myCareer TEC
Technology Education Center

Agile Development Using Java 8 Features: Unit Testing, Design Patterns, and RESTful Services

Case Study

ROITraining
Maximize Your Training Investment

Fidelity
INVESTMENTS

This page intentionally left blank.

TABLE OF CONTENTS

This page intentionally left blank.

OVERVIEW

This case study focuses on building a robust enterprise system utilizing REST-based web services, web security principles, and java streams. Test-driven development will be used in all software development. This will include both unit and integration testing. Security measures, such as basic authentication, will be incorporated into the application.

CASE STUDY PRESENTATION

On day five of this program, each case study group will make a presentation of their project. This 15-minute presentation should include a discussion of the technical aspects of the solution that the team designed and developed over the course of the training.

Each team member is responsible to showcase a work product and describe the role played in developing the solution. In particular, the design of the solution should be presented, as well as details on how that design was implemented. Any problems, challenges, difficulties, or "features" should be acknowledged and discussed. For those issues that were resolved, a discussion of the solution should be included. For any issues that are still unresolved, a discussion of what was attempted, and how the issue could be approached should be provided.

The presentation will be made in the classroom. Those in attendance will include the instructor, the other teams in the class, and additional invited guests.

After a team has made their presentation, there will be a period for any questions that the audience may have for the team, or for specific individuals, about the end deliverable or the role played to develop the deliverable. Expect to receive several interesting questions from the audience. In the unlikely event that no questions are asked, the instructor may assume the role of *Interrogator* and pose a few questions of his/her own.

THE INSTRUCTOR'S ROLE

During the case study sessions, your instructor will play a variety of roles depending upon what the specific deliverable or activity calls for. During the case study periods, the instructor will wear many different hats.

One primary role is simply an extension of the traditional instructor role—that of being a **Technical Resource/Coach/Mentor**. When a team encounters problems in determining how to approach a particular problem, the instructor may facilitate discussions to reach a solution, or may play the role of **Product Owner** to provide input to requirements and expected deliverables.

The instructor will regularly check in with each group to discuss how the group is progressing and proactively deal with any issues they are facing.

STOCK TRADING AND PORTFOLIO MANAGEMENT PLATFORM

Introduction

A stock exchange is an exchange where stock brokers and traders can buy and/or sell stocks (also called shares), bonds, and other securities. Securities traded on a stock exchange include stock issued by listed companies, unit trusts, derivatives, pooled investment products, and bonds. Stock exchanges often function as "continuous auction" markets, with buyers and sellers consummating transactions at a central location. To be able to trade a security on a certain stock exchange, it must be listed there. Usually, there is a central location at least for record keeping, but trade is increasingly less linked to such a physical place, as modern markets use electronic networks, which gives them advantages of increased speed and reduced cost of transactions. Trade on an exchange is restricted to brokers who are members of the exchange. The 'crown jewel' of any stock exchange platform is its matching engine—high throughput algorithms which match buy and sell orders with each other. The price at which equity has 'changed hands' most recently is called 'market price' and it is being sent out back to the brokers as part of continuous feed of 'completed trades'.

Many broker companies these days provide trading platforms for retail investors to place orders with stock exchange directly or via broker, analyze performance of the stocks in their accounts over time, set up various trading strategies to be executed on stock exchange, as well as test trading strategies by running trade simulations against historical or mock data. When trading strategy is defined, trader can set up *algorithmic trades* based on calculating various *technical indicators*. Once trades are set up, they would be triggered automatically when certain conditions are met (e.g., stock price or technical indicator value reaches some threshold).

The Project

Your task will be to build parts of functionality of the trading platform for ACME Broker Company.

For Application Administrator:

