Table of Contents

[Introduction 3](#_Toc137041823)

[Known Bugs 3](#_Toc137041824)

[Functionality 3](#_Toc137041825)

[Admin Page 6](#_Toc137041828)

[Technology Stack 7](#_Toc137041829)

[UML, ERD, Wireframes and Flowchart 7](#_Toc137041830)

[Gurdeep Inc. Stock Price generation 10](#_Toc137041832)

[The BIG Spring Security Change-Up of Friday Night 11](#_Toc137041833)

[Log of Thoughts: 12](#_Toc137041834)

[Ideas That Did Not Make The Cut 16](#_Toc137041835)

[Future Work 16](#_Toc137041836)

[Conclusion 16](#_Toc137041837)

[References 17](#_Toc137041838)

# Introduction

My final project idea was to create a platform to trade options, as options are relatively unknown to the general public, and this will allow them to get more familiar with them. To see my thought process, see the Log of Thoughts.

# Known Bugs

Sometimes the Login page will force you to login again, even if login information is correct. This is being looked into.

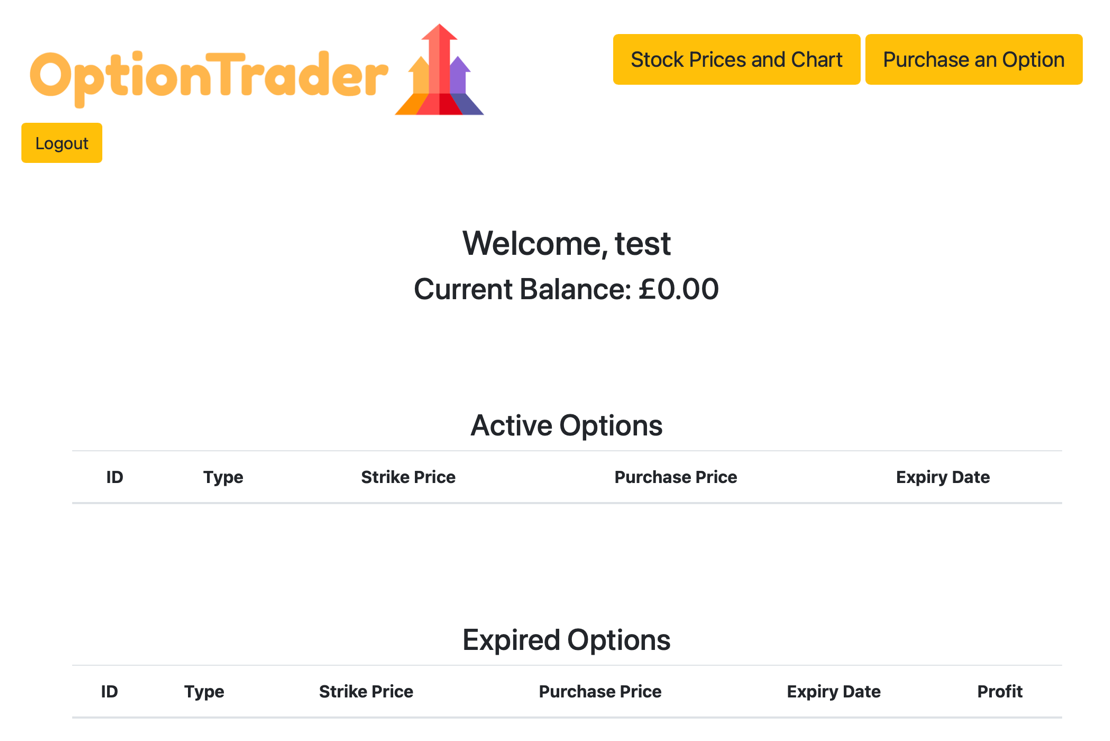
# Functionality

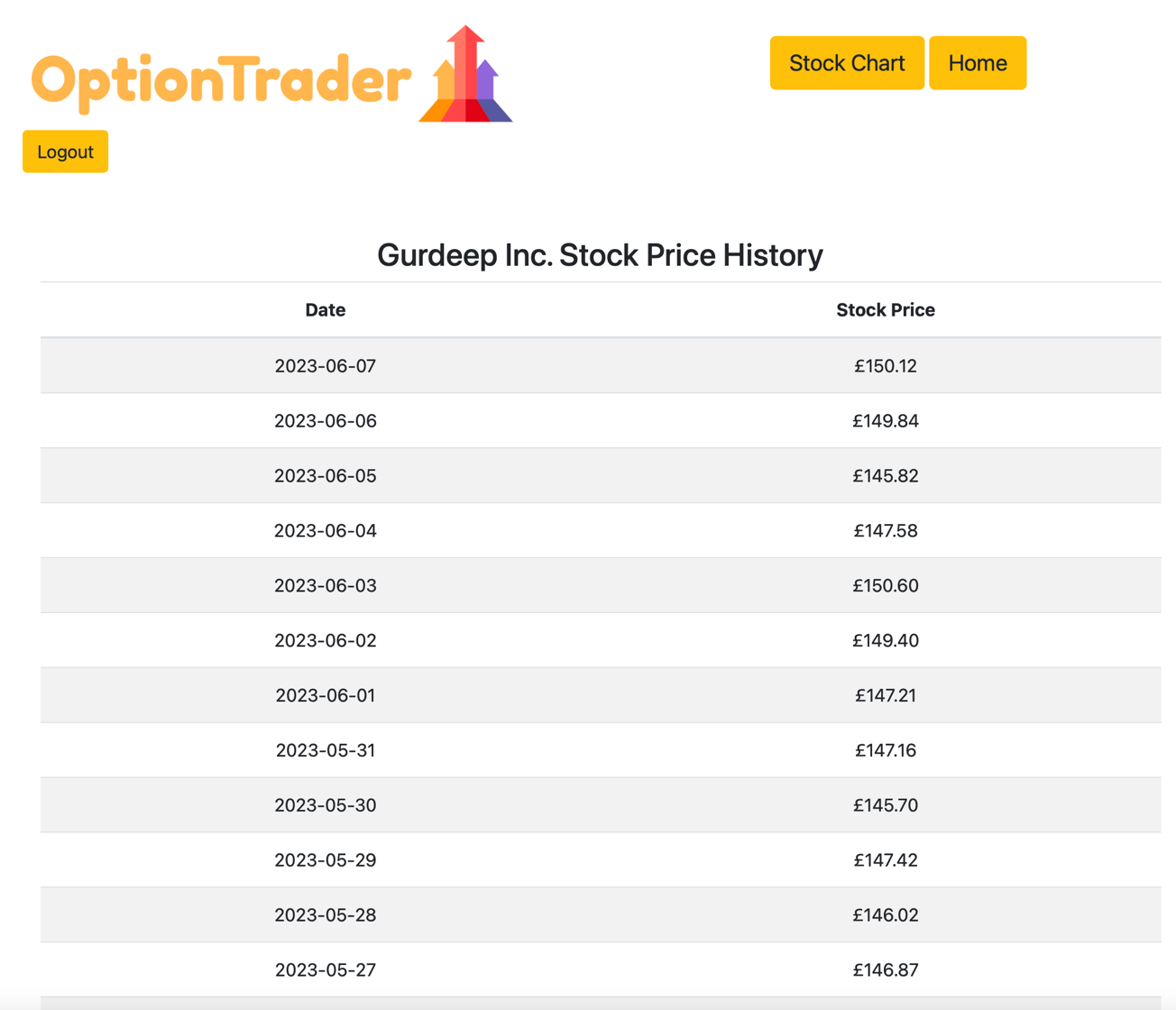
The project offers full integration of Spring Security for a secure login, complete with password encryption. When using the project, if the user has not logged in, they cannot access any pages other than the Login and Register pages. If they attempt to access any other pages, they will be redirected to Login.

Both the Login and Register have validation for their fields, as shown below.

