Final Project

OPTION TRADING

Gurdeep Lidder

Introduction

Stocks are commonly traded, options are less well known

Aim was to build a simple platform to buy options to learn more about them

Options

In their most basic form:

Options are based on stock prices, and give the *option* to buy the stock at a set date

They have:

- **a fixed expiry date (e.g. 1 year)**
- **strike price = price at which you buy the stock at the expiry date**

Options example

Current stock price = £100
You think over the next year it will go to £150

Buy an option for £7.50 with Strike Price = £100

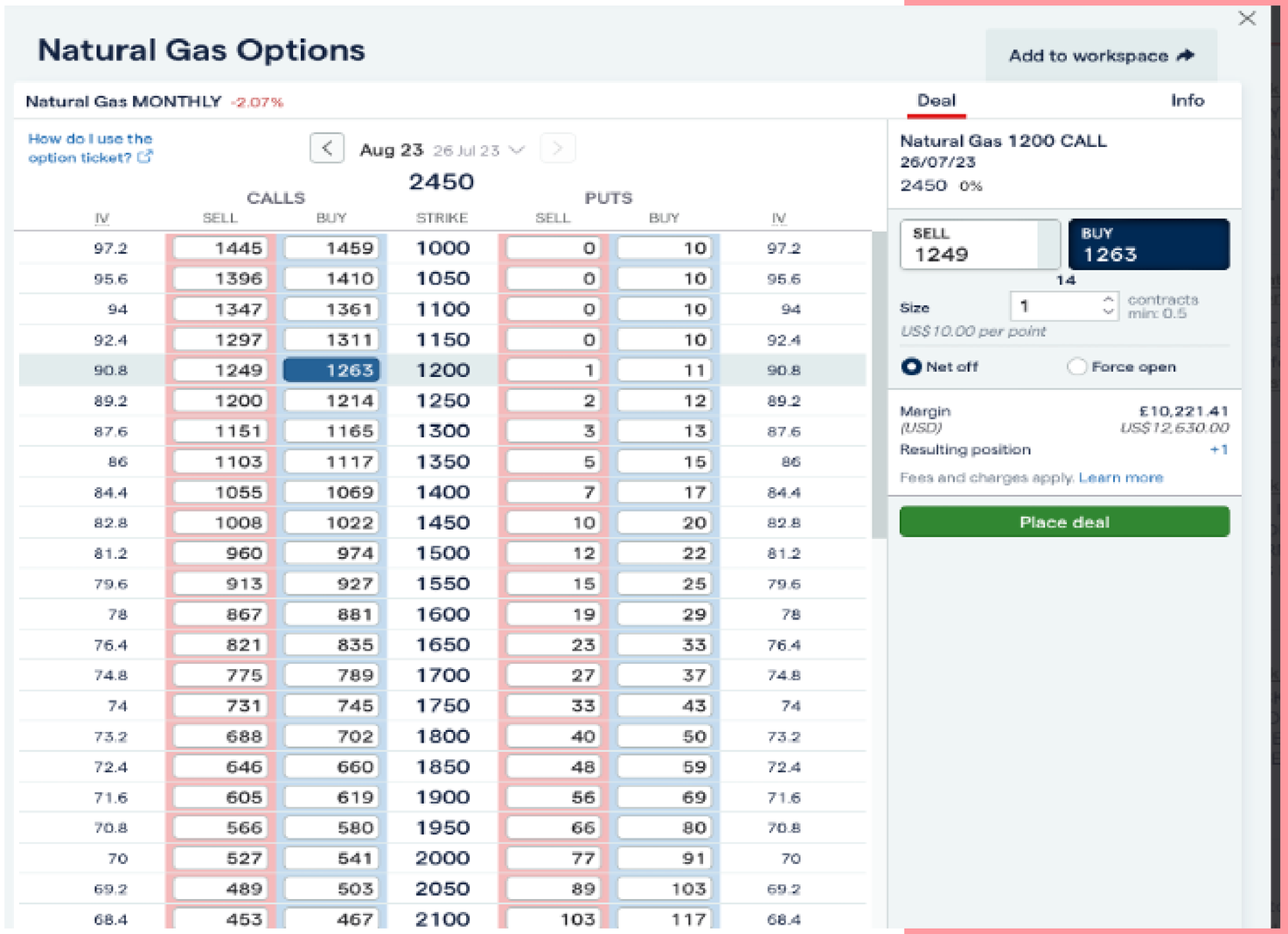
If the stock goes to £150 after 1 year, you can buy the stock at £100