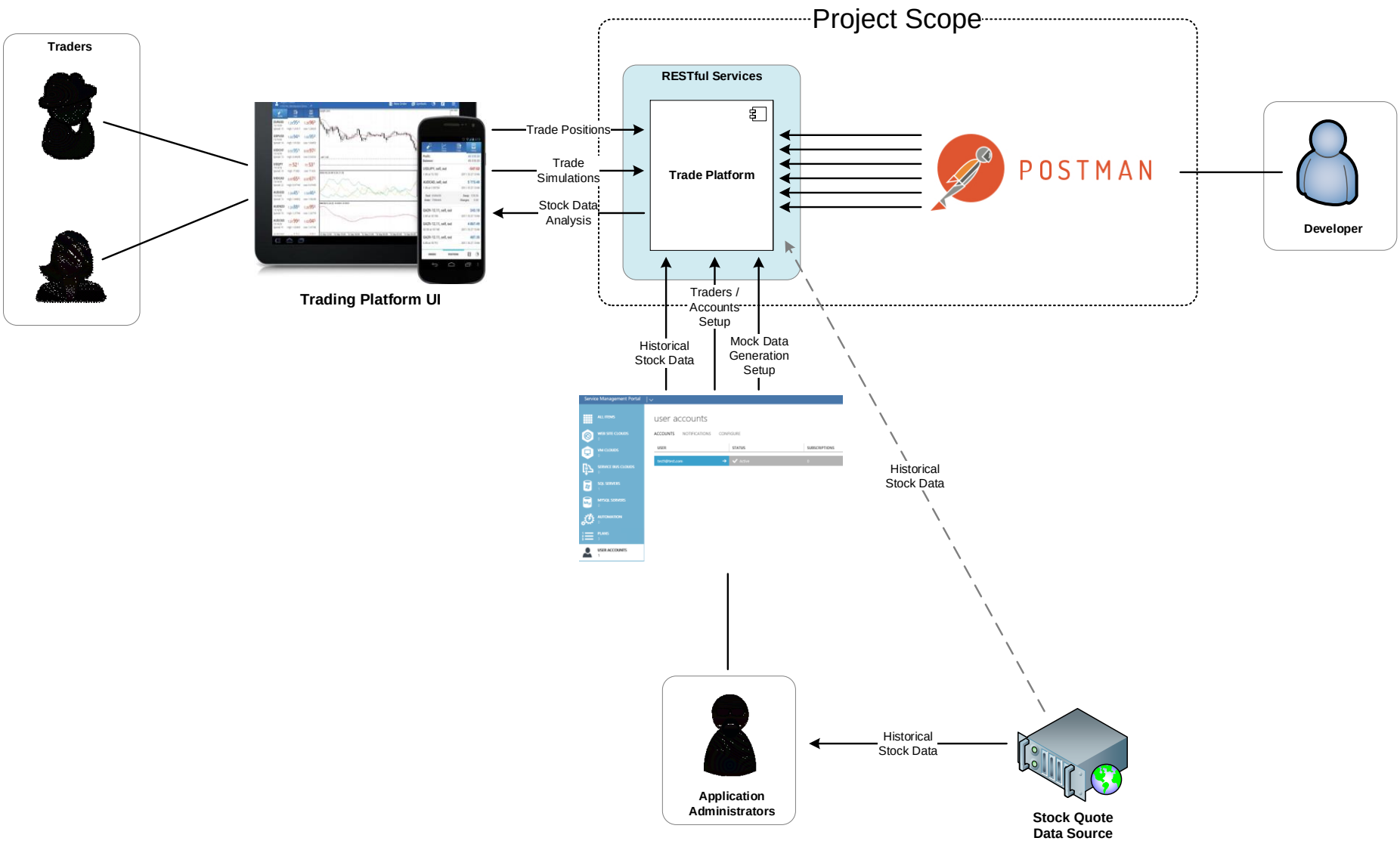
- Manage traders and traders' accounts
- Maintain historical stock quotes data
- Generate mock stock data used for testing trading algorithms

For Trader:

- Analyze historical and mock stock data
- Manage stock positions
- Add historical transactions to the ledger
- Analyze portfolio performance over time
- And, optionally, validate trading strategies by simulating trades against mock and historical data

All functionality has to be implemented as RESTful webservice and tested via Postman client.

Below is a high-level context diagram of key stakeholders and systems involved in the implementation scope of the Stock Trading Platform.



The system will be built incrementally, and as you are working in teams, you should aim to integrate early and often. The use of tests to verify your own code and the overall system's functionality are vital and mandatory. The requirements will be presented as a series of user stories that lead to the incremental development of the system. Acceptance criteria focused on automated tests will be the norm.

Important Note on Expectations

Completing all user stories is not a requirement. What is a requirement is that each story is delivered completed and with good quality code supported by relevant tests. It is better to have one solid story completed than say, three partially or not fully tested.