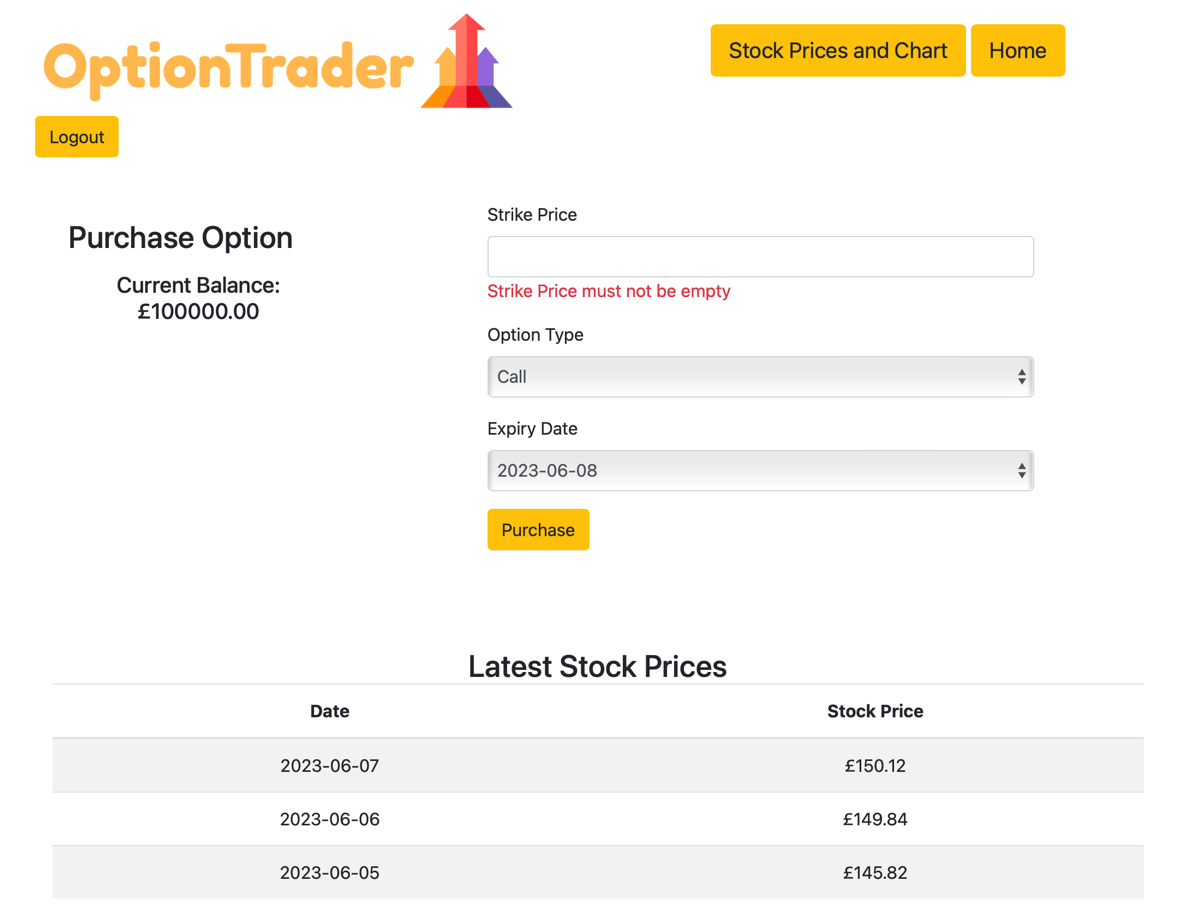
## 

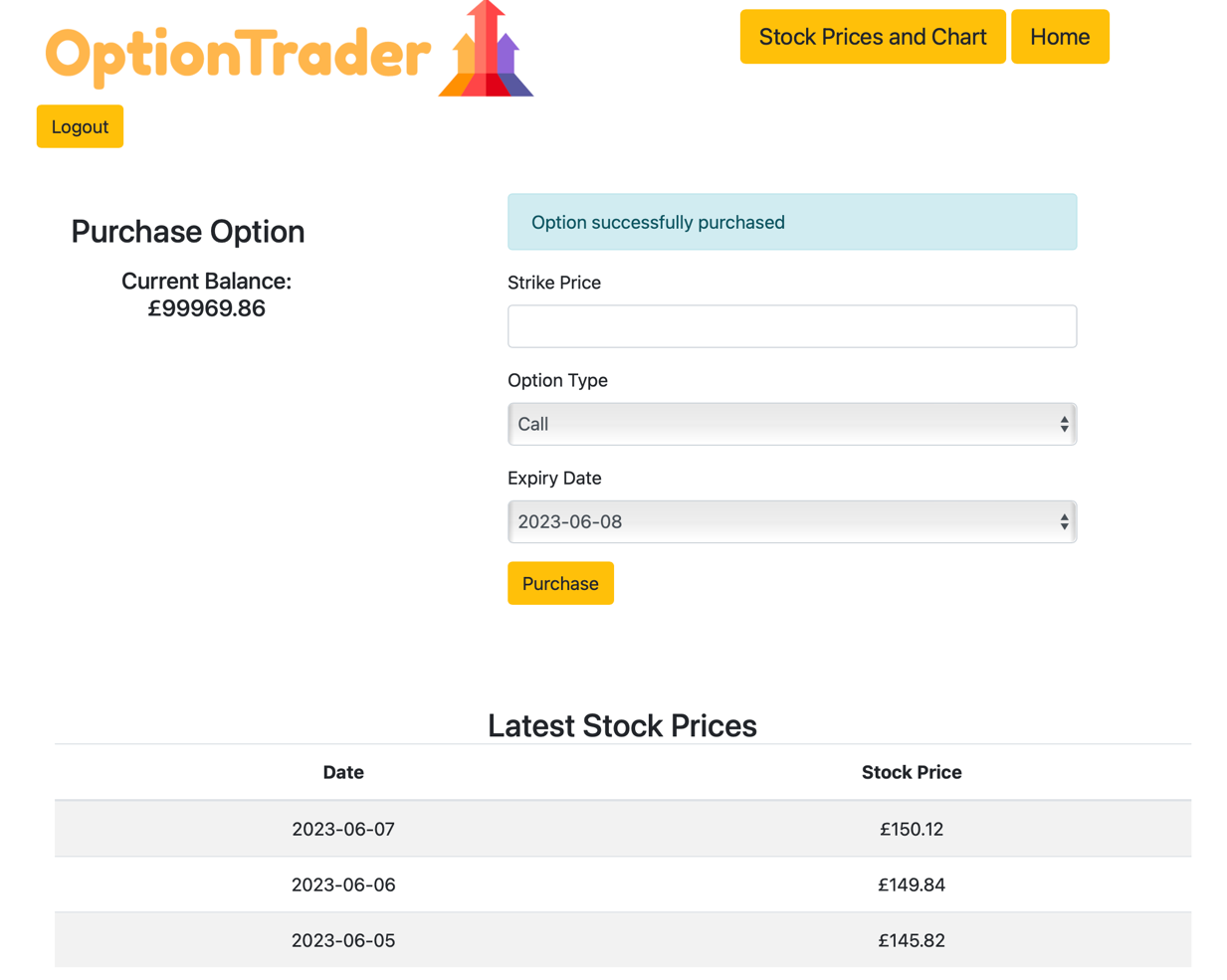
Upon successful login, the user is directed to the Home page. Here they can view their balance, view their active and expired options, along with go to another page to purchase an option, or view the stock prices and stock chart for Gurdeep Inc. They can also Logout.



The ‘Stock Prices and Chart’ button takes you to a full history of Gurdeep Inc. stock prices until today’s date, as shown below.

From here, the ‘Stock Chart’ button can be selected to see a graphical representation of the stock prices.

From the home page, a user can purchase an option.