Profit of £50 - £7.50

Inspirations

Options trading platforms:

https://www.ig.com/uk



Inspirations

Stock price displays:

https://finance.yahoo.com

Crude Oil Jul 23 (CL=F)
NY Mercantile - NY Mercantile Delayed Price. Currency in USD

Follow

70.13 +2.04 (+3.00%)
As of 04:34PM EDT. Market open.

Summary | Chart | **Historical Data** | Futures

Time Period: Jun 01, 2022 - Jun 01, 2023 | Show: Historical Prices | Frequency: Daily | Apply

Currency in USD

Date	Open	High	Low	Close*	Adj Close**	Volume
Jun 01, 2023	67.60	71.07	67.51	70.13	70.13	369,562
May 31, 2023	69.62	69.69	67.03	68.09	68.09	440,862
May 30, 2023	73.23	73.55	69.02	69.46	69.46	440,862
May 29, 2023	-	-	-	-	-	-
May 26, 2023	71.89	73.05	71.49	72.67	72.67	261,498
May 25, 2023	74.21	74.37	70.98	71.83	71.83	422,637
May 24, 2023	73.77	74.73	73.13	74.34	74.34	386,611
May 23, 2023	72.10	73.79	71.71	72.91	72.91	317,586
May 22, 2023	71.70	72.36	70.55	71.99	71.99	263,977