Implementation Requirements

The following are requirements for the system:

- All code should be written using Java 8
- The platform should be plain Java and capable of executing in any standard Java runtime
- The REST service must be a Spring Boot application using Spring REST
- REST services should use JSON data format
- Unit tests are mandatory
- Keep it simple—don't try to optimize code prematurely; in many cases, scanning and searching through 'list' structure is the easiest way to implement things
- Try to implement stories which would support the epic, but ultimately it's the team's decision whether to implement separate/discrete User Stories which might not contribute to a single epic. In any case, epic can be used for context setting and for better understanding of the end-to-end flow.

Dictionary of Terms

Term	Description
Position	A position is the amount of a security, commodity, or currency that is owned (a long position) or borrowed and then sold (a short position) by an individual, institution, or dealer. A position can be profitable or unprofitable, depending on market movements. Trader manages positions in one of his accounts. Positions can be: • 'Opened': trader buys security with the cash available in the account • 'Closed': trader sells full or part of this position Read more: Position https://www.investopedia.com/terms/p/position.asp For the purpose of the case study: • Short positions are NOT allowed • Only stocks can be traded • No commission is being collected by the broker

Account (Trading Account)	A trading account is similar to a traditional bank account, holding cash and securities, and is administered by an investment dealer. The account is held at a financial institution and administered by an investment dealer that the account holder uses to employ a trading strategy rather than a buy-and-hold investment strategy. Read more: Trading Account https://www.investopedia.com/terms/t/tradingaccount.asp Account can be in 'native' or 'foreign' currency. For example, for U.S. financial institution, native currency will be U.S. dollars. Trader can open account in U.S. dollars or any other currency, such as CAD dollar or British Pounds. Each account is financed separately with cash deposits done in the currency of the account. It is common to provide trader with a summary view of ALL of his accounts represented in the 'native' currency—currency conversions are done at the 'currently posted rate'.
Trader	A trader is an individual who engages in the buying and selling of financial assets in any financial market, either for himself or on behalf of another person or institution. Read more: Trader https://www.investopedia.com/terms/t/trader.asp Trader can open account with a broker company and buy/sell securities within that account with the cash he deposits to the account. Trader can open more than one account in native or foreign currency. Trader can open more than one account in the same currency.
Application Administrator	A team responsible for: • Setting up trader information in the system • Opening or closing trader's accounts • Various other maintenance tasks (such as uploading historical stock quotes, generating mock data, etc.) Application Administrator can help Service Desk team by looking up trader information in the system or any of the trader's accounts.
Ledger / Transaction history	Each operation on the account is recorded in the 'ledger' or 'transaction history'. This includes cash deposits and withdrawals, as well as stock purchases and sales.

This page intentionally left blank.

EPIC DESCRIPTIONS

EPIC 1: STOCK QUOTE FILES SETUP

Application Administrator of ACME Broker Company uploads initial historical stock quotes for last year to the trader application for the most popular stocks (AAPL, GOOG, MSFT, GS, QCOM, GE, etc.). He then validates the data was uploaded correctly and can be accessed by the traders by downloading some of the uploaded files, as well as querying subsets of the data.

EPIC 2: ACCOUNT SETUP

Trader John Smith currently has an account with another broker. He wants to move his accounts to ACME Broker Company. Trader calls in and asks Application Administrator to set up two accounts in USD currency. Application Administrator creates accounts and provides John with URL to access his accounts with. Trader John Smith deposits 100K dollars into each account, receiving transaction confirmation for each operation, with each operation recorded in the ledger.

EPIC 3: OPENING AND CLOSING POSITIONS

Trader John Smith locates his account #1 and opens a position in GOOG (e.g., 100 shares). He then opens a position in AAPL (50 shares) and sells 30 shares of GOOG at a profit. He checks the summary of his account to see the cash balance and validate that transactions have been properly captured.

EPIC 4: UPDATING HISTORICAL TRANSACTIONS IN THE ACCOUNT

Trader John Smith wants to recreate historical transactions in his accounts based on the records he has from his previous brokerage company:

- Opens and closes positions at some date in the past
- Deposits/withdraws cash at some point in the past

The system adds all historical transactions into the ledger disallowing transactions which might result in negative cash balance of the account, either at the timestamp of transaction or at any future point in time. For example, cash balance of the account as of Jan 1st , 2015, was 1K. There is already a transaction in the ledger with a timestamp of Feb 1st 2015 (buy 1 share of GOOG for $900), with resulting cash balance after the purchase being $100. If trader wants to enter transaction of 'buy 5 shares of AAPL at $100 dollars each' on Jan 15th, this transaction should be disallowed, since it would make the transaction on Feb 1st invalid (if the transaction on Jan 15th is entered, it would make the cash balance of the account equal to $500 as of Jan 31st, making the purchase of GOOGLE share tip the account balance into negative territory).

