

ST. MARY'S UNIVERSITY
FACULTY OF INFORMATICS
DEPARTMENT OF COMPUTER SCIENCE



**FOREX TRADING JOURNAL WITH PERFORMANCE
ANALYTICS**

BY

| Name | ID |
|------------------|---------------|
| Daniel Eshetu | RCD/1201/2014 |
| Natnael Feseha | RCD/1226/2014 |
| Tigistu Begashaw | RCD/1233/2014 |
| Yoseph Wegayehu | RCD/1241/2014 |

June 2025

SMU

ADDIS ABABA

**FOREX TRADING JOURNAL WITH PERFORMANCE
ANALYTICS**

BY:

**Daniel Eshetu, Natnael Feseha, Tigistu Begashaw, and Yoseph
Wegayehu**

**A SENIOR PROJECT DOCUMENT SUBMITTED TO THE
DEPARTMENT OF COMPUTER SCIENCE**

FACULTY OF INFORMATICS

ST. MARY'S UNIVERSITY

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF BACHELOR OF SCIENCE IN
COMPUTER SCIENCE**

JUNE 2025

SMU

ADDIS ABABA

ST. MARY’S UNIVERSITY

**FOREX TRADING JOURNAL WITH PERFORMANCE
ANALYTICS**

BY:

Daniel Eshetu, Natnael Feseha, Tigistu Begashaw, and Yoseph
Wegayehu

**FACULTY OF INFORMATICS
DEPARTMENT OF COMPUTER SCIENCE**

APPROVED BY THE COMMITTEE OF EXAMINERS

Department Head

Metages Ephrem

Advisor

Internal Examiner

External Examiner

Signature

Signature

Signature

Signature



PIP JOURNAL

CERTIFICATE OF AUTHENTICATED WORK

This is to certify that this project “**FOREX TRADING JOURNAL WITH PERFORMANCE ANALYTICS**”

” was carried out by group of students of computer science at St. Mary’s University as a requirement for senior project by:

Group Members:

Name

Signature

Daniel Eshetu

Natnael Feseha

Tigistu Begashaw

Yoseph Wegayehu

The matter embodied in this project is authentic and is genuine work done by the student and has not been submitted whether to this university or any other university / institute for the fulfillment of the requirement of any course of study.

This Project proposal was reviewed and approved (for examination) by:

Advisor Name

Signature

Date

Metages Ephrem

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to all those who supported us throughout the development of our senior project, Forex Trading Journal.

First and foremost, we thank our advisor Metages Ephrem, for their invaluable guidance, constructive feedback, and continuous encouragement.

Special thanks to our families and friends for their unwavering support and motivation during this journey.

Lastly, we appreciate the collaborative effort of our team, whose dedication and hard work were essential to the success of this project.

TABLE OF CONTENT

| | |
|--|----|
| Chapter one: Introduction | 1 |
| 1. Introduction..... | 1 |
| 1.1Background..... | 1 |
| 1.2 Statement of the Problem..... | 4 |
| 1.3 Objectives | 5 |
| 1.4 Methodology | 6 |
| 1.5 Hardware and Software Requirements..... | 8 |
| 1.6 Scope and Limitation..... | 9 |
| 1.7 Significance of the Project | 12 |
| 1.8 Feasibility Study | 13 |
| 1.9 Risk Assessment..... | 14 |
| 1.10 Work Breakdown Structure (WBS) | 17 |
| Chapter two: Business Area Analysis And Requirement Definition | 19 |
| 2.1 Introduction..... | 19 |
| 2.2 Current System | 19 |
| 2.2.1 Players of the Existing System | 21 |
| 2.3 Business Rules | 22 |
| 2.4 Proposed System | 24 |
| 2.4.2 Requirement Definition..... | 26 |
| 2.4.3 System Model | 34 |
| 2.4.3.2 Sequence Diagram | 38 |
| 2.4.3.3 ACTIVITY DIAGRAM | 44 |
| 2.4.3.4 State Chart Diagram | 48 |
| 2.4.3.2 Class Diagram..... | 50 |
| 2.4.3.6 Data Model | 53 |
| 2.5 User Interface Prototyping..... | 55 |
| Chapter three: system design | 68 |
| 3.1 Introduction..... | 68 |
| 3.2 Purpose of the System | 68 |
| 3.3 Design Goals | 69 |
| 3.4 Current Software Architecture | 70 |
| 3.5 Proposed Software Architecture | 71 |

| | |
|---|----|
| 3.5.2 Component Diagram | 73 |
| 3.5.3 Deployment Diagram | 74 |
| 3.5.4 Persistent Data Management | 76 |
| 3.5.5 Detailed Database Design..... | 78 |
| 3.5.6 Access Control and Security | 82 |
| 3.5.8 Boundary Conditions..... | 86 |

LIST OF FIGURES

| | |
|---|----|
| Figure 1 Agile Development Methodology | 7 |
| Figure 2 Laravel | 8 |
| Figure 3 Gantt Chart | 18 |
| Figure 4 Trader use case | 36 |
| Figure 5 performance Analyst use case | 36 |
| Figure 6 Admin use case | 38 |
| Figure 7 All Actors Use Case..... | 38 |
| Figure 8 Trader sequence diagram | 41 |
| Figure 9 performance Analyst sequence diagram..... | 42 |
| Figure 10 Admin sequence diagram | 43 |
| Figure 11 Performance analyst activity diagram..... | 46 |
| Figure 12 Trader activity diagram | 46 |
| Figure 13 Admin Activity diagram | 47 |
| Figure 14 State Diagram | 50 |
| Figure 15 Class Diagram..... | 53 |
| Figure 16 ER Diagram | 55 |
| Figure 17 UIWT1..... | 56 |
| Figure 18 UIWT2..... | 56 |
| Figure 19 UIWT3..... | 57 |
| Figure 20 UIWT4..... | 57 |
| Figure 21 UIWT5..... | 58 |
| Figure 22 UIWT6..... | 59 |
| Figure 23 UIWT7..... | 59 |
| Figure 24 UIWT8..... | 60 |
| Figure 25 UIWA1 | 61 |
| Figure 26 UIWA2 | 62 |
| Figure 27 UIWA3 | 63 |
| Figure 28 UIWA4 | 64 |
| Figure 29 UIWA5 | 65 |

| | |
|-----------------------------------|----|
| Figure 30 UIWM1 | 66 |
| Figure 31 UIWM2 | 67 |
| Figure 32 component diagram..... | 74 |
| Figure 33 Deployment diagram..... | 76 |
| Figure 34 EER Diagram..... | 80 |

LIST OF TABLES

| | |
|---|----|
| Table 1 Hardware requirements | 8 |
| Table 2 software requirements..... | 9 |
| Table 3 WBS Table | 17 |
| Table 4 Trader | 28 |
| Table 5 Performance Analyst | 28 |
| Table 6 Admin | 29 |
| Table 7 User Interface Prototyping | 55 |
| Table 8 Types of Persistently Stored Data | 76 |
| Table 9 user table | 78 |
| Table 10 trade table | 78 |
| Table 11 Feedback table | 79 |
| Table 12 ExportLog table | 79 |
| Table 13 Mapping table | 81 |

Abstract

This project outlines a web-based Forex Trading Journal with Performance Analytics that focuses on structured trade logging and data-driven feedback as a means of decision-making improvement for Forex traders. The primary aim of the Forex Trading Journal with Performance Analytics is to facilitate traders in decision-making by logging trades in a structured manner and receiving feedback based on data. A number of traders tend to use informal tools like spreadsheets or notebooks, which are often inconsistent, lack analytics, and do not have secure storage. The suggested system solves these problems by offering a single platform for the logging of trade details, tracking of performance trends, and receiving of feedback from performance analysts, in other words, an integrated platform.

The system is designed to have three user roles — Trader, Performance Analyst, and Admin — The roles determine the access and the functions that the users can perform. Traders can input trade data, go over previous trades, and interact with analytics visualized on dashboards. Analysts have the ability to go over a trader's history and give them a feedback that is tailored to them. Admins are responsible for user management, system monitoring, and data backup tasks.

The system design is based on a modular and scalable Model-View-Controller (MVC) architecture which uses PHP, MySQL, and Chart.js. The document specifies various design goals including usability, security, performance, and maintainability. Sections elaborate on the system specifications, database design, UML diagrams (such as use case, class, component, and deployment diagrams), and mechanisms for access control. The data schema complies with normalization, and all parts are explained based on software engineering standards and best practices.

The system not only reinforces the trader's discipline but also equips the trader for performance assessment and provides mentorship through collaboration in a structured manner — the system thus represents a contemporary and dependable solution to a void that exists in retail trading workflows.

Chapter one: Introduction

1. Introduction

The document is about the design and build of a web-based Forex Trading Journal with Performance Analytics that is the one that gets a trader to his trading performance by means of structured journaling and visual insights.

Forex trading need is the technical skills, but it is also emotional discipline and a strategic consistency. The new ones struggle a lot to recognize that they are lacking the skills of record keeping and recognizing their behavior in the past and review of their decisions. It is true that some platforms provide a simple trade history log. However, they lack the most important features like performance analysis, emotional tracking, or custom filtering.

Our solution does this by giving the user a journaling tool which allows to log trades both manually and via CSV, to tag trades, to enter the emotional/psychological part of the note, to use interactive analytics, and to get automated summary reports. Also, the records can be exported in PDF, and personal notes can be added, which leads to the platform being the place of both performance review and personal growth.

The document deals with the discussion of the background of the project, the problem statement, the objectives of the project, the methodology, the requirements, the scope, the limitations, and the feasibility in the following chapters - all underlining the system's planned development and its significance in the trading community.

1.1Background

Forex (foreign exchange) is the most liquid financial market in the world and it is accessible to anyone globally with a daily transaction volume that exceeds \$7 trillion [1]. A Forex trader is a person who operates in this market in a single day across multiple time zones. The Forex market is open 24 hours a day, hence Forex trading is highly liquid and attracts millions of individual traders who are looking to make money from currency fluctuations. The volatile and complex nature of this market, however, makes it difficult to succeed and it requires more than just market knowledge. In addition, success in Forex trading is also a matter of self-assessment, strategic decision-making and emotional control [2].

Among the various resources that a trader might use to facilitate his performance gain, the most frequently suggested one is a trading journal. The decision of the trader to keep a journal documenting all his trades will grant him access to insights into the past of his trading activities and, in turn, help him to identify new decisions that could be better than the old ones given the situations faced. Experts such as Mark Douglas [2] and Van K. Tharp [3] maintain that, for a trader to be self-disciplined and have a clear vision of making decisions, a trading journal is the key thus enabling them to keep the discipline and decision-making clarity over a long period of

time. In fact, a lot of traders either have no journal at all or depend largely on rudimentary instruments like a notebook or spreadsheet to record their trades. No analysis, no psychological monitoring, no structure, and fewer psychological signs characterise these manual methods hence they may not be as effective over time.

This initiative aims to close the gaps in personal Forex trading by creating a web-based Forex Trading Journal with Performance Analytics. The latter is specifically designed for individual Forex traders. Such a system is not a one-effort, one-solution type, but rather a hybrid of technical and behavioral features, hence, a more comprehensive approach to trader professional development.

The system will give users the option to enter trades manually or import them via a CSV, thus, it will not limit data entry methods. Additionally, each trade record requests not only the basic information but also many other aspects such as currency pair, session, trade type, risk-reward ratio, outcome, reasoning, and emotional state. Moreover, the users will be able to mark their trading operations with a certain strategy and thus, e.g., if they decide to employ the Breakout and Reversal only, the platform will assist them to keep record of which of the two is the most effective and the one that they should use henceforth.

The platform will also introduce a part meant for personal reflections that can be of great help to users when it comes to non-specific trades. Users also have the possibility to upload their screenshots or if preferred, connect online charts such as TradingView to visually reinforce their setups. Besides, a trade history interface with robust filtering options will allow users to search through the log of completed trades by outcome, date, session, pair, or strategy.

Apart from that, the system, being empowered by Chart.js will carry the interactive part through colorful and clear visual representations such as win/loss ratios, session performance, and strategy effectiveness. This will keep them abreast of the trends of their performance even when they are not directly involved in the trading activity as it will cover the periods that have passed. They can at the same time export and save the documents arising from the record of trades and from the summaries in the format PDF thus, they can decide to keep them safely archived or to share them with others.

This project is not simply recording trades, it is certainly about encouraging discipline, thoughtful, and self-improvement in the trading business. This system has been created with modern technologies such as Laravel (PHP framework), MySQL, JavaScript, Bootstrap, and it serves as a complete journaling tool for traders who desire to manage their development and efficiencies in the Forex market.

In the fiercely competitive forex market, currency traders are driven to incessantly assess and develop their performance if they want to remain profitable. They, however, find it very difficult to keep a record of their trades, analyze their strategies, and even reflect on their decision-making process. As a matter of fact, carrying out trades is a necessary skill but success over time

depends on a trader's capability to draw lessons from past experiences, his emotional regulation, and his further adaptations, which performance trends reveal [1]. The bad news is that most of those individual traders are not equipped with the necessary tools that enable them to participate in this kind of structured self-assessment.

Today, there are a lot of traders who depend on manual journaling means like notebooks or spreadsheets for recording their trades. Such tools are, however, prone to inconsistency and errors and also do not support the incorporation of functions such as filtering, summarization, or analysis. Besides that, these methods do not include psychological parts such as the emotional state or decision-making mindset. Psychology, as revealed by experts like Douglas [2] and Tharp [3], is of great influence in the trading results. Those who keep unstructured logs and are not able to identify the issues and work out strategies that can be front-running them or spot how the emotions may lean performance if they continue for a while are the ones.

Furthermore, a plethora of software solutions online that allow for trade journaling and also include analytical features—e.g., Edgewonk, Myfxbook, or Tradervue—are often costly, difficult to set up with broker platforms, or they are designed to cater the particular geographical areas or account types [4]–[6]. Hence, first-time or independent traders tend to see themselves out of what is accorded to them from decision aids that will foster their learning process.

This project is aimed at tackling those problems by designing a web-based Forex Trading Journal with Performance Analytics that is accessible and at the same time full of features. The new system, which is not like the traditional journals, will enable the traders to enter trades manually or bring them from CSV, thus reducing entry friction. In the same way, the journal system can be used to track finished trades or ongoing operations that are recorded with structured fields like pair, session, trade type, risk-reward ratio, outcome, and emotional state.

The traders besides having the possibility to assign tags to the trades, to give them the type of the strategy or to add visual evidences such as the screenshots or the TradingView links, can also filter the trading history by win/loss, day, session or strategy. In that case, they will have a better, more specific way to check what kind of trade it is or how they acted in a certain situation if they check the trade's emotional state.

Besides that, the system provides the user with a section designated for personal notes of a more extensive character and also offers an analytics dashboard that has interactive charts to facilitate the understanding of the performance trends. It is possible to introduce a time frame so recordings will be automatic and can be switched into different period formats such as weekly, monthly, or even yearly. Self-review and self-reflection become more frequent and, therefore, more powerful as the trader can now have the necessary information always at hand, along with an option to keep it.

With the mix of these techniques and tools, such as keeping track on the trader's mood and emotions through questions and support, allowing the trader to adjust the review to his or her

needs, and providing clear and diverse picture of a trader's performance, this project presents a total solution for a trader who wants to close the gap between the states of trade execution and the long road of improvement.

1.2 Statement of the Problem

In the cutthroat world of Forex trading, traders must never stop evaluating and improving their performance in order to remain profitable. However, a large number of them still have huge difficulties in keeping track of their trades, analyzing their strategies, and reflecting on their decision-making processes. Execution of trades is a vital skill, but continued success depends on the trader's capacity to learn from past experiences, control emotional reactions, and make changes according to performance trends. Unfortunately, most individual traders are not provided with the necessary tools to allow them to carry out these structured self-assessment tasks.

At present, many traders still rely on manual journaling methods such as notebooks or spreadsheets to record their trades. These approaches are often disorganized, prone to errors, and lack essential features like filtering, summarizing, or analyzing trade data. While the psychological side of trading — including emotional state and decision-making mindset — plays a critical role in performance, it is often overlooked or not tracked at all. Traders using unstructured logs may struggle to identify their behavioral patterns, evaluate which strategies are effective, or understand how their emotions influence their results over time.

Besides, most of the current digital resources that actually provide trade journaling and analytics—like Edgewonk, Myfxbook, or Tradervue—are either very costly, too hard to integrate with the broker platforms, or dedicated to specific regions or account types. Consequently, new or independent traders very often feel like they are outside their comfort zone. This is so, as they can hardly find the tools that are most beneficial to the learning process in their possession.

This initiative intends to overcome these obstacles through the creation of a web-based Forex Trading Journal with Performance Analytics that is not only feature-rich but also accessible and utilizing the performance analytics from Google Performance. The system, which is different from conventional journals, will give traders the opportunity to either manually log trades or import them using CSV, hence minimizing the entry friction. Each transaction will be described using the fields such as pair, session, trade type, risk-reward ratio, outcome, and emotional state. Besides this, traders can also tag the trades by the type of strategy, add the visuals like the screenshots or TradingView links, and they can also filter the trade history by the win/loss, day, session, or strategy.

Furthermore, the system has a personal notes section, where the traders can write their feelings and a dashboard with charts that are interactive to help the traders keep track of their performance. Additionally, the system will generate weekly and monthly reports to enable the

user to carry out self-review on a regular basis. Besides this, trade data can be exported not only in offline mode but also to personal archives, in the form of PDFs.

By linking structured data recording, emotional recognition, adaptable filters, and visualization, this project would be a truly all-around solution for traders to connect the execution of a trade with long-term development.

1.3 Objectives

1.3.1 General Objective

To create a web-based Forex Trading Journal system that facilitates individual traders in recording, organizing, analyzing, and exporting their trade data with performance analytics for better decision-making, strategic evaluation, and emotional awareness through structured journaling.

1.3.2 Specific Objectives

- Implement a secure user authentication system.
- Let users manually log detailed trade data such as currency pair, date, session, trade type, outcome, risk-reward ratio, and emotional state.
- Make the optional CSV file import available for powerful bulk trade entry.
- Implement strategy tagging for the trades that correspond to the chosen category (e.g., Breakout, Reversal).
- Enable uploading of screenshots or TradingView links for trade reference.
- Add a personal notes section for general journaling, goal setting, and reflection.
- Develop filtering options by date, session, trade outcome, and tag.
- Produce automated weekly and monthly summaries of trading performance.
- Use Chart.js for designing an analytics dashboard that visualizes win/loss ratios, trading trends, and session-based performance.
- Add PDF export feature for saving or sharing trade records and reports.
- Construct the backend using Laravel and MySQL, and the frontend using HTML, CSS, JavaScript, and Bootstrap.
- Make sure that users can access the site on different devices through responsive web design.

- For carrying out experiments to confirm reliability, usability, and accuracy of the system.

1.4 Methodology

The journey of the creation of the Forex Trading Journal with Performance Analytics is a well-defined route. The course outlines the techniques of the data gathering, the approach to system development, design patterns, and the technologies that were used.

1.4.1 Data Collection

In order to find out Forex traders' needs and design a trading journal that fits those needs, various data collection techniques will be used:

Document Analysis – Going through Forex trading journals, spreadsheets, and software solutions to spot limitations and good practices.

Interviews – Getting the most out of structured interviews with Forex traders to find their journaling habits, challenges, and feature requirements.

Online Research – Looking at trading forums and reading literature on trade journaling to get an idea of market trends and trader preferences.

These points that are gathered will serve as an outline of the system requirements and will also be used to check if the design choices are correct.

1.4.2 System Development Model: Agile

The design of the Forex Trading Journal with Performance Analytics utilizes the Agile software development methodology that perfectly fits the projects that are flexible, require continuous testing, and are carried out feature by feature [8]. Agile divides the whole endeavor into minor units called sprints, with each sprint dedicated to creating and verifying a particular module or functionality. This model facilitates the utilization of new information, flexibility, and constant progress by means of iteration and the input from stakeholders [9].

Every sprint is organized with a definite objective, a product, and a test period. As an example, one sprint can deal with the creation of the CSV import module, whereas another will be dedicated to implementing the summary reports and their corresponding visualizations. Such a repetitive scheme guarantees that the system will still be capable of embracing changes, helps open communication among team members, and permits the implementation of continuous improvement based on feedback and testing results.



Figure 1 Agile Development Methodology

1.4.3 Design Pattern: MVC

The project is developed using the Laravel framework, which in turn is built on the Model-View-Controller (MVC) design pattern. MVC divides the application logic into three main parts:

Model (data and business logic),

View (user interface), and

Controller (intermediary that handles user input and updates models and views) [10].

Such an arrangement of concerns enables several developers to work in parallel on different parts of the system without interfering with each other. For example, one developer can focus on the backend logic while another designs the user interface. This approach makes it easier to manage code, adapt it to changes, and maintain clarity especially for medium to large-scale web applications.

1.4.4 Programming Languages and Tools

The system has been designed employing open-source technologies from the present era. A PHP framework named Laravel, which is well known for its easy syntax, comprehensive documentation, reliability features, and a lively community, was selected for the backend [11]. Laravel also provides the following resources:

- Laravel Excel to import/export CSV
- DomPDF/SnappyPDF to create PDFs
- Laravel Scheduler to automate tasks

The frontend is constructed with HTML, CSS, JavaScript, and Bootstrap, a framework that provides ready-to-use components and responsive layout utilities. Chart.js was chosen for the analytics part due to its simple, interactive, and customizable features.