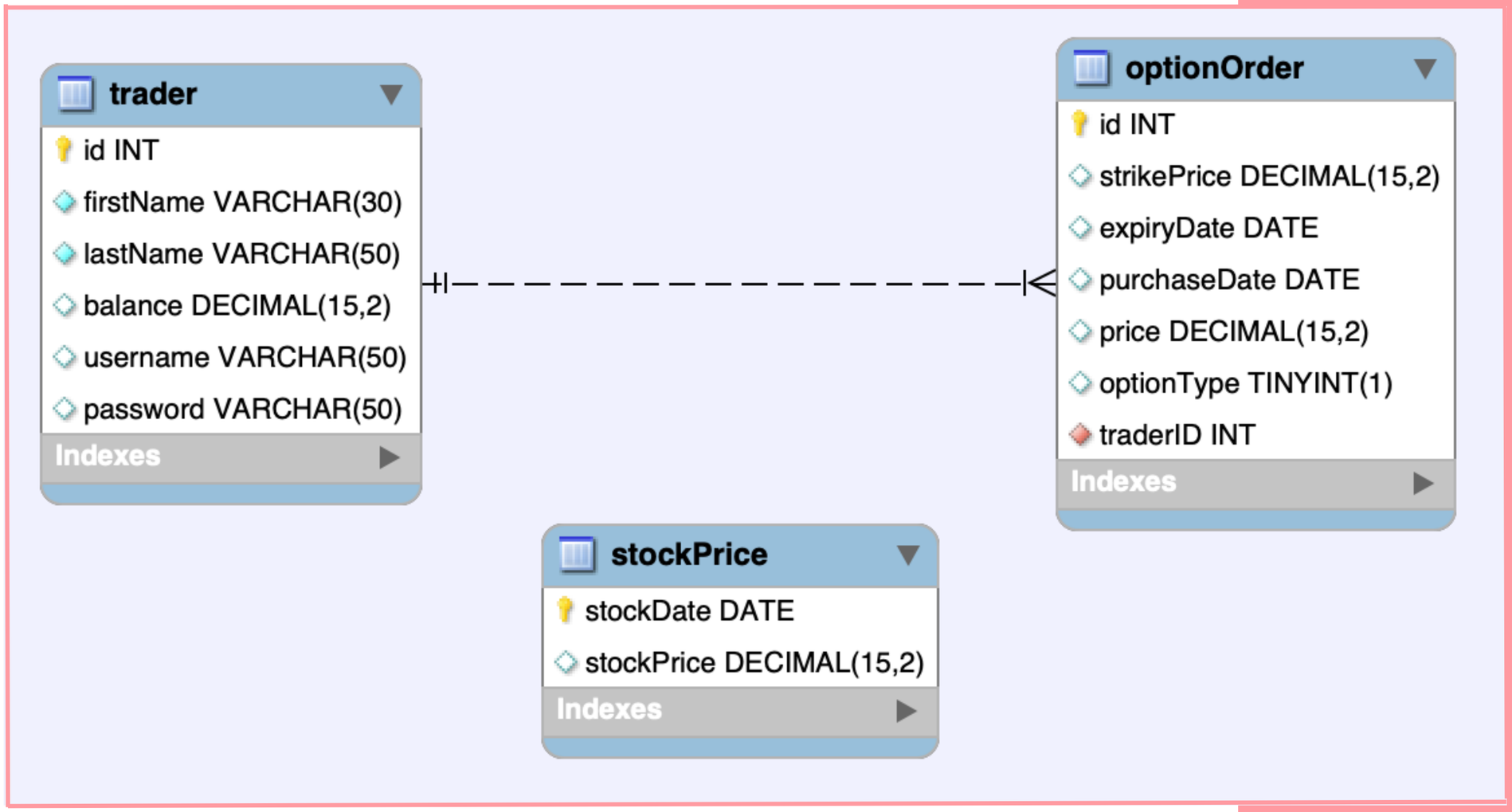
Details

The underlying stock is for Gurdeep Inc. and was generated myself

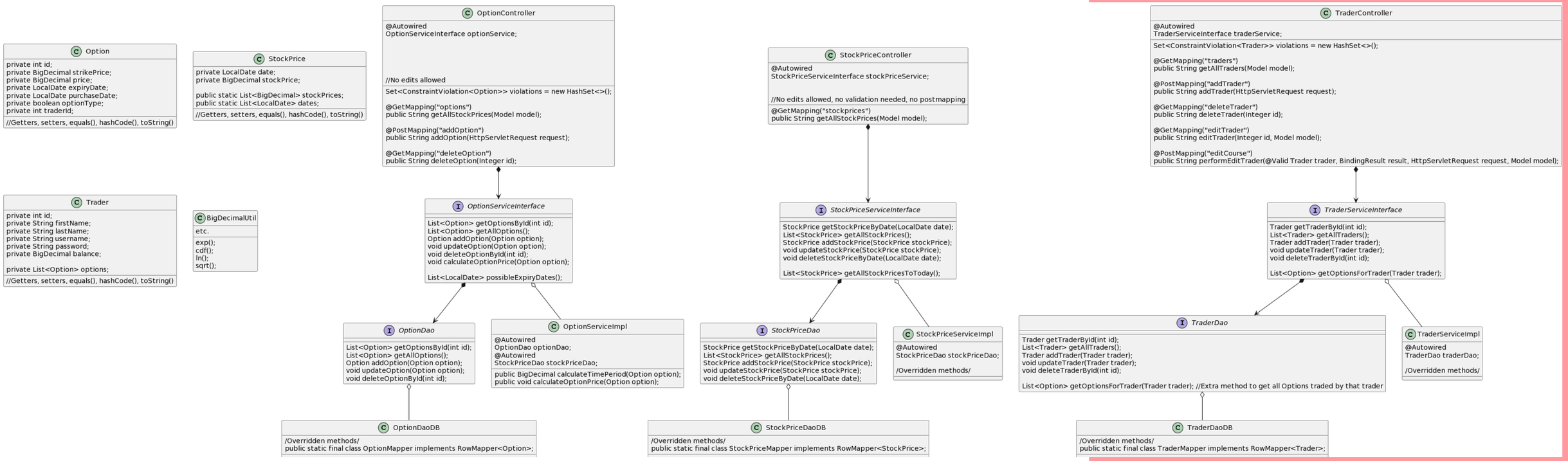
Stock simulated for 20 years from 2013, starting at £100