EPIC 5: REVIEW PORTFOLIO PERFORMANCE

Trader John Smith wants to check the performance of his portfolio. He runs a summary report on individual accounts and across all accounts. He also checks historical performance of his portfolio.

NB: for the historical performance of the portfolio, it is allowed to create 'mock transactions' manually in the ledger (e.g., 'prefill' one of the accounts with previously opened/closed positions), if updates to historical data service is not available.

EPIC 6: SMA GENERATION

Trader John Smith wants to start working out more sophisticated trading strategies. For that, he wants to explore various strategies involving SMA as technical indicator. He generates a SMA for several stocks over various time frames.

EPIC 7: TRADE SIMULATOR

As a trader, I want to set up and run trade simulations against 'fake' and 'historical' data by setting up algorithmic trades (based on SMA crosses, resistance and support levels, MFIs, MACDs, etc.) to work out the best trading strategy to maximize my profit.

This is a rather large epic which mostly has not been refined into user stories. Feel free to explore and refine as time allows. Consult product owner (your instructor) for ideas or clarifications.

USER STORY DESCRIPTIONS

STORY 1: STOCK QUOTES GENERATOR (MOCK DATA GENERATOR)

Requirement

As an application administrator, I want to be able to generate daily stock quotes based on predicable functions and save those quotes into a file on a server. Multiple files must be able to co-exist. This data will help me in the future to perform backtesting and fine-tune trader's automated trading algorithms, as well as calculate the total value of the account.

Date	Open	High	Low	Close	Volume
2018-01-24	160.36	160.63	158.40	158.97	20340900
2018-01-25	159.64	159.92	158.21	159.03	20290900
2018-01-26	159.53	159.69	158.77	159.60	16120300
2018-01-29	159.06	159.86	158.62	158.70	18293300
2018-01-30	157.66	158.22	156.73	157.18	24533800
2018-01-31	157.74	158.42	155.92	156.36	27468900

Stock quote generator must be able to generate data based on the following parameters:

- Starting date
- Number of days
- Stock symbol
- Type of function to be used for data generation
    - Can be simply some id

Filename should contain the symbol name the data was generated for.

Acceptance Criteria

At least three different generators must be available. Data format is left up to the implementation team.

More Details

Feel free to create stock quotes based on harmonic or linear functions. Functions can be as simple or as complicated as you want them to be, as long as output is 'predictable' and not completely random. For example, simple 'sin(x)' or something more complicated, such as 'sin(x) + 2*sin(x/2)+cos(x)'. Feel free to add random deviations at random spots, etc.

NB: think about URI structure when creating the endpoint—what should be passed as parameters and what should be part of the URI. Review other stories to determine the best naming convention for the file—you might want to include information about the algorithm/stock generator ID in the name of the file to simplify design.

Additional Resources

- http://fooplot.com
    - An online charting app that allows you to visualize various graphs

STORY 2: PROVIDING METADATA ABOUT AVAILABLE STOCK GENERATORS

Requirement

As a trader, I want to be able to get metadata about available stock generators. The data should provide the ID of the generator to be used in the 'stock quote generator' and description (what kind of function is being used, for example).

Acceptance Criteria

N/A

More Details

Implementation details are left for the team to decide on (e.g., whether to implement it as a single endpoint for multiple endpoints, etc.)

Bonus

As part of response, provide a link to the specific stock generator which can be used to retrieve metadata about the generator, generate mock data, etc.

STORY 3: LIST AVAILABLE STOCK QUOTE FILES

Requirement

As a trader, I want to be able to list all or a subset of the stock files based on the following search criteria (any combination of thereof):

- Type of file/quotes (whether quotes are historical data or mock data created by 'stock quotes generator')
- Stock symbol
- Type of generator function used (for files with mock data)

Information provided in the response should be enough to retrieve additional metadata about the specific file.

Acceptance Criteria

At least five files should be available.

More Details

Implementation details are left for the team to decide on (e.g., how much additional information to provide for each of the files, etc.).

