****

**MOUNT KENYA UNIVERSITY**

**SCHOOL OF COMPUTING & INFORMATICS**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**LIGHTNING TRADING PLATFORM.**

**STANLEY ANDREW KINYUA**

**BBIT / 2019 / 53703**

This project is submitted in partial fulfilment of requirement for the Mount Kenya University award of BACHELOR OF BUSINESS INFORMATION TECHNOLOGY

**DECLARATION**

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at Mount Kenya University

Signature : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID No. : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**SUPERVISOR**

I the undersigned do hereby certify that this is a true report for the project undertaken by the above-named student under my supervision and that it has been submitted to Mount Kenya University with my approval

Signature…………………………………………. Date……………………………….

**TABLE OF CONTENTS**

[**CHAPTER 1** **: INTRODUCTION** 5](#_Toc194063736)

[**1.1** **Background Study** 5](#_Toc194063737)

[**1.2**  **Problem Statement** 6](#_Toc194063738)

[**1.3** **Proposed Solution**. 7](#_Toc194063739)

[**1.4** **Project Objectives.** 8](#_Toc194063740)

[**1.5** **Project Scope.** 9](#_Toc194063741)

[**1.6** **Project Justification.** 10](#_Toc194063742)

[**1.7** **Project Risk and Mitigation.** 10](#_Toc194063743)

[**1.8**  **Hardware & Software Tech Requirements.** 11](#_Toc194063744)

[**1.8.1**  **Hardware Requirements** 11](#_Toc194063745)

[**1.8.2** **Software Requirements.** 12](#_Toc194063746)

[**1.9** **Proposed Project Budget** 13](#_Toc194063747)

[**1.10** **Project Time Scheduling** 14](#_Toc194063748)

[**CHAPTER 2:**  **LITERATURE REVIEW** 15](#_Toc194063749)

[2.1 INTRODUCTION. 15](#_Toc194063750)

[**2.2** **SYSTEM REVIEW.** 15](#_Toc194063751)

[**Spotware c-trader** 15](#_Toc194063752)

[**J FOREX4** 18](#_Toc194063753)

[**2.3** **SYSTEM ADAPTATION.** 19](#_Toc194063754)

[**CHAPTER 3:**  **METHODOLOGY.** 20](#_Toc194063755)

[**3.1** **Introduction** 20](#_Toc194063756)

[**3.2** **Waterfall Development Model** 20](#_Toc194063757)

[**3.3** **Target Population** 21](#_Toc194063758)

[**3.4** **Data Collection** 21](#_Toc194063759)

[**3.4.1** **Primary Data Collection** 21](#_Toc194063760)

[**3.4.2** **Secondary Data Collection.** 23](#_Toc194063761)

[**3.5** **Data Analysis Method** 23](#_Toc194063762)

[**3.6** **Design and Implementation Tools and Techniques** 24](#_Toc194063763)

[**3.7 System Testing.** 25](#_Toc194063764)

[**CHAPTER 4:**  **SYSTEM ANALYSIS AND REQUIREMENTS MODELLING** 27](#_Toc194063765)

[**4.1** **Use Case Diagram of the Current System** 27](#_Toc194063766)

[**4.4 Requirements definition and modelling of the current system** 28](#_Toc194063767)

[**4.5** **Requirement definitions and specifications of the current project** 29](#_Toc194063768)

[**CHAPTER 5:** **SYSTEM DESIGN** 32](#_Toc194063769)

[**5.1** **System Design** 32](#_Toc194063770)

[***Architecture: Multi – Tier Architecture.*** 32](#_Toc194063771)

[***Key System Components*** 32](#_Toc194063772)

[***Technologies*** 33](#_Toc194063773)

[**5.2** **Database Design** 33](#_Toc194063774)

[***Conceptual Database Design.*** 33](#_Toc194063775)

[***Logical Database Design.*** 34](#_Toc194063776)

[***Physical Database Model*** 37](#_Toc194063777)

[**Entity Relationship for DB** 38](#_Toc194063778)

[**CHAPTER 6: SYSTEM IMPLEMENTATION** 39](#_Toc194063779)

[**6.1 Tools used for coding and testing** 39](#_Toc194063780)

[**Coding Tools** 39](#_Toc194063781)

[**Testing Tools** 39](#_Toc194063782)

[**6.2 System test plan** 40](#_Toc194063783)

[**6.3 CHANGE OVER TECHNIQUES** 42](#_Toc194063784)

[**CHAPTER 7 : LIMITATIONS, CONCLUSIONS AND RECOMMENDATIONS** 43](#_Toc194063785)

[**7.1 Limitations** 43](#_Toc194063786)

[*7.1.1 Technical challenges* 43](#_Toc194063787)

[**7.2 Conclusions** 44](#_Toc194063788)

[**7.3 Recommendations** 44](#_Toc194063789)

[**REFERENCES.** 45](#_Toc194063790)

[**APPENDIX** 46](#_Toc194063791)

[**1.** **Organizational Structure** 46](#_Toc194063792)

[**2.** **Documents Reviewed** 48](#_Toc194063793)

[**3.** **Sample codes** 52](#_Toc194063794)

[**4.** **Technical Guide and User Manual** 56](#_Toc194063795)

[**User Manual** 56](#_Toc194063796)

[**Technical Guide** 58](#_Toc194063797)

**TABLE OF FIGURES**

[Figure 1 - cTrader Login Page 15](#_Toc194063047)

[Figure 2 - cTrader Trading Interface 16](#_Toc194063048)

[Figure 3 - J Forex Trading Interface 18](#_Toc194063049)

[Figure 4 - Waterfall model of development 20](#_Toc194063050)

[Figure 5 : Database Engine Screenshot 52](#_Toc194063051)

[Figure 6 : Kraken Class - Screenshot 53](#_Toc194063052)

[Figure 7 : Login Window Class - Screenshot 54](#_Toc194063053)

[Figure 8 : Chart Panel Class - Screenshot 55](#_Toc194063054)

# **CHAPTER 1 : INTRODUCTION**

## **1.1 Background Study**

According to the Bank of International Settlements (BIS) in 2023, the market cap of all crypto-currencies and currencies is estimated to be at about $84 Trillion. Every day, more than $5 Trillion changes hands in the Currency markets and another $300 Billion in the crypto -currency markets. Various stakeholders take part in the trading of currencies and crypto-currencies among them are Brokers, who are the firms that own and operate the software that allows exchanges to function, the Large global banks who act as liquidity providers that buy and sell large chunks of currencies and crypto-currencies so as to fulfil customer demand in their branches as well as Traders who are the individuals engaged in the active buying and selling of currencies and crypto currencies on their behalf or on the behalf of other organizations. Each of these plays a vital role in the market with only the traders engaging in the market with the motive of earning a profit.

Due to advancements in technology, most of the global trading of currencies and crypto-currencies occurs on computerized exchanges that use high speed computers and low latency networks to allow for the trading to occur.

These computerized exchanges utilize matching engines, which are special pieces of software that allow buyers and sellers to be matched together in real time. To be able to access this matching engine, traders, utilize a special piece of software that is referred to as a Trading platform. The purpose of this trading platform is to be able to receive real time market data about prices and display them to the user as well as be able to send orders to buy and sell whichever financial instrument is being trade at a specific price.

There are a variety of trading platforms such as CTrader developed by Spotware Systems, J Forex by Dukascopy Gmbh, Meta Trader by Meta Quotes as well as Trader Workstation by Interactive brokers. This trading platforms offer services such as managing orders and open positions, charting of market data in real time, displaying market news, transmitting buy and sell orders to the matching engine among others.

For any sort of trading to occur, a trader must have access to a trading platform because there is no way for them to be able to communicate with the matching engine without a trading platform. Historically, most of these trading platforms are the custom creations of companies that run these financial exchange matching engines and as such they are restricted for use only within the products offered by that organization. A clear example of this is the J Forex Trading platform, which can only be accessed by the Customers of Dukascopy. This means that traders are forced to utilize the matching engine of a certain broker even though it may not be the best. For Traders who want to access a different matching engine or multiple matching engines at the same time, there are few if any choices that are both cost effective and readily accessible.

Walter and Reeds is a proprietary trading firm that is located in Nairobi, that conducts its trading activities online. It is seeking a custom system that would allow its traders to trade both currencies and crypto-currencies across multiple exchange accounts as well as maintain proper records for all of its traders who are located all over the country. The reason that they are seeking this system is because previous experience with other platforms has allowed the company to identify a host of issues that hampers its traders from performing to the best of their ability.

A custom trading platform would allow the to be able to connect with multiple Currency matching engines as well as multiple crypto – currency exchanges thus allowing them to trade seamlessly across all these venues. It is also important that their custom platform is able to aggregate data into reports that allows them to monitor progress and performance across all their accounts.

Successful implementation of this platform would lead to the better performance of their traders as result of being able to buy and sell at convenient times as well as a reduction in the amount of funds being lost.

## **1.2 Problem Statement**

* Financial markets generate a lot of information in a very short time thus special software is needed for services such as Automated charting, plotting time series and indicators.
* Traders may generate profits as well as perform trades that may lead to accrual of taxable funds and thus it is essential to store information on the number and type of trades made as well as profits earned so that the correct amount of money can be paid in taxes
* Traders want to be able to access their trading information offline without having to connect to an online server, this is especially important because many traders live in locations with poor internet connectivity as well as in different time zones
* It is important to have tools in place that can automatically manage the risk on trader’s open positions as well as close positions automatically before they move too far into losses. Risk adverse market conditions caused by sudden positive or negative news, and can lead to large losses.
* It is essential for traders to be able to create, manage Trading orders across various execution venues and liquidity providers. Orders dictate how positions will be entered and exited thus without a manner to efficiently manage them; they lead to a situation where positions may be entered at unfavorable rates harming profitability.
* Position tracking and Portfolio performance tracking and evaluation are important too because they allow traders to be able to identify poorly performing positions and thus close them out before they can grow too large
* Traders also require tools that would allow them to monitor their trading costs which come in the form of Commissions and fees which if left unchecked lead to poor performance as a result of trading fees eating up all of trader’s profits. It is thus necessary to monitor and optimize trading costs especially in the long run.
* Another simple but highly effective tool is a Trader’s Journal that displays things like market news in real time as well as managing a calendar that tracks important market dates such as holidays, important announcements like dividend dates etc. This is important in that it allows traders to plan and adjust their trading styles well before hand.
* Walter reeds is seeking a complete package that would allow its traders to be able to access markets, parse information and manage knowledge all from one platform, all while allowing it to operate locally such that it may be able to operate even offline for purposes such as historical data analysis needing only an internet connection for charting and executing on positions

## **1.3 Proposed Solution**.

The development of the system combines the usage of JAVA, MySQL, APACHE HTTP libraries as well as SSL and various APIs from different providers to offer all the needed services

The Trader workstation is expected to have the following features:

1. Automated chart – plotting module that takes up data from different data feeds and plots real-time market data as well as computes time series and various indicators in real-time.
2. Execution capabilities across different venues and exchanges. It allows for the opening of positions in different exchanges and monitoring and managing them as if they were a singular position.
3. Storage of historical market data that allows traders to view historical market prices even offline.
4. Automated risk management checks that study market conditions and protect trader’s capital from adverse market conditions that would lead to loss.
5. A news feed that allows for traders to view important news that are moving the market as they come in.
6. A module that computes the performance of a trader’s portfolio in real time.
7. Database that stores all the information as it is being generated so that the information can be utilized at a later date.
8. View the trading costs associated with their trading activities.

## **1.4 Project Objectives.**

The main objective of the project is the design, documentation and successful implementation of an efficient Trader WorkStation that will allow traders to effectively participate in financial markets and improve their chances of success in both the long and short term.

The Secondary objectives are:

* To determine the ways in which the Trader workstation will increase the performance of a Trader in the markets.
* To determine the factors that lead to Failure that are caused by poor system design and attempt to mitigate them.
* To identify the ways in which Trader’s use trading software and attempt to come up with a design that is both user friendly while also not sacrificing performance and features.

## **1.5 Project Scope.**

The Scope of the Lightning Trading platform will allow us to keep on track during the development as well as allowing us to decide on the most important aspects of the development of the software.

1. The application is expected to be a desktop application because it is needed to work even offline thus it would not be able to work offline with a web-based app.
2. It was decided that we would develop the Application using Java and MySQL because of their relative simplicity as well as low cost. This would also reduce the number of tools needed during the development.
3. It is expected that we will implement an order management system to be able to keep track of all orders.
4. It is necessary to have a live feed of the international market news that allows traders to see what is occurring globally right from the platform without needing to open a browser.
5. It is essential for the application to be able to plot a variety of charts so that users may be able to track how the markets are performing globally.