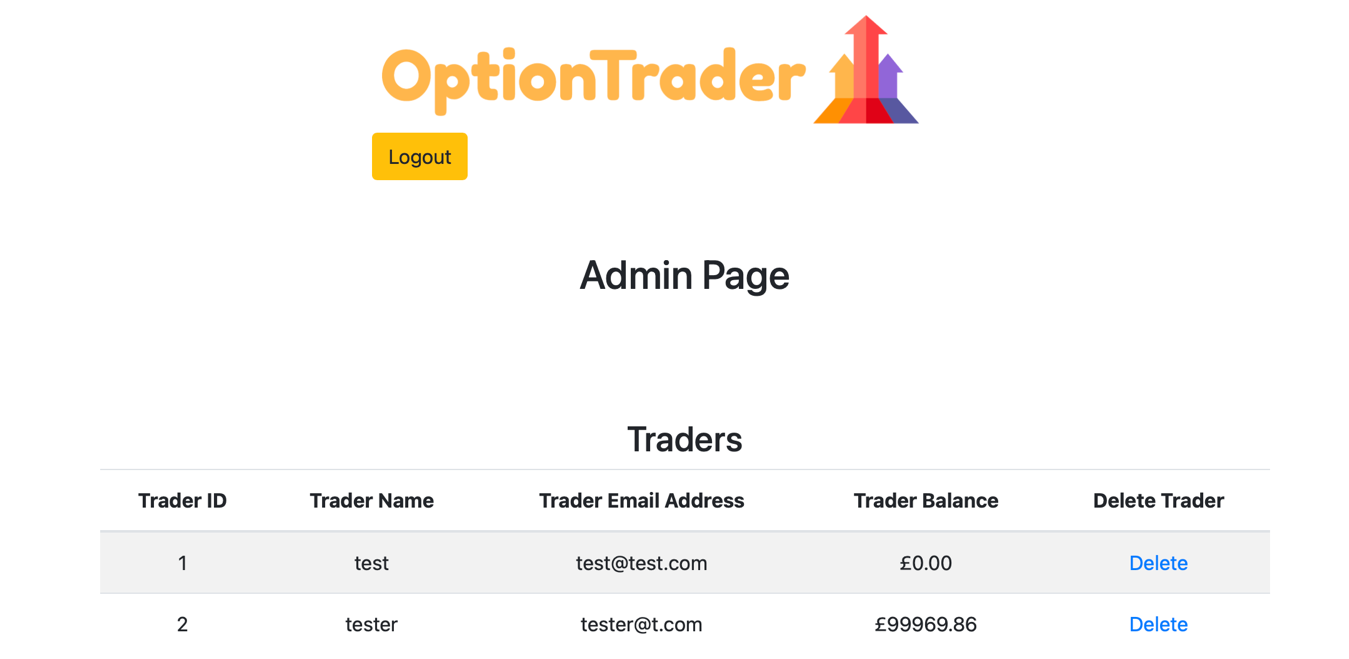
This page has validation for the Strike Price, along with displaying the user’s current balance, and the latest 3 stock prices, so they can make a more informed decision. Should they choose to purchase an option, the option price will be calculated and checked against their balance to ensure they can afford it. If they can, their balance will be changed to reflect the purchase.

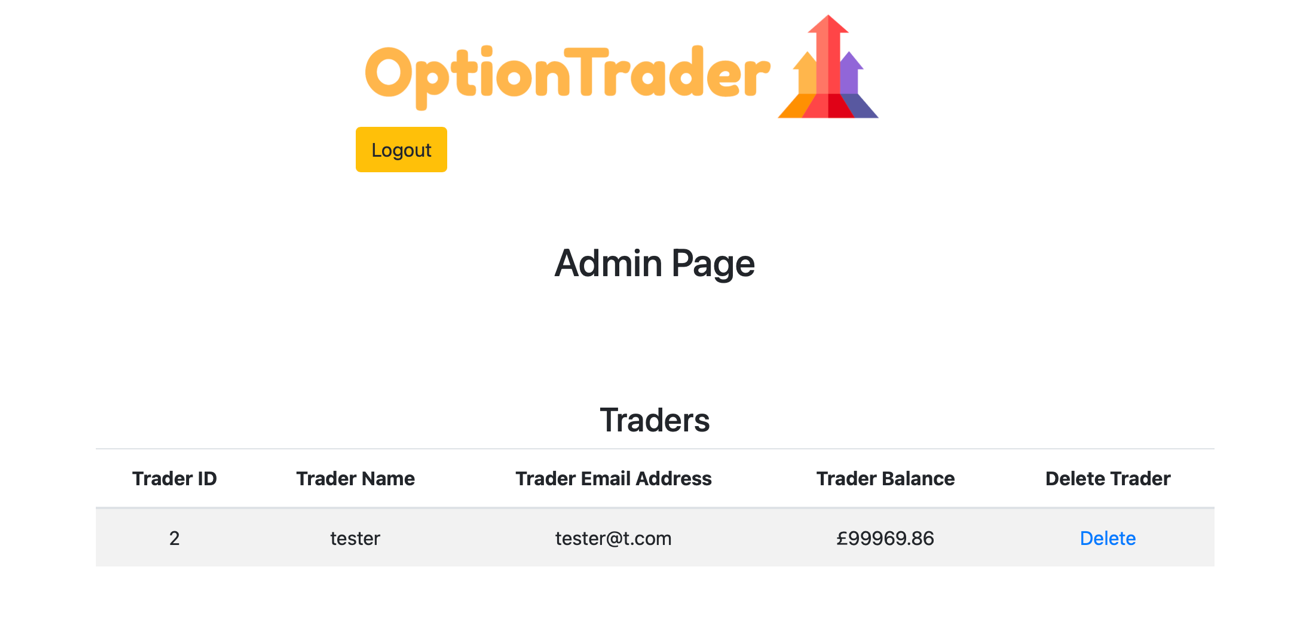
Returning to the home page, it can be seen that the balance has updated here also, along with the option now appearing in the ‘Active Options’ table.

## 

Every time the user logs in, there is a check to see if any active options have expiry date = login (today’s) date. If there is, it checks the stock price of Gurdeep Inc. and sees if the option makes any profit. If it does, it is added to the user’s balance, and the option moves to the ‘Expired Options’ table.

## Admin Page

To demonstrate the D in CRUD (delete functionality), there exists an admin page to view all non admin users. Currently this is available to view for all users, should they know the URL. This URL is <http://localhost:8020/admin>. 

From here, any non admin users can be deleted, as shown below.

The functionality to restrict this page to admins is in the works.

# Technology Stack

This changed halfway throughout the development process, but the final technology stack is as follows.

Backend:

JDK 17, Spring Boot 3, Maven, Lombok, Spring Data JPA, MySQL

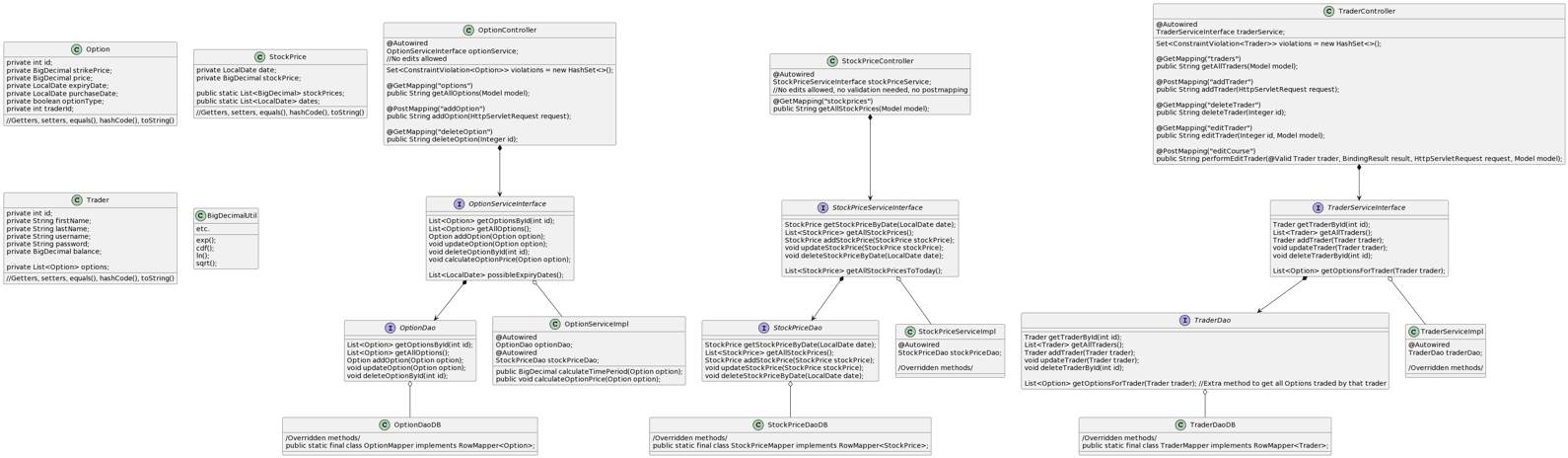
Front end:

Thymeleaf, HTML, CSS, JavaScript, Bootstrap 5

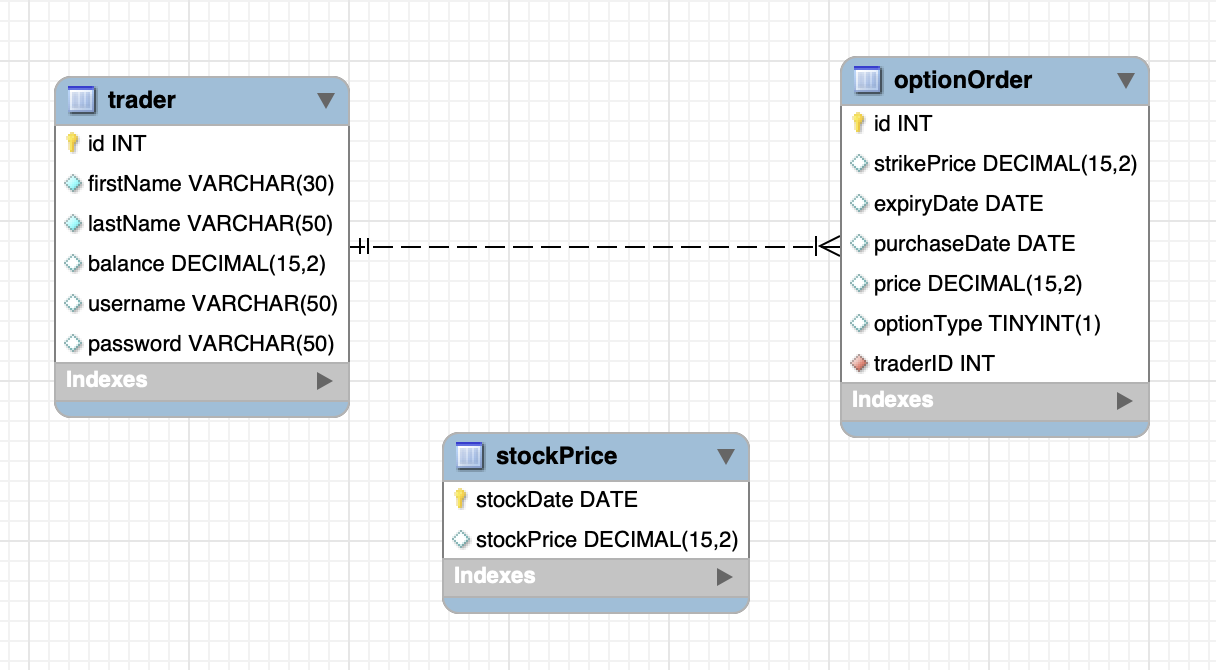
Stock price generation:  
C++

# UML, ERD, Wireframes and Flowchart

The initial UML looked as follows:



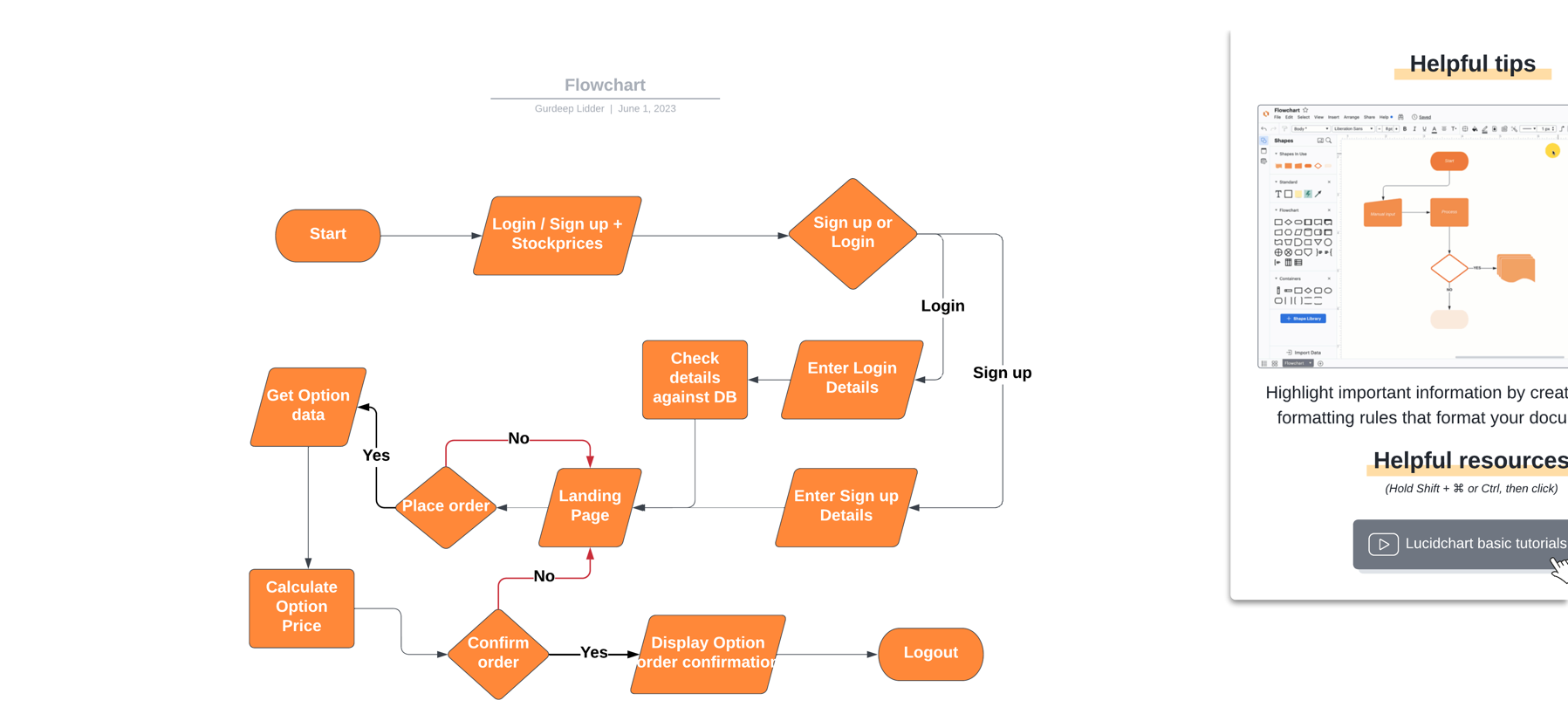
The initial ERD:



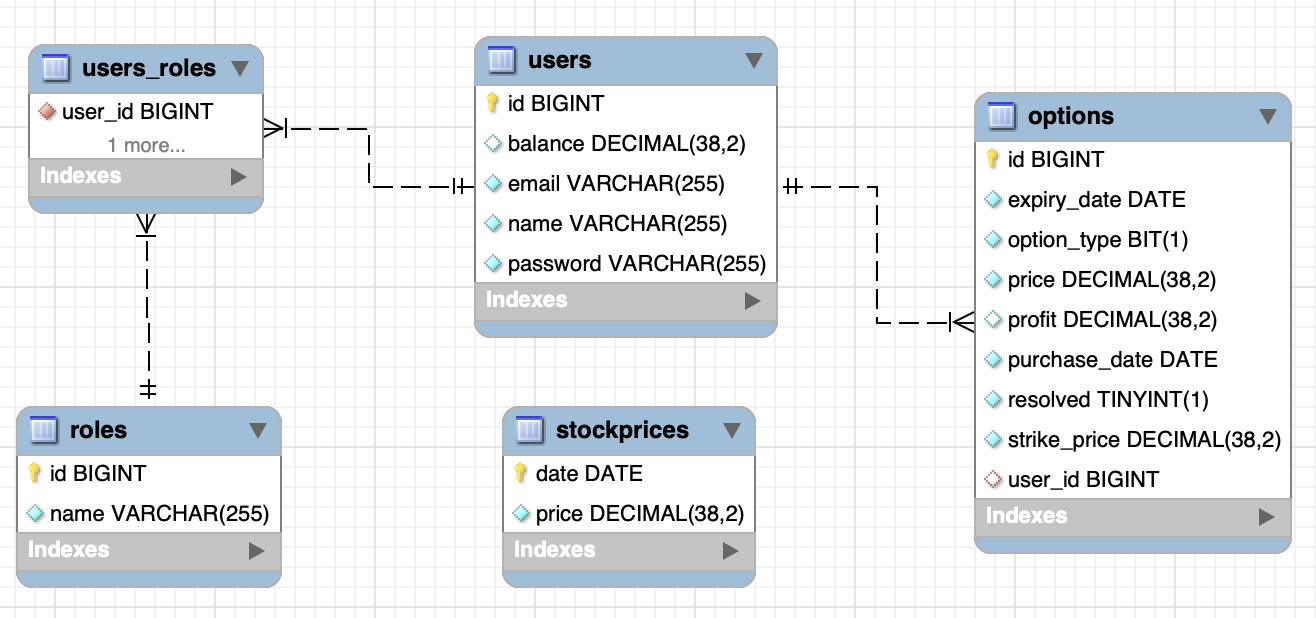
The initial Wireframe for a webpage of purchasing an option:

# 

Along with the associated flowchart:



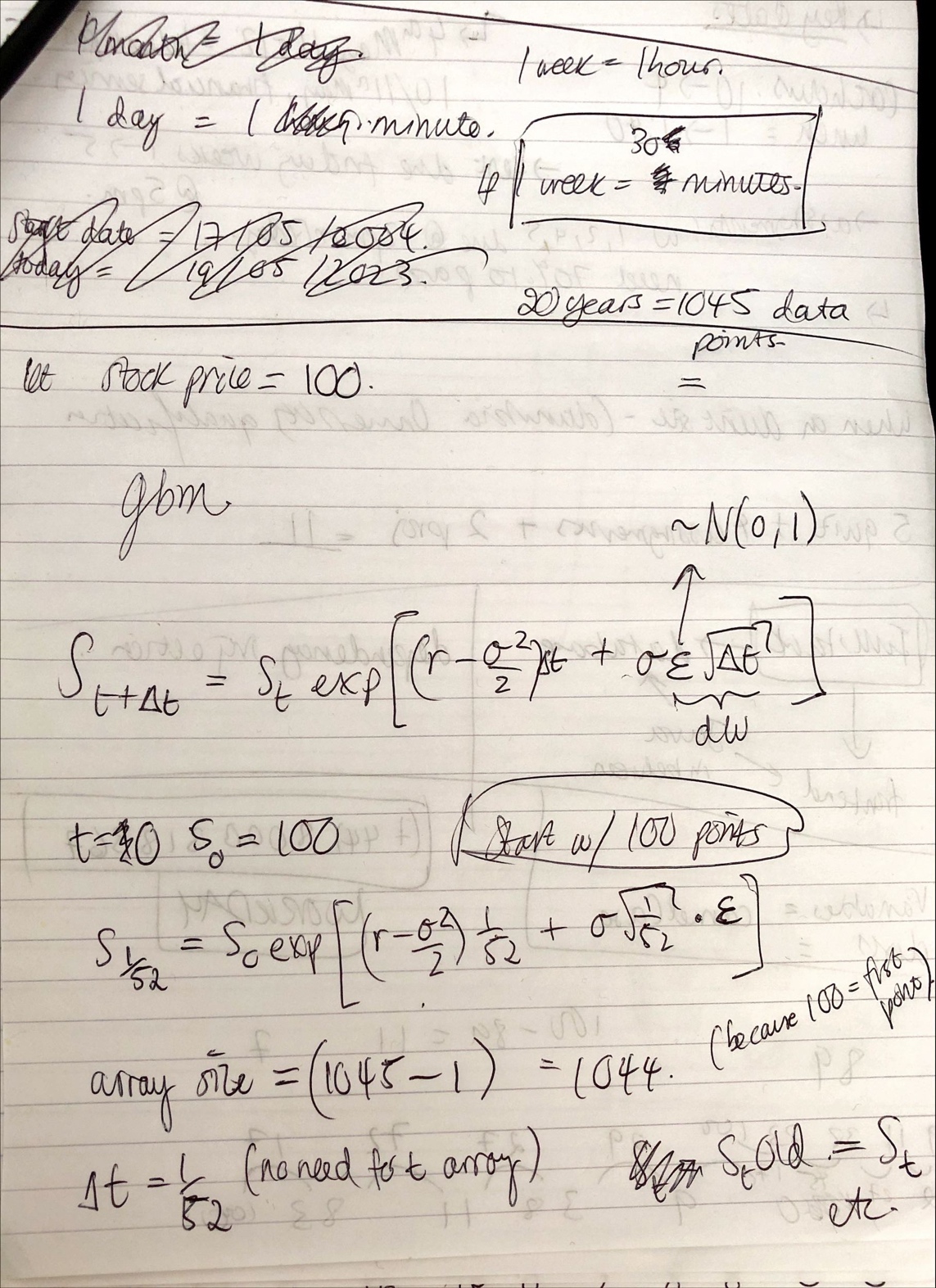
Since the project was restarted for easy integration of Spring Security, the ERD now looks like this:



# Gurdeep Inc. Stock Price generation

The considerations behind generating my own stock and the option pricing itself took the majority of the thinking time on this project.

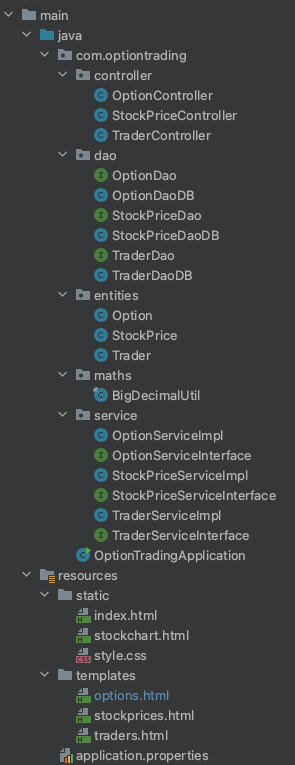
Option pricing is a very difficult process, which still has not been perfected, even in industry. Early considerations were to use real life stock price data and price options from there. However, because of the time constraints of the project, this was not possible, due to it requiring much more stochastic analysis to find parameters for pricing.

The pricing model used for this project was the most primitive and simple one, Geometric Brownian Motion, and prices used the Black-Scholes formula. Early workings with the formula are shown below.

The above formula was used to simulate 20 years of stock prices for Gurdeep Inc., in C++ because of its speed and accuracy in mathematical computation, with the parameters being created by myself. Option pricing was done using the Black Scholes formula (see e.g. <https://en.wikipedia.org/wiki/Black–Scholes_model>) and results were checked against https://goodcalculators.com/black-scholes-calculator/.