Stock prices are daily, retrieved from database everyday

ERD

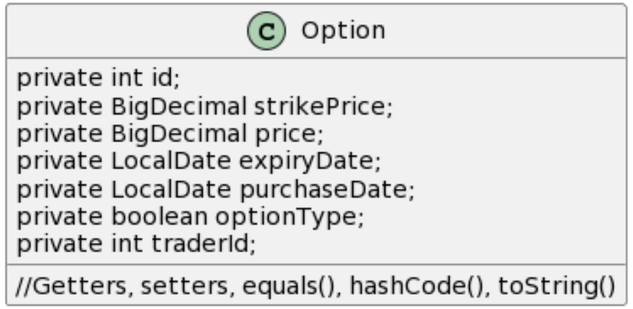

trader
id INT
firstName VARCHAR(30)
lastName VARCHAR(50)
balance DECIMAL(15,2)
username VARCHAR(50)
password VARCHAR(50)
Indexes
optionOrder
id INT
strikePrice DECIMAL(15,2)
expiryDate DATE
purchaseDate DATE
price DECIMAL(15,2)
optionType TINYINT(1)
traderID INT
Indexes
stockPrice
stockDate DATE
stockPrice DECIMAL(15,2)
Indexes

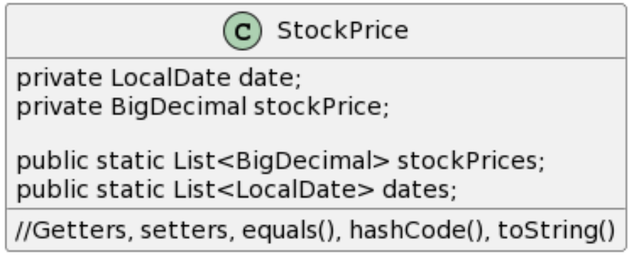
UML

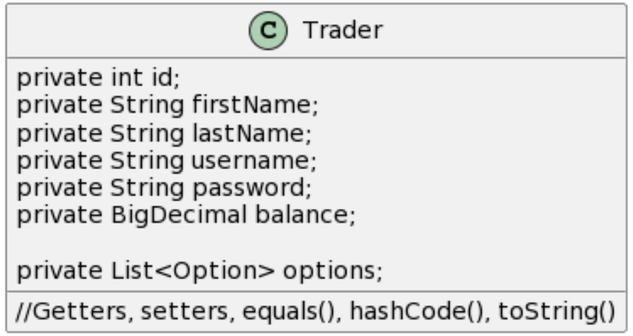

Option
private int id;
private BigDecimal strikePrice;
private BigDecimal price;
private LocalDate expiryDate;
private LocalDate purchaseDate;
private boolean optionType;
private int traderId;
//Getters, setters, equals(), hashCode(), toString()
StockPrice
private LocalDate date;
private BigDecimal stockPrice;
public static List<BigDecimal> stockPrices;
public static List<LocalDate> dates;
//Getters, setters, equals(), hashCode(), toString()
OptionController
@Autowired
OptionServiceInterface optionService;
//No edits allowed
Set<ConstraintViolation<Option>> violations = new HashSet<>();
@GetMapping("options")
public String getAllStockPrices(Model model);
@PostMapping("addOption")
public String addOption(HttpServletRequest request);
@GetMapping("deleteOption")
public String deleteOption(Integer id);
StockPriceController
@Autowired
StockPriceServiceInterface stockPriceService;
//No edits allowed, no validation needed, no postmapping
@GetMapping("stockprices")
public String getAllStockPrices(Model model);
TraderController
@Autowired
TraderServiceInterface traderService;
Set<ConstraintViolation<Trader>> violations = new HashSet<>();
@GetMapping("traders")
public String getAllTraders(Model model);
@PostMapping("addTrader")
public String addTrader(HttpServletRequest request);
@GetMapping("deleteTrader")
public String deleteTrader(Integer id);
@GetMapping("editTrader")
public String editTrader(Integer id, Model model);
@PostMapping("editCourse")
public String performEditTrader(@Valid Trader trader, BindingResult result, HttpServletRequest request, Model model);
Trader
private int id;
private String firstName;
private String lastName;
private String username;
private String password;
private BigDecimal balance;
private List<Option> options;
//Getters, setters, equals(), hashCode(), toString()
BigDecimalUtil
etc.
exp();
cdf();
ln();
sqrt();
OptionServiceInterface