Bonus

As part of the response, for each of the files provide a link to download the specific file and/or a link to get metadata for the quote file.

STORY 4: PROVIDE SUMMARY INFORMATION ABOUT RETURNED QUOTE FILES

Requirement

As a trader, as part of the stock quote files listing (see story above), I want to get a summary of the available stock quote files:

- Total number of files available
- Total number of files returned in the response
- Breakdown of the files returned in the response by:
    - Stock
    - Type of data (mock vs. historical data)

Acceptance Criteria

N/A

STORY 5: DELETE STOCK QUOTE FILES

Requirement

As an application administrator, I want to be able delete stock quote files I no longer need.

Acceptance Criteria

Indicated file deleted and can no longer be retrieved.

More Details

404 error should be returned if indicated file can't be located.

STORY 6: STOCK QUOTE DATA FILE DOWNLOAD

Requirement

As a trader, I want to be able to download a list of stock quotes from the files stored on the server in CSV format. I want to be able to retrieve a full list or a subset of the data based on the selected time frame. Furthermore, I want to be able to define which fields from the file are to be returned. It should not be allowed to filter out timestamp or stock price fields from the file. First row of the file is a 'header' and should list the field names.

Acceptance Criteria

Data returned in CSV file. If time frame is out of range, empty file is to be returned (with only header row).

STORY 7: STOCK QUOTE FILE METADATA

Requirement

As a trader, I want to be able to retrieve metadata about individual stock quotes file:

- Stock symbol
- Type of file (autogenerated or real historical data)
    - If mock data, what algorithm was used to generate it
- Start and end date of the data contained within the file

- File ID
- Total number of days contained within the file

More Details

404 error should be returned if indicated file can't be located.

STORY 8: STOCK QUOTE DATA BASIC STATISTICS

Requirement

As a trader, I want to be able to get statistics for the stock quotes data in the files stored on the server. For the stock quote file, I want to be able to retrieve the statistics for a full list or a subset of the data based on the selected time frame.

I want to retrieve the following stats (for the selected time frame):

- Max and min price
- Max and min volume
- Average volume
- Total number of days in the data set
- Number of days the stock closed higher than opened ('gain' days)
- Number of days the stock closed lower than opened ('loss' days)
- Total trade volume on 'gain' and 'loss' days (two separate numbers)

I expect the service will also be returning basic information about the origin of the data (e.g., quote data file ID).

STORY 9: STOCK QUOTE DATA ADVANCED STATISTICS – SIMPLE MOVING AVERAGE (SMA)

Requirement

As a trader, I want a REST service that calculates a simple moving average indicator for a full set or subset of the data stored in the stock data file based on the selected time frame.

SMA can be configured with the following parameters:

- Time period: number of data points used to calculate each moving average value (e.g., 20 or 50 days)
- Series type: the desired price type in the time series to be used in calculation ('open', 'close', 'high', 'low')

I want to be able to retrieve the SMA either in JSON or CSV format.

More Details

For more information on SMA, see http://www.investopedia.com/articles/technical/052201.asp and http://www.fmlabs.com/reference/default.htm?url=SimpleMA.htm.

STORY 10: SET UP HISTORICAL STOCK QUOTE DATA ON THE SERVER

Requirement

As a trader, I want historical stock quote data to be available for me to run stock trade simulations. Data can be downloaded from various available sources and uploaded to the server ***manually***. The uploaded data files must be available for retrieval, listing, generation of metadata, etc. (see stories 3 through 9).

BONUS: provide ability for an application administrator to upload file to the server via RESTful endpoint.

Acceptance Criteria

All 'read' operations on the uploaded files (see stories 3 through 9) are working correctly.

More Details

Historical data be downloaded from the YAHOO FINANCE website (https://finance.yahoo.com/quote/AAPL/history/ – select time period and press 'Download data'). Columns other than Date, Open, High, Low, Close, and Volume can be ignored during retrieval or deleted before saving.

For BONUS implementation, see examples on the internet for implementation of file upload functionality via RESTful service (e.g.: http://codophile.com/2015/05/27/how-to-upload-binary-file-to-spring-rest-service/).

STORY 11: MAINTAIN TRADER INFORMATION

Requirement

As an application administrator, I want to be able to maintain trader information:

- Create new trader
- Delete existing trader
- List existing traders
- Get information about specific trader

Trader information should contain at the very least:

- First/last name
- Contact information (email, phone)
- Address
- Trader ID
- Trader accounts
    - Can be just a list of account numbers. It can also contain additional information the implementation team considers useful to be included (such as account balance, total value, etc.).