After careful analysis, it was decided that it would not be possible to implement some features because they may require expensive licenses as well as hardware tools that are difficult to acquire. Some of these things are:

1. Multi – Factor Authentication – After analysis, we found that it would be difficult implementing multi – factor authentication without expensive licenses thus we decided to abandon that approach and instead use a Password as well as Secure System Pin for security.
2. A popular way to connect to Trading Brokerages is the FIX (Financial Information Exchange) Protocol because of the extreme complexity required to set up a Working FIX ENGINE to parse messages thus we decide to not utilize this and instead use the vendor supplied APIs because they are quite simple to implement and we can get right into development.
3. Multiple Themes – It was found that most users, rarely use the light mode themes and instead prefer dark themes because they are easy on the eyes. As a result, we do not need to implement multiple themes because we will implement only a dark theme.

## **1.6 Project Justification.**

* With the advent of working from home, such a project would lead to the development and realization of software that would help millions of people globally to be able to access global markets from the comfort of their homes @ a reasonable price.
* The project poses an interesting challenge because a lot of the important bits such as charting would have to be written from scratch to optimize for performance as well as include features that are not usually common in charting packages such as depth of market. Solving these issues would work to give the learner an even better understanding of the programming language and tools implemented
* With the continuing advancement of Crypto-Currencies and digital money, it is very timely project that would allow traders to be able to trade and interact with crypto – currencies from a single platform.

## **1.7 Project Risk and Mitigation.**

**RISK**

1. Financial constraints due to economic downturn caused by the weakening of the Kenyan shilling may cause the Organization to shift Funds to other Areas thus starving the project and leading to the premature death of the project.
2. Scope Creep, which is where the requirements and the feature list of the application keep changing leading to development constantly halting and restarting thus leading to delays that always push the release date forward.
3. Poor internet connectivity can hamper the development process and make it to take longer.
4. Client can change their mind mid-way through the development lifecycle and halt the project leading to an unfavorable situation where the project is not completed and thus the developer has not been paid.

**MITIGATION.**

1. Enough finances should be set aside for the project before development has begun so as to avoid future hiccups and the smooth development and completion of the project.
2. A list of features and expected functions should be clearly listed and known prior to the start of development. Any new features that may be needed or identified during the course of development will be added at a later date after the completion of the system.
3. A legally – binding contract should be signed as well as part of the development costs paid before the beginning of the project so as to mitigate against client changing their mind mid-way and refusing to pay for completion of the project.
4. Seeking and alternate internet provider will allow for a much stable network connection during the development process thus allowing for development to proceed smoothly

## **1.8 Hardware & Software Tech Requirements.**

### **1.8.1 Hardware Requirements**

|  |  |
| --- | --- |
| Hardware | Requirement |
| Laptop | Intel Core i5, 8GB ram, 500gb SSD |
|  |  |
|  |  |

Table 1 - Sample Hardware Requirements

### **1.8.2 Software Requirements.**

|  |  |
| --- | --- |
| Software | Minimum Version |
| Operating System | Windows 10 |
|  |  |
| Database | MySQL version 8.0 |
| Database Administration | MySQL Workbench Version 8.0 |
|  |  |
| Java Virtual Machine | JDK 20 |
|  |  |
| Maven | Maven Version 3.9.4 |
|  |  |
| JAVA IDEs | IntelliJ Idea Community version 2023.1.2 |
|  |  |
| JAVA Testing Tools | Visual VM version 2.15 |
|  | Rest Assured Version 5.3.2 |
|  | JUNIT version 5 |
|  |  |
| Version Control Tools | GIT version 2.4.3 |
|  | GIT Kraken Free – Latest Available Version |
|  |  |
| BROKER APIs | Dukascopy SDK - Latest Version at time |
|  | Binance Rest Api - Latest Version at time |
|  | OKX Api - Latest Version at time |
|  |  |
|  |  |
|  |  |

Table 2 - Sample Software Requirements

## **1.9 Proposed Project Budget**

|  |  |
| --- | --- |
| **ITEM DESCRIPTION** | **AMOUNT** |
| Laptop: Core i5 -8GB RAM – 750gb hdd | 45,000 |
| WI-FI Internet | 4,000 |
| Java IDEs and Profiling Tools | Free |
| Apache HTTP Library | Free |
| Maven | Free |
| Windows 10 | Kshs. 10,000 |
| MySQL Server | Free |
| GitHub Hosting | Free |
| Git and GitKraken Version Control Software | Free |
| TOTAL | Kshs 59,000 |

Table 3 - Proposed Budget Table

## **1.10 Project Time Scheduling**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ACTIVITY | JANUARY 2024 | FEBRUARY 2024 | MARCH 2024 | APRIL 2024 | MAY  2024 | JUNE 2024 |
| Problem Identification, Requirements Collection and Analysis |  |  |  |  |  |  |
| System analysis, Use Case Diagrams, UML, Class and Sequence Diagrams |  |  |  |  |  |  |
| System Development using Java, MySQL, Apache HTTP, SSL |  |  |  |  |  |  |
| System Testing and Debugging |  |  |  |  |  |  |
| System Implementation and User Training |  |  |  |  |  |  |
| System Review, Maintenance and Improvement |  |  |  |  |  |  |

Table 4 - Project Time Schedule Table

# **CHAPTER 2: LITERATURE REVIEW**

2.1 INTRODUCTION.

This chapter details the literature of other scholars on the problems and issues encountered when developing custom trading Software.

The aim of this study is to gain additional information by the analysis already existing bodies of work and as such seeking to learn what makes a good trading system as well as pitfalls to avoid while at the same time seeking to improve on the design and adding more useful features.

Here I attempt to critique my findings in as neutral a manner as possible while also seeking to learn as much as possible so that I can develop an even better trading platform.

## **2.2 SYSTEM REVIEW.**

### **Spotware c-trader**

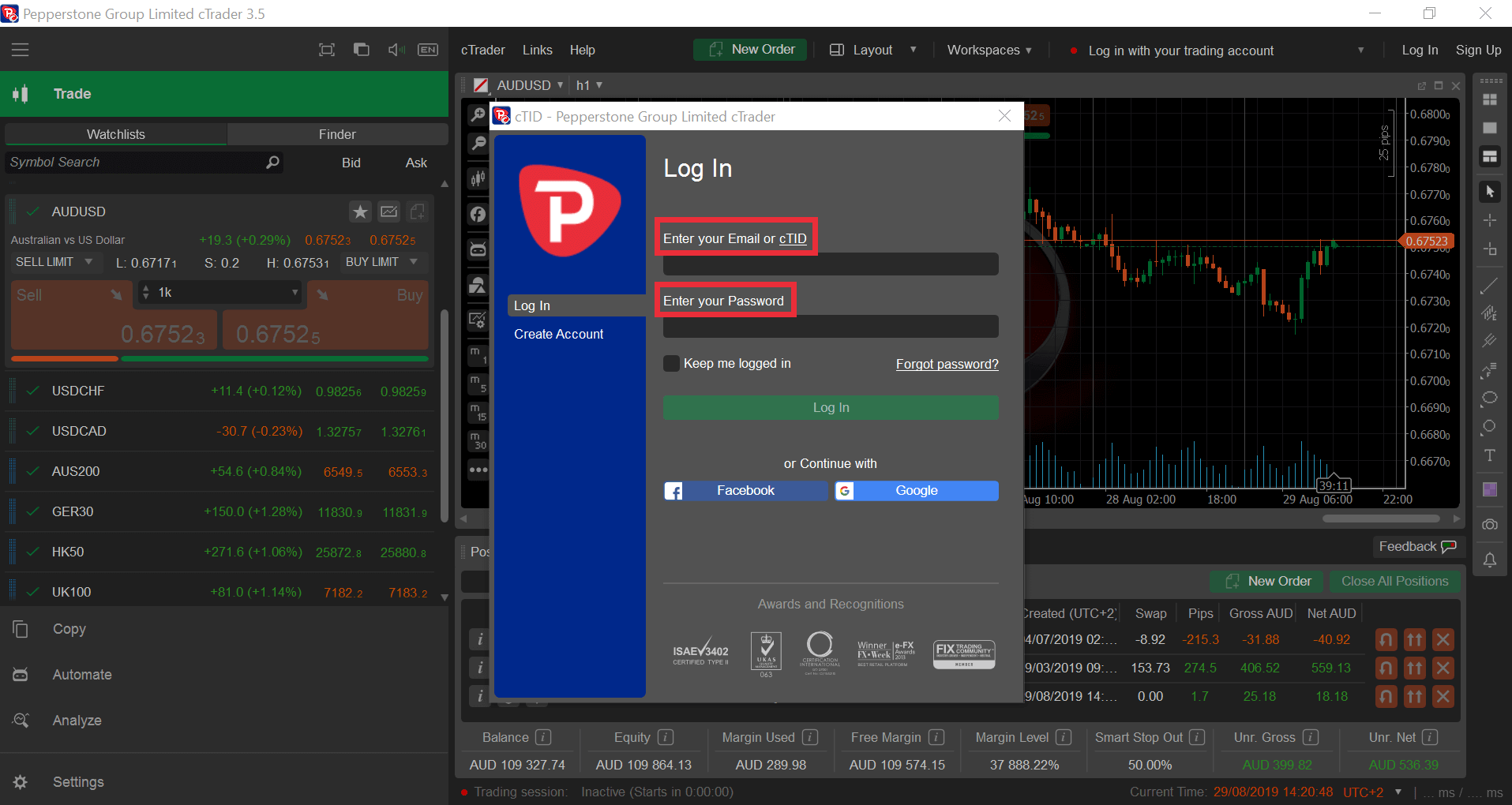


Figure 1 - cTrader Login Page

cTrader is a trading platform that was developed by Spotware Systems. It is used to trade all manner financial instruments ranging from currencies, commodities, indices and even crypto-currencies. It is developed using the C# programming library as well as the Windows Presentation Foundation (WPF) for its user interface.

cTrader is a very well-designed trading platform that makes it easy to carry out trading operations. Its user interface is designed to allow you to be able to access all operations with as few clicks as possible.

Despite being a very well-developed software package, it is faced by a glaring problem in that it is designed primarily to be integrated with brokerage companies and not with traders. This means that in order for a trader to use the program, they need to have a brokerage account with the broker that offers it.

Being broker- neutral is one of the core reasons why I want to build this platform because it makes it easy to trade with whichever brokerage firm that a trader chooses to.

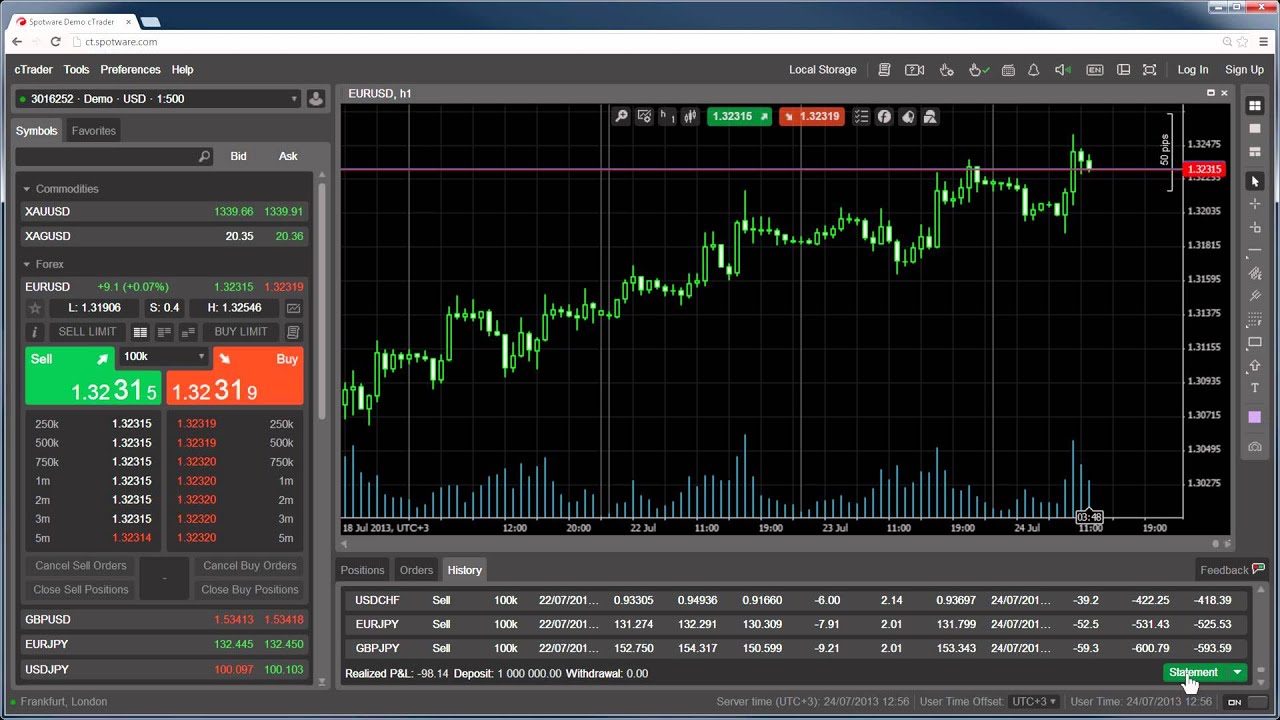


Figure 2 - cTrader Trading Interface

By picking all the good things that cTrader has to offer and making it broker neutral as well as able to cache data offline, I will be able to provide a product that meets the demands of the clients wholly while also offering some features that are not found inside cTrader such as the ability to connect to crypto – currency exchanges.\

**C TRADER PROS.**

1. The user interface is very clean and simple thus it reduces the complexity and makes it easy for traders to jump right into financial markets with just some short training.
2. It offers a variety of display options that allows even people with poor eyesight to use it such as display scaling that makes UI items larger.
3. It is very stable and rarely crashes thus it can be relied upon by serious organizations

**CONS.**

1. It is not a cost-efficient trading platform and requires large payments for use because it is primarily developed for brokers to offer to their clients instead of being software for individual clients.
2. The choice of Financial Instruments you can trade with the platform is limited to what SPOTWARE systems deems necessary, as such there is no execution for futures, Options or even crypto-currencies.
3. There is no way to access CTrader as an individual without requiring the setup of expensive broker-dealer software because it is designed mostly for brokers and not individuals thus it is sold as a complete suite of products.
4. It does not allow working offline.
5. It does not integrate with Crypto – Currency exchanges.

