Princess Sumaya University for Technology
King Hussein School for Computing Sciences

TradeOpedia

Prepared By:

Yaman Khattab
Ahmad Zalmout

Supervised By:
Dr. Anas Abu Taleb

Project Submitted in partial fulfillment for the degree of Bachelor of Science in Computer Science

Semester-Year

2021-2022

Declaration of Originality

This document has been written entirely by the undersigned team members of the project. The source of every quoted text is clearly cited and there is no ambiguity in where the quoted text begins and ends. The source of any illustration, image or table that is not the work of the team members is also clearly cited. We are aware that using non-original text or material or paraphrasing or modifying it without proper citation is a violation of the university's regulations and is subject to legal actions.

Names and Signatures of team members:

Yaman Khattab

Ahmad Zalmout

Acknowledgments

We would like to thank the university of Princess Sumaya for providing this honorable opportunity to study and learn from experts in computer science. This opportunity provided us with enough knowledge and experience to help us build the system we explained and discussed in this document.

We could not have made a well-structured project without the supervision of Dr Anas Abu Taleb and his huge experience in graduation projects. We were pleased that he managed to help us in the decision of so many confusing points and to walk us through every chapter individually.

Also a special thanks for Dr.Nailah Al Madi and Dr. Rawan Ghnemat for helping and guiding us in a prototype we worked on as a first option of our graduation project which was courses scheduling for students.

We also would like to thank Dr Ammar Odeh for his kind words to keep us going forward without any hesitations. In addition, we highly appreciate his beneficial opinions on our requirements and guiding us to be the best project amongst our peers.

Special thanks to our families and friends who supported us along the journey and tolerated our mixed feelings and hesitations to continue the pursuit of the degree in the best way possible. Also, we are grateful for providing us with the studying atmosphere required to excel in our college life.

Summary

TradeOpedia is a trading platform website which allows the user to trade in the financial market and benefits from an intelligent system which will help increase profits. The purpose of making this system is to train young investors and help anyone who would like to start investing, especially people with no experience in the financial market and it is difficult for them to interpret regular trading platforms. Hence, there is no need to deposit any money into the wallet; rather, there is a fixed amount of virtual money deposited for every user to see how trading translates to profits or losses in the real world. This way money is not trapped in huge risks and could unluckily be lost due to lack of exposure to the market.

The intelligent system is based on well tested machine learning algorithms which are trained on large datasets that run back to more than 3 decades. The algorithms create models that can show predicted prices based on fundamental analysis. Meaning, our models are trained on historical prices to study each industry's fluctuations separately to produce such predictions. Rather than studying news headlines and experts' expectations written in articles; which is called sentimental analysis.

Each user has a fixed amount of virtual money to start using the trading website and test his skills on real live data presented on live charts. Also, there exist charts representing the predicted prices for each industry that allows the user to have a picturesque idea about where the stock is going. Additionally, the website has risk management tool; on one hand, it prevents the user from taking huge losses by setting a lower limit in the trade and cannot lose more than half of his investment, on the other hand, it prevents greedy profits and setting an upper limit to the trade in order to lower the chances of the risk of a sudden drop of the stock after a huge rise.

List of Abbreviations

Table 1 Table of abbreviations

Abbreviation	Definition
UML	Unified Modeling Language
NLP	Natural Language Processing
AI	Artificial Intelligence
ML	Machine learning
DDoS	Distributed Denial of Service attacks
CSV	Comma Separated Values
UI	User interface
TB	Terabyte
CVV	Card Verification Code
SQL	Structured Query Language

Table of Contents

Table of Figures

Table of Tables

Chapter 1
Introduction

1.1 Overview

The digital world is now evolving more than ever. New technologies are being invented. Technologies that are well integrated in our lives. One of the domains computers are getting into is finance. This project will be focusing on stocks, cryptocurrencies and currencies prices.

1.1.1 General description

There exist platforms that allows investments to be done from any computer. It is not stopping there rather; bots are now making investments by their own using data analysis on stocks and how prices are expected to change. What we are interested to do is not a simple data analysis; but a system that uses machine learning algorithms that study the market from previous stocks and predict their prices for a better judgment and maximize the profit.

1.1.2 Technicalities

When it comes to financial market, there are two important concepts:

1) Fundamental Analysis which is a method to measure company's financial statements, revenues and how politics effect the stocks as an example.

2) Technical Analysis which studies historical prices, charts patterns and trending market that include support and resistance.

1.1.3 Platform

Our system will be focusing on the technical analysis. Users can access the system through a web site that allows users to test the model on real data by allowing every user to have fixed amount of virtual money to use the predictions to learn more about the process of trading.

1.2 Problem Statement

Investments in stocks is one method of creating a secondary income or for some people the main income. Investments have helped numerous people grow a fortune. However, making the choice of when to buy or sell a stock is the most vital move in this domain, and so companies exist with main purpose of helping and advising investors on managing their investments called brokers.