List<Option> getOptionsById(int id);
List<Option> getAllOptions();
Option addOption(Option option);
void updateOption(Option option);
void deleteOptionById(int id);
void calculateOptionPrice(Option option);
List<LocalDate> possibleExpiryDates();
StockPriceServiceInterface
StockPrice getStockPriceByDate(LocalDate date);
List<StockPrice> getAllStockPrices();
StockPrice addStockPrice(StockPrice stockPrice);
void updateStockPrice(StockPrice stockPrice);
void deleteStockPriceByDate(LocalDate date);
List<StockPrice> getAllStockPricesToToday();
TraderServiceInterface
Trader getTraderById(int id);
List<Trader> getAllTraders();
Trader addTrader(Trader trader);
void updateTrader(Trader trader);
void deleteTraderById(int id);
List<Option> getOptionsForTrader(Trader trader);
OptionDao
List<Option> getOptionsById(int id);
List<Option> getAllOptions();
Option addOption(Option option);
void updateOption(Option option);
void deleteOptionById(int id);
OptionServiceImpl
@Autowired
OptionDao optionDao;
@Autowired
StockPriceDao stockPriceDao;
public BigDecimal calculateTimePeriod(Option option);
public void calculateOptionPrice(Option option);
StockPriceDao
StockPrice getStockPriceByDate(LocalDate date);
List<StockPrice> getAllStockPrices();
StockPrice addStockPrice(StockPrice stockPrice);
void updateStockPrice(StockPrice stockPrice);
void deleteStockPriceByDate(LocalDate date);
StockPriceServiceImpl
@Autowired
StockPriceDao stockPriceDao;
/Overridden methods/
TraderDao
Trader getTraderById(int id);
List<Trader> getAllTraders();
Trader addTrader(Trader trader);
void updateTrader(Trader trader);
void deleteTraderById(int id);
List<Option> getOptionsForTrader(Trader trader); //Extra method to get all Options traded by that trader
TraderServiceImpl
@Autowired
TraderDao traderDao;
/Overridden methods/
OptionDaoDB
/Overridden methods/
public static final class OptionMapper implements RowMapper<Option>;
StockPriceDaoDB
/Overridden methods/
public static final class StockPriceMapper implements RowMapper<StockPrice>;
TraderDaoDB
/Overridden methods/
public static final class TraderMapper implements RowMapper<Trader>;

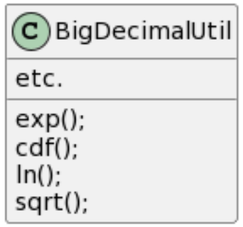
UML

Classes


Option
private int id;
private BigDecimal strikePrice;
private BigDecimal price;
private LocalDate expiryDate;
private LocalDate purchaseDate;
private boolean optionType;
private int traderId;
//Getters, setters, equals(), hashCode(), toString()