### **J FOREX4**

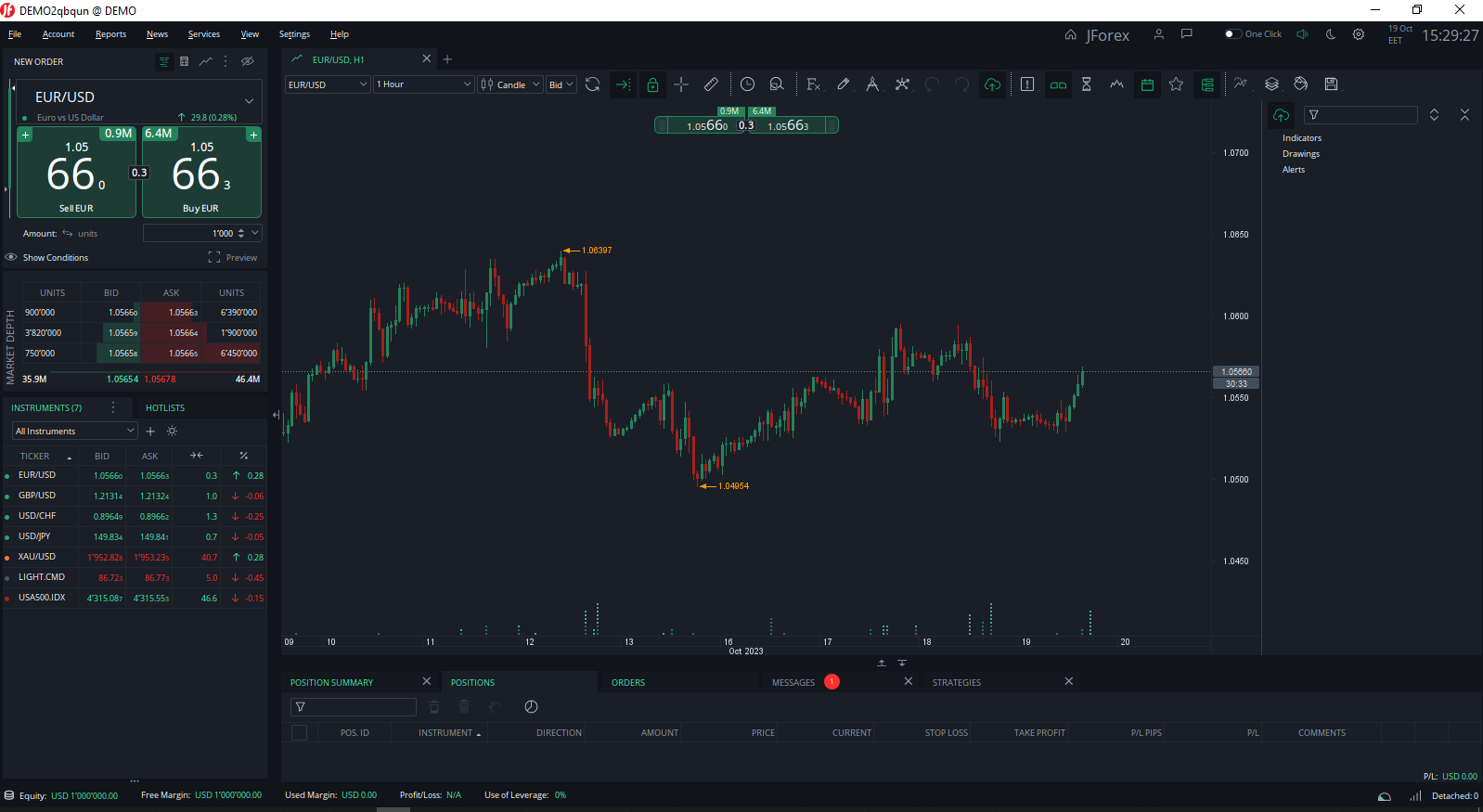


Figure 3 - J Forex Trading Interface

**J FOREX SYSTEM REVIEW**

J Forex is a Java – Based Trading platform that was developed in 2013 by DUKASCOPY gmbh, a Swiss based Brokerage Firm, for its clients. It is used to trade all the most common financial instruments such as currencies, metals, commodities etc.

It provides a much cleaner interface than cTrader while also providing a slew of features such as being able to work with market data offline as well as the capability to view exposure in multiple currencies.

The platform is one of the best trading packages out there but it is held back by being vendor – locked to it’s creator’s brokerage firm. This together with it’s lack of access to risk management tools makes it unsuitable for professional traders.

My aim is to use J Forex as a guide and acquire some of it’s useful features such as a wonderful order management system as well as exposure management tools when designing the Lightning Trading platform. I intent to remedy the short comings like lack of risk management tools so that I can implement a platform that is both capable as well as advanced enough for professional traders.

**PROS.**

1. It allows for the saving of data and manipulation of data even when there is no internet connection.
2. It is highly efficient and uses very little memory as it is running.
3. It has a simple user interface that makes it easy to work with.

**CONS.**

1. It is vendor locked to the Broker that developed it and cannot be used outside of their special execution venue.
2. It does not support usage of multiple accounts.
3. It does not support connecting to Crypto – currency exchanges.
4. It is limited to operating in a single base currency – Us Dollar thus it is not suitable for use by people who do not use the USD as their Local Currency.

## **2.3 SYSTEM ADAPTATION.**

The Lightning Trading Platform is a project whose aim is to help retail traders such as those at Walter and Reed access international markets in a cheap, effective and efficient way. It makes all operations needed to run smoothly as well as introducing automation by printing daily and weekly reports that show performance of each trader as well as various metrics such as profitability, commissions and fees paid amongst others

It includes a simplified User Interface that makes it easy to use even for traders without a background in Finance while also remaining feature rich enough to appeal to even seasoned veteran traders.

It also introduces encryption that allows user data to be stored locally e.g., Account passwords and keys in a manner that is both safe and stress free for traders. It is highly efficient and allows for the retrieval of price feeds as well as allowing a trader to execute across multiple execution venues simultaneously.

# **CHAPTER 3: METHODOLOGY.**

## **Introduction**

An incremental development approach is the most preferrable design choice because it allows both a linear and iterative development methodology. It allows for the identification of services that are the most heavily used and having them be completed earlier so that there is a functional prototype much earlier during the development lifecycle. This means that users do not need to have the entire system be fully complete before they have a chance to use it. The users are also able to use the program much earlier and get a feel for it as well as offer opinions on what they think needs to be improved. It also allows for users to gain working experience with the system even before it is fully launched thus making it easier to implement the system.

## **Waterfall Development Model**

The Waterfall model also referred to as the ***linear- sequential life cycle model*** is among the earliest process models to be introduced and it was favored for its ease in both its use as well as understanding. In a waterfall model, each phase must be fully completed before the next phase can begin. It is typically used for projects that are small and have uncertain requirements because it allows for the continuous assessment at the end of each project phase that determines whether the project is still on track or whether to make changes or even discard the project as a whole. These project phases do not overlap and are each distinct in the manner of what they comprise of and what their objectives are.

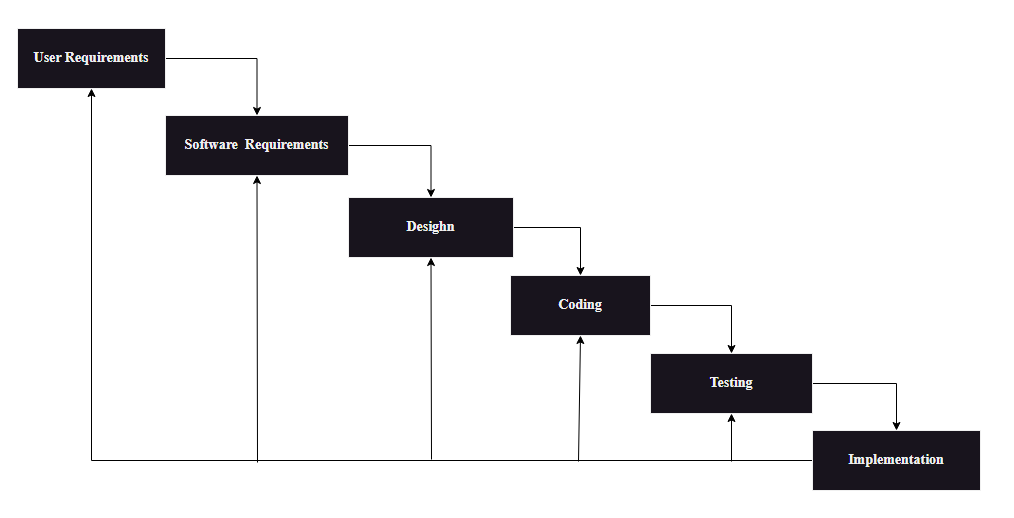


Figure 4 - Waterfall model of development

## **Target Population**

The target population is defined as an entire group of individuals, events, objects or items having common observable characteristics with information that a researcher is interested in.

The target population will be the users of the system who are primarily the traders at Walter Reed. Due to the fact that the number of traders at Walter and Reeds is small, at only 35, we can be able to use the entire group of traders as our sample. This is a significant advantage because we do not need to expend a significant amount of time and resource to obtain the information we need. This system will be tailored towards the needs and demands of the traders allowing for a more personal touch unlike common off the shelf solutions.

## **Data Collection**

This section lays out the various tools and methods that will be used during the research and analysis of the data collected. This involves deriving facts from a collection of information about the nature of business operations at the firm as well as processes that go into preparation for trading as well as actual trading. It also goes into depth about best practices as well as areas of critical importance and risk. It also outlines the nature and flow of information within the System.

There are two primary types of data collection:

* Primary data collection
* Secondary data collection

### **Primary Data Collection**

It looks into the collection of information from end-users of the system first hand.

The following tools will be used in conducting this research

* Interviews
* Questionnaires

#### **Interviews**

In this research, face to face interviews will be conducted at the company premises. It will be conducted by asking the traders a series of questions and from their responses, an analysis of the data will be made.

The following is a sample of Interview questions that will be asked to the traders.

1. *How long have you been trading Financial Instruments?*
2. *How long have you worked at Walter and Reed?*
3. *What do you feel are the weaknesses of your current trading setup?*
4. *What do you think are important features that would help you as a trader?*
5. *In your opinion, what is the most critical part of your Trading activities?*
6. *What are the risks that you encounter while in the process of trading?*
7. *What are your expectations for the upcoming system?*

#### **Questionnaires**

A Questionnaire is a series of printed questions that are handed out to people so that they can give information about a variety of issues related to the current system as well as the new expected system. The questions are usually open ended so as to provide detailed responses as well as some being closed -ended to so that they give a clear understanding on the current system as well as expectations of the new system.