Thus, by employing the Agile framework, following the MVC pattern, and using this particular toolkit, the project becomes highly adaptable, secure, and scalable, which are key features for delivering a performance-tracking system that is able to satisfy the requirements of individual traders.

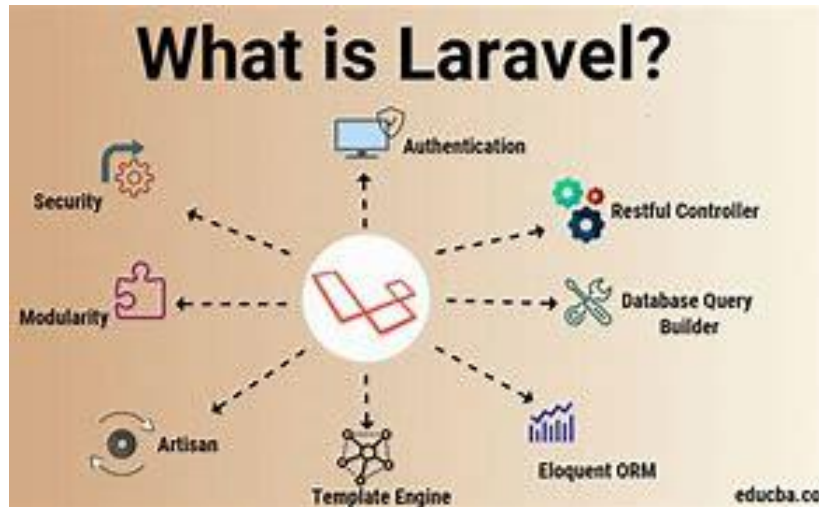


Figure 2 Laravel

1.5 Hardware and Software Requirements

This section outlines the hardware and software resources required for developing and running the Forex Trading Journal with Performance Analytics system.

1.5.1 Hardware Requirements

The following hardware components are necessary for system development and testing:

| Hardware | Specification/Model | Purpose |
|---------------------|-----------------------------------|-------------------------------------|
| Laptop | 8GB RAM, intel core i5, 222GB SSD | Development and testing |
| Mobile Devices | Android Device | Testing the system's responsiveness |
| Internet connection | WIFI and Mobile Data | Online research and deployment |

Table 1 Hardware requirements

1.5.2 Software Requirements

The system will be developed using open-source and widely used technologies to ensure efficiency and maintainability.

| Category | Software tool | Purpose |
|----------------------|----------------------------|---|
| Programming language | PHP, Javascript, HTML, CSS | Core system development |
| Backend Framework | Laravel | Simplifies backend development |
| DBMS | MYSQL | Storing and managing trade records |
| Frontend Framework | Bootstrap | Enhancing UI design and responsiveness |
| Charting library | Chart.js | Data visualization for analytics |
| Development tool | Visual studio code, XAMPP | Writing, testing and running code |
| Testing tool | postman | API testing and debugging |
| Diagram tools | Draw.io | Creating system architecture and system diagram |

Table 2 software requirements

1.6 Scope and Limitation

This section it Specifically outlines where and in what manner the system will operate (scope) as well as the restrictions (limitations) that apply to the system.

1.6.1 Scope of the Project

The main aim of the current project is to create a web-enabled Forex Trading Journal with Performance Analytics that is most suitable for solo Forex traders who desire to keep track of their trading performance in a systematic way by logging, analyzing, and reflecting on their trades.

The plan of the system is to permit users to enter trades manually or to upload trade data in CSV format. A typical entry will have comprehensive fields like currency pair, date, day, session, trade type (buy/sell), risk-reward ratio, trade reasoning, market condition, emotional state, and trade outcome. In order to facilitate the search, the users will be able to classify the trades by strategy type besides the words (such as Breakout, Reversal, News-based) in the present system.

Traders will be able to provide visual references in the form of screenshots or by inserting TradingView links for each trade. A trade history page will allow users to browse past trades by selecting different filters such as win/loss, session, tag, or currency pair. Besides this, a personal notes section will also be available in the system for users to write down trading insights, weekly reviews, and goal-setting.

The system will also supply a thorough analytics dashboard, employing Chart.js to represent in graphical form the main performance indicators like win/loss ratio, session-based performance, and strategy effectiveness. It will be an automatic agent that creates performance summaries for a week or month, which the traders can either go through or download them as PDF files for offline analysis or sharing.

This app will be reachable by means of contemporary web browsers and also it will be made suitable for both desktop and mobile uses by means of a responsive design. The system will be built with the Laravel (PHP framework) for the backend, MySQL for data management, and HTML, CSS, JavaScript, Bootstrap, and Chart.js for the frontend.

This project version of this undertaking is intended for personal and independent use, not for commercial brokerage integration or real-time trade execution. This is meant to be a self-directed performance upgrading tool for retail traders.

The Forex Trading Journal with Performance Analytics is created in a way that allows the user to log, track, and analyze these trades efficiently. The project's ambit comprises...

- Target Users: Individual Forex traders who are determined to keep track of their investing, analyze it, and thus improve their trading performance.
- Platform: A web-based application, accessible via desktop and mobile browsers.
- Geographical Scope: The system is built for traders globally, however, it cannot be used with regional brokerage regulations for the specific areas of the world.
- ❖ Core Features:
 - Trade Logging: Users can fill in the details of the trades they make, such as currency pair, trade type, session, and risk-reward ratio.
 - Trade History & Filtering: Users can look back and also filter the trades they have made in the past according to various parameters such as (win/loss, best-performing pairs).
 - Analytics & Visualization: Such charts can be used to visualize performance trends, win/loss ratios, and trading patterns.
 - User Authentication & Security: It guarantees that a trader's data will not be shared with anyone.

- **Data Storage:** The system manages trade data, performance analytics, and user activity securely via a MySQL database.
- **Time Frame:** For the first version, the system will only allow you to enter data for 1 year of trading history per user but it is possible to increase the storage in the next versions.

1.6.2 Limitations of the Project

Although the Forex Trading Journal with Performance Analytics system is certainly useful in terms of features it has for trade tracking, analysis, and self-reflection, it is also limited in several aspects, that make it different from more advanced or commercial trading platforms.

- On one hand, it is worth mentioning that the system is not designed to connect directly with live trading platforms like MetaTrader 4 (MT4), MetaTrader 5 (MT5), or TradingView via APIs. Therefore, the only ways to place trades on the system are manual entry or upload from a CSV file, which restricts the level of automation and real-time data synchronization. This might cause an inconvenience to professional or high-frequency traders who are heavily dependent on automated journaling solutions as a result.
- On the other hand, the system definitely helps the user keep track of their progress, but it does not offer real-time Forex market data feeds or charting functions. Users will not be able to open price charts, check economic events, or make trades directly on the platform. The platform is meant only for analyzing and reflecting on the trades after they have been executed, so it does not allow active and real-time trading nor does it give assistance in making the decisions on market execution.
- At the same time, the system is intended for individual traders. It lacks the ability to execute certain functions such as creating multi-user accounts, brokerage integration, or portfolio management as well as collaborating with others through team tools. Hence, it may not be the right fit for those trading firms or institutions whose main characteristic is having centralized account access or those who need to work together in multi-trader dashboards feature.
- The system is also designed as a web-based platform, which means it is continuous in requiring internet access and cannot be used offline. Moreover, although the system is mobile-responsive and can work on a typical smartphone browser, it does not have a mobile application thus the user may find it less convenient if he/she prefers using the app.
- The platform in terms of its users' inclusivity capabilities is at present supporting only English. Traders who are not good in English language might have a hard time finding the system that is easy for them to use. Besides, those traders who are unfamiliar with

structured journaling practices might need some time to get used to the detailed data input process before they can use the system smoothly.

- The system does not give users the option of advanced quantitative tools such as drawdown tracking, equity curves, correlation matrices, or risk-adjusted performance metrics like the Sharpe ratio. The system is mainly created to be a lightweight tool that guides the trader in understanding the basic patterns, emotional behavior, and trade performance in a general sense.
- The platform is self-improvement and educational product clearly, rather than a live trading or institutional analytics platform if we compare its specified extent. Although it contains a variety of practical journaling and performance-tracking features, it can never be used as a full-scale trading management software or a live-trading dashboard.

1.7 Significance of the Project

- Gives individual Forex traders an organized platform to enter and analyze trades, which enables them to train their discipline and get better long-term performance.
- Facilitates traders to register the emotional and psychological factors that affect them, thus, they have the ability to uncover behavioral patterns that influence their trading decisions.
- Gives filtering and analytics tools to locate the strengths, weaknesses, and performance trends over various trading sessions, strategies, and pairs.
- Enables strategy assessment via trade tagging, so that users perform the analysis of which techniques (e.g., Breakouts, Reversals) give more success.
- Perfect for busy traders who would like to help grow their trade journaling habits, as they may be able to return weekly and monthly summary reports generated by the tool.
- It offers an import/export CSV feature to make that extra bit of journaling lighter on the schedule, especially for those using MT4 or spreadsheets.
- Anyone who wants to be able to share data can easily extract and add trades, reports, and mentoring in a variety of versatile end-use options such as PDFs, reviews, or archives.
- Gives users a space for personal notes where they can establish goals, complete weekly reviews, and record improvements - thus, facilitating self-awareness and keeping track of goals.

- Serves as a self-learning and self-improvement instrument for novice and intermediate traders who have not yet adopted complicated analytical tools [3].
- Provides a no-cost substitute for commercial platforms that are beyond the affordability of individual traders and may be too complicated for them.
- Available on any browser or device, hence, it can be a convenient journaling tool for traders who are always moving.
- Encourages traders to become more data-driven in their decision-making process thus decreasing the influence of emotional or impulsive trades [2], [3].
- Exhibits simple, open-source software like Laravel, Chart.js, and MySQL, which can be used to generate practical, real-life examples.
- Maps out the role of self-evaluation in Forex trading, bridging the gap between trade execution and long-term strategy development [2].

1.8 Feasibility Study

A feasibility study defines that the Forex Trading Journal with Performance Analytics can be successfully developed, implemented, and maintained without any technical, economic, and operational factors limitations.

1.8.1 Technical Feasibility

This project is technically feasible due to the following reasons:

- JavaScript, and Chart.js, all of which are well-documented and widely supported.
- Scalability :A representative system architecture of the MVC (Model-View-Controller) design pattern is making it easy to maintain and expand.
- Developer Skillset: The development team has experience with PHP, database management, and front-end technologies, thus Availability of Required Technologies: The system will be built with PHP (Laravel), MySQL, implementation will be smooth.
- Hosting Requirements: The system can be hosted on a local server (XAMPP) during development and later deployed on a cloud-based hosting service.

1.8.2 Economic Feasibility

The project is financially realizable because:

- Low Development Costs: The system is based on open-source technologies, largely PHP, MySQL, Chart.js, Bootstrap which lower the cost of licensing.

- **Minimal Hardware Investment:** Only the development of the required computers and internet connection is needed, no expensive infrastructure.
- **Cost-Effective Maintenance:** The system is predicted to have negligible operational costs, as it does not require real-time data feeds or third-party API subscriptions.
- **Potential for Monetization:** The project is to be monetized via premium services, data storage in the cloud, or subscription-based models in later versions.

1.8.3 Operational Feasibility

- **Ease of Use:** The system will have a user-friendly interface that is clear and intuitive, thus enabling traders to efficiently carry out the tasks of logging, filtering and analyzing trades with minimum effort.
- **Minimal Training Required:** Because the system is designed in a way similar to standard web application, therefore users will not need a lot of training.
- **Reliable Performance:** The system is created to allow multiple users to access it without losing speed or usability.
- **Future Support & Updates:** The system is capable of being extended with additional features consonant with users' requests.

1.8.4 Legal Feasibility

- The project is not engaged in dealing with money nor does it involve real-time trading, therefore it is not subjected to stringent financial regulations.
- **Data Privacy:** The system will implement minimum acceptable security requirements in order to ensure that no infringement of users' rights to privacy takes place.
- **No Licensing Issues:** All the development tools (PHP, MySQL, Chart.js) are free and open-source software, thus there are no legal restrictions.

1.9 Risk Assessment

- Lack of experience with advanced Laravel features or third-party packages (e.g., Laravel Excel or PDF libraries) may cause development to be delayed or result in erroneous implementation of parts of the code.
- ✓ In order to reduce these effects, the team will set up periods of time which will be dedicated solely to studying the official documentation, watching tutorials, and accessing YouTube. Breaking advanced features into smaller and simpler parts and testing each one independently before integration will also be part of our strategy.

- Time restrictions resulting from academic deadlines and overlapping coursework might make it impossible to implement all the features planned.
- ✓ First, the tasks that have the highest priority, such as trade logging, filtering, and basic analytics will be implemented. We will also plan the implementation of the CSV import and PDF export in the last sprints, which will be changed if there is not enough time.
- Mistakes in data handling or inconsistencies during CSV import could make the data incorrect or cause users to become angry.
- ✓ We will ensure the correctness of the data before importing it and we will inform users in case there is any formatting error by providing a clear error message. Accuracy will be assured by relying on Laravel Excel's validation features.
- Security risks like unauthorized data access, especially for personal trade notes and performance history, could compromise user trust.
- ✓ To mitigate this threat, the system will employ hashed passwords, session-based authentication, and basic access control (middleware) to guarantee that each user only reads their own data.
- If there are bugs or the system crashes during file uploads, rendering of analytics, or report generation, then it may affect usability that can slow down progress and decrease the speed of the process.
- ✓ We will carry out unit and integration testing, specifically focus on file handling and data visualization parts. User responses in these testing phases can expose unpredicted behaviors and will enable their removal.
- The fact that the use of the Internet is a main requirement and that offline access is not available may be an inconvenience to those users whose connection is not stable.
- ✓ At least this is a part of the project scope that is known as a limitation, however, the fastest loading time will be prioritized for the platform, which a clean design and lightweight code will allow it to perform well even on low bandwidth.
- If it is not easy to organize team collaboration, especially with many tasks that include learning new things, then it can be a reason for miscommunication or delay.
- ✓ We will utilize GitHub for version control. Also, Nous will organize weekly planning meetings to divide tasks, track progress, and set clear responsibilities for the team members, hereby making it easier to collaborate, communicate, and be on the same page.
- Not knowing well the libraries that are used for the creation of charts or doing the data visualization incorrectly might cause analytics to be less meaningful or even unclear.

- ✓ Since it is simple and has good documentation, we will employ Chart.js. Besides, we will test several chart types with real and sample data to be sure visual clarity and accuracy are at the highest level.

1.9.2 Assumptions

The project is founded on the below-mentioned points:

- Traders are optimistic that they will manually input their trading data.
- Users will have minimal Forex trading knowledge and will comprehend the terms related to trades.
- The system will be used over a reliable internet connection.
- The team will be able to learn Laravel and other necessary tools within the project timeline.
- Besides those security measures like authentication and encryption, they will be enough to protect user data.

1.9.3 Constraint

In the course of the Forex Trading Journal project, the development group are faced with a number of constraints which they can't change. For instance:

- Internet Connectivity: This system is based on the internet and thus needs a strong connection. Service interruptions in the internet (for example, if Ethio Telecom has no service) will cause users not to access the system and also the system performance to be affected.
- Third-Party Services: Integrating such features as PDF export, chart libraries, or CSV handling that are dependent on third-party tools or APIs is quite risky. Reputationally, the changes or downtime can cause our system to lose some of its features without our being aware of it.
- Device Compatibility: Indeed, making sure that the journal is accessible on all devices is quite challenging. Users may access the journal from various browsers or devices with different screen sizes and operating systems. Ensuring perfect compatibility across all of them is not fully within our control.
- User Behavior: The users, for instance, may intentionally or unintentionally provide incorrect or partial information about trades. While we can provide validation checks, we cannot control how users interact with the system.

- Time and Academic Schedule: Therefore, the commitment to be efficient throughout the academic calendar, to submit all your assignments on time, and to do the different tasks requested in class is fundamental to make good progress as a student.

1.10 Work Breakdown Structure (WBS)

The Work Breakdown Structure (WBS) diagram represents the project tasks and phases, thus, it is very helpful for the team to effectively manage time, resources, and responsibilities. The WBS identifies the five major phases of the project: Planning, Analysis, Design, Implementation, and Testing & Deployment.

1.10.1 WBS Table

| Phase | Tasks |
|------------------------|--|
| Planning | Finalize project topic- assign team roles- draft project proposal |
| Requirement analysis | Gather functional and non-functional requirements- conduct literature review-select development methodology (Agile) |
| Design | Design UI wireframes and user flow- create database schema- plan system architecture |
| Implementation | Setup development environment (PHP, MYSQL, LARAVEL)- build authentication module- implement trade entry from- develop trade history and filtering- integrate chart.js for analytics- implement file upload and data validation |
| Testing and deployment | Conduct unit and integration testing- fix bugs and optimize performance- deploy webserver (XAMPP) |

Table 3 WBS Table

1.10.2 Gantt Chart

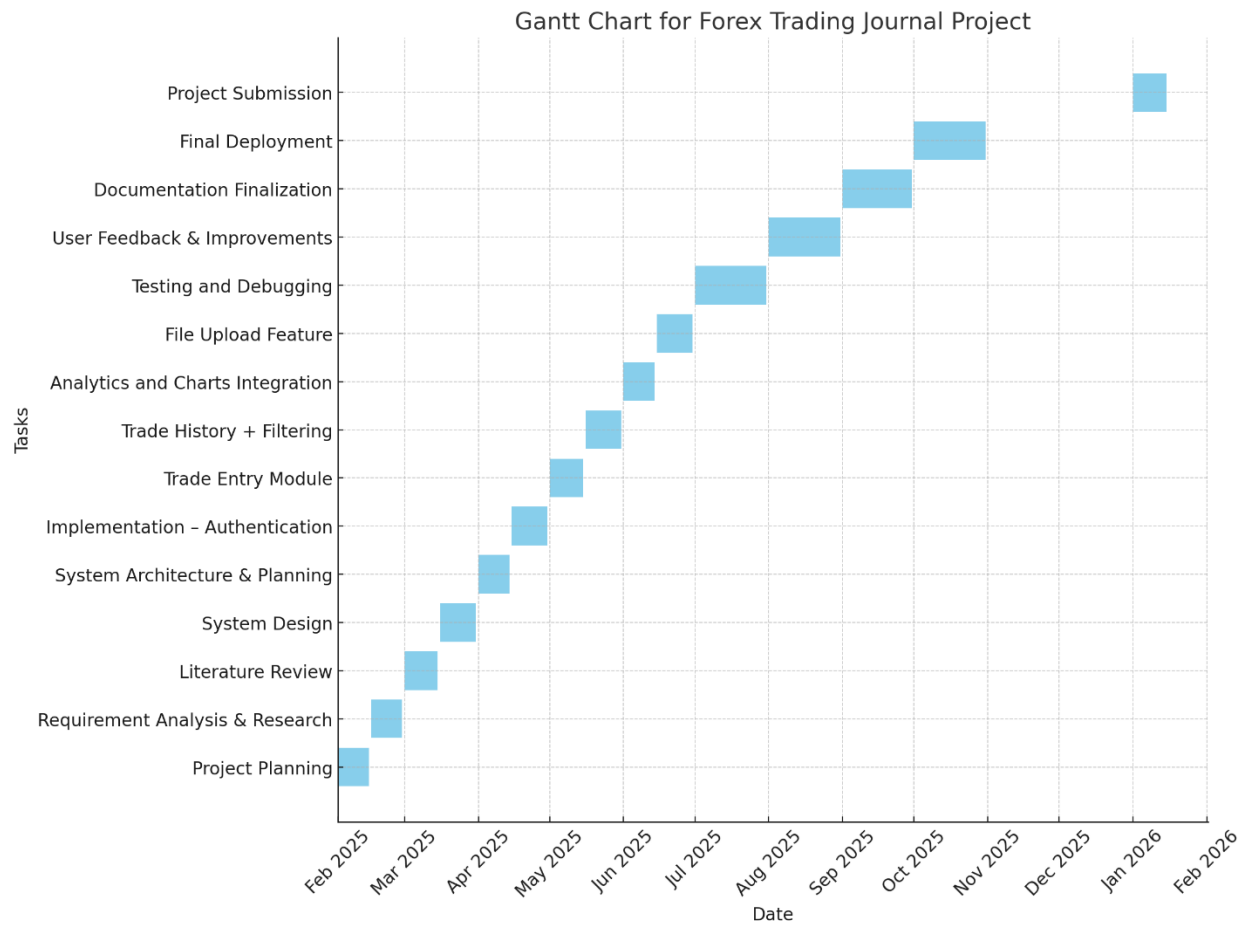


Figure 3 Gantt Chart

Chapter two: Business Area Analysis And Requirement Definition

2.1 Introduction

This chapter closely examines the business field and requirement structure of the Forex Trading Journal with Performance Analytics system. It starts by highlighting the common problems that Forex traders encounter such as inconsistent journaling, lack of structured feedback, and the absence of meaningful performance metrics. Thousands of traders worldwide still rely on spreadsheets or notebooks that may limit their capabilities in evaluating long-term performance, identifying emotional or strategic weaknesses, or getting thoughts from experts. Besides, such conventional means also don't provide data visualization and psychological analysis features, which are vital to trading discipline improvement and strategy polishing [2], [3].