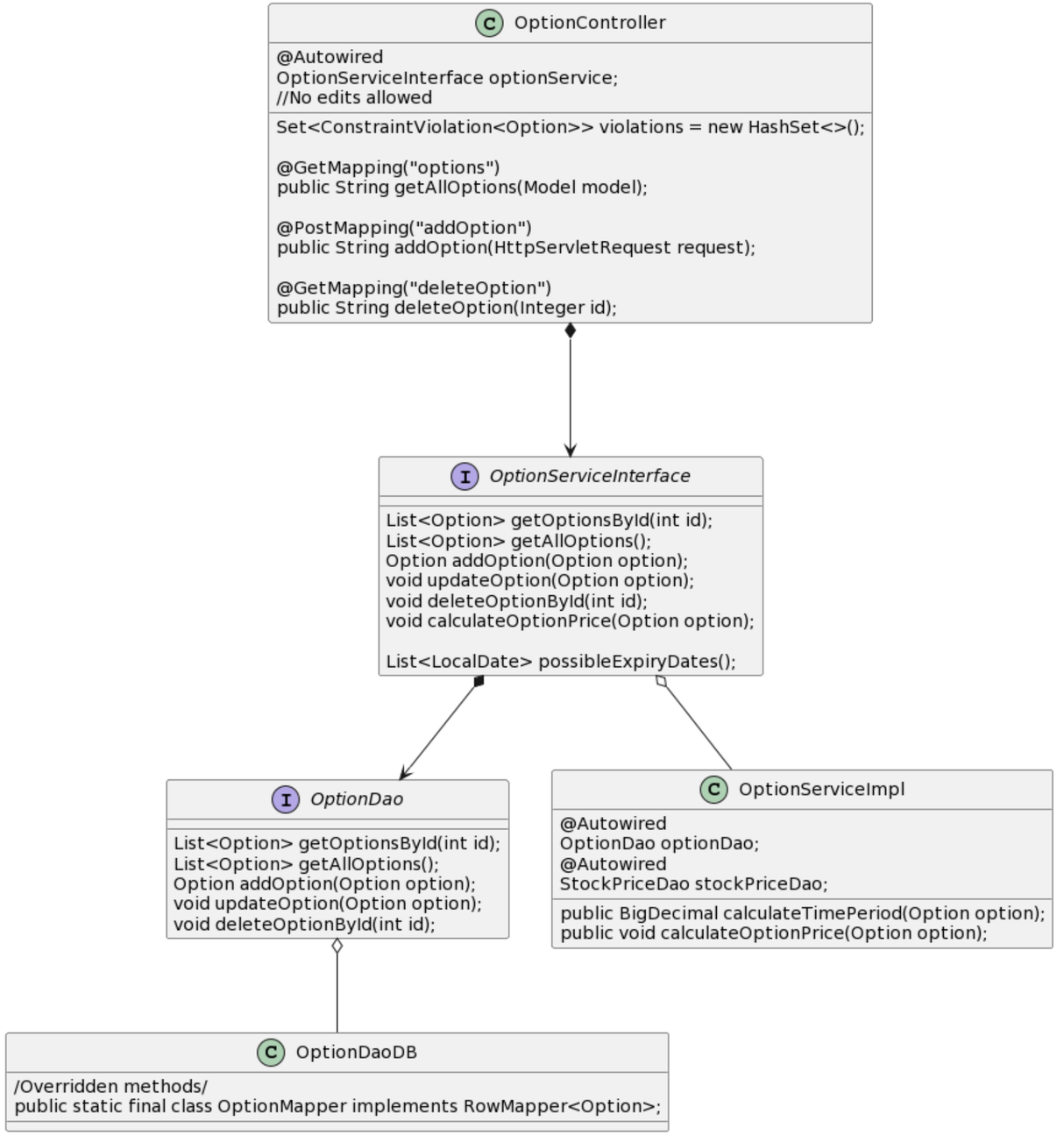

StockPrice
private LocalDate date;
private BigDecimal stockPrice;
public static List<BigDecimal> stockPrices;
public static List<LocalDate> dates;
//Getters, setters, equals(), hashCode(), toString()


Trader
private int id;
private String firstName;
private String lastName;
private String username;
private String password;
private BigDecimal balance;
private List<Option> options;
//Getters, setters, equals(), hashCode(), toString()


BigDecimalUtil
etc.
exp();
cdf();
ln();
sqrt();