Below is a sample Questionnaire:

*Tick the Answers that Apply and Answer the question*

1. *How long have you been in the Finance industry?*
   * *Less than 1 year.*
   * *1 – 2 years*
   * *More than 2 years*
2. *Are you proficient with Computers?*

* *Yes*
* *No*

1. *Do you have a personal Computer?*
   * *Yes*
   * *No*
2. *How does your typical day look like? ................................................................................................................................................................................................................................................................................................*
3. *How do you get the information that you need while you trade? ................................................................................................................................................................................................................................................................................................*
4. *What is your opinion on the effectiveness of the current system? ………………………………………………………………………………………………………………………………………………………………………………………………*

### **Secondary Data Collection.**

##### **The Internet**

The internet is also a valuable research tool and it has a vast sea of information related to trading as well as trading systems. This information comes in the form of peer- reviewed documents, news, university Journals as well as other reputable publications

##### **Books.**

Books and Printed Journals offer a vast array of information that is publicly available to all people.

## **Data Analysis Method**

The data collected in the field is meaningless and can not be used to reach any conclusion until it is transformed into a form where it can be analyzed. A number of tools is used to process and transform collected data ranging from quantitative techniques such as statistical analysis using means, averages, standard deviations and output graphical views that are easy to understand such as pie charts, graphs. A researcher can also use qualitative analysis to read and draw meaningful information from the data set.

## **Design and Implementation Tools and Techniques**

***IDE - Integrated Development Environment***

IntelliJ Idea is a popular software program that is used for writing and editing java source code. It is an extremely useful tool because it allows developers to create and edit Java Source files, perform Java Software testing as well as allow for easy debugging.

It has both a paid version and an unpaid version. For my use case, the free version will work just fine because most of the paid features are for enterprise companies an I will never need them.

***Programming language - JAVA***

Java is an open-source programming language that is used to develop both local desktop apps as well as webapps. Java runs on a Virtual Machine called the JVM that makes it portable across different operating systems.

Java bundles a bunch of features that allow me to work effectively for the development of the Trader Work station. Key among this feature is: a mature UI framework called JAVA SWING that is used to develop user interfaces, Mature HTTP libraries that can be used to integrate with and consume REST APIs, Database connection drivers in the form of the JDBC library that allows communication with different Database engines, Cryptographic Libraries that allow for things like File encryption as well as a host of other features. Java is also easy to work with and there exists a wealth of information for any kind of hurdle you may encounter during the development process.

***Database - MySQL***

MySQL is an SQL database that is both free and open source and is suitable for both small and large projects. It is available on a variety of operating systems including Linux, Unix and windows. It allows for the effective storage and manipulation of data even programmatically using language wrappers like JDBC that allows for custom apps to be built with support for MySQL.

It is used to store data easily in a manner where it can be rapidly accessed as well as in a structured format so that in order to carry out operations on the data, very little change is needed. This is important for us because we will be saving a lot of price and market data inside the database and once, we require this information we can readily get it from the database. Also being open source will allow us to keep our development costs low.

***Version Control***

***Git &GitHub***, Git is a local version control system used to manage different versions of software source files during the development process on the local machine, whereas GitHub is an online platform that allows us to upload our software repositories and back them up online to prevent loss should something happen to our local machine

***GitKraken*** – This is a piece of software that is used to manage GIT software repositories as well as make it easier to push Software repositories to GITHUB for cloud storage.

***Testing Tools***

***Visual VM & Java Mission Control*** - A visual java testing suite that allows for the testing and profiling of java applications. It is especially useful when testing software that is multi – threaded as well as software that sends or receives data over the internet.

***JUNIT*** - Junit is a Java testing tool that is used to write automated tests that can be continuously run during the development process so as to ensure that bugs are detected and fixed during the development process. It creates and manages tests that run every time a Java project is built thus identifying any bugs and fixing them before deployment

***REST Assured*** - A testing tool used to test Applications that utilize RESTful Services and APIs. It is very useful when developing applications that consume APIs because during testing, we can validate that the application is conforming to the REST standards and is bug free before shipping.

**Other tools**

***TRADING APIs -*** These are the APIs that grant as access to the matching engines of different execution venues and they perform functions such as receiving price feeds, executing trading positions, receiving account information i.e., Account Balance as well as Opening and Closing Accounts. All of the needed APIs are free and easily accessible on the Internet

## **3.7 System Testing.**

Functional testing involves testing the system against business requirements of the organization. These testing methods include:

* Unit Testing
* Integration Testing
* Validation Testing
* System Testing

Unit Testing.

Unit testing focuses on the testing of a unit, which is the smallest unit of software design. Using a detailed description of the design as a guide, important execution paths are tested so as to uncover bugs and errors within the boundary of the module and to solve them before it comes time to implement the system. A module is not a stand-alone program and thus each module has to have its own test case written.

Integration Testing

Integration testing is a type of systematic testing that attempts to construct the program structure while at the same time testing to uncover errors associated with the structure. Integration testing is an incremental approach and the success of each phase depends on the success of the previous phase.

Validation Testing.

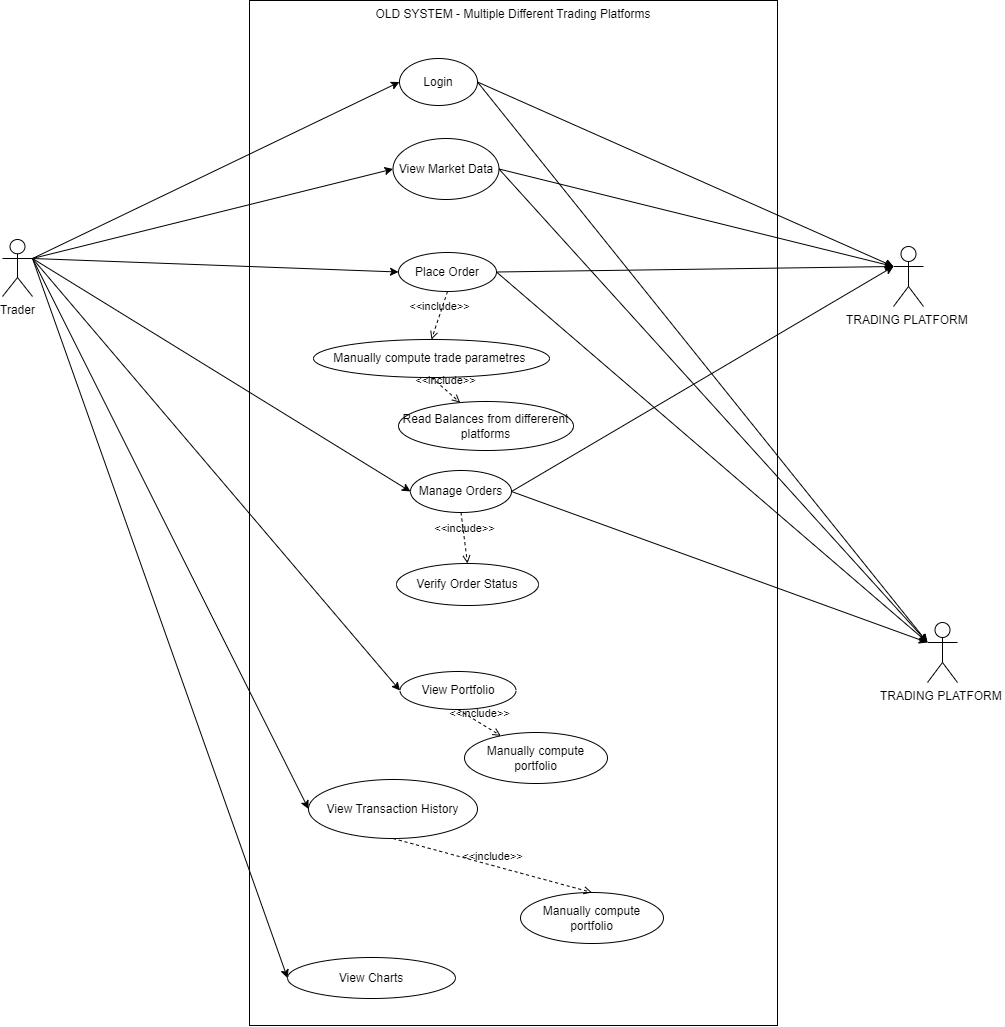
Validation testing is where a series of tests are run to ensure that the system conforms with the requirements. It consists of two phases – alpha and beta testing which are conducted with the end-users. Alpha testing is conducted in a controlled environment at the developer’s site by a customer with a developer supervising, while beta testing is conducted at one or more customer sites by the end users of the software without the developer present and in a live environment.

System Testing.

There are 3 different types of System testing. These are, Recovery testing, Stress Testing and Performance testing. Recovery testing forces the system to fail in a variety of ways so that they can verify that system recovery is properly performed. If the recovery is automatic, data recovery as well as initialization mechanisms need to be checked for correctness. Stress testing on the other had is where programs are put through abnormal conditions such as an unusual quantity or Volume of resource demand. The reason for this is to determine how stable the program is and whether it can continue to run even in periods of great stress. Performance testing on the other hand is used to test for the run time performance of the software, it is essential so as to uncover situations that lead to degraded performance and even possibly system failure.

# **CHAPTER 4: SYSTEM ANALYSIS AND REQUIREMENTS MODELLING**

### **Use Case Diagram of the Current System**



### **4.4 Requirements definition and modelling of the current system**

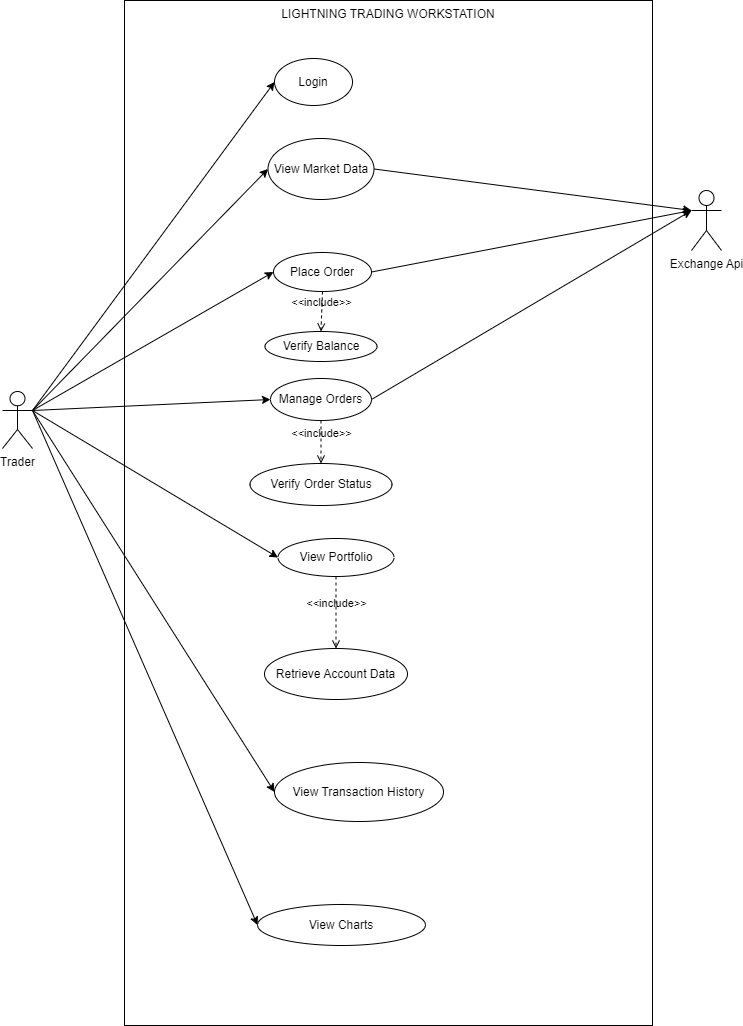
Before the current system, a set of different tools were used. These tools were a combination of spreadsheets, manual processes and limited separate trading tools.