The solution is a web-based, structured logging system for trades, and it keeps all the technical, emotional, and strategic details. It is also designed to provide the trader with automated visual performance reports such as win/losing ratios, session-based outcomes, and trading trends. It also brings in the idea of a collaborative learning loop with a Performance Analyst role, who can use the trade histories and analytics to provide written feedback based on the observed issues. An Administrator role, intended for managing user accounts and watching out for system-wide activity, has also been added. This section of the thesis goes on to outline the system's functional and non-functional requirements along with business rules, and also use case models, user interface prototypes, collaboration diagrams, and system modeling diagrams. At the end of this chapter, readers will grasp the system's functional scope, three primary user roles—Trader, Administrator, and Performance Analyst, and how each part deals with real-world issues in trading performance analysis.

2.2 Current System

In the present Forex trading market, unchecked and misshaped methods are being utilized by individual and self-educated traders, authors of these trades unwittingly assuming a very limited and unreliable understanding of changes taking place in markets. **Informal notes in the form of notebooks, Excel, or Google Sheets, and tools like Notion, Evernote, or Google Keep are very common means among the unstructured data case.** Certainly, these apps provide users with the opportunity to customize and make their own, but on the flip side, they suffer from many structural, data, usability, and security issues, and perhaps have limited capabilities in analytical functions at later stages.

Take for example the case where a trader opts to use spreadsheets as a tool for recording of trades. Here, **manual calculation of profit/loss, win/loss ratios tracking, and chart creations**

will be required. The problem is that this manual approach is time-wasting, troublesome, and does not imply the habit of consistent journaling. Moreover, qualitative data such as trade reasoning, psychological state, and market conditions, which-by their nature-are difficult to standardize and evaluate, become harder to manage within these tools [1].

On the other hand, several professional journaling platforms have been introduced to cover these needs and are as follows:

- **Edgewonk** is a complete, albeit paid, app specifically aimed at experienced traders that offer strategy tagging, and performance metrics, among other features [4].
- **Trademetria** – Proffers analytics and broker integration but part of it is free access with limiting only a few trades per month [5].
- **Myfxbook – Automatically.** Yet, no one psychological feature is present. [6].

On one hand, the functionality of the tools is highly appreciated; however, **local traders, student developers, or educational settings face several limitations.**

- **Cost barriers:** The majority of the tools aimed at professionals need subscriptions on a monthly or yearly basis, which might be beyond the budget or unrealistic in places where there are no efficient online payment systems.
- **Over-complexity:** The utilization of advanced functions and dashboards in Edgewonk-like software may confuse inexperienced or average users that just want a systematized journaling and some feedback.
- **Unavailability of educative features:** These platforms do not facilitate interaction activities that are learning oriented between the trader and the performance analyst, the mentor, or the instructor within the system.
- **Dependence on broker integration:** Users who have demo accounts, are engaged in manual journaling, or make manual journaling as their primary habit, will find these systems lacking or less adequate.
- **Absence of direct feedback mechanism:** There is no internal way for a third party to access the performance of the trader and give comments, suggestions, or coaching inside the system.

Besides, the means by which traders and mentors communicate are as a rule outside any journaling system—via well-known platforms like **WhatsApp, Telegram, Discord, or e-mail.** This disconnection hinders the ability to trace feedback of experts in relation to specific trades. More often than not, past feedback is not well organized and may lose its context if it is separated from the actual trading data.

This means, however, that they lack the opportunity to become their own masters and to receive structured guidance if they don't get performance analytics and contextual feedback. Without

these two elements, it is common that traders are repeatedly making the same emotional or strategic mistakes but are unaware of it.

2.2.1 Players of the Existing System

In the existing trade journaling market, a lot of individuals and the use of different tools play a major role in influencing or assisting a trader's performance in monitoring in an indirect manner. These players' roles give insight into their collective energy, location, and importance in supporting individual traders' operations. Understanding their roles helps justify the need for an integrated platform.

1. Individual Trader

The individual trader is the primary actor in the current system of trading. They are those who make trades and also try to record their trades informally through the use of tools like notebooks, Excel, Google Sheets, or note-taking applications such as Notion or Evernote [4], [5]. Most of the traders, however, are in a situation where they cannot find a good habit of regularly doing the following:

- Recording emotional states
- Evaluating trading strategies
- Tracking results over time

Consequently, a decline in discipline due to the inability to get structured feedback or performance metrics and the repeated mistakes happening result in the situation [8], [9].

2. Trading Mentors or Coaches

One group of traders actually find mentoring to be a driving force through the mentorship they get from the online community, Telegram groups, or paid coaching sessions [7]. A sample of such a scenario is that the mentors are willing to request the traders to give them the necessary information such as screenshots, spreadsheets, or even a brief summary of their trades that they might have at hand for them to make the review. But,

- Feedback is often unstructured and informal
- There is no consistent link between feedback and specific trades
- Mentors cannot interact with a centralized trade journal
- Learning opportunities are lost due to fragmented communication

3. Analyst Role (Unsystematized)

In more advanced or private trading groups, experienced members sometimes act as informal performance analysts. They review screenshots or logs and offer feedback based on behavioral patterns or trading results. However:

- This role is not integrated into any journaling system
- Feedback is often delayed, scattered across chats, or lacks context
- Analysts cannot access a trader's full performance history or spot trends
- This limits the depth and long-term usefulness of the advice

4. Community Platforms

People often go to Telegram, Discord, and Facebook Groups for quickly sharing and receiving support on their trades. Platforms such as Telegram, Discord, and Facebook Groups are popular among traders for sharing trade ideas, charts, and emotional support [7]. However:

- These environments lack structure
- There are no privacy controls for sensitive trade data
- There is no feedback tracking or performance integration

Such platforms may be socially useful but they are not intended for journaling or analytics in the systematic manner.

5. Journaling Tools (Used as Workarounds)

This is the case when the traders make use of:

- Excel/Google Sheets for data tracking
- Notion/Evernote for journaling and note-keeping
- Myfxbook or Trademetria for automated analytics via broker integration [4], [5], [6]

These tools may be good for supporting the functions that relate to the emotional logging, analyst feedback workflows, and real-time performance improvement strategies. Therefore, they remain fragmented solutions that do not correspond to the evolving needs of traders.

2.3 Business Rules

Business rules specify standard procedures, limitations, and policies, which set the tone for how users interact with the system and how the system changes its data, as well as, operations. The rules are indispensable for ensuring data consistency, operational integrity, and Forex Trading Journal with Performance Analytics logical flow of the system.

The business rules below are those that relate most directly to this project:

1. User Authentication Rules

- Only registered users (Trader, Admin, Performance Analyst) can access the system.
- Traders and Analysts must give valid usernames and passwords to be able to log in.
- User sessions must be automatically logged out after a period of inactivity (e.g., 15 minutes) for security purposes.
- Passwords must be at least 8 characters long and include numbers and symbols.

2. Trade Data Entry Rules

- Traders have to be logged in if they want to record, edit, or delete trades.
- Every trade has to contain the required fields:
 - Currency Pair
 - Trade Type (Buy/Sell)
 - Trade Date & Day
 - Risk-Reward Ratio
 - Session
 - Optional fields like Emotions, Market Conditions, and Screenshot Upload can be left empty but are recommended.
 - Trades are final after a certain time period (e.g., 48 hours) and hence, cannot be modified or deleted unless approved by Admin.
- **3. Trade History and Filtering Rules**
 - The trader who has done the trade and is the only one who can see is the one who submitted the trade.
 - The filters need to have the following characteristics: pair, date range, win/loss, session, and performance tag.
 - If a trade is being deleted, it will be designated as “archived” but not erased for it is needed for the continuation of the historical analytics.

4. Analytics and Reporting Rules

- Analytics dashboards must be always updated if a new trade happens or an existing one is amended.
- Visual representations (e.g., win/loss ratio, best pairs, performance trends) should display only the data of the logged-in trader.

5. Feedback Rules

- Performance Analysts are only able to submit feedback for those traders whom they have been assigned to review.

- Each time a feedback is submitted, it should contain a brief of observations, strengths, and areas for improvement.
- Traders have the option of looking at the feedback given to them, but they are not allowed to reply or make any changes in the system.
- The feedback must be accompanied with a timestamp, and it should be possible to change only within a limited period (for example, 24 hours after the submission).

6. Admin Management Rules

- An Admin is entitled to perform any of the following operations: create, suspend, or delete any user account.
- Admin is in charge of saving trade and user data throughout a given period on a regular basis (e.g., every week).
- An Admin can access all trade records for observation and control work.
- Only Admin has the operation of restoring deleted data or making system-wide configuration changes.

7. Data Integrity and Validation Rules

- Before sending, the system should check all the form inputs for validity.
- Duplicate entries (e.g., same trade recorded twice in the same minute) should be flagged.
- During submission, all the fields marked as mandatory cannot be omitted.

The business rules that are mentioned here are the functional constraints and the logical guidelines that are intended for the user to follow, for the system to be intact and for the ease of the system being maintained and expanded.

2.4 Proposed System

Forex Trading Journal with Performance Analytics, the proposed system, is a web-based application that assists Forex traders in recording, managing, and analyzing their trading activities. The system also enables the performance evaluations of analysts, and administrative management by the system admins. The platform aims to provide a centralized, data-driven platform that improves trading discipline, enables performance tracking, and enhances decision-making for traders.

First, the system tackles the drawbacks of journaling through paper or digital scattered methods by introducing a more organized and user-friendly workplace where trades can be stored with detailed attributes like currency pair, session, trade type, reasoning, risk-reward ratio, emotional state, and additionally screenshots or links for support. After the entry has been made, the system

can produce lots of graphical reports like win/loss ratios, performance trends, and summaries by day, session, or pair.

Secondly, the same system enables Performance Analysts to go through profiles of individual traders, trade history, and analytics. They are able to give personalized comments for every trader, thus, they will be able to spot their faults and change the trading psychology and strategy that is compatible with them [2],[3]. This response thus turns into the trader's journaling section, the very self-awareness and learning it invites thus being made.

Administrators have complete control over the system and can thus carry out activities such as managing user accounts, tracking the platform, backing up, and restoring data. Access provisions based on role help users to get involved only with those areas of the system that relate to their duties.

The system is architected in a three-tier structure, which includes:

- **Presentation Layer:** It leverages HTML, CSS, JavaScript, and the bootstrap framework. Bootstrap provides responsive components and utility classes for rapid front-end development [7].
- **Application Layer:** Implemented with Laravel, a PHP framework widely known for its elegant syntax, MVC pattern, routing, validation, authentication, and large community [8].
- **Data Layer:** Utilizing MySQL, an open-source relational database system that is perfect for storing structured trade data, user accounts, and performance analytics, as well as being widely used by the open-source community to build the system [9].

To represent trader performance visually, the system includes Chart.js, a free JavaScript library which makes it possible to construct vivid and interactive charts and figures [6]. Win/loss ratios, session performance, and strategy outcomes from recorded trades are just the reports that can be generated. Traders may use the export libraries that have been implemented in the Laravel environment to download their data in PDF or CSV formats.

This architecture facilitates modularity, maintainability, and scalability, which are the key features for supporting secure and efficient performance, growth with user demand, and at the same time being energy-efficient. The solution proposed in the new paper can automatically do the journaling task of Forex trading, and additionally, it can convert the raw trading data into a good decision-making material, which makes it a very useful tool for both beginners and experienced Forex traders Section.

2.4.2 Requirement Definition

The requirement definition phase outlines what the proposed system must accomplish to meet the expectations of its users. The Forex Trading Journal with Performance Analytics system is designed to provide a structured platform for traders to log, manage, and analyze their trading data while allowing analysts and administrators to perform their roles effectively. The system

requirements are derived based on a thorough understanding of the business process and feedback from potential users.

The system must provide three types of users—Traders, Performance Analysts, and Admins—with appropriate interfaces and capabilities based on their roles. Traders are the primary users of the system. They require functionality to register, log in, and manage their trades by recording details such as currency pair, trading session, trade type, risk-reward ratio, reasoning, psychological state, and attachments like screenshots. Additionally, they should be able to review their trade history, filter it using various criteria (such as date, session, or outcome), and visualize performance trends using analytics dashboards. The system must also allow traders to export reports in PDF or CSV format and view feedback given by analysts.

Performance Analysts should be able to log in securely, view a list of traders, access trade histories and performance charts for each trader, and submit structured feedback. Their feedback must be stored and made visible to the respective trader. Optionally, they can update their submitted feedback within a restricted time window (e.g., 24 hours).

Admins are responsible for overseeing the entire system. Their duties include creating and managing user accounts, monitoring system usage, performing data backups, and restoring data when necessary. Admins must also be able to view system logs and ensure the platform runs reliably and securely.

2.4.2.1 Functional Requirements

Functional requirements define the specific behaviors, actions, and operations the proposed system must perform to meet the needs of its users. These requirements are based on the features expected by the three main user roles in the system: Trader, Performance Analyst, and Admin. The following list outlines the core functionalities that the system must provide to ensure a complete and effective solution.

General System Functionalities

- The system shall allow users to register and log in securely based on predefined roles: Trader, Performance Analyst, or Admin.
- The system shall manage role-based access, ensuring that each user type only accesses authorized features.
- The system shall provide a user-friendly dashboard with access to all relevant modules based on the user's role.

Trader Functionalities

- The system shall allow traders to record new trades by entering details such as currency pair, trade type, session, risk-reward ratio, date, reasoning, psychological state, and optional screenshot or chart links.
- The system shall validate all required fields before saving a trade entry.
- The system shall allow traders to view, edit, or delete their own trade entries, with editing or deletion restricted to a configurable time window (e.g., 48 hours).
- The system shall provide filtering options on the trade history page, including filters by date, session, currency pair, and trade result (win/loss).
- The system shall display analytics dashboards, including win/loss ratios, performance over time, and best/worst trading sessions and pairs.
- The system shall allow traders to export their trade history and performance reports in PDF or CSV format.
- The system shall allow traders to view feedback provided by performance analysts.
- Performance Analyst Functionalities
 - The system shall allow performance analysts to log in using their assigned credentials.
 - The system shall provide access to a list of registered traders.
 - The system shall allow analysts to select a trader and view their trade history and analytics dashboards.
 - The system shall allow analysts to submit performance feedback for a specific trader.
 - The system shall allow analysts to update previously submitted feedback within a limited time window (e.g., 24 hours).
- Admin Functionalities
 - The system shall allow admins to manage all user accounts, including creating, updating, deactivating, or deleting trader and analyst accounts.
 - The system shall allow admins to view system activity logs and monitor login sessions.
 - The system shall allow admins to perform system-level tasks such as database backup and data restoration.

- The system shall enable admins to access all trade records and feedback entries for auditing purposes.
- These functional requirements serve as the operational backbone of the system. They guide the system's architecture, ensure role-based interactions, and establish the foundation for system modeling and implementation.

2.4.2.2 Essential Use Case Modeling

Essential use case modeling describes what the users want to achieve through the system, abstracting away the implementation details. It helps identify the core user interactions and system responsibilities in terms of goals rather than physical interfaces or UI design. This approach supports early requirement validation and lays the foundation for detailed system design.

For the Forex Trading Journal with Performance Analytics system, the essential use cases are derived from the three primary actors: Trader, Performance Analyst, and Admin.

Essential Use Cases by Actor

| Use Case | goal |
|----------------------|---|
| Record new trade | To log trade details such as currency pair,session,reasoning |
| Edit trade entry | To modify trade data within the allowed time |
| Delete trade entry | To remove the trade from history if permitted |
| View trade history | To access and review previous recorded trades |
| Filter trade history | To search trades based on pair,date,session and result |
| View analytics | To view win/loss ratio,trends and best/worst days or pairs |
| Export trade report | To download trading records and performance data in PDF or CVS format |
| View feedback | To read feedback submitted by a performance analyst |

Table 4 Trader

| Use Case | Goal |
|---------------------------|--|
| View Trader List | To access a list of registered traders |
| View Trader Performance | To review trade history and analytics for individual traders |
| Submit Feedback | To give personalized performance evaluations for a trader |
| Update Submitted Feedback | To revise feedback within a limited time frame |

Table 5 Performance Analyst

| Use Case | Goal |
|------------------------------|--|
| Manage user Accounts | To create, deactivate, or delete trader and analyst accounts |
| View system logs | To monitor login activities and overall system usage |
| Perform backup | To create backups of user data and trade records |
| Restore data | To recover data in case of failure or loss |
| View all trades and feedback | to audit trader and analyst activity across the system |

Table 6 Admin

These essential use cases reflect the user's intent without specifying how the system will implement them. This abstraction is especially useful during early design and system validation stages, as it focuses on user goals and value rather than technical solutions.

2.4.2.3 Essential User Interface Prototyping

User interface prototyping represents the early visual layout of how users will interact with the system. It helps stakeholders and developers understand the look, flow, and functionality of the system before implementation. These prototypes are not final designs but serve as conceptual blueprints for the actual user interface.

For the Forex Trading Journal with Performance Analytics system, prototyping focuses on interfaces that serve the three user types: Trader, Performance Analyst, and Admin. The goal is to ensure each role has access to intuitive, well-organized, and role-specific views.

A. Trader Interface Prototypes

1. Trader Dashboard

- Displays a welcome message, quick links (Record Trade, View History, Analytics)
- Sidebar menu: Dashboard, Add Trade, Trade History, Analytics, Feedback, Export, Logout

2. Trade Entry Form

Fields:

- Currency Pair (dropdown)
- Date and Day (auto and dropdown)
- Trade Type (Buy/Sell)
- Session (Asian, London, New York)
- Risk-Reward Ratio (numeric input)
- Entry Reasoning (textarea)

- Market Conditions (textarea)
- Emotions/Psychology (textarea)
- Screenshot Upload or TradingView Link (optional)
- Personal Notes/Strategy (optional)
- Buttons: Submit, Reset

3. Trade History View

- Table format showing recent trades
- Filtering options (pair, session, result, date)
- Buttons: Edit, Delete per row

4. Analytics Dashboard

- Graphs generated using Chart.js
- Metrics shown:
 - Win/Loss Ratio
 - Best Trading Days
 - Most Traded Pairs
 - Monthly Performance

5. Export Reports Page

- Options to download as PDF or CSV
- Filter range by date

6. Feedback Viewer

- Display analyst feedback with date and analyst name
- Read-only area

B. Performance Analyst Interface Prototypes

1. Analyst Dashboard

- Sidebar menu: Dashboard, Trader List, Feedback Submissions, Logout

2. Trader Review Page

- Select trader from dropdown or search bar
- View trade history and analytics for selected trader
- Form to write performance feedback

3. Feedback Management Page

- View submitted feedback
- Edit if within 24-hour limit

C. Admin Interface Prototypes

1. Admin Dashboard

Quick stats (active users, recent exports, system uptime)

2. User Management Page

- List of users (filter by role)
- Options to Add, Deactivate, or Delete users

3. Backup & Restore Page

- Button to create a new backup
- Upload and restore from a previous file

4. System Log Viewer

- Timeline view of login history and admin actions

These interface prototypes serve as the foundation for front-end development. They emphasize clarity, ease of use, and role-specific access, ensuring the system remains intuitive and task-oriented for each type of user.

2.4.2.4 Collaboration Modeling

Collaboration modeling describes how different system components and users (actors) interact with each other to accomplish specific tasks. It focuses on the structural relationships between objects and the sequence of messages exchanged between them within a particular use case. This model is useful in identifying roles, responsibilities, and object communication paths.

For the Forex Trading Journal with Performance Analytics system, collaboration modeling highlights interactions among objects such as Trader, Trade Controller, Database, Analytics Engine, and Feedback Module, depending on the scenario. Below is an example collaboration scenario involving a trader recording a new trade.

Scenario: Trader Records a New Trade

Actors/Objects Involved:

- Trader
- Trade Entry Form (UI)

- Trade Controller (Back-end Logic)
- Database
- Analytics Engine

➤ Interaction Summary:

1. The Trader fills out the trade entry form and submits it.
2. The UI captures the form data and forwards it to the Trade Controller.
3. The Trade Controller validates the data and sends it to the Database for storage.
4. Upon successful storage, the controller triggers the Analytics Engine to update performance metrics.
5. The result (success or error) is returned to the trader via the UI.

This collaboration shows both structural links (who talks to whom) and dynamic message flow between components. It ensures clear responsibilities for data processing, logic handling, and response generation.

2.4.2.5 Non-Functional Requirements

Non-functional requirements specify the quality characteristics of the planned system that refer to the factors of usability, performance, reliability, and maintainability. While functional requirements describe what the system should do, non-functional requirements specify how well the system carries out the tasks.

1. Usability

The system must have a clear simple user interface that is nevertheless consistent across different pages. All users, without exception, must be able to move around the platform with minimum training. Form validations and error messages must be clearly presented to users as they perform each task, thus adhering to Nielsen's usability principles [10].

The system shall execute user commands (for example, form submission, page loading) within 3 seconds under normal network conditions. This target performance is in accordance with usability guidelines which indicate that users perceive responses within 2–3 seconds as acceptable [10]. Furthermore, chart rendering and trade history filtering shall be eased so as not to interrupt the user experience.