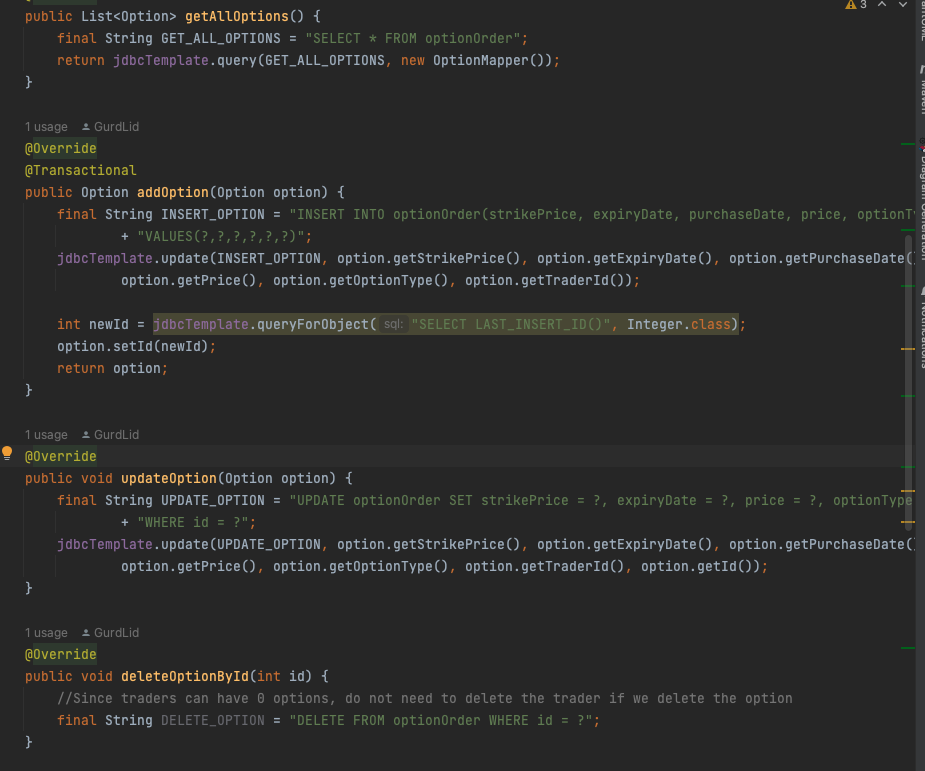
# The BIG Spring Security Change-Up of Friday Night

The initial version of the project as was presented in Friday’s presentation was an admin view of the platform. There were 4 pages, one for listing all users, one for listing all options, 1 for stock price history and 1 for stock chart. This was done using JDBCTemplate for database interactions and had file structure as follows:

This was functional, but I wanted secure logins as part of the project, as this would produce a more useful project at submission date.

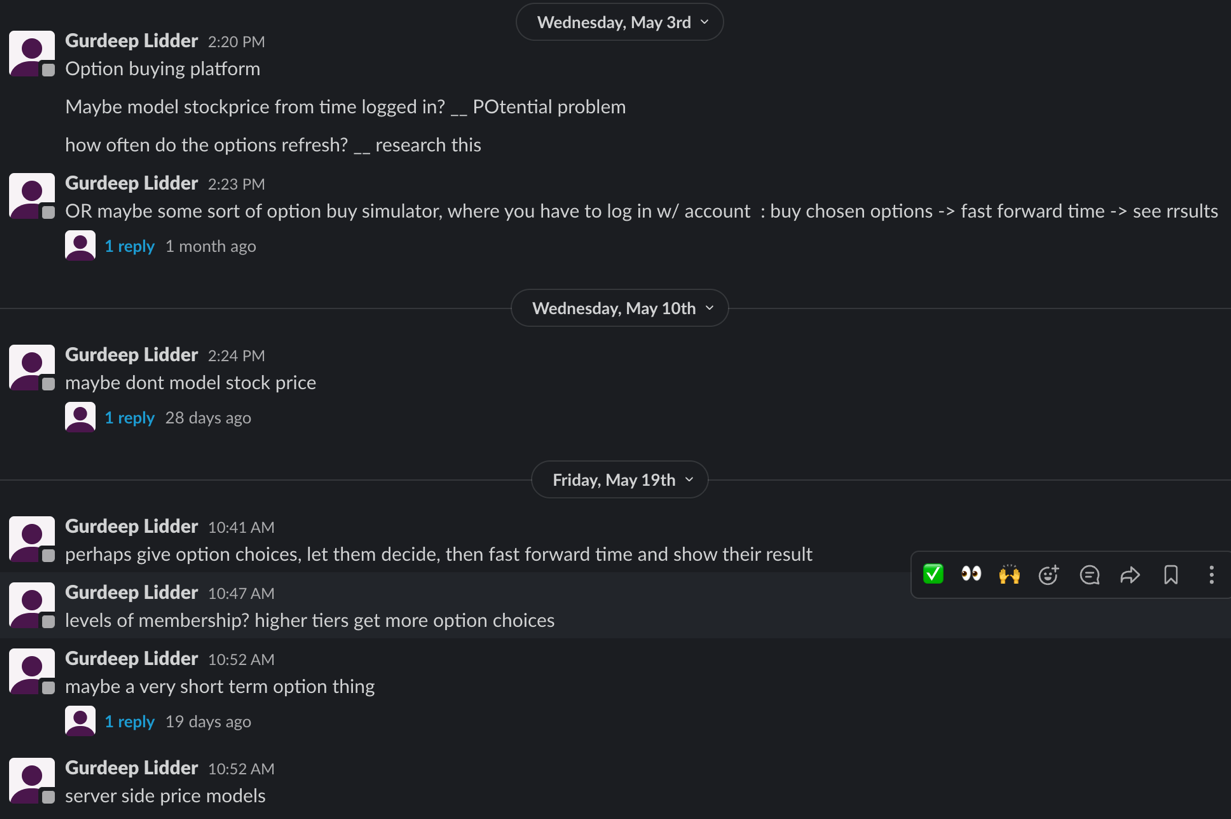
For this reason, a restart was done to learn and incorporate Spring Security, and this brought the introduction of Lombok and Spring Data JPA to reduce boilerplate code and interact with the database.

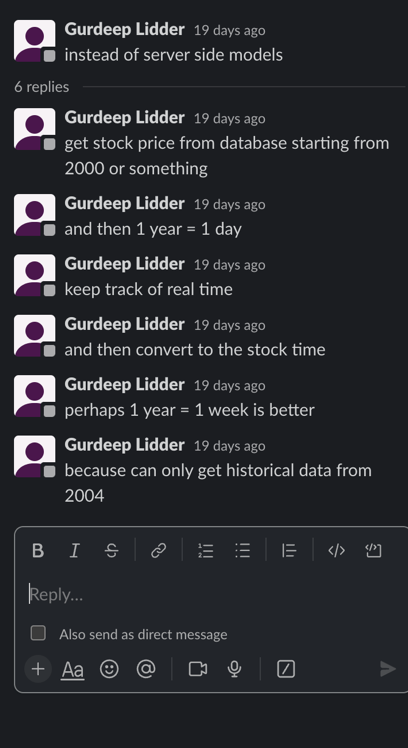
An example of part of one of the DAO’s is shown below for reference.



# Log of Thoughts:

The following section is a log of thoughts that I had during the planning and coding phases. It started with a Slack conversation I had with myself on the 3rd May 2023, as shown below.





This then moved to this document as follows:

Choosing the date translation (1 day = 1 week etc)

Choosing stock

Whether to model stock price

If choosing real historical stock, getting r and sigma?

If making own stock, deciding on r and sigma

Java: normal cdf and pdf, plus bigdeicmal w/ exponential etc

BigDecimalUtils from <https://github.com/javadev/calculator>

Just not accurate enough, trying <https://arxiv.org/pdf/2206.12601.pdf>

For problems with LocalDate:

<https://medium.com/@pdouvitsas/global-localdate-format-in-spring-boot-and-thymeleaf-29ff83b8f4c8> **https://www.baeldung.com/dates-in-thymeleaf**

‘Gurdeep stock’ takes no notice of IRL events e.g. global financial crisis 2008- could be something to implement in the future.

Something like as follows to handle the deletion of a customer and it’s associated Option orders: (likely a one to many relationship)

———————————————

In this case, let's just delete any Meeting associated with the Room. To that end, we need a separate delete statement for each:

final String DELETE\_MEETING\_BY\_ROOM = "DELETE FROM meeting WHERE roomId = ?";

jdbc.update(DELETE\_MEETING\_BY\_ROOM, id);

final String DELETE\_ROOM = "DELETE FROM room WHERE id = ?";

jdbc.update(DELETE\_ROOM, id);

We start out by deleting any Meeting with the Room ID we want to delete and then follow up by deleting the Room itself. Now if the Meeting had any other relationships with other objects, we would also have to deal with those, but we'll look closer at that when we look at many-to-many relationships.