**1. Process Mapping:**

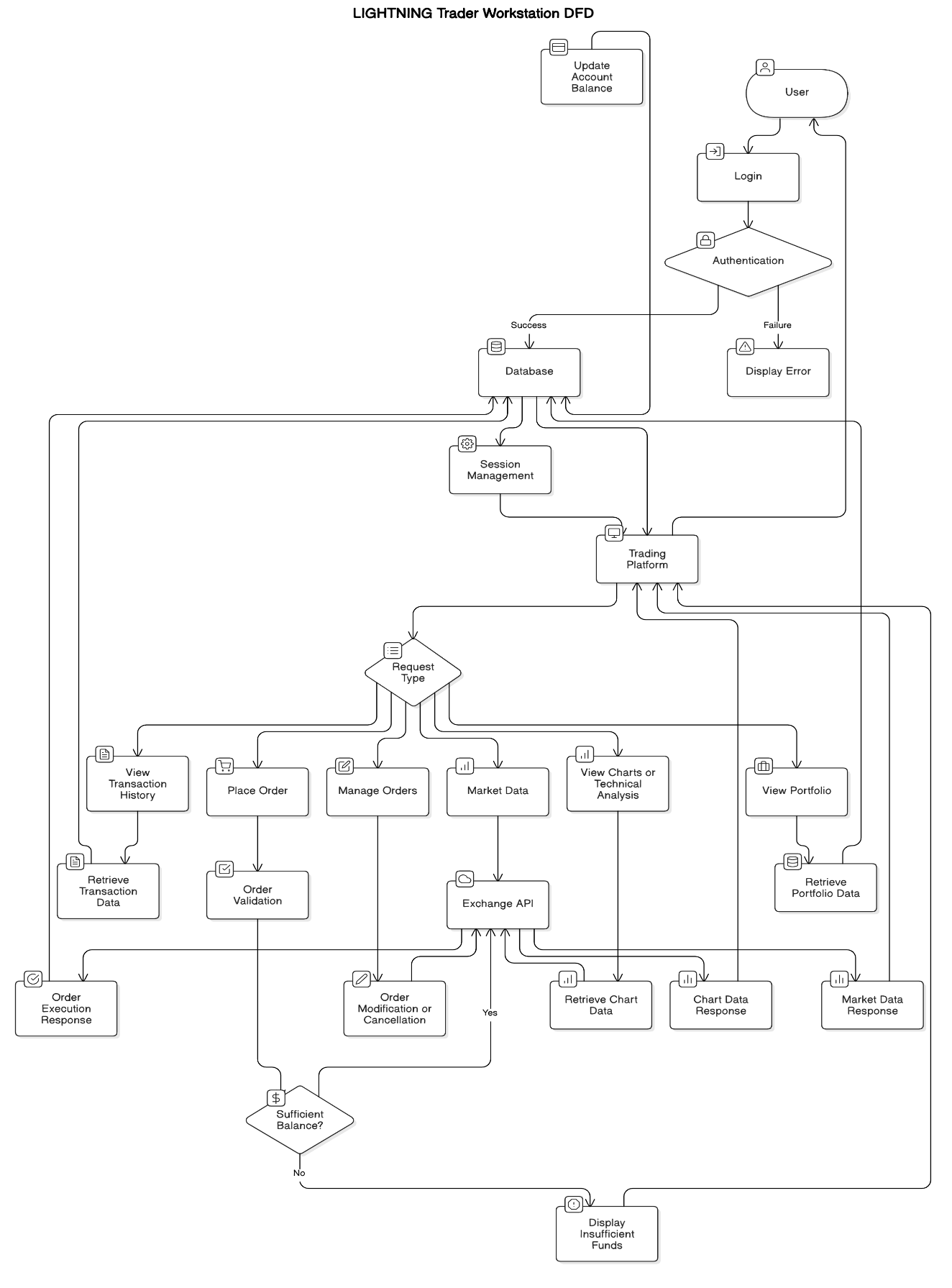
* **Order Placement:** Traders manually enter orders into individual exchange platforms or broker terminals.
* **Portfolio Tracking:** Spreadsheets are used to track positions, balances, and profits/losses.
* **Risk Management:** Manual calculations of risk exposure, often delayed and prone to errors.
* **Reporting:** Manual generation of reports, often time-consuming and inconsistent.

### **Requirement definitions and specifications of the current project**

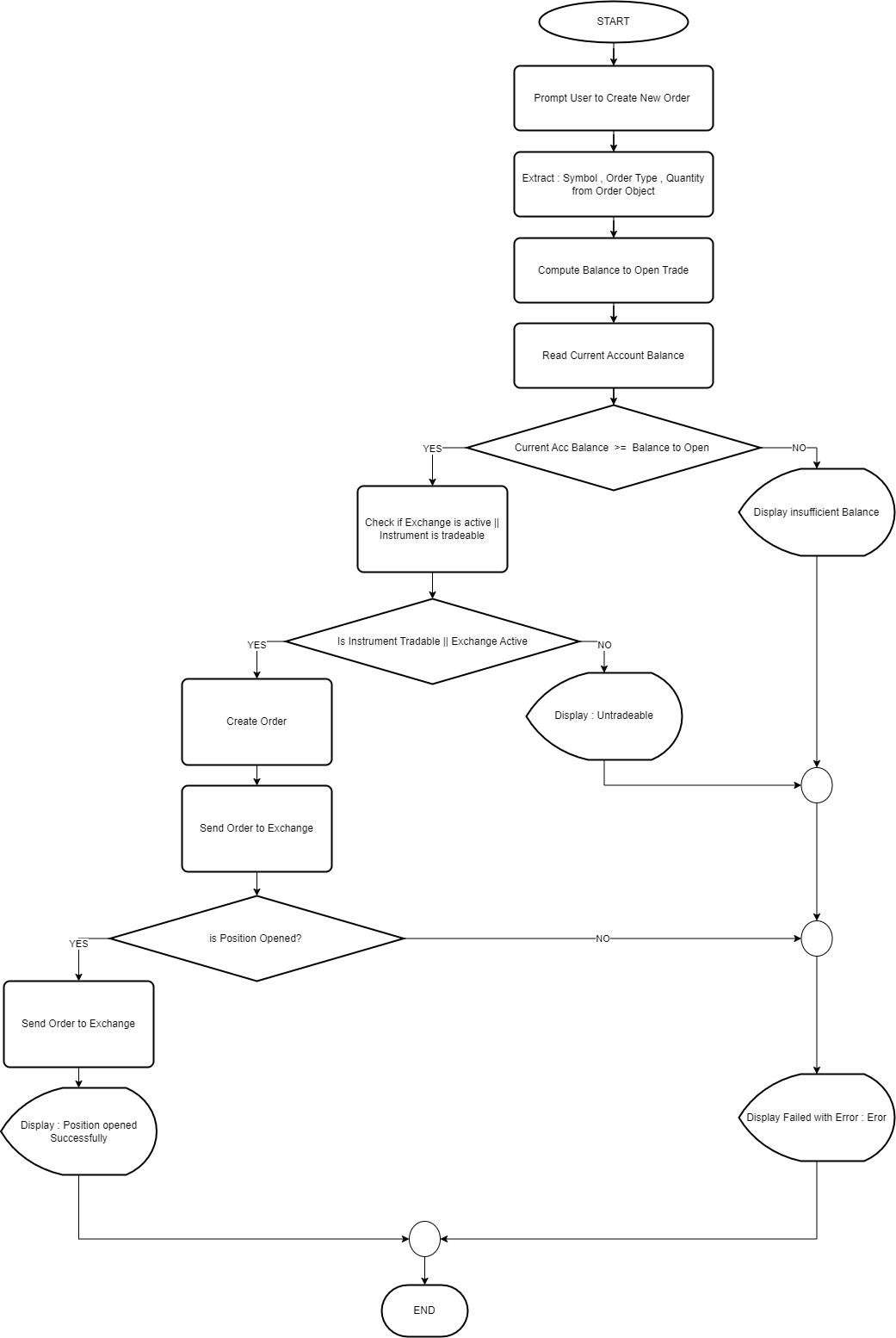
#### **4.5.1 Use Case Diagram of our Current Project**



#### **4.5.2 Data Flow Diagram of Our current Project**



#### **4.5.3 Flow Chart of Current System – Position Opening**

* 1. 

# **CHAPTER 5: SYSTEM DESIGN**

## **System Design**

### ***Architecture: Multi – Tier Architecture.***

Our system is built on a multi – tier architecture that breaks down the system in a manner that allows for the separation of concerns as well as grouping similar functions together.

Our system tiers are:

* Presentation Tier - (U.I): The user interface that a trader interacts with and allows a user to feed commands as well as view outputs. It is built on ***Java – Swing.***
* Application Tier – The application logic that includes order processing, market data handling and business rules
* Data Tier (Database): Used to Store Data transactions as well as Market Data. It is built on MySQL.

### ***Key System Components***

* User Interface
  + Allows the Display of real time market data.
  + Allows users to enter and manage orders
  + Displays Charts as well as variety of technical analysis tools
  + Allows users to perform Account management.
* Order Management System
  + Performs order validation and routing
  + Performs the execution and tracking of active orders
  + Handles risk management
* Market Data Handlers
  + Receives and stores real time data from various exchanges
  + Stores and retrieves historical data for charting as well as for research.
* Account Management Systems
  + Performs user authentication and Authorization
  + Tracks user Account balances across different exchanges
  + Manages portfolio details and displays to user
* Exchange API integration
  + Allows the system to connect to different exchanges for execution and market data.
  + Performs API requests and handles responses from Exchange Servers
* Database
  + Stores user data, market data as well as user transactions.

### ***Technologies***

* ***Java***: Used for development of the core application
* ***Swing***: Java UI library that allows for complex UIs suitable for trading applications.
* ***Flatlaf*** : Modern Swing Library that modernizes the Look and feel of java swing as well as making it easy to skin and theme applications
* ***Apache HTTP Client***: rest request and response
* ***JDBC***: Allows database Access
* ***Relational Database***: MySQL for data storage and easy retrieval

## **Database Design**

### ***Conceptual Database Design.***

* Entities
  + USERS : User accounts, credentials and profile information
  + ACCOUNTS : User account balances and portfolio
  + ORDERS : Order details (type, symbol, quantity, price, status)
  + TRANSACTIONS : Trade executions and Account transactions
  + MARKET DATA : Real time and Historical Market Data
  + SYMBOLS : Information about tradable Assets
* Relationships
  + All users have accounts.
  + Users can place and modify orders.
  + Orders generate transactions
  + Market data is associated with Symbols
  + Accounts have transactions

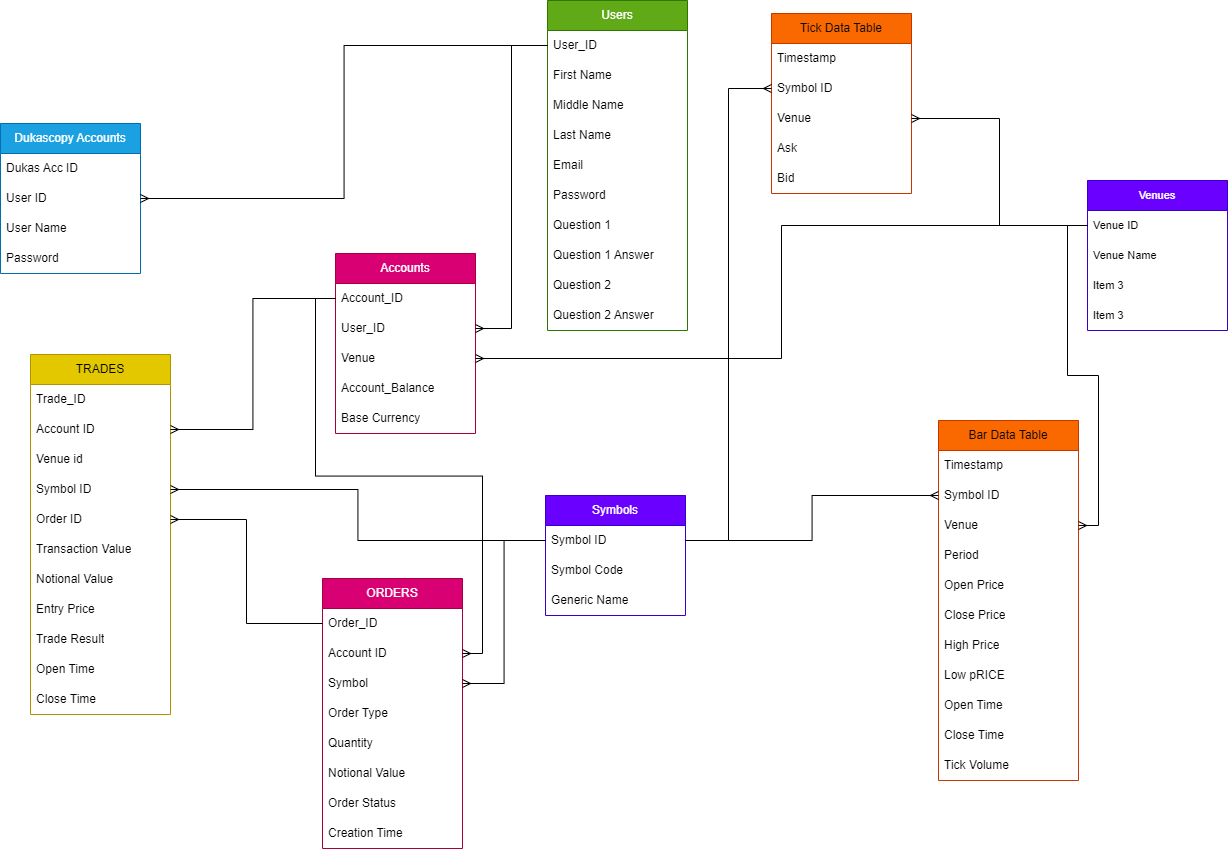
### ***Logical Database Design.***

* Users Table:
  + User ID - Primary Key
  + First Name
  + Middle Name
  + Last Name
  + User Name
  + Email
  + Password
  + Question 1
  + Question 1 Answer
  + Question 2
  + Question 2 Answer
* Accounts Table
  + Account ID - Primary Key
  + User ID - Foreign Key
  + Venue
  + Account Balance
  + Base Currency
* Tick Data Table
  + Timestamp - primary key
  + Symbol
  + Venue
  + Bid
  + Ask
* Bar Data Table
  + Timestamp
  + Symbol
  + Venue
  + Period
  + Open Price
  + High Price
  + Low Price
  + Close Price
  + Open Time
  + Close Time
  + Tick Volume
* Trades Table
  + Trade ID - Primary Key
  + Account ID - Foreign Key
  + Venue ID
  + Order ID
  + Transaction Type
  + Notional Value
  + Entry Price
  + Result
  + Open Time
  + Close Time
* Orders Table
  + Order ID - Primary Key
  + Account ID
  + Symbol
  + Order Type
  + Quantity
  + Notional Value
  + Price
  + Status
  + Timestamp
* Symbols Table
  + Symbol ID - Primary Key
  + Symbol Code
  + Generic Name
  + Exchange
* Dukascopy\_Accounts
  + Account\_id - Primary Key
  + User\_ID
  + User Name
  + Password
* Venues
  + Venue\_id
  + Venue Name
  + Connection Type