3. Security

The authentication of users will be done using encrypted passwords and the handling of sessions in a secure manner. Role-based access control will provide the guarantee that the users will only be able to carry out those operations, which they have been authorized. These controls follow the

guidelines of the Open Web Application Security Project (OWASP) [11] security best practices. Operations that are very sensitive—like deleting trades, switching off users, and restoring data—will still need the confirmation or the higher privilege of an admin.

4. Scalability

The system shall be developed in such a way that it can handle an increased number of users and trade records without significant degradation in performance. Code designing in a modular way and the correct use of indexing in the database will greatly help in supporting both horizontal and vertical scaling. Laravel's MVC structure and Eloquent ORM aid in scalability and well-structured development [8].

5. Maintainability

The system shall consist of modular, well-documented code that makes it easy to implement future changes as well as to debug. Adopting the Laravel framework will inevitably result in a clean architecture, reusable components, and consistency across the codebase [8].

6. Reliability

The system shall have an uptime of 99%, excluding scheduled maintenance periods. Appropriate exception handling, input validation, and error logging mechanisms will be put in place to limit crashes and ensure that recovering from unexpected problems is smooth. This level of availability is in line with the general consensus of what is expected for non-critical business applications [12].

7. Backup and Recovery

The system shall be capable of automated weekly backups as well as manual ones which administrators can carry out. In the event of data loss, corruption, or migration, admins will be able to restore system data from backup files.

8. Portability

The system shall not only be reachable through leading web browsers, such as Chrome, Firefox, and Edge, it shall also be flexible enough to work on different screen sizes, which include desktop, tablet, and mobile. No operating-system-specific features shall prohibit deployment on common platforms.

2.4.3 System Model

2.4.3.1 Use case Model

The Use Case Model represents the manner in which users externally interact with the Forex Trading Journal System. Trader, Performance Analyst, and Admin: it defines what the system does according to them. When each actor interacts with the system, specific tasks aligned with their role as well as permissions are accomplished. Prior to when implementation begins, use case modeling ensures the system supports each of the core user interactions as well as helps clarify user requirements. In relation to use case diagrams, this model follows after the Unified Modeling Language (UML) standard [13].

➤ Trader Use Cases

It is the Trader's responsibility to record each trade in addition to reflecting on the trades themselves because they are the system's main user. The system can offer functionalities for this one actor. These are the following.

- Make a new user to register account details. An account can then be accessed by someone.
- Authenticate to log in to the system for access to the trader dashboard.
- With a new trade, input all required fields: pair, date, session, strategy, and emotion.
- Trade Entry Edit: Data on Trade Modified.
- Deletion of trade: Trades that are saved are then removed.
- View Trade History: Get all trades performed in chronological order.
- Filter Trade History: Use criteria such as date, session, strategy, or outcome to filter your trade history.
- View Analytics Dashboard: Get an overview of performance such as the win/loss ratio, strategy efficiency, and session results.
- Export Trade Report: Make and save a copy of your trades in PDF or CSV format.
- View Feedback from Performance Analyst: Read performance improvement points starred by the analyst.
- Update Profile: Modify your personal information such as password or email.

➤ Performance Analyst Use Cases

A Performance Analyst is a person who monitors trader activities and sends him/her personalized insight. The system allows the analyst to:

- **Login to System:** Use secure authentication to get to the analyst side of the system.
- View List of Traders: See the list of traders present on the platform.

- **Select Trader Profile:** Define a trader you want to investigate.
- **View Trader's Trade History:** Get access to records of trader's executed trades.
- **View Trader's Analytics Dashboard:** Go over graphical summaries of the trader's decision.
- **Submit Performance Feedback:** Compose your argumentation of the performance plus your impressions of the data.
- **Update Submitted Feedback:** You can change your words in case of misinterpretation or add new information if it occurs.

➤ **Admin Use Cases**

Admin is a person responsible for the area of the system and the management of the data(core). The system extends the following utilities to the Admin.

- **Login to System:** Sign in to the admin panel with elevated privileges.
- **Manage User Accounts:** Modify, create, or deactivate trader and analyst accounts.
- **Monitor System Activity:** See user logins, operations, and overall activity logs.
- **Backup Trade Data:** Carry out backups of trade records for recovery and migration purposes.
- **Restore Trade Data:** Use backup files to restore data in case of system failure.
- **View All Trade Logs:** Get the trade history in entirety across all users.
- **View System Usage Reports:** Produce reports on platform activity, user statistics, and system health metrics.

This Use Case Model makes sure that all system roles are well understood and the core functionalities are covered. The division of roles between Trader, Analyst, and Admin not only allows a modular system design but also reduces the complexity of both implementation and testing.

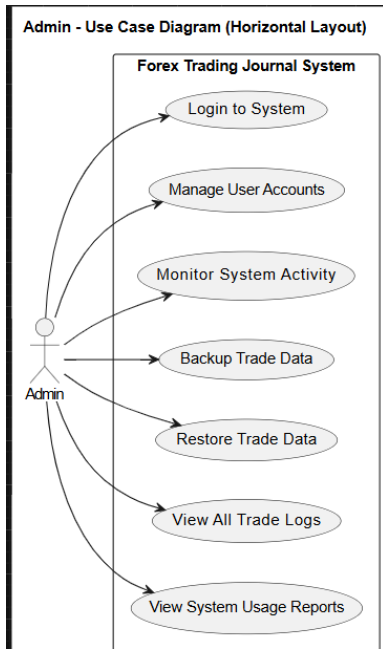


Figure 4 Trader use case

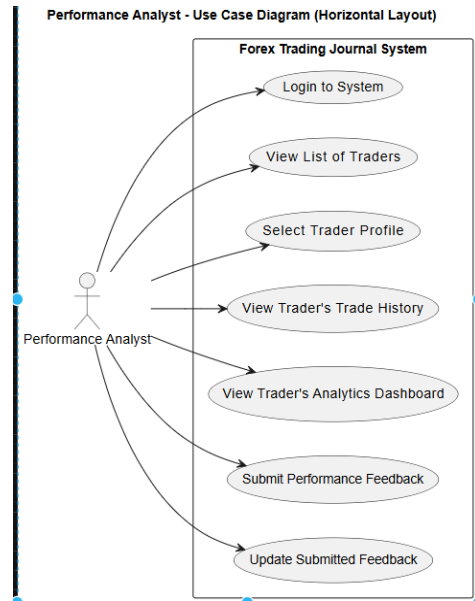
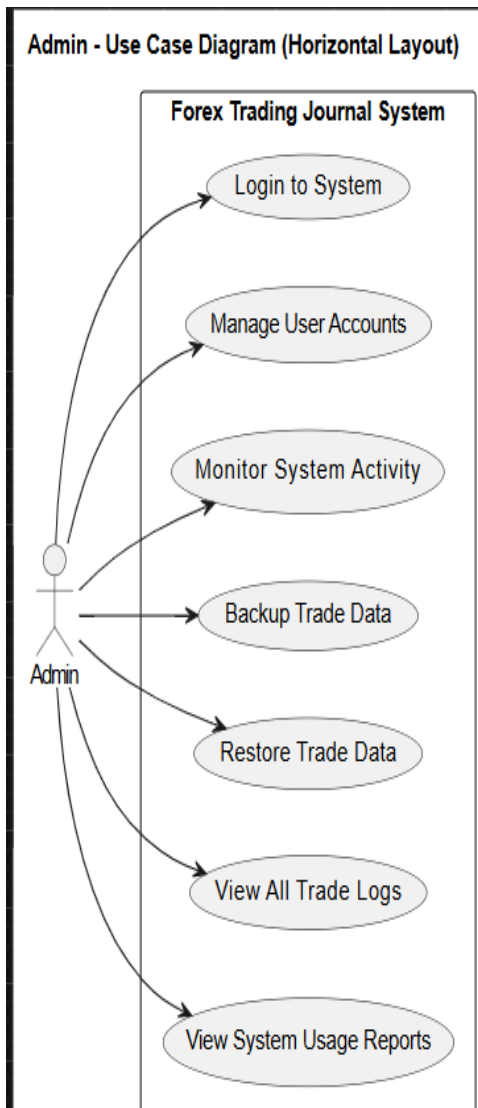


Figure 5 performance Analyst use case



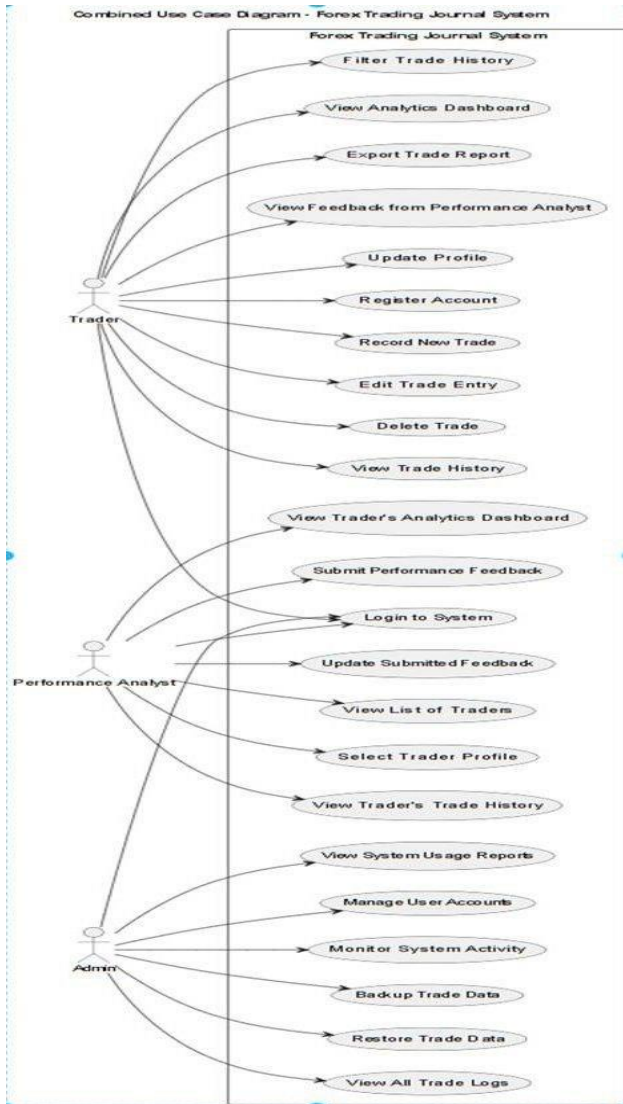


Figure 6 Admin use case

Figure 7 All Actors Use Case

2.4.3.2 Sequence Diagram

2.4.3.2.1 Sequence Diagram (Final Explanation with UML Reference)

The sequence diagrams manifest the system components and user actors' dynamic interactions that are time-ordered sequences. The diagrams represent the features of the

Forex Trading Journal System that behave differently depending on the user actions. They are very helpful in understanding the message flow, execution of logic in the order and integration of different system layers such as UI, controller, database, and external modules (e.g., analytics engine, feedback module) because of their visual interpretation.

The sequence modeling adheres to the UML conventions, which is a standardized method to represent the behavior of the system over the period of time [13].

1. Trader Interactions

The Trader sequence diagrams portray the following major interactions:

- **Record New Trade:** Trader submits a form → Controller validates → If valid, data is saved → Confirmation is shown.
- **Edit Trade:** Trader selects a trade → Trade data is retrieved → Form is updated → Saved changes are confirmed.
- **Delete Trade:** Trade selection triggers a soft delete (archival) → Confirmation returned.
- **Filter Trade History:** Filter criteria are submitted → Controller queries the DB → Filtered results are shown.
- **View Analytics:** Controller fetches trade history → Analytics engine generates charts → Visuals are displayed.
- **Export Report:** Export option triggers data retrieval → Report generator produces file → User downloads report.
- **View Feedback:** Feedback controller retrieves comments from DB → Displayed to trader.

This diagram shows the relationship between the presentation layer (UI) and the controller, as well as with the backend modules, which manage trader actions in an efficient, easy-to-use manner.

2. Performance Analyst Interactions

Performance Analyst diagrams illustrate the analytical and feedback-related use cases:

- **Login:** Analyst credentials are verified → Access granted.
- **View Trader List:** A request is sent to the DB to return all user data for all traders.
- **Select Trader & View History:** Stored trade data for a particular trader is displayed.
- **View Analytics Dashboard:** The analytics engine receives the data → Returns charts that are then visualized.
- **Submit Feedback:** The analyst enters the comments → The feedback module registers the data → A confirmation appears.

- **Update Submitted Feedback:** The copy of the previously saved feedback goes through the updating process → It is saved and the system sends a confirmation.

These flows depict the system's support for professional analysis and feedback submission without the need of direct interaction with the trade records, thereby maintaining the division of roles.

3. Admin Interactions

Admin use cases pinpoint the maintenance of the system, management of accounts, and security of data as their main focus:

- **Login:** Admin credentials are verified → Dashboard access is granted.
- **Create New User:** The form data is sent to the DB and a new record is created → The system confirms that it is successful.
- **Deactivate/Delete User:** The user who is deleted or deactivated is chosen → The DB changes the status of the user or deletes the record.
- **Backup/Restore Data:** Backup causes a script to be executed → The system returns confirmation of the operation. Restore operates in a similar way using backup files that were selected.
- **View System Logs:** Admin needs the logs and asks the system to retrieve them → System gets the logs and displays them for reading.

The Admin workflows show us that administrative actions are protected and verified by multi-step conversations that usually go along with the database and controller layers.

Such sequences are a big factor in grasping the scheduling, rationale, and validity of the system functions of various roles, which in turn helps to be sure of the proper flow control, limiting of access, and modular interaction.

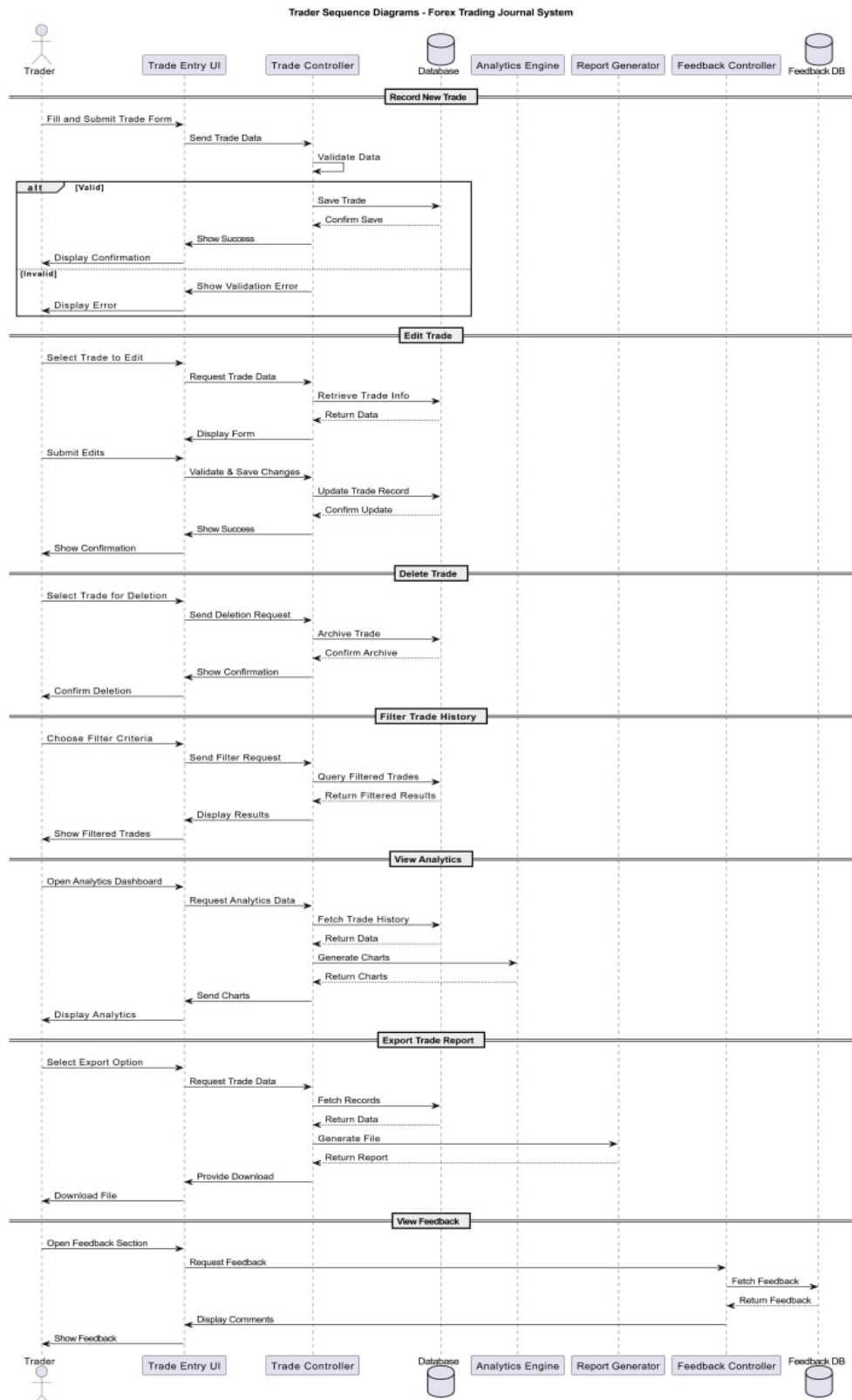


Figure 8 Trader sequence diagram

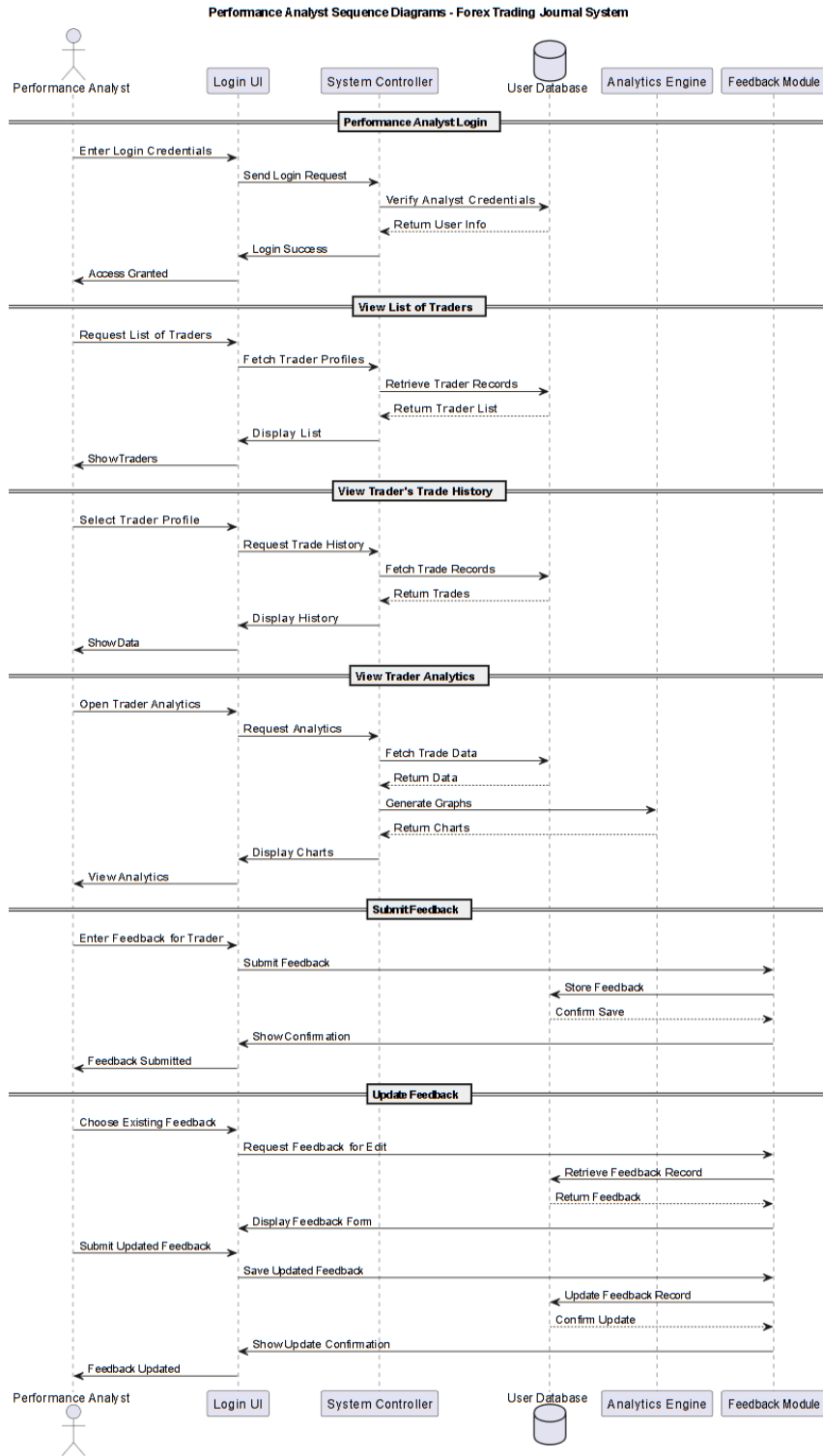


Figure 9 performance Analyst sequence diagram

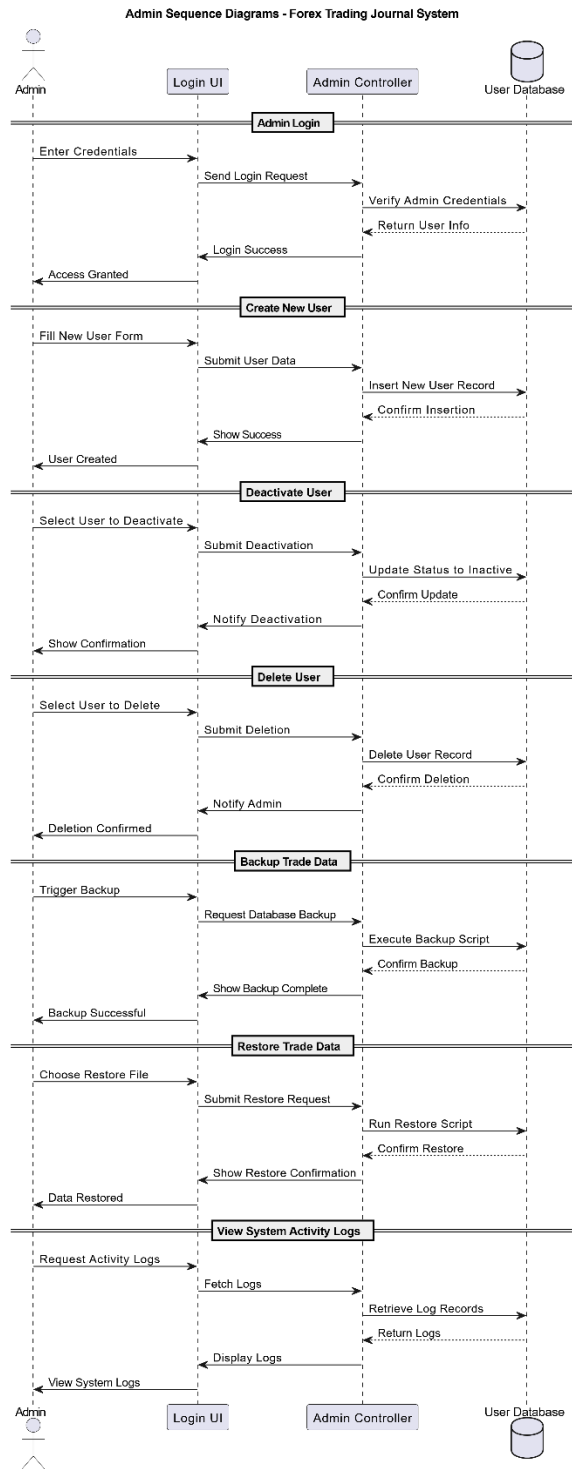


Figure 10 Admin sequence diagram

2.4.3.3 ACTIVITY DIAGRAM

The activity diagrams provide a visual representation of the individual workflows of each primary actor within the Forex Trading Journal System. Such diagrams capture the business conditions and the system's response to each user role. They also come in handy when spotting the sequences of tasks, the decisions, and the parallel flows, thus, facilitating the verification of the system design's adequacy to the expected user behaviors.