STORY 12: MAINTAIN TRADER ACCOUNTS

Requirement

As an application administrator, I want to be able to maintain traders' accounts:

- Create new account
- Delete existing account
- List existing accounts
- Get information about specific account

The account information should contain at the very least:

- Account currency
- Account number
- Trader the account belongs to
- Available cash balance
- Account open positions (aggregated by stock)
    - Can be just a reference to positions which additional information can be retrieved from OR can contain additional information the implementation team considers useful to be included about the position.

STORY 13: TRADER LOOKUP

Requirement

As an application administrator, I want to be able to search for trader by one of the following attributes:

- Last name
- Email address
- Phone number
- Trader ID

STORY 14: ACCOUNT LOOKUP

Requirement

As an application administrator, I want to be able to get information about a specific account by account number **only** (i.e., without necessarily knowing any information about the trader the account belongs to).

STORY 15: ADD TO/CREATE NEW POSITION

Requirement

As a trader, I want to be able to create a new position, add to, or close fully or partially any existing position.

To add to or create a new position, trader will provide:

- Stock ticker symbol
- Quantity of the position (number of shares)
- Date and time of the purchase
- Price of the purchase
- Account the position should be opened/added to
- Trader ID

If historical quote information for the stock is available, the price and quantity of the position will be validated (price of the purchase must be between 'low' and 'high' range on the date of the purchase, quantity of the position can't be negative or exceed the trade volume on the date of the purchase). There should be enough cash available in the account to make the purchase.

Position can't be opened if there is not enough cash as of the date of transaction.

STORY 16: CLOSE THE POSITION

Requirement

As a trader, I want to be able to close the existing position fully or partially. Trader will provide:

- Stock ticker symbol
- Quantity of the position to be sold (number of shares)
- Trader ID
- Date and time of the sale
- Price of the sale
- Account the trade commenced in

If historical quote information for the stock is available, the price and quantity of the position will be validated (price of the sale must be between 'low' and 'high' range on the date of the sale, quantity of the position can't be negative or exceed the trade volume on the date of the purchase), a corresponding open position must exist in the account (i.e., can't close the position that doesn't exist), and sufficient quantity must be available within the open position (i.e., can't sell more than what you own—short selling is not allowed).

Position that doesn't exist can't be closed.

STORY 17: RETRIEVING ACCOUNTS INFORMATION

Requirement

As a trader, I want to be able to list all my accounts.

The account information should contain at the very least:

- Account currency
- Account number

- Trader this account belongs to
- Available cash balance
- Account open positions (aggregated by stock)
    - Can be just a reference to positions which additional information can be retrieved from OR can contain additional information the implementation team considers useful to be included about the position.

STORY 18: GET DETAILED ACCOUNT INFORMATION

As a trader, I want to be able to see detailed information about my accounts:

- Listing of all open positions as of today with the following information:
    - Average cost of the position (weighted average price)
    - Current market value of the position
        - Calculated as: numOfShares * lastKnownPriceOfStock
    - Book value
        - avg cost * numOfShares
    - Gain/unrealized loss
        - (marketValue – bookValue) / bookValue
    - Percentage of portfolio
        - Percentage of this position within the account: marketValue of the position / totalValue
- Cash balance
- Total investment value (sum of market values for all open positions)
- Total value of the account (cash balance + total investment value)
- Percentage of cash holdings relative to total value of the account
- Percentage of investment holdings relative to total value of the account
- Transaction history
    - History of all open/closed positions (basically, activity history on the account)

More Details

Weighted average price is calculated as:

$$Weighted\ average\ price = \frac{(first\ price\ x\ shares) + (second\ price\ x\ shares)\ ....}{total\ number\ of\ shares}$$

See https://www.fool.com/knowledge-center/how-to-calculate-weighted-average-price-per-share.aspx for more details.

This, as any other story, can be implemented in single or multiple endpoints.