————————————————

This is complex: perhaps just track people’s option trading.

Or perhaps not, make sure they sell the option to someone else: as per the example on https://www.investopedia.com/terms/o/option.asp#toc-example-of-an-option

MIN 2 TABLES, MAX 6 TABLES in SQL

NEED A WIREFRAME (basic strcutre)

Include a State of Art.

<https://codepen.io/> for front end stuff?

For now trades probably only allow one type of option to be traded per transaction (in certain quantities). Future work could be to implement an ‘order’ table, with different option orders in place. With <https://vertabelo.com/blog/a-data-model-for-trading-stocks-funds-and-cryptocurrencies/> for reference

If a customer closes their account, we should probably close that option.

MINIMUM Option length = 1/365.25 (or should I only allow trading days)

**— PROBLEM: trades are allowed on weekends, but have to model stock price only on weekdays**

**Gurdeep stock is based on GMT time zone**

*Writing options not allowed, only buying*

*Limited to a selection option lengths : 1 day, 1 week, 1 month, 6 months, 1 year, 10 Years?*

*—Since stock prices are only daily, trading is instantaneous*

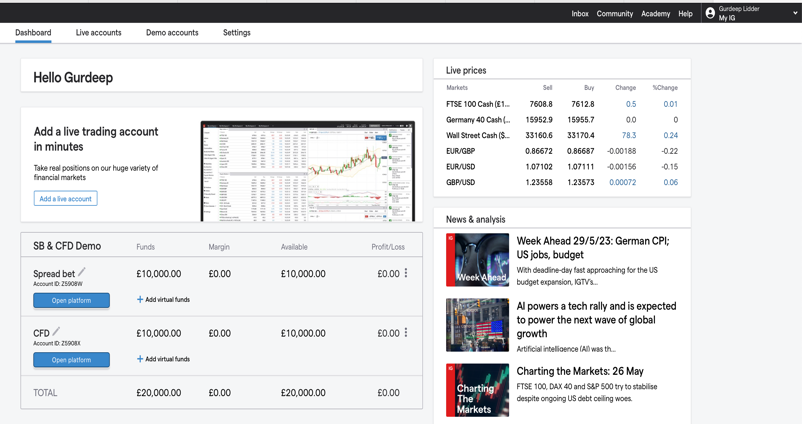
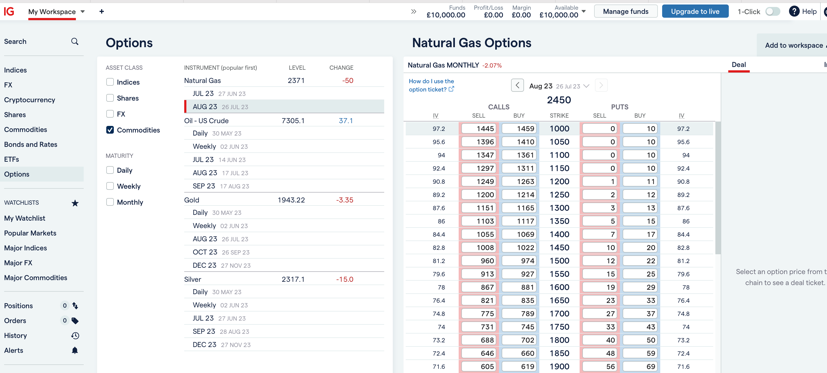
Perhaps consider a stock price graph (front end): this? <https://www.anychart.com/blog/2022/02/24/stock-chart-js/>

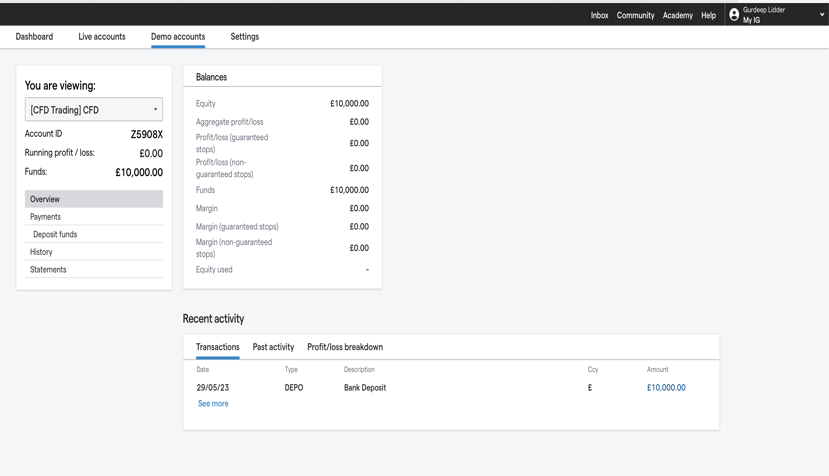
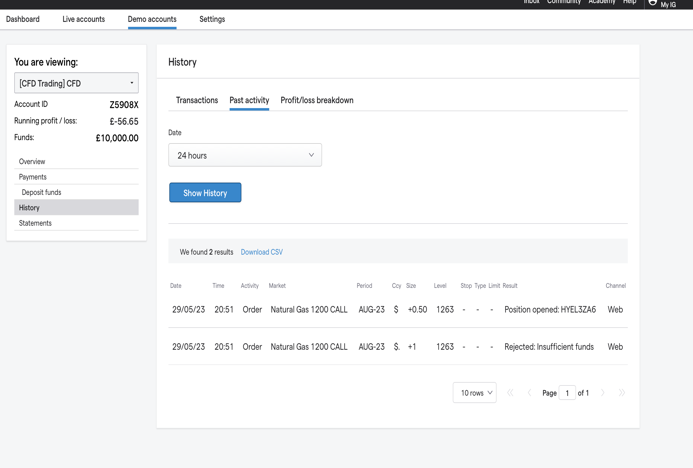
Generated SQL database and in this process, perhaps might be nice to have an ‘active’ boolean for each option, but then again maybe not, since it is only not active if (in this app) it hasn’t expired yet.