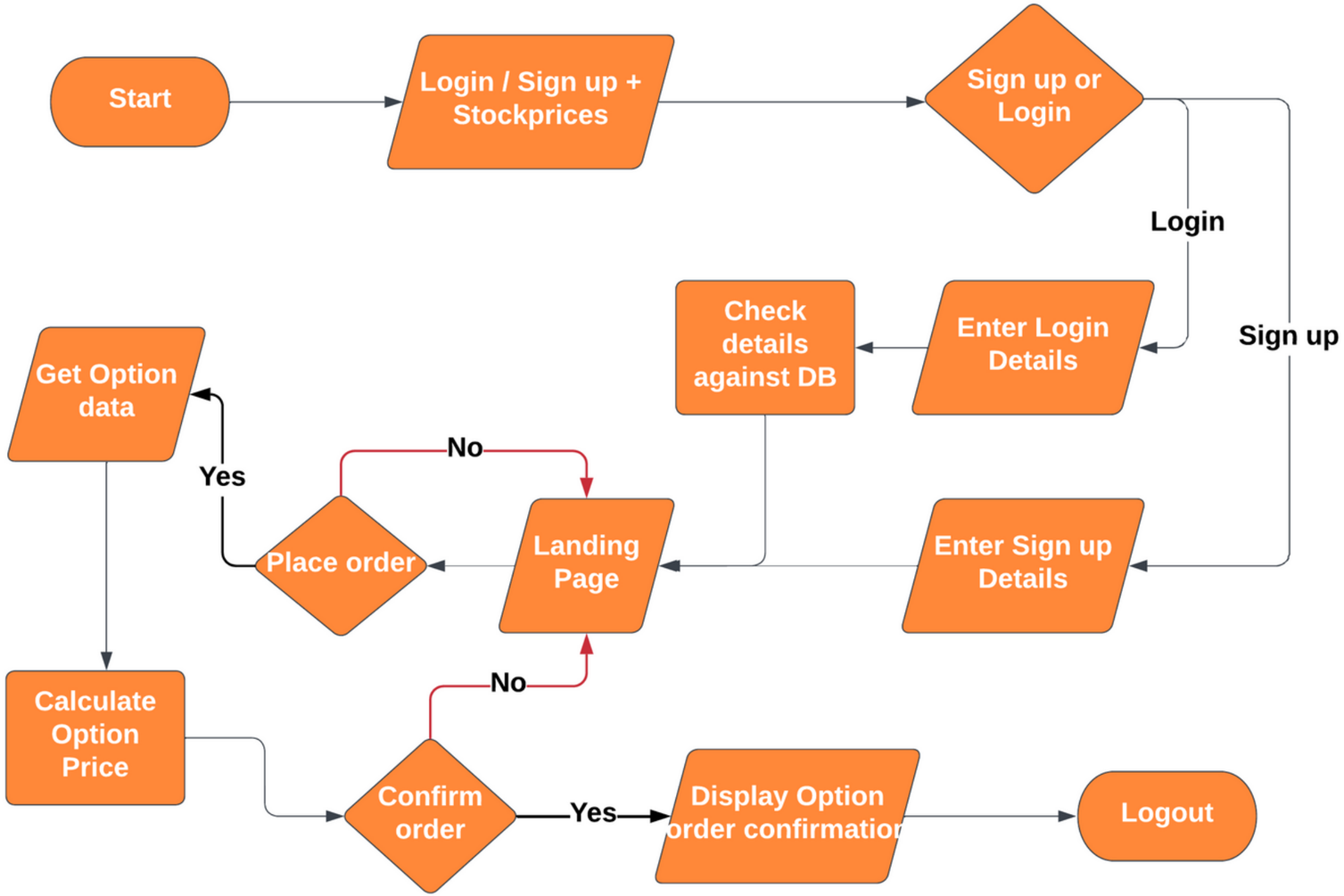
UML

Controllers



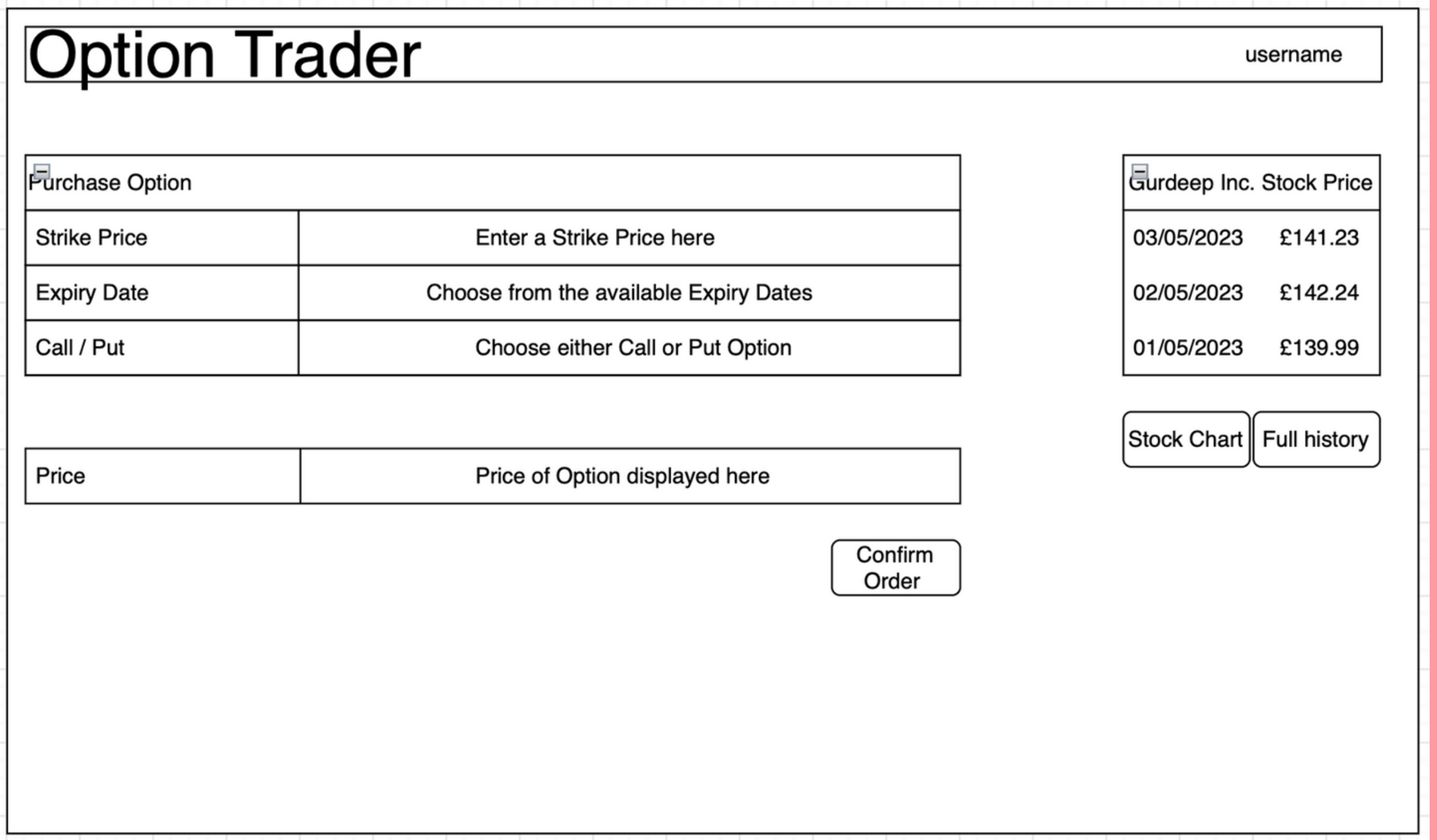
Flowchart

Option buying:



Wireframe

Option buying:



Technology Stack

Backend: Java, Maven, Spring Boot, JDBCTemplate, C++

Frontend: HTML, CSS, JavaScript, Thymeleaf

Database: MySQL

VCS: Git, Github

Live walkthrough time

Conclusions

Problems:

- BigDecimal and accuracy of mathematical functions
- Spring Boot Security implementation

Things I could do differently next time:

- Learn some more Frontend technologies/ languages
- Branding

Further work:

- Selling options
- Real stocks with live prices, not daily like Gurdeep Inc.

Thank you for listening

Any questions?