The activity modeling follows standard UML notation. The control flow is depicted through decisions (if statements), partitions (logical groupings), and system responses [13].

1. Trader Activity Diagram

The Trader activity diagram is arranged in four functional partitions: Journaling, History, Analytics, and Feedback.

- **Login and Dashboard:** The trader logs in. If the credentials are correct, the dashboard is shown.
- **Journaling:** The trader clicks “Record New Trade,” completes the form, and submits it. If the data is correct, the trade is saved, and the analytics are updated. In case of unsatisfactory data, an error is displayed.
- **History:** The trader is able to see and filter the trade history (by session, result, etc.). In addition to that, trades are editable and can also be deleted (archived).
- **Analytics:** The trader runs the analytics dashboard to check charts and create reports in PDF/CSV to be exported.
- **Feedback:** The trader sees the analyst performance comments.
- **Logout:** When the trader finishes, the session is terminated.

This activity diagram makes sure that the journaling process is interactive, error-free, and aware of the analytics.

2. Performance Analyst Activity Diagram

An analyst is the one who performs work with the partitions Review and Feedback.

- **Login and Dashboard:** After a successful login the analyst sees the list of traders on the dashboard.
- **Review:** The analyst chooses a trader and looks over his trade history and performance analytics.
- **Feedback:**

Now the analyst writes the feedback and submits it to the trader.

If the validation is successful, then the feedback is written to the database.

If the validation is not correct (e.g. some fields have not been filled in), then the system will show an error and after this, the form will be repeated.

The analyst also has the possibility to enter a new version of the old feedback, of course, this should be done within a reasonable time.

- **Logout:** After the work is finished, the application logs the user out.

This rigorous activity model illustrates allowances for feedback loops that can be used to enhance trading performance without changing raw trade data.

3. Admin Activity Diagram

The User Management, Data Maintenance, and System Monitoring are the three functional partitions into which the Admin's diagram can be divided.

- **Login and Dashboard:** The admin signs into the system and has access to the control panel.
- **User Management:**

The process of user account management includes creation, validation, deactivation, and deletion.

- **Data Maintenance:**

The backup can be made at any time by the manual method and the result can be saved to a file.

The restore process can be carried out by uploading a previously saved backup file.

- **System Monitoring:**

Admin can perform system monitoring by checking system logs, observing login attempts, and generating system usage logs.

- **Logout:** End session securely.

This activity flow definitely indicates that the system is controllable, fixable, and trackable by the administrators.

In combination, these activity diagrams establish a detailed picture of user actions and system input, which makes sure that the application can manage error, continue work processes, and separate roles.

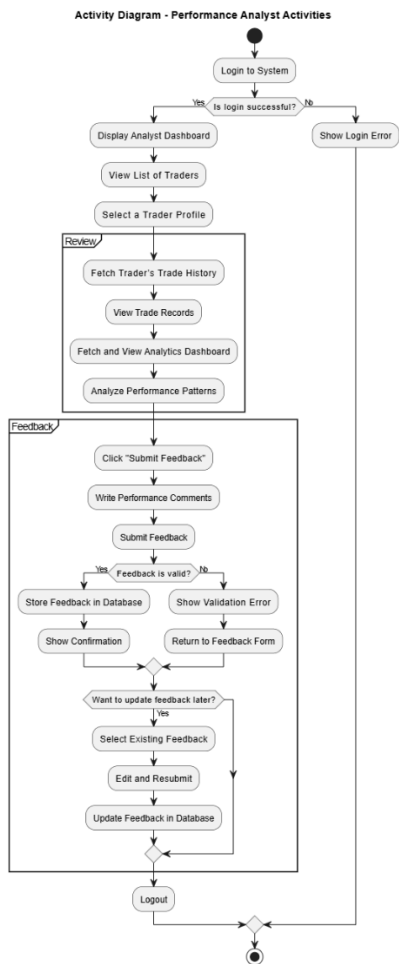


Figure 11 Performance analyst activity diagram

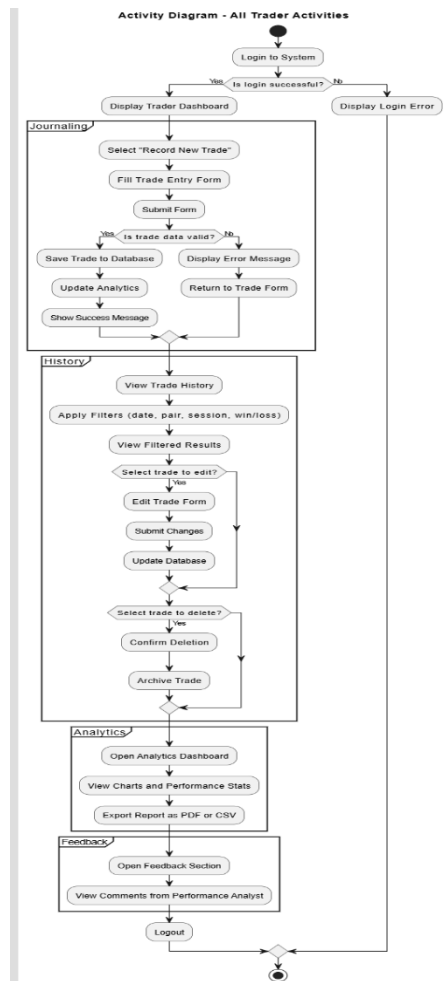


Figure 12 Trader activity diagram

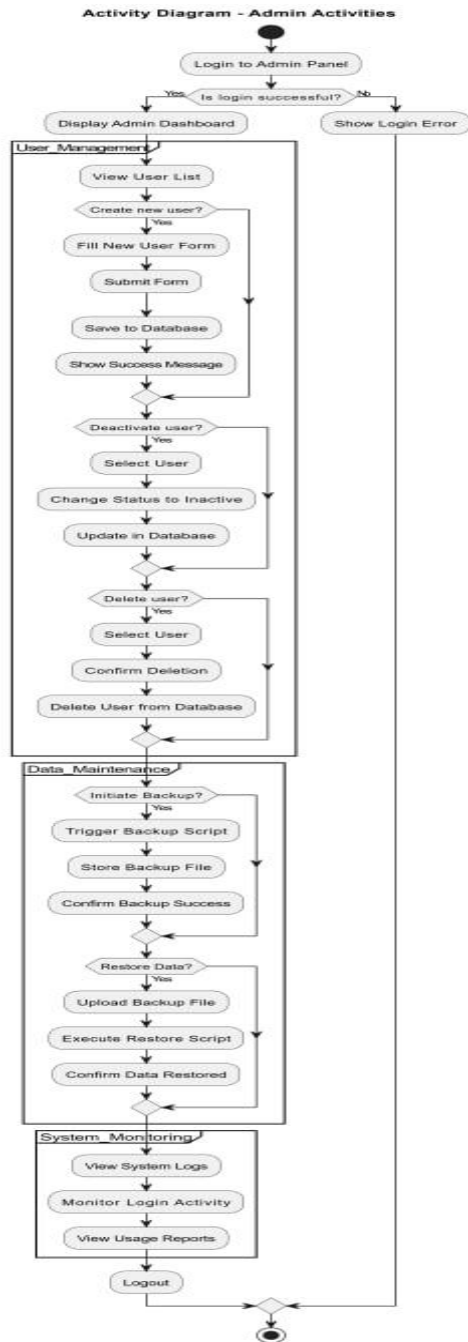


Figure 13 Admin Activity diagram

2.4.3.4 State Chart Diagram

State diagrams offer a visual representation of the various stages that primary components in the Forex Trading Journal System may go through, due to internal events or user actions. Each one depicts the birth and the life of a certain process or object, outlining how it react to various trigger events, such as submissions, edits, errors or timeouts. These models are instrumental in grasping the character of objects, making sure there are no contradictions and thus opt business rules throughout the system.

The state diagrams use the same notation as the UML state machine modeling language [13].

1. Trade Lifecycle

The Trade Lifecycle diagram represents the different phases a trade can go through starting from creation up to archiving:

- **Draft:** A new trade is created but yet to be submitted.
- **Submitted:** A trader sends the form to be processed.
- **Validated:** The system verifies the trade information for accuracy.

If correct → goes to Saved

If wrong → changes to Error

- **Saved:** The trade is recorded in the database.
- **Updated:** The trader changes the trade.

It can be resubmitted → goes back to Saved

Or be archived → goes to Archived

- **Error:** If the result of the validation is negative, the trade will become Draft to be corrected.
- **Archived:** The trade can be deleted by the user or the system can auto-archive it. This is a terminal state.

The diagram guarantees that each step of the procedure is followed and that the trades could only reach the storage, if they are correct.

2. Feedback Lifecycle

Feedback Lifecycle diagram represents the life of a feedback from the analysts, according to their performance:

- **Draft:** The analyst is work on feedback.
- **Submitted:** The system receives the feedback.

- **Editable:** The feedback can be changed during the first 24 hours.

Transitions to Updated → resubmitted → back to Submitted

- **Locked:** The edit function is not available after 24 hours.
- **Updated:** The analyst has given a different version of the feedback.

This state logic not only sets a time-limited editing policy but also helps to keep track of professional evaluations.

3. User Session State

The User Session diagram illustrates the authentication and session management process:

- **LoggedOut:** The user is in the default state and has not logged in yet.
- **LoggingIn:** This represents the state where the user has submitted the login form.

Correct credentials → changes state to LoggedIn

Incorrect credentials → goes back to LoggedOut

- **LoggedIn:** The session is active.

If no activity for 15 minutes → changes to Inactive

If user performs logout → changes to LoggingOut

- **Inactive:** After the timeout period the user will be automatically logged out → back to LoggedOut
- **LoggingOut:** When a session is ended explicitly → back to LoggedOut

This diagram is perfect for navigating session handling in a safe and controlled way for all users.

4. Report Export State

The Report Export diagram outlines the process of exporting trade data:

- **Requested:** The trader initiates export (PDF/CSV).
- **Generating:** The system executes the export command.

If success → Completed

If failure → Failed and option to Retry

- **Completed:** The state where the file is ready to be downloaded.
- **Failed:** The state where the user decides to continue and reverts to Requested.

This state diagram not only tracks exports but also validates and retries them in case of failures.

Collectively, the states illustrate the flow of operations and behavior in a system while maintaining the integrity of the workflow that is consistent with the functional and non-functional requirements of the system.

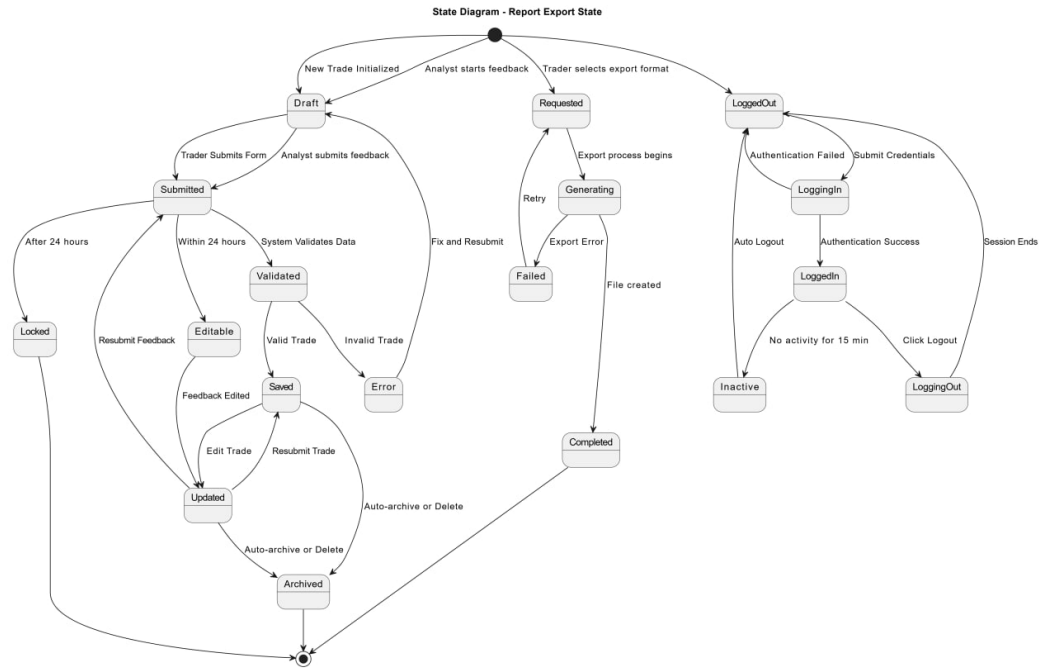


Figure 14 State Diagram

2.4.3.2 Class Diagram

The Class Diagram represents the static structure of the Forex Trading Journal System by specifying its main entities, their attributes, methods, and the relationships between them. It illustrates the way data is structured and the interaction of different components in the application's logic backend. The model adheres to the principles of the Unified Modeling Language (UML) for object-oriented system design [13].

The system includes four main classes: User, Trade, Feedback, and ExportLog. Each class represents specific functionality related to trade journaling, performance analysis, or administrative operations.

➤ User Class

The User class is the parent class for all system actors: Traders, Analysts, and Admins. A role attribute allows differentiating user types.

Attributes:

- **userId:** Uniquely identifies a user
- **fullName, email:** Data for login and communication
- **passwordHash:** Secured authentication via encrypted password
- **role:** Indicates user type (trader, analyst, or admin)
- **createdAt:** Date and time when the user account was registered

Methods:

- **login():** Logs the user in
- **logout():** Terminates the user session

Each user is allowed to possess several recorded trades, submitted feedback if they are an Analyst, received feedback in case they are a Trader, and export logs.

➤ Trade Class

The Trade class is the place where all the information of a trader's single trade entry is kept.

Attributes:

- **tradeId, userId:** Unique strings that identify the trade and the trader who created it
- **currencyPair, tradeType, session:** Market characteristics of the trade
- **riskRewardRatio, result:** Numbers that describe the trade
- **date, createdAt:** Fields that specify time

reasoning, psychology, screenshotLink, notes: Qualitative inputs for trader self-reflection

Methods:

- **editTrade():** Change an already existing trade
- **deleteTrade():** Get rid of a trade record

One trade is definitely correlated to the only one user(trader) and it is also possible to find the same user in analyst feedback.

➤ Feedback Class

Feedback class is meant to represent the link between a Performance Analyst and a Trader and it also allows one to capture comments on performance to the question: "How was the period of work?".

Properties:

- **feedbackId, traderId, analystId:** IDs of users to link
- **feedbackText:** Performance review text
- **submittedAt:** The date when the review was given

Functions:

- **updateFeedback():** After having submitted the feedback analyst can change it

Feedback is the answer of an analyst to the performance question of a trader, hence the bi-directional association they form.

➤ **ExportLog Class**

ExportLog class records the moments when a user exports trade data or reports.

Properties:

- **logId, userId:** Identifiers of export events and the user carrying out the export
- **exportType:** Export format (PDF or CSV)
- **exportedAt:** The time when the export was performed

Utilizing this record can help with auditing and monitoring who has accessed the data and when.

Relationships**A User can:**

- Record multiple Trades
- Receive or submit multiple Feedbacks (depending on role)
- Export multiple reports, tracked by ExportLog
- Roles currently assigned via role attribute and enforced through application logic, not through inheritance in the diagram.

This object-oriented design ensures data encapsulation, separation of concerns, and supports Laravel's Eloquent ORM structure for relational mapping.

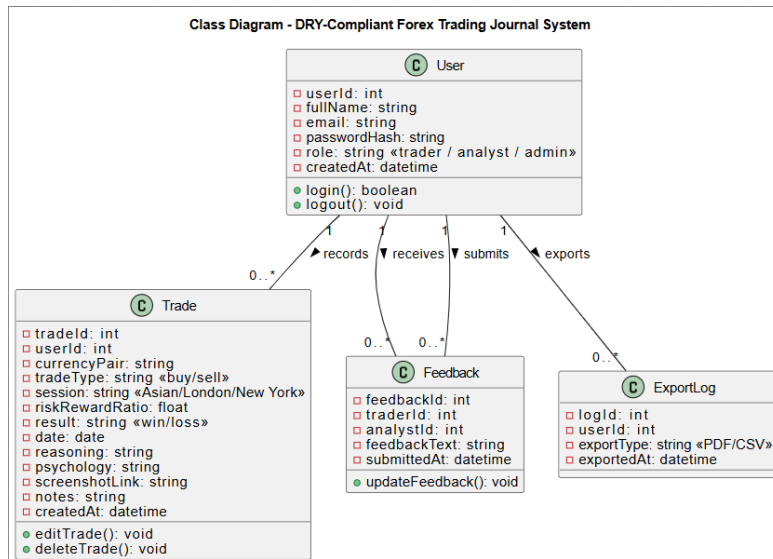


Figure 15 Class Diagram

2.4.3.6 Data Model

2.4.3.6.1 ER Diagram

The Entity-Relationship (ER) diagram illustrates the database structure underlying the Forex Trading Journal System. It depicts the main entities, their characteristics, and the interrelations between them. The diagram guarantees that the information is arranged, normalized, and compatible with the principles of relational database design [16].

The ER diagram facilitates all three user roles—Trader, Performance Analyst, and Admin—while using a centralized data structure that not only enforces referential integrity but also supports role-based operations.

Entities and Their Attributes

1. User Table

Holds authentication and profile data for users who can be traders, analysts, or admins.

- **user_id (PK):** Unique code
- **full_name, email:** Means of contact
- **password_hash:** Login credential saved securely
- **role:** ENUM to specify user type: trader, analyst, or admin
- **created_at:** The moment the account was created

2. Trade Table

The table that describes each trade that a trader has submitted.

- **trade_id (PK):** A unique number to identify the trade
- **user_id (FK):** Points to the User table
- **Market details:** ccy_pair, trade_type (buy/sell), session (Asian, London, New York)
- **Evaluation metrics:** risk_reward_ratio, result (win/loss), date
- **Qualitative inputs:** reasoning, psychology, notes, screenshot_link
- **created_at:** The moment trade was entered

Only one trader is allowed per trade, however, a trader can have many trades.

3. Feedback Table

Represents evaluative comments made by analysts to specific traders.

- **feedback_id (PK):** Unique identifier
- **trader_id (FK):** References the trader being evaluated
- **analyst_id (FK):** References the submitting analyst
- **feedback_text:** Qualitative content of the evaluation
- **submitted_at:** Timestamp of submission

A trader can have several feedback entries, and an analyst can send feedback to a number of traders.

4. ExportLog Table

Records the instances of export (PDF/CSV) actions performed by users.