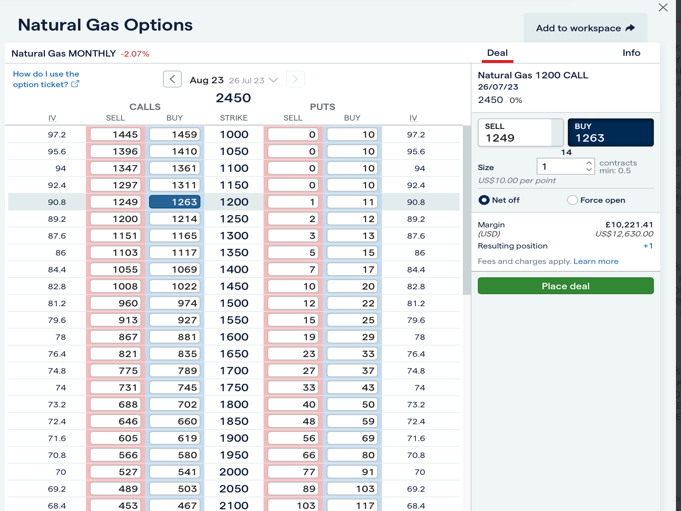
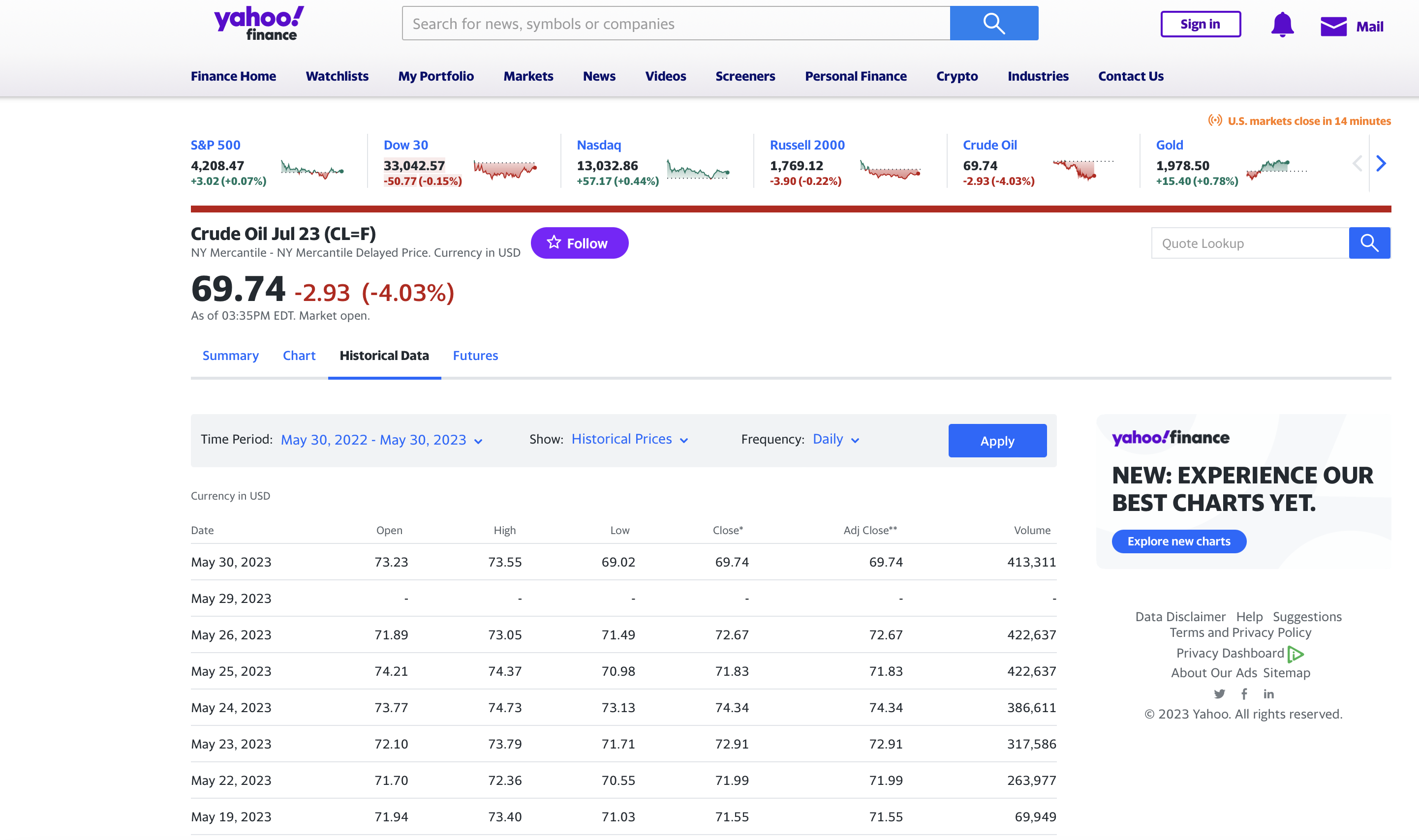
For now, only allowing one option per order.

Looked at <https://tastytrade.com> for inspiration of landing page and SOA, along with <https://www.ig.com/uk>

Opened a demo account at IG:

****

****

****

This yahoo finance is inspiration for my historical data showing (especially the date formatting)

**The idea: good tool for teaching how options work, people know how to trade stocks, but many people are unaware of options, explain their advantages etc.**

Include a service method that sorts the options by expiry date.- completed

**PROBLEM:::**HV000030: No validator could be found for constraint 'javax.validation.constraints.NotBlank' validating type 'java.math.BigDecimal'. -> validation for bigdecimal not easily solved

Currently no unique usernames in sql database, need to do this in the database

Passwords was changed to password type in html doc, now hides the characters!

Whether the option was a call or a put was never included!

Time period creates problems, need a column for purchase date too.

Used github gists to store the stock price data for the stock chart.

To get today’s date in CSS: <https://www.w3resource.com/javascript-exercises/javascript-basic-exercise-3.php>

Big problem: Spring Security has deprecated WebSecurityConfigurerAdapter class. Need to follow tutorial: <https://www.youtube.com/watch?v=7HQ-x9aoZx8>

Also using <https://spring.io/guides/gs/securing-web/> - not used

Help with the login stuff: https://www.codejava.net/frameworks/spring-boot/user-registration-and-login-tutorial

STILL NEED TO RESOLVE THE OPTIONS

This: <https://www.javaguides.net/2018/10/user-registration-module-using-springboot-springmvc-springsecurity-hibernate5-thymeleaf-mysql.html> and especially this:

<https://www.codeburps.com/post/spring-boot-form-login> might save my login worries.

***Massive change up:***

Changing the project completely to incorporate Lombok and JPA. This is to accommodate for Spring Boot Security. Problems so far: stockchart.html cannot be loaded from the home page, suspect this maybe because there is no get mapping: it is trying to go to home/stockchart.html (which doesn’t exist).

For now only allowing them to go to ‘Stockprices and Chart’ and then go to the chart from there.

Still suffering from the having to login twice bug.

Can now go to home page, but have to think about the GetMapping on Homecontroller (“/”) as from stockprices to go to home I need to go to “home/” in the html file

Many to one queries are very problematic for getting the options for one guy- maybe resort to JDBCTemplate?\\

No need for JDBC template, JPA is quite clever.

Problem getting the details for the logged in user. Found code from <https://dzone.com/articles/how-to-get-current-logged-in-username-in-spring-se> --but apparently this isnt needed, can use the CustomUserDetails class in a roundabout fashion.

Might transition to a user focused app, remove the admin side of things.

Live updating of a page is not possible in Thymeleaf according to <https://stackoverflow.com/questions/63661485/updating-webpage-element-with-java-spring-mvc-thymeleaf-without-refreshing>. This was desired to give a live update of the price when purchasing

Have to add a column to the Option table: BigDecimal profit. Default will be null, when the option expires, it is calculated.

Logo created on logocreator.io, inserted into webpage via uploading it to imgur, as the source path is not working/

<https://getbootstrap.com/docs/4.0/components/buttons/> used for changing branding

Maybe we can change the orange on the logo to the yellow on the buttons for better matching/

# Ideas That Did Not Make The Cut

* To make the project an option pricing simulator, such that users could more easily see an option expire. This would work such that, for example, 1 hour in real life would represent 1 week in the platform. This proved too complicated for the time constraints
* Using live real world stock price data. This opened a can of worms when considering pricing the options, as parameters such as risk-free rate and volatility would have to be calculated from historical data, which was not possible in the time constraints

# Future Work

* Firstly, to iron out bugs, and fix the database situation
* To incorporate real world data. This is much more complicated but would give the user a more fun experience
* Improve the visuals of the platform. Whilst not bad at the moment, there is much room for improvement, which would require learning more about front-end technologies and tools
* Have a more seamless integration of the stock chart
* Allow users to write (sell) options. Current functionality only allows the purchase of a call option, not a put options nor selling of either type
* An educational page which has a good description of what options are

# Conclusion

This project has been an incredible learning experience and has produced a result that has a few bugs, but on the whole has a good level of the desired functionality. There is lots of room for further work, and this project will continue to be one that I improve consistently.

# References

<https://github.com/javadev/calculator>: Mathematical functions in Java BigDecimal

<https://arxiv.org/pdf/2206.12601.pdf>: Used this research paper for an approximation for Cumulative Distribution Function for Option pricing

<https://vertabelo.com/blog/a-data-model-for-trading-stocks-funds-and-cryptocurrencies/>: Inspiration for initial database layour

<https://www.ig.com/uk> : SOA for option platform

finance.yahoo.com: SOA for displaying option prices

<https://www.anychart.com/blog/2022/02/24/stock-chart-js/>: for the StockChart

<https://www.w3resource.com/javascript-exercises/javascript-basic-exercise-3.php>: For getting current date into the StockChart

<https://www.codeburps.com/post/spring-boot-form-login>: Implementing Spring Security

<https://stackoverflow.com/questions/39930876/spring-get-custom-userdetails-in-securitycontextholder> : For getting current logged in user’s information

<https://getbootstrap.com/docs/4.0/components/buttons/> : Changing button style

<https://goodcalculators.com/black-scholes-calculator/>: Option price accuracy checks