### ***Physical Database Model***

* ***Database System*** – ***MySQL*** – Allows for easy integration with Java as well as allows for rapid development. It is also open source and very easy to work with.
* ***Indexing*** - Indexing of Frequently Accessed Columns allows for very fast searches and retrieval i.e. from: timestamp, symbol etc.
* ***Partitioning*** - For table Market data, we need to partition it due to the sheer amount of data that it will hold. By partitioning it, we can improve performance greatly.
* ***Data types*** – By choosing the appropriate data types for each column, we can ensure accuracy of our data. This is especially critical in our case i.e. We need to Use DECIMAL to hold financial data because crypto currencies have very high precision and using DOUBLE or FLOAT would lead to wrong values because of data truncation. We also need to store our Unix Timestamps in TIMESTAMP so as to avoid a scenario where we need to keep converting back and forth from STRING so as to be able to utilize it effectively.
* ***Constraints*** – We can implement foreign and private keys as well as other constraints to ensure data integrity.
* ***Connection Pooling*** - We can utilize connection pooling to reuse connections and eliminate the need to create a new database connection for each transaction. This allows us to improve performance and reduce latency associated with creating connections.
* ***Sorting*** – Most Tables will have their data sorted by the timestamp

### **Entity Relationship for DB**



# **CHAPTER 6: SYSTEM IMPLEMENTATION**

## **6.1 Tools used for coding and testing**

### **Coding Tools**

* ***IntelliJ Idea Community Edition*** - A professional Java IDE with Maven Support.
* ***Maven -***  A java build tool that allows us to manage our dependencies as well as build and package our project
* ***MySQL Work Bench*** - A user interface-based software that allows us to interact with the MySQL server installed on the computer as well as to view, modify schema as well as database settings.

### **Testing Tools**

* ***Apache Surefire Maven Plugin*** - A tool that is used to execute unit tests during the last stage of a maven build lifecycle.
* ***Junit5 -*** a modern testing framework for Java, used for writing unit and integration tests. It offers a modular architecture and improved extensibility. It is integrated into a project via the Maven pom. It is used to write clear and concise test cases using annotations, utilizing assertions to verify expected outcomes. It can also be used to write parameterized tests.
* ***Mockito -*** a mocking framework for Java, used to create mock objects for dependencies during unit testing. It allows for isolated testing of components without relying on external resources or complex setups.
* ***TestNG -*** Testing Next Generation) is a testing framework inspired by JUnit and NUnit, but with additional features and flexibility. It is suitable for unit, integration, and end-to-end testing.

## **6.2 System test plan**

1. **Introduction**

The system test plan for the Lightning trading platform, which facilitates cryptocurrency and forex trading. The objective of this plan is to define the scope, methodology, resources, and schedule for system testing to ensure the platform meets the specified requirements and functions reliably in a production-like environment.

**2. Objectives**

* Verify the end-to-end functionality of the platform, including user registration, account management, trading, and reporting.
* Validate the integration of all system components, including APIs, databases, and external services.
* Assess the platform's performance, scalability, and stability under load.
* Evaluate the security measures implemented to protect user data and assets.
* Confirm compliance with relevant regulatory requirements.

**3. Scope**

The system test will cover the following areas:

* **User Management:** Registration, login, profile management, KYC/AML verification.
* **Account Management:** Deposits, withdrawals, balance tracking.
* **Cryptocurrency Trading:** Order placement, execution, market data streaming.
* **Forex Trading:** Order placement, leverage, margin, market data streaming.
* **Trading Interface:** Usability, responsiveness, and functionality.
* **API Integration:** Communication with cryptocurrency exchanges and forex liquidity providers.
* **Security:** Authentication, authorization, data encryption, vulnerability testing.
* **Performance:** Load testing, stress testing, response time measurement.
* **Reporting:** Transaction history, portfolio reports, financial statements.

**4. Test Environment**

* A dedicated test environment that mirrors the production environment as closely as possible.
* Realistic test data, including user accounts, trading pairs, and market data.
* Tools for performance testing (e.g., JMeter, Visual VM, Java Mission Control).
* Monitoring tools to track system performance and resource utilization.

**5. Test Strategy**

* **Functional Testing:** To verify that the platform functions according to the specified requirements.
* **Integration Testing:** To ensure that all system components work together seamlessly.
* **Performance Testing:** To evaluate the platform's performance under various load conditions.
* **Security Testing:** To identify and mitigate potential security vulnerabilities.
* **Usability Testing:** To assess the ease of use and user experience of the platform.
* **Regression Testing:** To ensure that new changes do not introduce regressions into existing functionality.
* **Compliance Testing:** To verify adherence to regulatory requirements.

**6. Test Data**

* Realistic test data will be used to simulate real-world scenarios.
* Test data will include various user accounts, trading pairs, and market data.
* Test data will be anonymized and protected to ensure data privacy.

## **6.3 CHANGE OVER TECHNIQUES**

As a result of the sensitive nature of trading systems as well as financial markets, the proposed changeover technique is ***Parallel Change over***.

This will allow both systems to run parallel to each other for an amount of time such that we have made any improvements that need to be made as well as having a fallback system in case the platform crashes thus preventing the stalling of business.

It will also allow us enough time to see the platform operating in a production environment and perform bug fixes until the platform reaches a start of stability and performance where it can be relied in solely to perform day to day operations of the company.

# **CHAPTER 7: LIMITATIONS, CONCLUSIONS AND RECOMMENDATIONS**

## **7.1 Limitations**

### *7.1.1 Technical challenges*

* ***Real time data integration.*** This is the cornerstone of an effective trading platform and it entails integrating multiple APIs and Exchange protocols to allow for handling of Realtime data, orderbook updates and trade executions
* ***Performance and scalability.*** The trading platform must be able to handle large amounts of data, process and store it concurrently in the database as well as keep handling incoming data without freezing or crashing
* ***Security Vulnerabilities***. Being a financial application, it presents itself with certain vulnerabilities such as secure storage of API keys as well as network security and the secure authentication. Also, the various dependencies that are used to add functionality to the project add vulnerabilities that we cannot patch.
* ***Poorly Documented Exchange APIs and SDKs***. This made development harder because of lack of a clear direction.
* Demo Accounts don’t support all Features. We were unable to test all features such as opening positions because they were unsupported in Demo Mode and thus we resorted to simulating them.

*7.1.2 Operational Challenges*

* High Cost. Development proved costly as a result of fast internet needed to consume large amounts of market data.
* Poor Internet, this made it a challenge during development because we need to constantly keep reestablishing and authenticating with the exchange after each disconnection which is tedious
* Uncooperative responses from Traders made it harder to optimize the platform towards serving their needs.

## **7.2 Conclusions**

This project aimed to develop a robust and scalable trading platform in Java, capable of handling both Foreign Exchange (FX) and cryptocurrency transactions. The development process highlighted the inherent complexities of building such a system, particularly in managing real-time data feeds, ensuring secure transaction processing, and opening and managing positions

While significant progress was made in establishing the core architecture, including the API integration, data handling, and basic trading functionalities, the project also revealed the challenges associated with integrating with diverse market APIs. The successful implementation of Java's concurrency features was crucial for managing simultaneous user requests and data streams. However, the project underscored the importance of rigorous testing and optimization to ensure stability and reliability in a live trading environment.

The project provided valuable insights into the practical application of Java in the financial technology sector, specifically within the dynamic and demanding realm of FX and crypto trading.

## **7.3 Recommendations**

Based on the findings and experience that I have gained in this project; the following recommendations are proposed:

* Enhance Performance optimization via thorough performance testing to seek and eliminate performance bottlenecks.
* Continuous testing and monitoring to catch bugs before production as well as being able to monitor and track system performance and ensure uptime.
* Prioritize the development of a user-friendly UI to enhance the trading experience. Gather User feedback so as to decide on key areas that need improvement.
* Future development and feature expansion, we can explore the addition of extra features in the future so as to enhance the platform

By implementing these recommendations, the project can be termed a success and we can have a competitive and effective Trading Platform

# **REFERENCES.**

1. Andrew H., David T. and Matt H (2007), ***Pragmatic Unit testing in C#***, Dallas
2. Buschmann (2000), ***Pattern Oriented Software Architecture***, Wiley Publishing
3. Djamel K. and Francine H. (2007***) Advances in Enterprise Information Technology Security.*** Information Science Reference, London.
4. Hans V. (2001) ***Software Engineering Principles and Practices, 2nd Edition***, Wiley limited, Amsterdam
5. Johnson, K. (2020). ***Algorithmic Trading & DMA: An Introduction to Direct Access Trading***. Harriman House, London.
6. Kaspersky (2023), ***What is Cryptocurrency and how does it work***, [online] Available at : [*https://www.kaspersky.com/resource-center/definitions/what-is-cryptocurrency*](https://www.kaspersky.com/resource-center/definitions/what-is-cryptocurrency)
7. Kendall K (2007) ***Electronic and Algorithmic Trading Technology***, Academic Press, San Diego
8. Martin F. (2002), ***Patterns of Enterprise Application Architecture***, Addison Wesley,
9. Michael R. (2001) ***Secure Communication Systems: Design, Analysis and Implementation***, Cambridge University Press, New York
10. Oracle. (2023). Java Platform Standard Edition Documentation. [online] Available at : *https://docs.oracle.com/en/java/javase/index.html* [Accessed: 2025-01-22]
11. PWC (2017), ***Making sense of Bitcoin, Cryptocurrency and blockchain,*** [online] Available at : [*https://www.pwc.com/us/en/industries/financial-services/fintech/bitcoin-blockchain-cryptocurrency.html*](https://www.pwc.com/us/en/industries/financial-services/fintech/bitcoin-blockchain-cryptocurrency.html)
12. Roy O. (2014), ***The Art of Unit Testing, 2nd Edition***, Manning, New York.
13. Subramaniam, V. (2019). ***Functional Programming in Java: Harnessing the Power of Java 8 Lambda Expressions.*** Raleigh, North Carolina

# **APPENDIX**

## **Organizational Structure**

* ***Chief Executive Officer*** – Provides the overall strategic direction and management as well as coordinates all other department heads
* ***Chief Operations Officer*** – Performs management of overall operations and manages the core activities of the firm.
* ***Chief Technology Officer*** - Manages the technological hardware and software infrastructure as well as the cyber security of the firm.
* ***Trading Operations Manager*** – Manages all the trading desks for both Crypto and forex as well as the traders and oversees general trading operations.
* ***Head of Risk Management*** – Tasked with overseeing risk analysis as well as risk mitigation and adherence to regulatory frameworks.
* ***Chief of Finance*** - Manages the financial operations of the firm as well as Accounting and book keeping.
* ***Infrastructure Manager*** – Tasked with management of computer hardware and networks.
* ***Cybersecurity Manager*** – Manages the cybersecurity of the firm as well as set up security policies and frameworks that keep the company secure from both internal and external threats.
* ***Tax Office*** - The tax office mainly focuses on ensuring tax compliance as well as keeping up to date on tax law and continuously seeking tax strategies that can lower the tax bill for the organization.
* ***Trading Desks: Crypto and Forex*** - These are the actual centers where trading takes place and traders place buy and sell orders in the market.
* ***Compliance Officer –*** *The* compliance officer ensures that the company adheres to the regulatory framework of the home country of the firm as well as the regulations for all the exchanges that I trade on

## **Documents Reviewed**

This section lists sample documents that were reviewed and utilized during the different development phases of Lightning Trading platform.