- **log_id (PK):** Unique identifier
- **user_id (FK):** References the exporting user
- **export_type:** ENUM (PDF, CSV)
- **exported_at:** Timestamp of export

Can be used for routing an activity, auditing, and user behavior analytics.

Relationships

An User is capable of:

- Establishing more than one Trades
- Being the recipient of more than one Feedback (in the capacity of Trader)
- Giving more than one Feedback (in the capacity of Analyst)
- Creating more than one ExportLog events
- The Feedback entity establishes a many-to-one relationship going from Feedback to User two times—once for trader_id, and another for analyst_id.

- All foreign keys incorporate referential integrity, thus making the system compatible with implementation using relational DBMS, for example, MySQL.

The ER model presented here is a normalized, scalable design, which removes redundancy and at the same time provides proper interaction of all user roles with the system. It also guarantees compatibility with the Laravel ORM structure for convenient query building and data relationships.

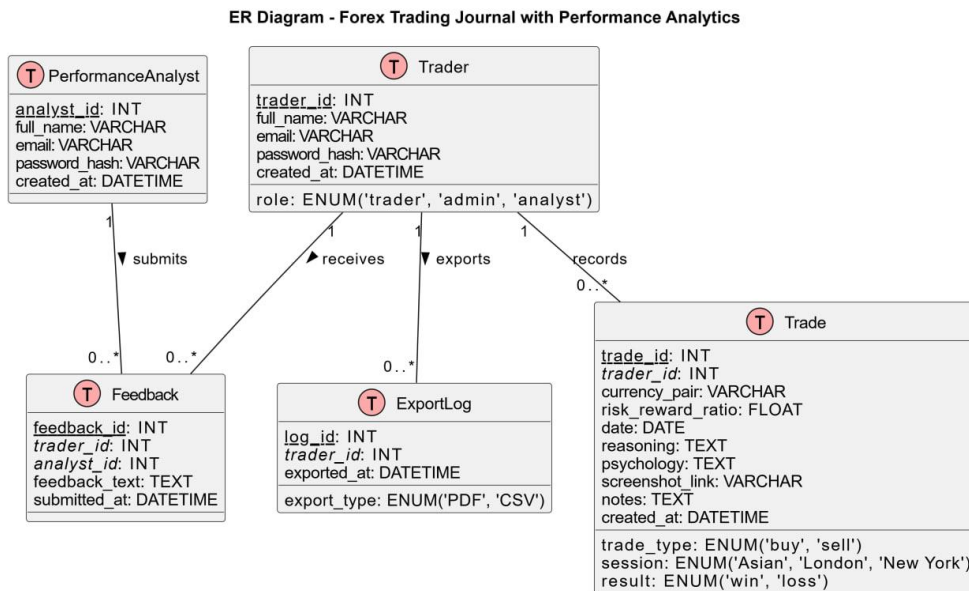


Figure 16 ER Diagram

2.5 User Interface Prototyping

| User Type | ID | Page Name |
|---------------------|-------|---------------------------|
| Trader | UIWT1 | Login Page |
| | UIWT2 | Dashboard |
| | UIWT3 | Trade Entry Form |
| | UIWT4 | Trade History Page |
| | UIWT5 | Analytics Dashboard |
| | UIWT6 | Export Reports Page |
| | UIWT7 | Feedback Viewer |
| | UIWT8 | Profile / Settings Page |
| Performance Analyst | UIWA1 | Login Page |
| | UIWA2 | Analyst Dashboard |
| | UIWA3 | Trader List Page |
| | UIWA4 | Submit Feedback Page |
| | UIWA5 | Feedback History |
| Admin | UIWM1 | Login Page |
| | UIWM2 | Admin Dashboard |
| | UIWM3 | User Management Page |
| | UIWM4 | System Logs / Backup Page |

Table 7 User Interface Prototyping

TradeLog

New Account

Sign in to TradeLog

Email

example@domain.com

Password

Enter password

Sign in

[Forgot your username or password?](#)
[Need an account?](#)

Figure 17 UIWT1

Forex Journal

Sign up

Log in to Forex Journal

Email

Password

Log in

[Forgot your password?](#)
Don't have an account?

Create account

Figure 18 UIWT2

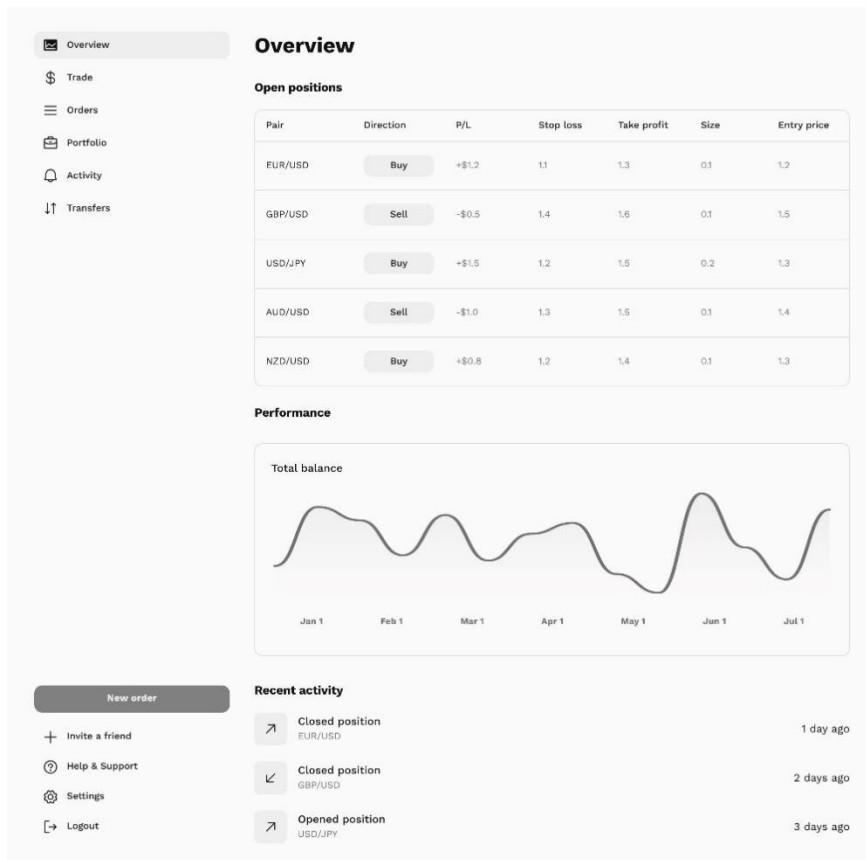


Figure 19 UIWT3

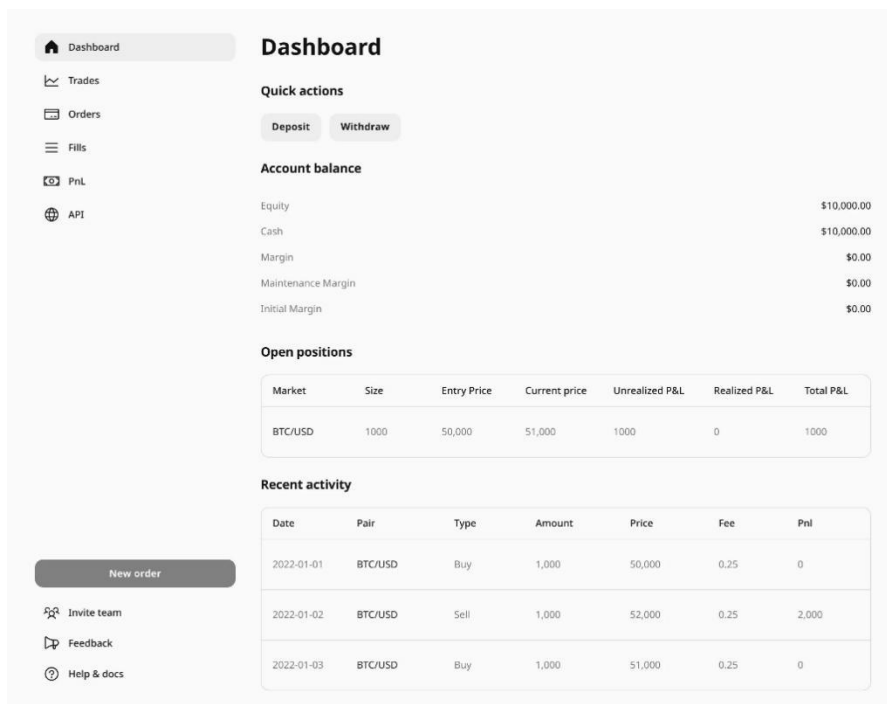




Figure 20 UIWT4

Forex Trading Journal



Trade Entry Form

Date

Currency Pair

EUR/USD

Entry Price

1.2345

Stop Loss

1.2345

Take Profit

1.2345

Risk Reward



2

Trade Rationale

Please describe the reason for this trade

Figure 21 UIWT5

Trade Journal



New Trade

Date

Currency Pair

Select currency pair

Entry Price

SL

TP

RR

Trade Rationale

Enter your trade rationale here...

Submit Trade

Figure 22 UIWT6


Trade Journal


Dashboard


Trade

Journal

Community







Trade History

Your trade history and performance analytics

Filter

Today

Yesterday

This week

Last week

Last month

This year

Custom date

All currency pairs

EUR/USD

GBP/USD

USD/JPY

AUD/USD

USD/CAD

Figure 23 UIWT7

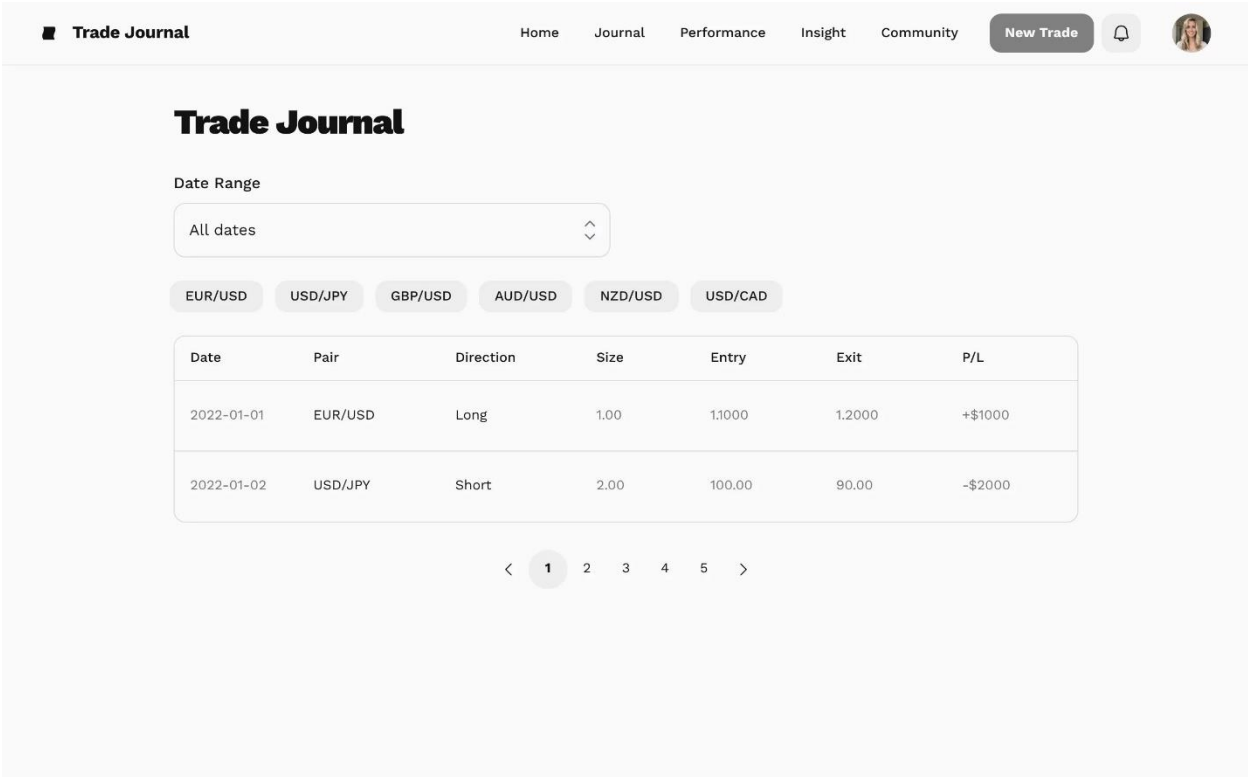
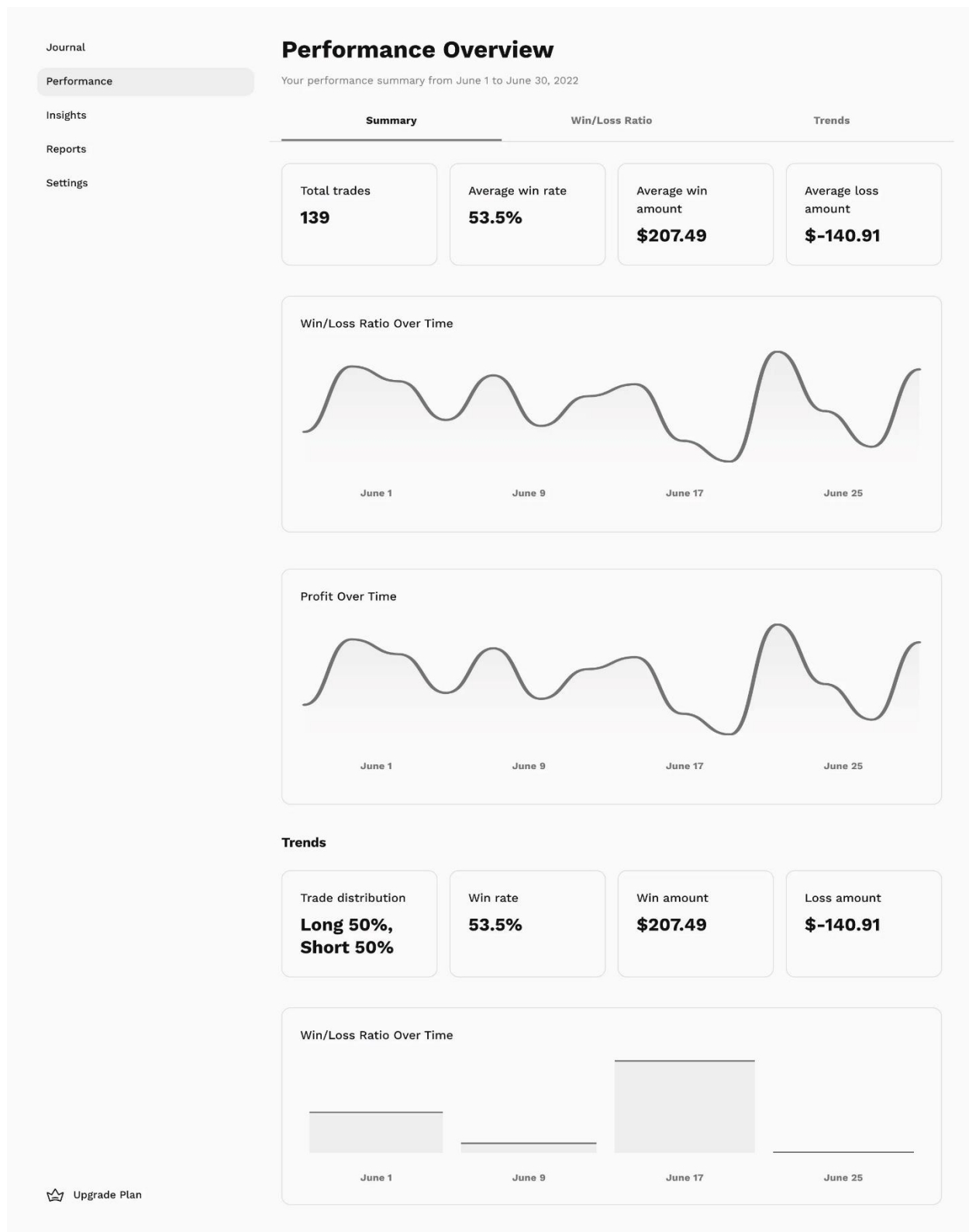