STORY 19: GET SUMMARY INFORMATION FOR ALL TRADER'S ACCOUNTS

Requirement

As a trader, I want to be able to see summary information across **all** my accounts:

- Top 5 positions by current market value
  - If I have an open position for the same stock in multiple accounts, I want to make sure total value is calculated across accounts
- Total available cash across all accounts
- Total investment value (sum of market values for all open positions)
- Total value of the account (cash balance + total investment value)
- Percentage of cash holdings relative to total value of the account
- Percentage of investment holdings relative to total value of the account
- Last 10 transactions across all accounts

I want all data to be reported in my 'home' currency (value of cash balance and investments in the accounts which are not in the home currency should be converted on the fly at the current posted exchange rate).

More Details

Exchange rates can be 'pegged' and set up via configuration, though more robust solutions are preferred (such as getting the current rate from a proper interface; implementation can be 'static').

STORY 20: ACCOUNT VALUE PERFORMANCE TRACKER

Requirement

As a trader, I want to get information about specific account performance over a specific time frame. I want to provide 'from' and 'to' dates and expect to see how the total value of the account changed over time based on the historical prices of the stocks.

If historical information for any of the stocks within account holdings is not available for the specified time frame, I want to calculate the linear approximation of the price between two known points. For example if 10 shares of AAPL were bought on November 1st for 150 dollars per share and then sold on December 1st for 200 dollars per share, the price per share on any of the days between November 1st and December 1st will be defined as: 150 + (200 – 150) / 30 * numOfDaysFromNov1st.

STORY 21: PORTFOLIO PERFORMANCE TRACKER

Requirement

As a trader, I want to get information about performance of all of my accounts over a specific time frame. I want to provide 'from' and 'to' dates and expect to see how the total value of all of my accounts changed over time based on the historical prices of the stocks.

If historical information for any of the stocks within account holdings is not available for the specified time frame, I want to calculate the linear approximation of the price between two known points. For example, if 10 shares of AAPL were bought on November 1st for 150 dollars per share and then sold on December 1st for 200 dollars per share, the price per share on any of the days between November 1st and December 1st will be defined as: 150 + (200 – 150) / 30 * numOfDaysFromNov1st.

I want all data to be reported in my 'home' currency (value of cash balance and investments in the accounts which are not in the home currency should be converted on the fly at the current posted exchange rate).

More Details

Exchange rates can be 'pegged' and set up via configuration, though more robust solutions are preferred (such as getting the current rate from a proper interface; implementation can be 'static').

STORY 22: SECURING REST SERVICES

Requirement

As a product owner, I want to make sure only privileged users have access to the corresponding functionality of the application.

Operation	Role(s)
Create/Update/Delete historical stock quote data or mock data	Application Administrator
Retrieving historical or mock trading data or various summaries and statistics of thereof	Application Administrator, Trader
Create/Update/Delete/List/Search traders or various summaries or statistics of thereof	Application Administrator
Create/Update/Delete/Read accounts for ALL traders	Application Administrator
Access account details for specific trader	Application Administrator, Trader (can access only his own accounts)
Get detailed and summary information on the account	Application Administrator, Trader (can access only his own accounts)
List positions in the account, search and get various stats on the positions	Application Administrator, Trader (can access only his own accounts)
Open or Close positions in the account	Trader (can access only his own accounts)
Set up algorithmic trades and run trade simulations	Trader (can access only his own accounts)

The above is an ideal 'wish list'. Depending on the design and implementation of the services, this can be easy or hard(er) to achieve. Team is encouraged to get as close to this set of privileges as possible. Rule of thumb:

- Any 'destructive' (CUD) operations on the stock quote files, accounts, or traders are available only to the application administrator
- Any 'destructive' (CUD) operations on the positions within the account are available only to the trader this account belongs to
- Any non-destructive operation (R) is available to anyone, with exception of Read operation on the individual trader account (trader can access only his own account).

More Details

Check documentation for the form of 'antMatcher' call which accepts HTTP method used to access the endpoint.

To retrieve username of currently logged in user: http://www.baeldung.com/get-user-in-spring-security (easiest way is to pass 'Principal' object; numerous other ways exist).

STORY 23: BONUS – RETRIEVING HISTORICAL QUOTE DATA FROM ONLINE SERVICE

Requirement

As an application administrator, I want historical quote data to be downloaded ahead of time from one of the online services, such as: https://www.alphavantage.co/documentation/, and saved to the server. Furthermore, I want a local copy of the data to be automatically updated on an ongoing basis from the online source, at least once a day.

I will provide the following parameters:

- Stock ticker and stock exchange
- Time frame for data download