#### **I. Project Initiation & Requirements Gathering:**

* **1.1. Initial Project Proposal/Statement of Work (SOW):**
  + Outlined the project's goals, scope, and deliverables.
  + Reviewed to ensure alignment with stakeholder expectations.
* **1.2. Business Requirements Document (BRD):**
  + Defined the high-level business needs and user requirements.
  + Reviewed to understand the core functionalities and business context.
* **1.3. Market Analysis Reports:**
  + Provided insights into the crypto and forex trading market.
  + Reviewed to inform feature prioritization and market positioning.

#### **II. System Design & Architecture:**

* **2.1. System Architecture Design Document:**
  + Detailed the overall system architecture, including the Swing client, Spring Boot backend, and database interactions.
  + Reviewed for scalability, maintainability, and performance considerations.
* **2.2. Database Entity-Relationship Diagram (ERD):**
  + Visualized the database schema and relationships.
  + Reviewed for data integrity and optimization.
* **2.3. User Interface (UI) Design Specifications:**
  + Defined the layout, design, and user experience of the Swing application.
  + Reviewed for usability and adherence to design principles.

#### **III. Development & Implementation:**

* **3.1. Java Swing API Documentation:**
  + Referenced for Swing component usage and UI development.
  + Reviewed for correct implementation of UI elements.
* **3.2. MySQL Documentation:**
  + Used for database design and query optimization.
  + Reviewed to insure proper time series data management.
* **3.3. Maven Project Object Model (POM) Files:**
  + Managed dependencies and build processes.
  + Reviewed for dependency conflicts and build configurations.
* **3.4. Coding Standards and Style Guides (Google Java Style Guide):**
  + Ensured code consistency and readability.
  + Reviewed during code reviews.
* **3.5. Version Control System (Git) Documentation:**
  + GIT branching strategy documentation.
  + Reviewed for proper version control procedures.

#### **IV. Testing & Quality Assurance:**

* **4.1. Test Plans and Test Cases:**
  + Defined testing strategies and test scenarios.
  + Reviewed for test coverage and effectiveness.
* **4.2. Unit Test Reports (JUnit):**
  + Documented the results of unit tests.
  + Reviewed for code quality and functionality.
* **4.3. Integration Test Reports:**
  + Documented the results of integration tests.
  + Reviewed for system wide data flow.
* **4.4. Performance Test Reports:**
  + Documented the results of performance tests.
  + Reviewed to insure system stability.
* **4.5. Security Audit Reports:**
  + Documented the results of security vulnerability scans.
  + Reviewed to address security concerns.

#### **V. Deployment & Operations:**

* **5.1. Deployment Configuration Documents:**
  + Detailed the deployment process and configurations.
  + Reviewed for accuracy and completeness.
* **5.2. JPackage Configuration Files:**
  + Used to create native installers.
  + Reviewed to insure proper installer creation.
* **5.4. Log4j2 Configuration Files:**
  + Used for logging and error handling.
  + Reviewed to insure proper logging.

#### **VI. User Documentation & Support:**

* **6.1. User Manual (This Document):**
  + Provided instructions for end-users.
  + Reviewed for clarity, accuracy, and completeness.
* **6.2. Frequently Asked Questions (FAQ) Document:**
  + Addressed common user questions.
  + Reviewed for relevance and accuracy.
* **6.3. Troubleshooting Guide:**
  + Provided solutions to common user problems.
  + Reviewed for accuracy and user friendliness.