Figure 24 UIWT8



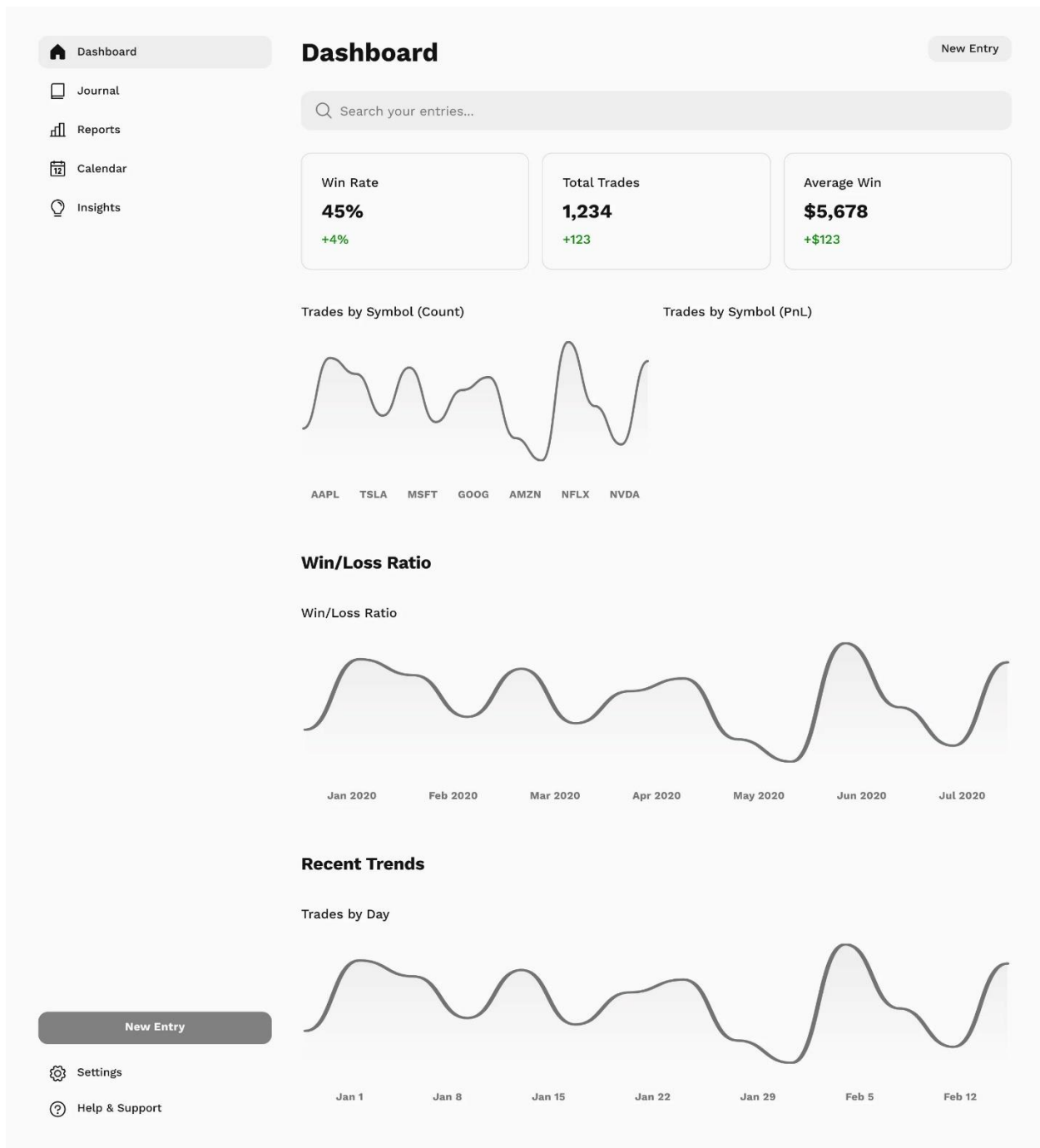


Figure 26 UIWA2

Forex Trading Journal

Dashboard

Journal

Statistics

Reports

Settings

Export Reports

Generate and export your reports

Report Type

☒ Trade

☐ Performance

☐ Summary

☐ Calendar

Time Period

Custom

Last 7 days

Last 14 days

Last 30 days

Last 3 months

Last 6 months

Last 12 months

Year to Date

Last Year

Additional Filters

All Pairs

EUR/USD

USD/JPY

GBP/USD

Generate report

Figure 27 UIWA3

Forex Trader

DashboardTradesReportsJournalEducation

Export Reports

Download a CSV of your trades or a performance report

Report Type

TradesPerformance

Time Period

All TimeLast 30 DaysCustom

Additional Filters

NonePairStrategyTag

Generate Report

Figure 28 UIWA4

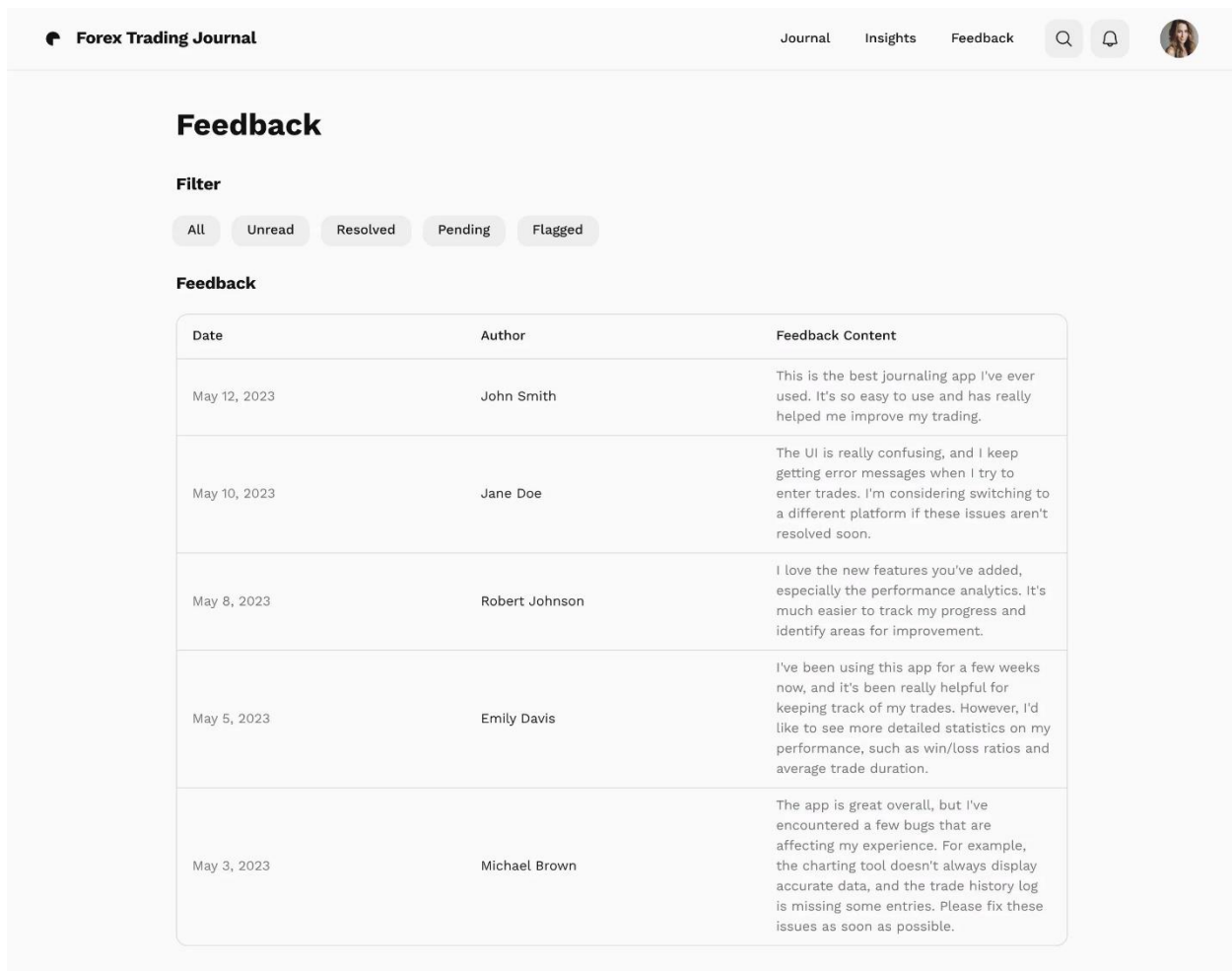


Figure 29 UIWA5

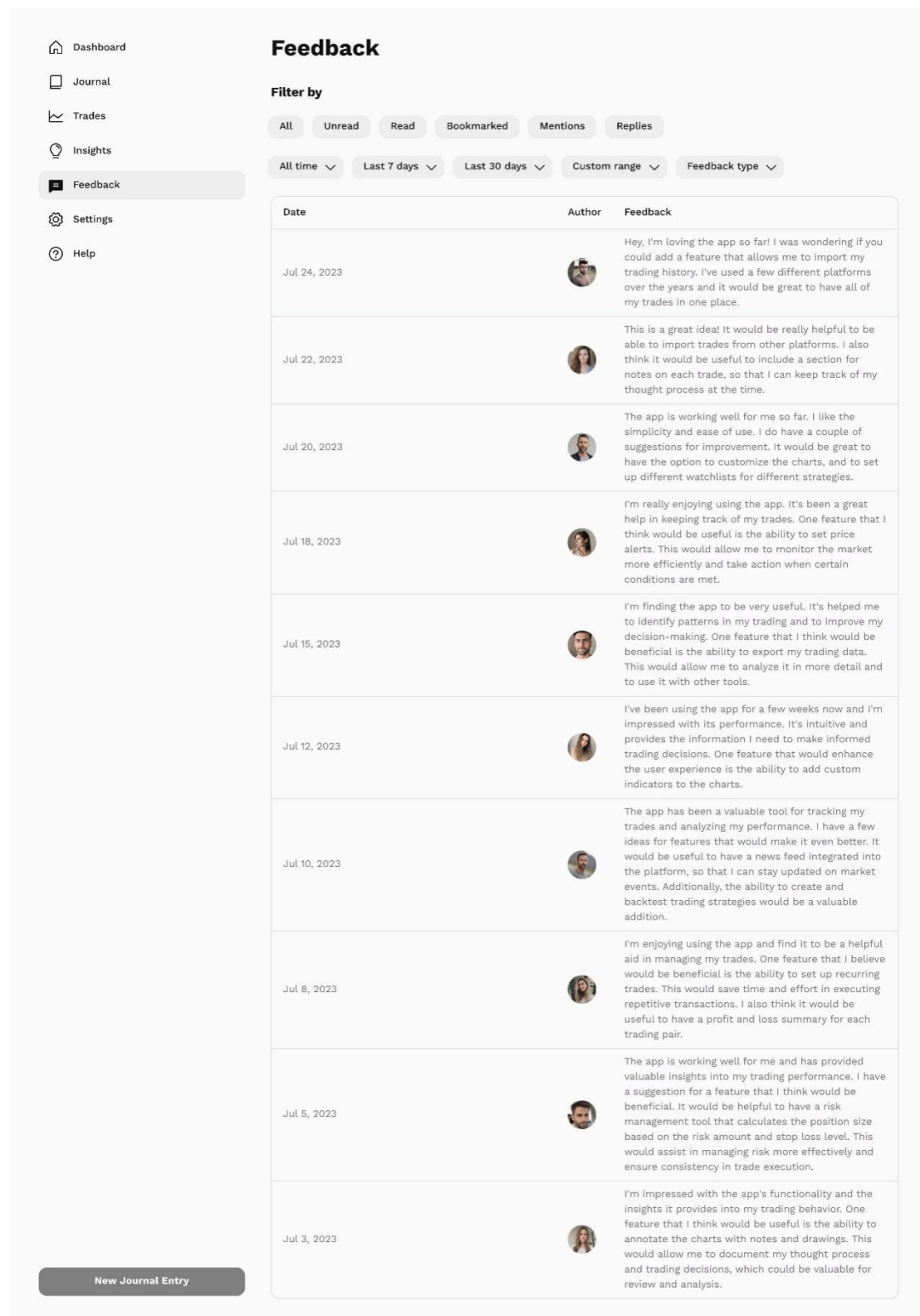


Figure 30 UIWM1

Forex Journal

Dashboard

Journal

Insights

Calendar

Trades

Watchlist

New Trade

Settings

Profile

Username

Email

First Name

Last Name

Timezone

Save Profile

Account Security

Password

Update Password

Notifications

New Followers

Notify me when someone follows me

New Replies

Notify me when someone replies to my post

Tagged in a Profile

Notify me when someone tags me in their profile

Connectivity

Slack

I'm connected to the following services

Disconnect

Connect Slack

Figure 31 UIWM2

Chapter THREE: system design

3.1 Introduction

System design is a very important stage in the software development lifecycle where the high level and architectural outline of the system is described. It connects the resource requirements with the system implementation by specifying the various parts, modules, data flows, and interfaces that will be compatible to deliver the required functionality.

The part of this book covers the designing phase of the system that was planned to be built as the Forex Trading Journal with Performance Analytics. This section also covers the system's objective, primary design targets, suggested software framework, division of parts, models of components and deployments, and also methods for managing data, security, and system control.

The objective is to make a detailed plan which confirms that the system is well formed, able to be sustained ,re-scalable, and in conformity with customer requirements and technical challenges. By explaining the defining elements of the system, this chapter offers a base for the continuous building, verifying, and supporting the system in the future.

3.2 Purpose of the System

The Forex Trading Journal with Performance Analytics system aims to be a central, intuitive, and clever web-based platform for Forex traders to systematically record, organize, and assess their trading operations. The system also enables performance analysts to track trader progress and provide targeted support, while administrators have the ability to manage users and monitor platform security.

The primary intention is to empower traders to make decisions that are based on data by documenting their trades in an organized way, and including emotional, strategic, and contextual information. Using visual analytics like win/loss ratios and trend charts, traders can pinpoint habits, adjust their strategies, and develop consistency. Analysts help to increase the capacity of traders by providing professional feedback based on the data, and admins are there to make sure the platform runs without any problems and is safe.

To summarize, the platform is designed to be:

- An organized, interactive platform to substitute scattered or manual journaling methods
- Support trader discipline, introspection, and development of new skills
- Encourage partnership between traders and performance analysts
- Provide secure, role-based access and easy user management

3.3 Design Goals

The Forex Trading Journal with Performance Analytics system's architecture is influenced by a handful of key objectives that ensure it is reliable, effective, and user-centered. The very same objectives that have decided the response actions and are behind the design of both functional and non-functional parts throughout the entire system, also, are the fundamental principles envisioned in the system.

First design goals of the system are below-listed:

1. Usability

System must offer a smooth, natural, and reachable user interface. Traders, analysts, and admins should be able to travel across the platform with minimal training and do tasks efficiently.

2. Security

Strengthening the authentication and authorization areas is a must to ensure that only protected data of a user are kept. Besides a login name and password, only trade history is a piece of highly confidential information that should be securely stored and transmitted via encryption and access control.

3. Scalability

The system must be able to accommodate a growth in the user base and the number of transactions without affecting the quality of the service. The architecture should allow for both horizontal and vertical scaling with ease.

4. Performance

The most important activities, including trade entry, filtering history, generating analytics, and submitting feedback, should be done rapidly even if there is a moderate or high load on the system. The system should reduce response time and improve data access efficiency.

5. Maintainability

The codebase needs to be organized and in a reusable way. The availability of well-organized documentation, harmonized code structure, and clear division of responsibilities will facilitate future updates, debugging, and the addition of new features.

6. Reliability

The system should be reliable when it comes to its performance and accuracy in all the modules. Data validation, error handling, and transaction control mechanisms must guarantee that no data will be lost or changed in the process.

These design goals are the basis of the proposed system architecture and they also direct the choice of technologies, frameworks, and modeling strategies in the following parts of the chapter.

3.4 Current Software Architecture

Most retail Forex traders employ a very fragmented and manual software architecture that is intended for journaling and performance analysis. Traders usually use general-purpose tools such as Microsoft Excel, Google Sheets, Notion, or Evernote to enter a trade data, write some notes, and try to track some basic patterns [17], [18]. These tools, while easily accessible and customizable, however, are insufficient in structure that they can support long-term analytical consistency and trading discipline.

For instance, if a trader is using Excel or Google Sheets, he/she personally decides and manages the formulas, charts, and filters that are going to be used — which may increase the chances that errors in data entry and inconsistency occur. These platforms do not have features that perfectly fit tracking psychological states, tagging strategies, or submitting performance feedback that are critical for deeper trading reflection [17].

In addition, myfxbook and trademetria, which are existing solutions, provide the convenience of automated trade tracking service that can be facilitated by the direct connection to the broker. However, they mainly focus on the statistical summaries rather than personalized journaling. Those solutions do not have the integrated performance feedback as well as the features that allow users to record their emotions and reflective note-taking, which are an essential part of behavioral development [17].

The current setup lacks the standardization of the backend, the user role management system, and the feedback framework in the architecture. The parts of the communication between traders and mentors or analysts go on over third-party messaging platforms such as Telegram, Discord, or email. In those places, feedback is easily lost, it is disconnected from the original trade context, and still, it is not in any way structured.

On the whole, the existing structure, which is still in use by individual traders, is fragmented, full of mistakes, and not suitable for traders who are looking for sustainable development. It also indicates an evident void of a devoted, centralized and role-aware trading journal system that comes with analytics, feedback mechanisms and safe data handling features.

3.5 Proposed Software Architecture

The system architecture proposed for the Forex Trading Journal with Performance Analytics is a modular, web-based, and multi-tiered design that is capable of supporting scalability, maintainability, and secure role-based access. The design uses the Model-View-Controller (MVC) pattern, which implies that the business logic, data handling, and user interface are dealt with as separate components. This arrangement allows flexible development, simpler debugging, and improved long-term support.

In general, the architecture is broken down into the following layers:

- **Presentation Layer (View):** This layer was created with HTML, CSS, JavaScript, and Bootstrap and it is responsible for all user interactions. It changes continually according to user roles (Trader, Analyst, Admin) and device types (desktop, tablet).
- **Application Layer (Controller):** This part was done with PHP (Core PHP or Laravel), it takes care of request routing, input validation, role-based access logic, and the communication with the backend.
- **Data Layer (Model):** This part of the system is MySQL-based and it is
 - used for storing all the data that has to be kept permanent such as user accounts, trades, feedback, and report logs.

All parts of the system — like trade management, performance analytics, feedback submission, and user administration — are done as modules with very clear interfaces, good for reusability and maintainability.

How the Architecture Meets Design Goals:

- **Performance:** Due to its loosely coupled components and asynchronous communication between tiers through APIs and web services, the modular MVC architecture significantly increases system response speed, thus efficiently utilizing distributed computing and persisting data caches.
- **Reliability:** The architect designed each part of the architecture with the network redundancy and failover capabilities that significantly increase service uptime, hence providing a high subjected availability service (24/7). Furthermore, the distributed load across the independent loads within the controller and model tiers reduces the chances of a single point of failure, thus achieving the highest possible availability of services.
- **Usability:** Clear division of front-end components for role-specific dashboards.
- **Security:** Centralized authentication and role-based access enforcement.
- **Scalability:** Database normalization and modular structure support growth.
- **Maintainability:** MVC separation enables isolated updates without breaking other parts.
- **Performance:** Efficient routing and minimal query overhead optimize response time

When compared with the unstructured current environment (manual spreadsheets, informal feedback channels), this architecture presents a sturdy framework that not only ensures data integrity but also enhances collaboration and automates the analytics.

A visual representation of the architecture due to subsystem decomposition and component diagrams is given

3.5.1 Subsystem Decomposition

To ensure clarity, modularity, and ease of maintenance, the proposed system is divided into six key subsystems. Each subsystem is responsible for a specific set of functionalities, and they interact through clearly defined interfaces. This modular decomposition supports scalability, simplifies testing, and enables parallel development.

1. User Management Subsystem

Handles user registration, login, role assignment, and access control. This subsystem ensures secure authentication and role-based authorization for Traders, Performance Analysts, and Admins.

2. Trade Management Subsystem

Manages all operations related to trade logging, editing, deletion, and retrieval. It captures trade details including currency pair, session, risk-reward ratio, reasoning, and psychology, and ensures data validation before storage.

3. Analytics Engine Subsystem

Processes stored trade data to generate performance visualizations such as win/loss ratios, session-based success rates, and monthly trends. This subsystem provides graphical insights that support data-driven decision-making.

4. Feedback Subsystem

Allows performance analysts to select a trader, review their trade history, and submit structured feedback. This feedback is then linked to the trader's profile and can be reviewed anytime by the trader.

5. Report and Export Subsystem

Enables traders to export their trade history and analytics in PDF or CSV format. This subsystem also logs export actions for accountability and reporting purposes.

6. Admin Control Subsystem

Provides system administrators with tools to manage users, monitor activity logs, perform backups, and restore system data when necessary. This subsystem ensures the system remains secure and operational.

Subsystem Dependencies and Communication

- The User Management subsystem is foundational and interacts with all others to enforce role-based access.
- The Trade Management subsystem supplies data to the Analytics Engine and Feedback subsystems.
- The Feedback subsystem depends on both User Management (for role access) and Trade Management (for trade context).
- The Report and Export subsystem pulls data from the Analytics Engine and Trade Management modules.
- The Admin Control subsystem operates at a supervisory level, accessing and interacting with all core modules to manage data and security.
- This structured decomposition promotes a clean separation of concerns and prepares the groundwork for the following component diagram.

3.5.2 Component Diagram

The component diagram below illustrates the major functional modules of the Forex Trading Journal with Performance Analytics system and the dependencies between them. Each component corresponds to a specific subsystem described in the previous section and is responsible for a distinct set of features.

At the core is the User Management component, which controls authentication and enforces role-based access. It interacts directly with all other modules to ensure secure and appropriate access to system functionalities.

The Trade Management component allows traders to record, update, and delete trade data. It serves as a primary data source for the Analytics Engine, which generates performance charts and trend analyses based on historical trade entries. Both of these components feed into the Report & Export Module, which enables traders to export their data in PDF or CSV format.

The Feedback System enables performance analysts to review trade histories and submit structured evaluations. It interacts with both User Management for access control and Trade Management to retrieve relevant trade information.

The Admin Control Panel provides oversight tools for system administrators. It connects with all other components to facilitate user management, activity monitoring, backup operations, and overall system integrity enforcement.

This modular structure promotes reusability, scalability, and separation of concerns, supporting a maintainable and secure software architecture.

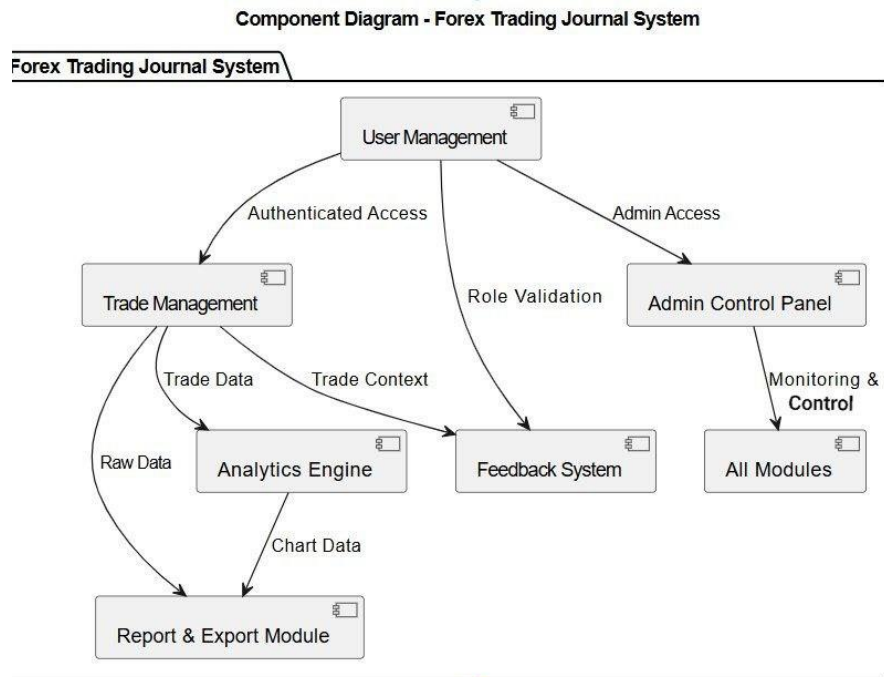


Figure 32 component diagram

3.5.3 Deployment Diagram

The deployment diagram illustrates how the system's software components are physically distributed across the computing infrastructure. For the Forex Trading Journal with Performance Analytics, the system follows a typical web-based deployment model, involving a client-server architecture supported by a database server.

This design ensures performance, scalability, and maintainability, while allowing remote access through any modern web browser.

Deployment Architecture Overview:

1. Client Side

- Users (Traders, Performance Analysts, Admins) access the system through a web browser.
- No installation is required — the UI is rendered dynamically using HTML, CSS, and JavaScript.

2. Application Server

- Hosts the backend application developed in PHP (Laravel or Core PHP).
- Handles user authentication, trade logic, feedback processing, report generation, and business rules.

3. Database Server

- Runs a MySQL relational database engine.
- Stores persistent data: user credentials, trade records, analytics data, feedback logs, and export history.

4. File Storage

- Stores uploaded screenshots or linked images (if applicable).
- May use local disk storage or cloud-based storage like AWS S3, depending on deployment.
- **Deployment Strategy**
 - All backend services are hosted on a secure web server (e.g., Apache or Nginx).
 - The application and database can reside on the same server for small-scale deployments or be separated for scalability and performance in production environments.
 - Regular backups and monitoring are enabled to ensure high availability and disaster recovery.
- **Deployment Benefits:**
 - **High Availability:** Accessible from any browser 24/7
 - **Security:** Encrypted connections via HTTPS
 - **Scalability:** Components can be scaled independently if needed
 - **Maintainability:** Modular structure supports efficient updates

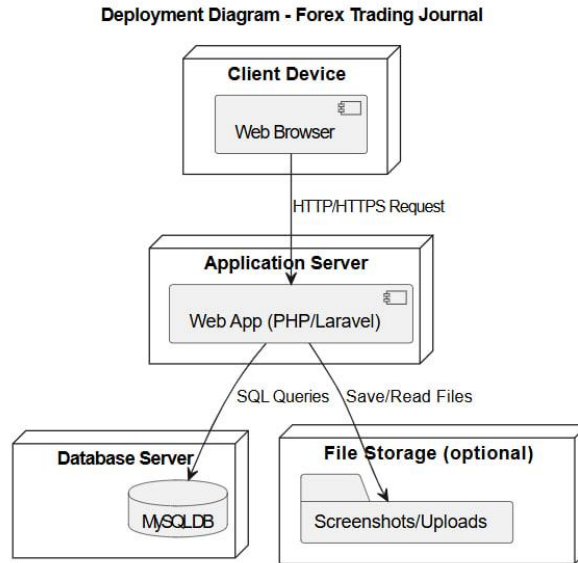


Figure 33 Deployment diagram

3.5.4 Persistent Data Management

The Forex Trading Journal with Performance Analytics system is designed to manage a diversity of user and system data which require durable and secure storage. The data is broad and covers user credentials, trade records, feedback messages, analytical summaries, and export logs. Efficient data management is essential to guaranteeing the system's dependability, performance, and user confidence.

