOCK STOCKR SIGN IN

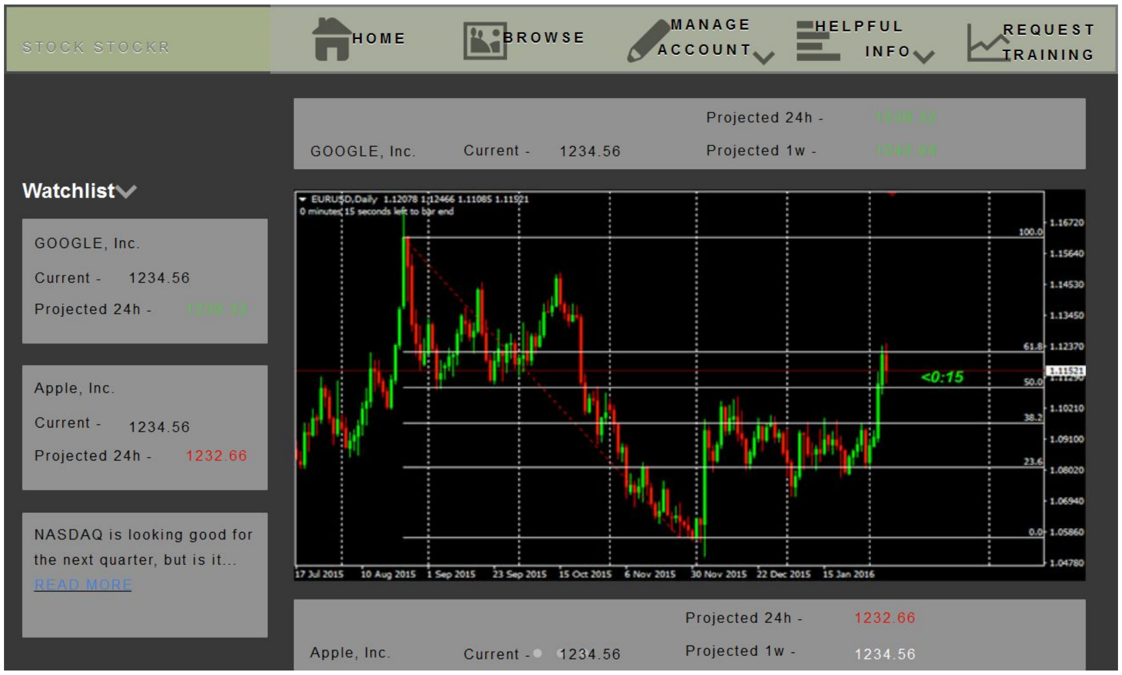
Username

Password

Forget password?

Copyright All Rights Reserved

Web Application User Home Screen



STOCK STOCKR

HOME BROWSE MANAGE ACCOUNT HELPFUL INFO REQUEST TRAINING

GOOGLE, Inc. Current - 1234.56 Projected 24h - 1239.32 Projected 1w - 1242.00

Watchlist

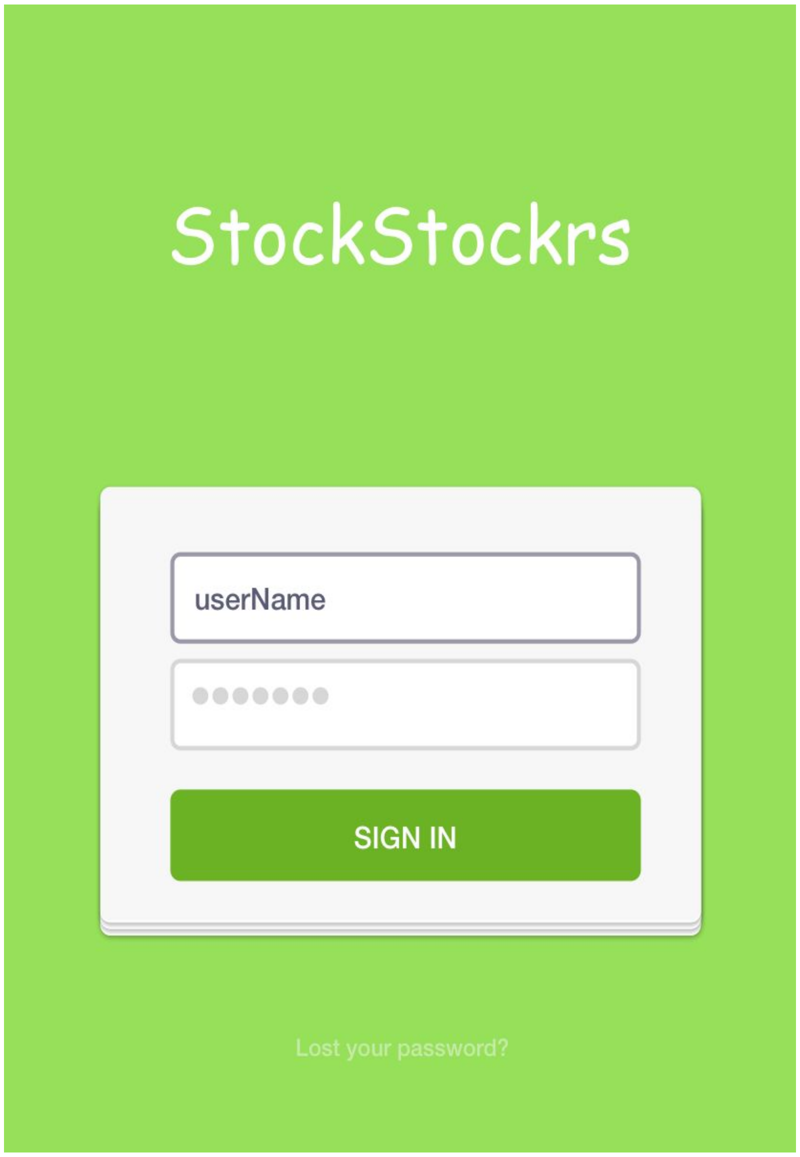
GOOGLE, Inc.
Current - 1234.56
Projected 24h - 1239.32

Apple, Inc.
Current - 1234.56
Projected 24h - 1232.66

NASDAQ is looking good for the next quarter, but is it...
READ MORE

Apple, Inc. Current - 1234.56 Projected 24h - 1232.66 Projected 1w - 1234.56

Mobile Application Login


StockStockrs
userName
●●●●●●●
SIGN IN
Lost your password?