## **Sample codes**

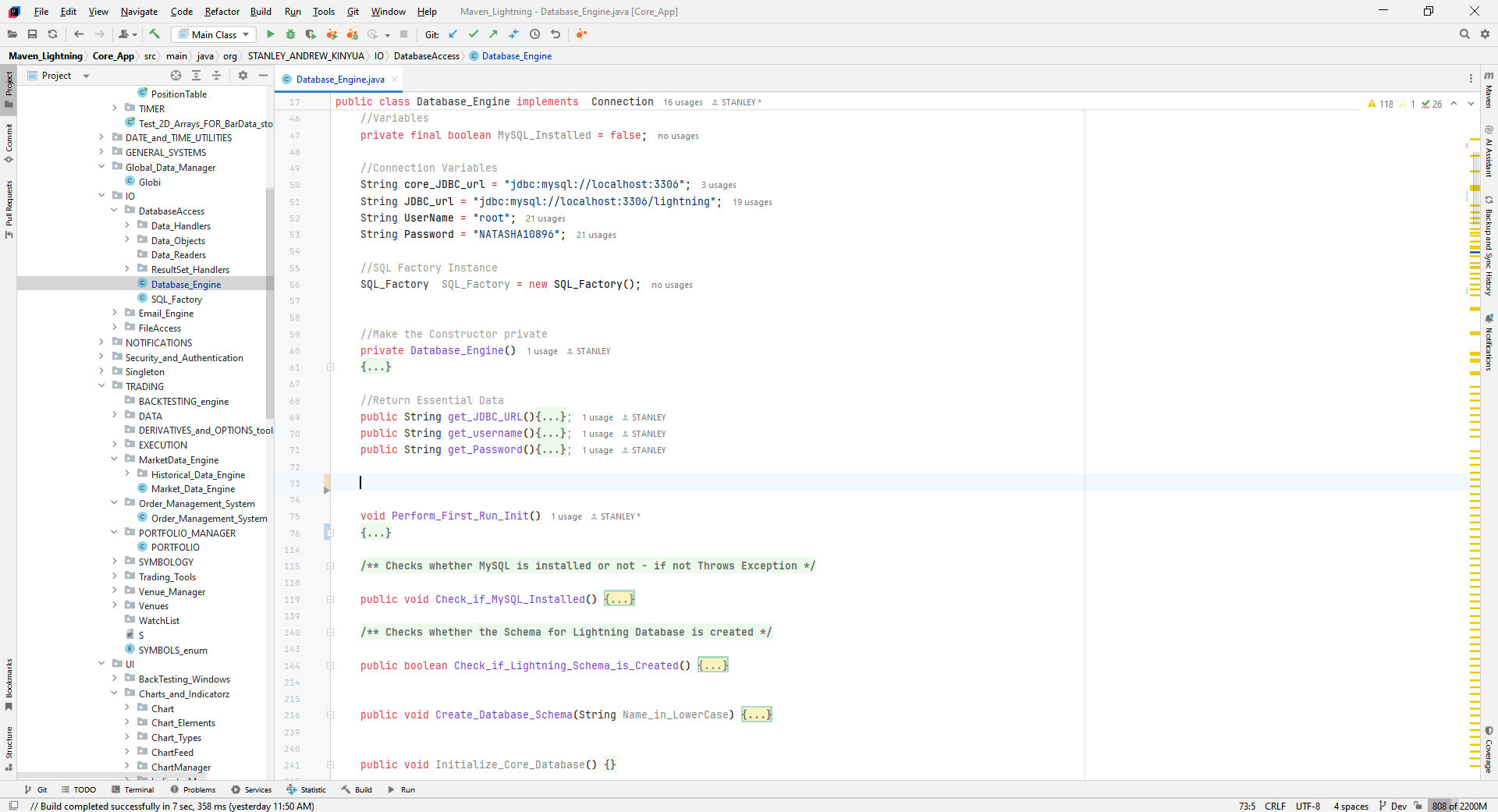


Figure : Database Engine Screenshot

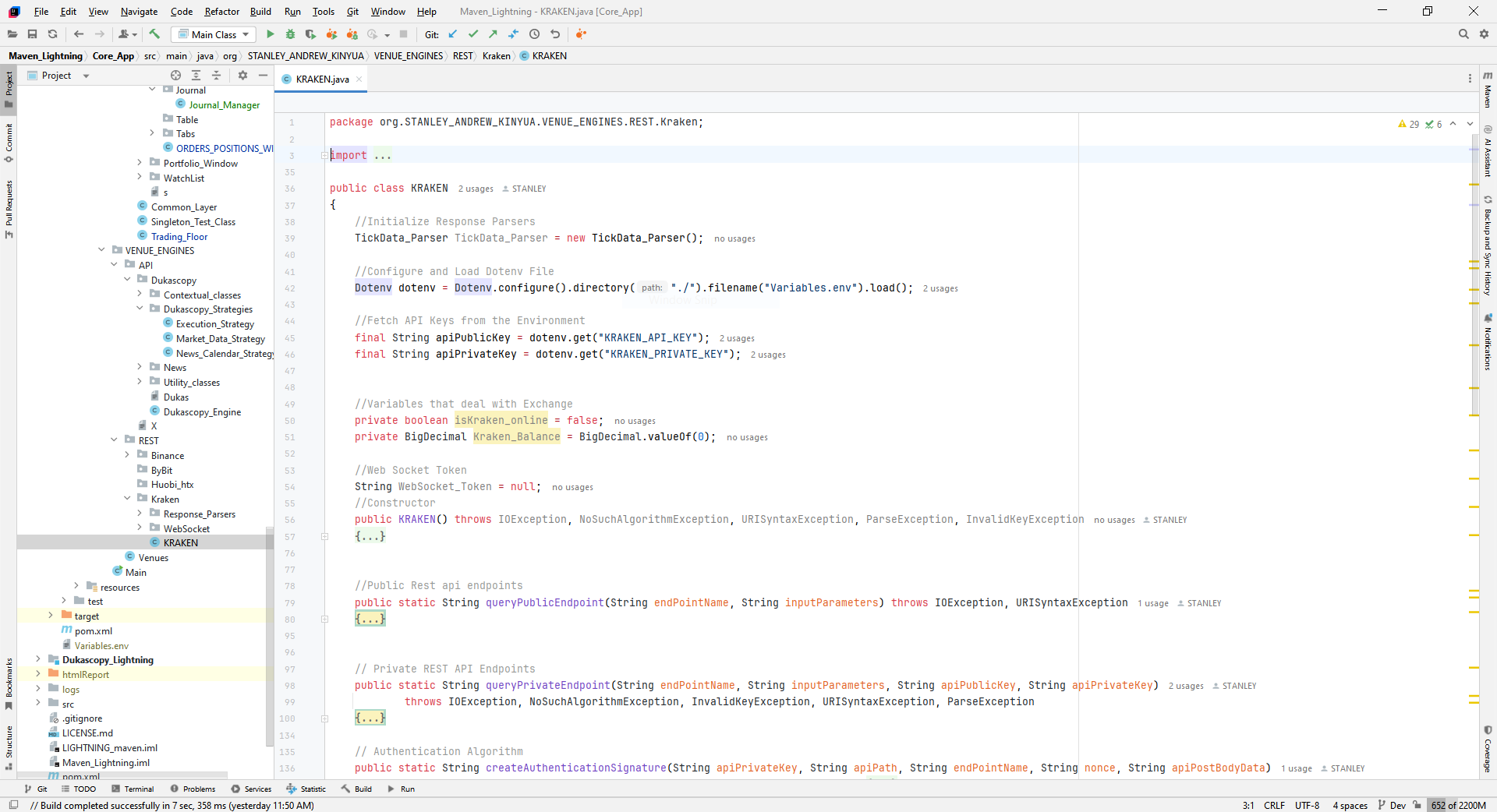


Figure : Kraken Class - Screenshot

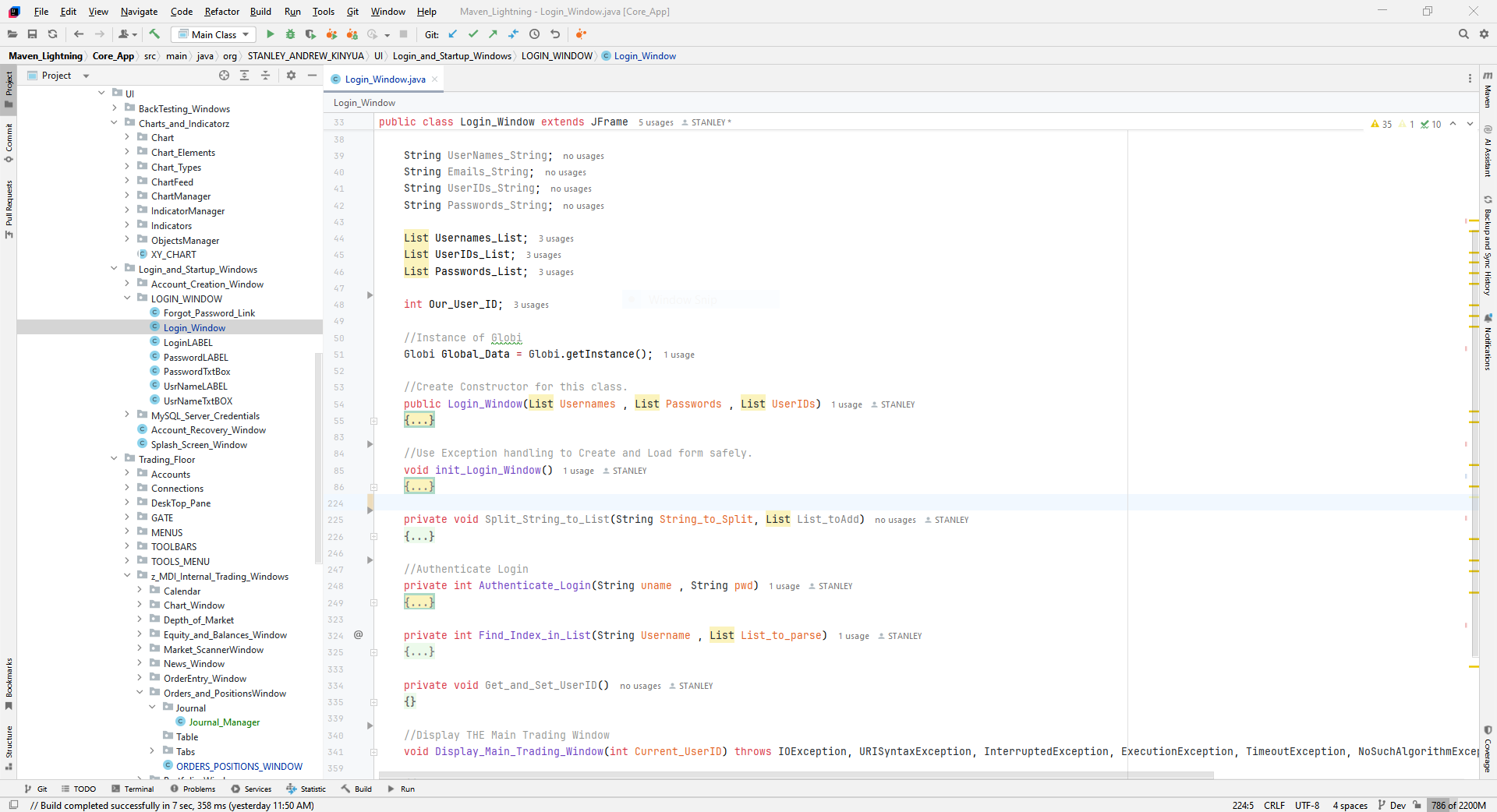


Figure : Login Window Class - Screenshot

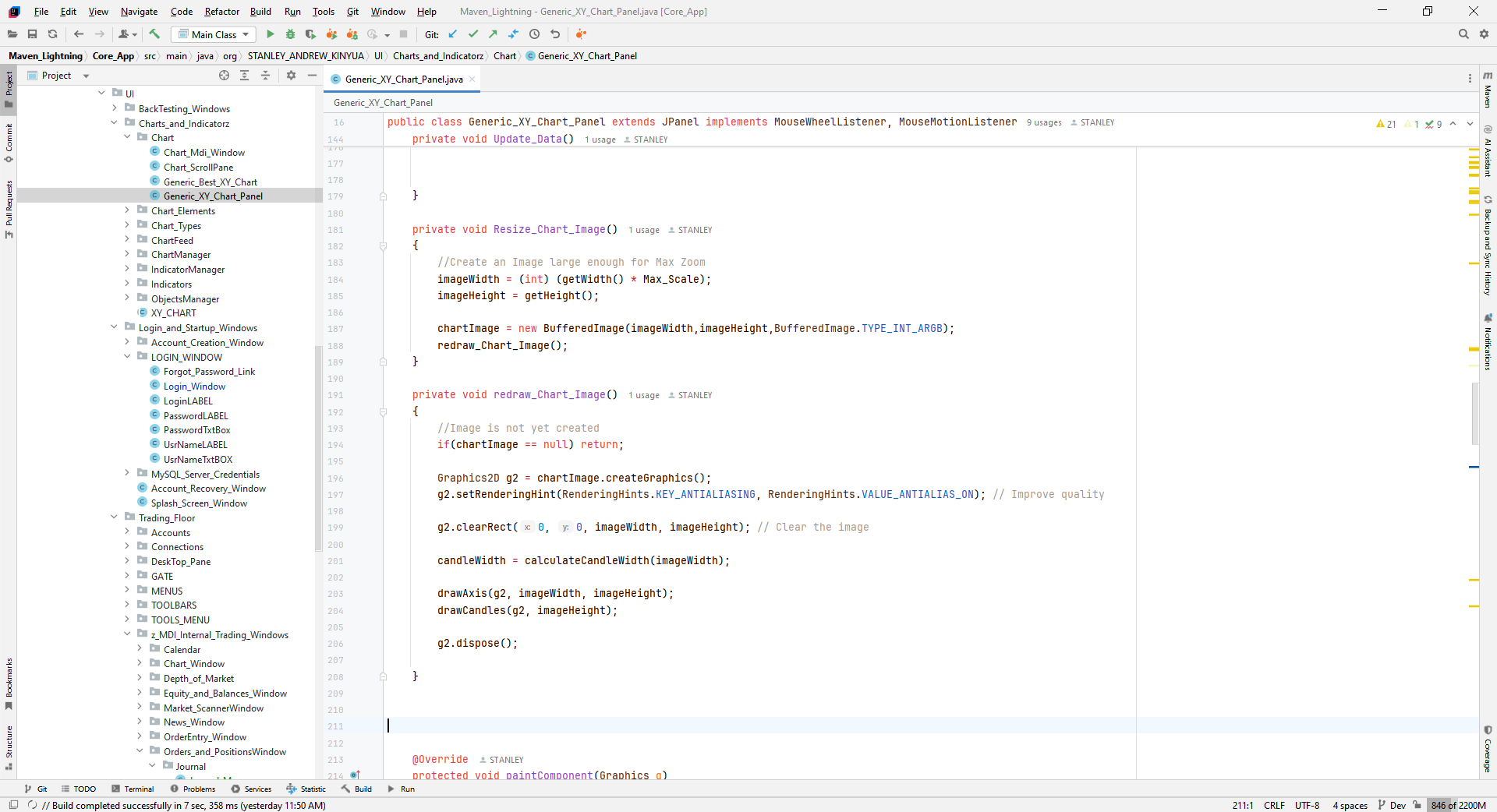


Figure : Chart Panel Class - Screenshot

## **Technical Guide and User Manual**

### **User Manual**

**1. Introduction**

The Lightning Trading Platform is a desktop application designed to provide traders with the tools they need to efficiently participate in financial markets. It aims to offer a user-friendly interface with powerful features for both new and experienced traders.

**2. Getting Started**

**2.1 System Requirements:** The application is a Java-based desktop application. Ensure your system meets the minimum Java Runtime Environment requirements.

**2.2 Installation:**

* + Download the installer or executable JAR file.
  + Run the installer and follow the on-screen instructions, or execute the JAR file.

**2.3 Account Creation and Login:**

* + Upon launching the application, you will be prompted to create an account or log in with existing credentials.
  + Follow the registration process, which may include email verification.

**3. User Interface Overview**

**3.1 Main Window:** The main window provides access to key trading information and tools. It includes sections for:

* + Market data display (real-time price feeds, charts)
  + Order management (placing and managing buy/sell orders)
  + Account information (balances, portfolio details)
  + News feed (market news)

**3.2 Navigation:** The application uses a combination of menus, toolbars, and panels for navigation between different features.

**3.3 Key Features:**

* + **Real-time Charting:** View and analyze market data with interactive charts and technical indicators.
  + **Order Management:** Place market orders, limit orders, and manage your open positions.
  + **Account Monitoring:** Track your account balances, transaction history, and portfolio performance.
  + **News Feed:** Stay updated with the latest market news and events.

**4. Core Functionalities**

**4.1 Viewing Market Data:**

* + Real-time market data is displayed in charts and data grids.
  + You can select different financial instruments and timeframes.

**4.2 Placing Orders:**

* + To place an order, use the order entry panel.
  + Specify the instrument, order type (e.g., market, limit), quantity, and price (if applicable).
  + Confirm the order details before execution.

**4.3 Managing Positions:**

* + View your open positions and order history in the positions panel.
  + You can modify or close open positions as needed.

**4.4 Account Management:**

* + View your account balances and transaction history.
  + The application may also provide tools for managing your account settings.

**5. Advanced Features**

**5.1 Historical Data:**

* + The platform allows users to view historical market data for analysis.
  + This data can be accessed even when offline.

**5.2 Risk Management:**

* + The application includes automated risk management checks to help protect your capital.
  + These tools monitor market conditions and provide alerts or automated actions.

**5.3 Reporting:**

* + Generate reports on your trading activity, portfolio performance, and trading costs.
  + These reports can help you analyze your trading strategies and track your progress.

**6. Frequently Asked Questions (FAQ)**

**Q: Can I use this platform to trade both currencies and cryptocurrencies?**

* + A: Yes, the platform is designed to support trading in both forex and crypto-currencies.

**Q: Can I access my trading data offline?**

* + A: Yes, the platform supports offline access to historical market data and account information.

**Q: How do I get help or support?**

* + A: Contact the developer for customized support

### **Technical Guide**

**1. Introduction**

This technical guide provides an overview of the system architecture, technologies, and development process for the Lightning Trading Platform.

**2. System Architecture**

**2.1 Overview:** The system uses a multi-tier architecture to separate concerns and improve maintainability.

**2.2 Tier Breakdown:**

* + **Presentation Tier (UI):** Java Swing is used to create the desktop user interface.
  + **Application Tier:** Java is used for the application logic, including order processing, market data handling, and business rules.
  + **Data Tier (Database):** MySQL is used to store market data, user transactions, and account information.

**2.3 Key Components:**

* + **User Interface:** Handles user input and displays market data.
  + **Order Management System:** Manages orders, execution, and risk.
  + **Market Data Handlers:** Retrieves and stores real-time and historical market data.
  + **Account Management Systems:** Manages user accounts, balances, and portfolio details.
  + **Exchange API Integration:** Connects to various exchanges for data and order execution.
  + **Database:** Stores application data.

**3. Development Environment Setup**

**3.1 Prerequisites:**

* + Java Development Kit (JDK)
  + Maven
  + MySQL
  + IntelliJ IDEA (or other Java IDE)
  + Git

**3.2 Installation and Configuration:**

* + Install the necessary software (JDK, Maven, MySQL).
  + Set up the database and configure connection properties.
  + Clone the project repository using Git.
  + Import the project into IntelliJ IDEA and configure the project SDK and dependencies.

**4. Code Structure and Conventions**

**4.1 Directory Structure:** The project follows a standard Maven project structure to organize source code, resources, and dependencies.

**4.2 Coding Standards:** Code follows Java coding conventions for readability and maintainability.

**4.3 Version Control:** Git is used for version control, with a branching strategy to manage development, releases, and hotfixes.

**5. Deployment Instructions**

**5.1 Backend Deployment:**

* + Package the application using Maven.
  + Deploy the application to a server with a Java Runtime Environment.
  + Configure the application to connect to the database.

**5.2 Client Distribution:**

* + The client application can be distributed as a JAR file or packaged into an OS-specific installer.

**6. Testing**

**6.1 Testing Tools:**

* + JUnit: For unit and integration testing.
  + Mockito: For mocking dependencies during testing.
  + REST Assured: For testing RESTful APIs.

**6.2 Testing Strategy:**

* + **Unit Testing:** Testing individual components.
  + **Integration Testing:** Testing the interaction between different components.
  + **System Testing:** Testing the entire system to ensure it meets requirements.
  + **Functional Testing:** Testing the system against business requirements
  + **Performance Testing:** Load testing, stress testing, response time measurement.

**7. Limitations**

**7.1 Technical Challenges:**

* + Real-time data integration and handling.
  + Performance and scalability.
  + Security vulnerabilities.
  + Poorly documented exchange APIs.

**7.2 Operational Challenges:**

* + Development costs.
  + Internet connectivity issues.
  + Gathering user feedback

**8. Future Development**

* Enhance performance optimization.
* Continuous testing and monitoring.
* Prioritize user-friendly UI development.
* Gather user feedback for improvements.
* Explore the addition of extra features.

Sources and related content