1. Types of Persistently Stored Data

The system stores the following data types:

| Data Type | Description |
|------------------|---|
| User Information | Username, hashed passwords, email addresses, roles (Trader, Analysts, Admin) |
| Trade Records | Currency pairs, trade type, session, date, risk-reward ratio, reasoning, etc. |
| Feedback Analyst | Analyst comments linked to specific traders and trades |
| Export Logs | Record of when and how users exported their data (PDF/CSV) |
| Upload Files | Optional screenshot URLs or chart links uploaded by users |

Table 8 Types of Persistently Stored Data

2. Database Technology

- MySQL is the RDBMS of choice.
- It is good practice to normalize tables up to 3NF to eliminate redundancy and keep the data consistent.
- Implementing foreign keys and indexing not only makes data access faster but also keeps the relational integrity.

3. Data Access Methods

- The system executes client-side script (PHP) to send SQL queries and receive data from the database.
- In the case of Laravel, the ORM Eloquent is in charge of performing data operations of the OOP type.
- The procedure of each interaction is validation and sanitization so that no SQL injection can be committed and the data remain intact.

4. Backup and Recovery

- The system allows for the carrying out of both automated and manual backups. The admin interface makes manual backups accessible.
- Backups are not only the database but also the uploaded files that help in recovery of the whole.
- If something goes wrong be sure that the data is there thanks to the latest backup.

5. Data Integrity

- Input validation even guarantees that all the data written into the database are in the correct format and complete.
- Transaction management is employed in those parts of the work that are critical (for example, the process of submitting a trade and updating the feedback) so that the occurrence of partial writes is not allowed.
- Failure handling ensures that the interruptions occurring during write operations do not result in an inconsistent state of the database.

This long-term data management technique makes sure that the service is running smoothly, is scalable to different users, it is reliable, and it gives traders and analysts a safe place to keep track of the progress and work on performance enhancement.

3.5.5 Detailed Database Design

This section presents the detailed structure of the relational database used in the Forex Trading Journal with Performance Analytics system. The design follows principles of normalization and aligns closely with the class diagram and ER model to ensure data consistency, scalability, and performance.

3.5.5.1 Relational Tables

Below is a list of the core relational tables, including field names, data types, and key definitions.

| Field Name | Data type | key | Description |
|---------------|---------------------------------|-------------|----------------------------|
| User_id | INT(Auto-increment) | Primary Key | Unique ID for each user |
| Full_name | VARCHAR(100) | - | User's full name |
| email | VARCHAR(100) | Unique | User's email address |
| Password_hash | VARCHAR(255) | - | Hashed password |
| Role | ENUM('trader';analyst','admin') | - | Role of the user |
| Created_at | DATETIME | - | Account creation timestamp |

Table 9 user table

| Field Name | Data Type | key | Description |
|-------------------|----------------------------------|-------------|-----------------------------|
| Trade_id | INT(Auto-increment) | Primary key | Unique ID for each trade |
| User_id | INT | Foreign key | References user(user_id) |
| Currency_pair | VARCHAR(10) | - | e.g. EUR/USD |
| Trade_type | ENUM('Asian','London','Newyork') | - | Type of trade |
| Session | ENUM('Asian','London','Newyork') | - | Market session |
| Risk_reward_ratio | FLOAT | - | Numeric value |
| Result | ENUM('win','loss') | - | Outcome of traders |
| Date | Date | - | Date of trade |
| Reasoning | TEXT | - | Trade reasoning |
| Psychology | TEXT | - | Emotional state |
| Screenshot_link | VARCHAR(255) | - | Optional link to screenshot |
| Notes | TEXT | - | Optional notes |
| Created_at | DATETIME | - | Timestamp of entry |

Table 10 trade table

| Field Name | Data type | key | Description |
|---------------|---------------------|-------------|-----------------------------|
| Feedback_id | INT(Auto-increment) | Primary key | Unique ID for feedback |
| Trader_id | INT | Foreign key | References Uer(user_id) |
| Analyst_id | INT | Foreign key | References user(user_id) |
| Feedback_text | TEXT | - | Text feedback fom analyst |
| Submitted_at | DATETIME | - | Date and time of submission |

Table 11 Feedback table

| Field Name | Data Type | key | Description |
|-------------|---------------------|-------------|--------------------------|
| Log_id | INT(Auto-increment) | Primary key | Unique ID for entry |
| User_id | INT | Foreign key | References User(user_id) |
| Export_type | ENUM('PDF','CSV') | - | Format of export |
| Exported_at | DATETIME | - | Time of export |

Table 12 ExportLog table

3.5.5.2 Normalization

Normalization is a systematic approach used in relational database design to reduce data redundancy and improve data integrity. The tables in the Forex Trading Journal with Performance Analytics system are designed to conform to the Third Normal Form (3NF), ensuring that each table contains data that is logically grouped and functionally dependent on the primary key.

Normalization Applied

1. First Normal Form (1NF)

- All attributes contain atomic (indivisible) values.
- Each record is unique and there are no repeating groups.
- All tables meet 1NF — for example, currency_pair, session, and result are stored as individual fields, not as lists or arrays.

2. Second Normal Form (2NF)

- The database is in 1NF, and all non-key attributes are fully functionally dependent on the primary key.
- Example: In the Trade table, fields like currency_pair and risk_reward_ratio depend only on trade_id, the primary key — not on any subset of a composite key (because there is none).

3. Third Normal Form (3NF)

- The database is in 2NF, and there are no transitive dependencies (i.e., no non-key attribute depends on another non-key attribute).
- Example: In the User table, role depends only on user_id, and no field depends on email or full_name.

➤ Conclusion

- All core tables in the system are normalized to 3NF, ensuring:
- Data consistency and accuracy
- Easy updates and deletions without causing anomalies
- Reduced redundancy and optimized storage

3.5.5.3 EER – Extended Entity Relationship Diagram

The Extended Entity Relationship (EER) diagram for the Forex Trading Journal with Performance Analytics system outlines the system's data structure in a conceptual way. It extends the basic ER model by incorporating generalization and optional participation.

The primary entity is the User, who is generalized to allow for three roles: Trader, Performance Analyst, and Admin — via a role attribute. Users may publish trades, provide and get feedback, and also export reports. The diagram represents one-to-many relationships with clearly delineated participation and constraints, which ensure that referential integrity is maintained throughout all modules.

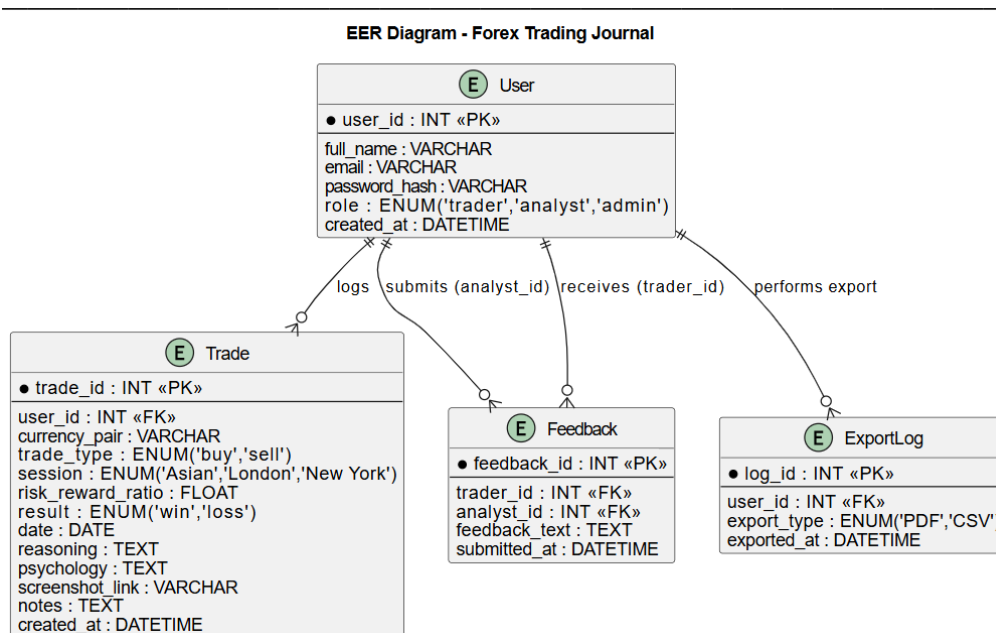


Figure 34 EER Diagram

3.5.5.4 Object-Oriented Relational Mapping (OO-Relational Mapping)

Object-Oriented

Relational Mapping (ORM) is a design technique used to bridge the gap between

object-oriented programming languages and relational databases. It allows class instances in code to be stored and retrieved as records in a relational database, simplifying data manipulation, abstraction, and consistency between backend logic and persistent data storage. In the Forex Trading Journal with Performance Analytics system, the OO-relational mapping aligns

closely with the DRY-compliant class diagram and the normalized relational

database design. The system is implemented using PHP, and optionally Laravel's

Eloquent ORM, which automatically handles this mapping between classes and tables.

| Class Name(OOP) | Mapped Table | Attributes (OOP ↔ DB Fields) |
|-----------------|--------------|--|
| User | user | userId ↔ user_id fullName ↔ full_name email ↔ email passwordHash ↔ password_hash role ↔ role createdAt |
| Trade | trade | tradeId ↔ trade_id userId ↔ user_id currencyPair ↔ currency_Pair tradeType ↔ trade_type session ↔ session riskRewardRatio ↔ risk_reward_ratio result ↔ result date ↔ date reasoning ↔ reasoning psychology ↔ psychology screenshotLink ↔ screenshot_link notes ↔ notes createdAt ↔ createdAt |
| Feedback | feedback | feedbackId ↔ feedback_id traderId ↔ trader_id analystId ↔ analyst_id feedbackText ↔ feedback_text submittedAt ↔ submitted_at |
| Export log | export-log | logId ↔ log_id userId ↔ user_id exportType ↔ export_type exportedAt ↔ exported_at |

Table 13 Mapping table

Key OO-Relational Concepts Used

- **Encapsulation:** Each class encapsulates the data and behaviors relevant to that entity (e.g., User has login() and logout() methods).
- **Associations:** Foreign keys in tables are mirrored as object associations (e.g., User has Many Trades, Feedback belongs To User).
- **Inheritance (Role Field):** Instead of using separate classes for Trader, Analyst, and Admin, a single User class uses a role attribute — a form of single-table inheritance.
- **Persistence Abstraction:** If Laravel is used, methods like save(), find(), and update() can be used directly on model instances without writing raw SQL.

This OO-relational mapping strategy ensures tight synchronization between the system's business logic and its persistent data layer, enabling faster development, clearer structure, and easier maintenance.

3.5.6 Access Control and Security

Access control and security are very important to ensure that user data is safe, the trust is maintained, and also it is guaranteed that each actor within the Forex Trading Journal with Performance Analytics system can only carry out those actions, which they are authorized to. The system implements security at both authentication and authorization levels, employing features that are available in Laravel (or can be copied in Core PHP).

1. User Authentication

- The system utilizes a login interface that confirms user credentials by comparing them with hashed passwords stored in the user table.
- Password hashing is done through Laravel's bcrypt() function or PHP's password_hash() if Core PHP is used.
- Laravel's Auth system, which is already in place, makes tasks such as session handling, login, logout, and middleware protection easier.
- Access control and security are very important to ensure that user data is safe, the trust is maintained, and also it is guaranteed that each actor within the Forex Trading Journal with Performance Analytics system can only carry out those actions, which they are authorized to. The system implements security at both authentication and authorization levels, employing features that are available in Laravel (or can be copied in Core PHP).

➤ **Laravel Example:**

```
Hash::make($request->password); // Hashing Auth::attempt(['email' => $request->email, 'password' => $request->password]); // Login
```

2. Role-Based Authorization

- Each user has a role field (trader, analyst, admin) stored in the database.
- Based on the role, users are granted access only to permitted features (e.g., only Admin can manage users).

Laravel's middleware can restrict routes like:

```
Route::middleware(['auth', 'role:admin'])->group(function () { Route::get('/admin-dashboard', 'AdminController@index'); });
```

In Core PHP, role-based checks can be done using session variables:

```
if ($_SESSION['role'] !== 'admin') { header('Location: unauthorized.php'); }
```

3. Data Encryption and Privacy

- Passwords are hashed securely before being stored — they are definitely not saved as plain text in the database.
- At the same time, HTTPS provides for encrypted communication between clients and servers (this is mandatory at deployment).
- The optional sensitive fields (for instance, feedback or notes) are cleaned (sanitized) before the database gets the insertion.

4. Session Management

- Sessions during the authentication process are controlled by Laravel's session drivers or PHP `$_SESSION`.
- For safer use, auto-logout or session expiration can be set up.
- By default, Laravel implements token-based protection (CSRF tokens) for all form submissions.
- 5. Input Validation and Sanitization
- No form is processed if its inputs are not validated first.
- Laravel utilizes the Validator class while Core PHP sanitizes inputs with `filter_input()` and `htmlspecialchars()`.

6. SQL Injection Prevention

- Laravel's Query Builder or PDO parameterized queries are used in Core PHP.
- Only sanitized user input is allowed in SQL statements.

7. File Upload Security (for Screenshots)

- Files are checked to be of the certain size and formats (e.g., PNG, JPG).

- Directories for storing files are secured with controlling access done through the database and changing access paths.

This access control and security design ensures that only authorized users interact with system features, all data is handled securely, and sensitive operations are protected from common web vulnerabilities — all using tools and practices fully supported in Laravel or vanilla PHP.

3.5.7 Global Software Control

Global software control defines how the overall control flow of the system is coordinated — including how requests are initiated, routed, and managed across the various components of the application. In the Forex Trading Journal with Performance Analytics system, global control is implemented using a role-based request-routing mechanism combined with modular controllers and middleware, following the MVC (Model-View-Controller) pattern

1. Request Flow

- A user action (e.g., logging in,
- submitting a trade, exporting a report) starts from the frontend (View).
- The request is routed to the appropriate
- Controller using Laravel's route definition or PHP request handling.
- The Controller interacts with the Model
- (e.g., User, Trade, Feedback) to process data or trigger business logic.
- A response is returned to the View, which

updates the user interface.

2. Role-Based Control Flow

Each user role accesses the system through different interfaces and with different permissions. The control flow ensures they are automatically routed to the correct dashboard and functionality.

Laravel Example:

```
if (Auth::user()->role == 'trader') {
    return redirect('/trader/dashboard');
} elseif (Auth::user()->role == 'analyst') {
```

```

        return redirect('/analyst/dashboard');
    } elseif (Auth::user()->role == 'admin') {
        return redirect('/admin/dashboard');
    }

```

In Core PHP:

```

switch ($_SESSION['role']) {
    case 'trader':
        header('Location: trader_dashboard.php');
        break;
    case 'analyst':
        header('Location: analyst_dashboard.php');
        break;
    case 'admin':
        header('Location: admin_dashboard.php');
        break;
}

```

3. Module Synchronization

Subsystems, like a Trade Management, an Analytics Engine, and a Feedback Module, could be independently operated; however, they are still able to communicate each other by means of the shared data models:

- The Analytics module is the one pulling data from the Trade module.
- The Feedback module is the one querying the same data that is being used by Analytics, hence the consistency.
- Export actions are the ones that access the two kinds of data, i.e., analytics and trade in real time.

Each controller and model are both independent, however, they communicate through the clearly defined relationships (for example, Eloquent relationships or foreign keys in queries).

4. Middleware and Global Guards

Laravel has a middleware for the purpose of intercepting and dealing with global conditions:

- User is authenticated (auth)
- User has correct role (role:admin)
- Request is protected against CSRF

In Core PHP, the same kind of checks is made by hand at the beginning of each script using session variables and custom functions.

5. Error and Exception Handling

- Laravel takes care of errors globally by the use of try/catch blocks and exception handlers.
- Core PHP uses `set_error_handler()` and try/catch structures for this.
- The system is the one that shows the users the error pages that are user-friendly in case they get unauthorized access, validation fails or there are no records.

This global control design allows the system to behave in a predictable way, to observe role-based boundaries, and at the same time to enable each module to be in coordination with others without interrupting the operation of others.

3.5.8 Boundary Conditions

Boundary conditions are characteristics of a system in its non-normal modes of operation or under abnormal inputs. These include the startup, shutdown, and erroneous states, as well as extreme usage scenarios. Forex Trading Journal with Performance Analytics system robustness is guaranteed by dealing with such conditions, which are undoubtedly the core of its reliability and resilience.

1. System Startup

- On application startup, the server goes through the process of initializing routing configurations, making a connection to the MySQL database, and loading models, controllers, and middleware that it can use.
- Laravel's automatic service providers or Core PHP initialization scripts (`config.php`) make sure
- that environment variables and session management are properly set up by registering and managing them.
- The system keeps a record of the error should it occur and shows a custom page

- of “System Unavailable” if the error log file reading or the config file is missing during this process.

2. System Shutdown

- During server shutdown or application maintenance, running sessions are terminated in a polite manner and completed if scheduled backups (if any) exist before disconnection.
- Laravel employs the down command to show a “Maintenance Mode” notification, thus, it does not accept any input during the update process and there is no data corruption.
- A short-term flag has been used in Core PHP to send users to the maintenance page.

3. Error Handling

- Errors on the user side (invalid input, unauthorized access) are handled by means of form validation and redirects with error messages.
- Errors on the server side (database failure, missing records) decide to do logging and offer a friendly error page situation for the users.
- On the one hand, Laravel handles all errors in a centralized way through Handler.php, however, on the other hand, try-catch blocks and error_log() are the instruments Core PHP utilizes.

4. Extreme Load Conditions

- When there is the highest number of users at the same time (e.g., many traders logging trades simultaneously), the system will continue operating without any problem as it uses optimized SQL queries, indexed tables, and Laravel’s Eloquent caching to retain response time.
- In such a case, if it is found useful to scale up horizontally or transfer the setup to a cloud platform (e.g., Laravel Forge, AWS), this option will be chosen.
- Queue systems (e.g., Laravel Queues or Redis) can run non-blocking jobs such as report generating.

5. Invalid or Unexpected Input

- Input fields are validated using Laravel’s Validator class or Core PHP filters.
- For file uploads (screenshots), file size and MIME type restrictions are enforced.
- Invalid form submissions return helpful validation messages without disrupting the user session.

6. Session Timeout and Auto-Logout

- Inactive users are logged out after a specified period (e.g., 30 minutes).
- Laravel manages this via session. lifetime config, while Core PHP uses time checks in `$_SESSION`.

7. Backup and Recovery

- Admins are notified of upcoming backup tasks or errors via the admin dashboard.
- In the event of data loss, a full restoration from backup includes both database and uploaded assets.
- **Laravel's scheduled tasks or Cron jobs ensure backup reliability.**

By anticipating and designing for these boundary conditions, the system ensures a stable, user-friendly experience even under abnormal or extreme circumstances, which is crucial for long-term reliability and user trust.

REFERENCES

- [1] M. Douglas, **Trading in the Zone**, Penguin, 2000.
- [2] V. K. Tharp, **Trade Your Way to Financial Freedom**, 2nd ed., McGraw-Hill, 2010.
- [3] G. Booch, J. Rumbaugh, and I. Jacobson, **The Unified Modeling Language User Guide**, 2nd ed., Addison-Wesley, 2005.
- [4] Myfxbook, "Automated Analytical Tool for Forex Trading Accounts," 2023. [Online]. Available: <https://www.myfxbook.com>
- [5] Trademetria, "Trading Journal & Performance Tracking," 2023. [Online]. Available: <https://www.trademetria.com>
- [6] Google, "Google Sheets for Business," 2023. [Online]. Available: <https://workspace.google.com/products/sheets/>
- [7] BabyPips, "How to Keep a Trading Journal," 2022. [Online]. Available: <https://www.babypips.com/learn/forex/how-to-keep-a-trading-journal>
- [8] Notion, "Your Connected Workspace for Docs, Projects, and Knowledge," 2023. [Online]. Available: <https://www.notion.so>
- [9] Laravel, "The PHP Framework for Web Artisans: Documentation," 2024. [Online]. Available: <https://laravel.com/docs>
- [10] R. Elmasri and S. B. Navathe, **Fundamentals of Database Systems**, 7th ed., Pearson, 2016.
- [11] OWASP Foundation, "Top 10 Security Risks," 2023. [Online]. Available: <https://owasp.org/www-project-top-ten/>
- [12] I. Sommerville, **Software Engineering**, 10th ed., Pearson, 2015.