1.2.1 Problem description

One problem is, brokers have a very bad reputation in the financial market as not all of them focus on the profit of their clients. Each broker gets paid differently; and one method is on the number of investments made per day/week. Which results in convincing clients to carry out more investments than needed so they can get paid more. Another problem is that brokers are human beings thus, they can be biased in their judgments and their decisions about a certain stock.

In addition, the stock market is not an easy world to get into and make profits from. As a new comer, it is a complex world full of confusion and extremely hard to understand.

1.2.2 Solution

Our solution is to create a website that is connected to a system that predicts prices of stocks based on previous prices dating back to at least 20 years. The system will generate reports guiding the investor for the most appropriate method to maximize his/her profit without relying on the someone else like brokers, and helping the less experienced investors.

1.2.3 Target audience

The audience comprises of any person who does not have experience in stock exchange or confused of where to start. Our users could be investors who would like to have a deep insight in the market and its future prices which will help them to decide more effectively.

1.3 Related Work

There exist numerous systems that facilitates technology in the sector of finance, including but not limited to:

Trading Technologies: it is a trading platform that allows trading in futures, options, and cryptocurrencies. The system can be accessed anywhere as long as there is internet connection. This system shows real-time data and sales through widgets and charts using technical analysis strategy. It uses automated trading strategies for faster trading decisions. It also provides the ability for users to build their own algorithm and deploy it in the client's portfolio. (Geannopulos, 2021)

GreenKey Technologies: it uses NLP to analyze speeches, conversations, and notes to see how reputations of companies are changing and in turn affect their stock price.

WOA: facilitates real time market analysis through AI algorithms but only to a strictly selected users including but not limited to high-net-worth traders.

Table 2 Companies in finance (Iac, 2021)

Company Name\Features	Trading Platform	Produce trading strategies	Use NLP to analyze text and conversations	Automated trading	Predicted prices	Pay to use system
Epoque	☒	☐	☐	☒	☐	☐
Trade Ideas	☒	☐	☐	☒	☐	☐
GreenKey	☐	☒	☒	☐	☐	☐
EquBot	☐	☒	☐	☐	☐	☐
WOA	☒	☒	☒	☒	☐	☒

What this project is focusing on is price prediction using ML algorithms on historical data. It was then discovered that ***Kavout*** is doing a very close functionality which is "uncovering hidden, dynamic, and nonlinear patterns in the financial markets" as it is said on their website. (unknown, 2021)

Our system takes the idea from ***Kavout*** of uncovering patterns of stocks to help users to have an insight of where a specific stock is going. It does not limit the audience of its users like ***WOA*** though. Our platform is considered a trading platform which cannot benefit directly from ***EquBot***'s capabilities. Also, it is interested in teaching less experienced traders and so cannot use automation. However, he idea of producing strategies and trading path sounds appealing and it is considered to be added in the future.

1.4 Document Outline

Table 3 Description of documentation structure

Chapter Title	Description
Chapter 1: Introduction	An overview of the project and a description of the solved problem. Also, it shows the related work of previous projects and the targeted audience.
Chapter 2: Project Plan	Detailed description of time completion for every task, consequent risk and its distribution to each member in the team. Also, it shows the total cost estimation and the tools that helped building the project.
Chapter 3: Requirements Specification	Description of stakeholders of the project and technicalities which consist of functional, non-functional and platform requirements.
Chapter 4: System Design	Description of logical model design which contain both low and high-level system design using object-oriented approach. Also, physical model design that describe the user interface, database design and reports design.

Chapter 2
Project Plan

2.1 Project Deliverables

Our system will be running on a website integrated with python scripts that run the ML algorithms that uses data sets in CSV files, the result will be screens showing data trends along with short reports related to each user individually and email notifications.

2.2 Project Tasks

Table 4: Tasks' Phases breakdown

Phase	Task Number	Task Name	Task Description	Dependencies	Period
Research and Analysis	T1	Team Formation.	Finding team members with adequate knowledge and passion in ML.		01/8 – 08/8
	T2	Team meeting.	Meeting with the team to decide the nature of the project to work on.	T1	10/8 – 11/8
	T3	Building prototype.	Working on a prototype of a scheduling system that creates semester courses schedules for students.	T2	12/8 – 1/10
	T4	Having different points of view.	Meeting with a number of Drs to show them the prototype. Having another project in mind and having a number of meetings to decide which project to implement.		01/10 – 18/10
	T5	Project idea decided.	Planning on building a trading platform that implements ML algorithms to predict future prices.	T4	18/10 – 19/10
	T6	Building technical ground bases.	Learning python libraries (like: pandas, NumPy, Matplotlib, yfinance, etc.) related to data science and ML to be implemented in the chosen project.	T5	29/8 – 24/10

	T7	Requirements gathering.	Searching for data sets for the system to train on. Searching for an appropriate ML algorithm that gives the best accuracy.	T5	01/10 – 24/10
	T8	Start documentation.	Beginning on writing overview and introduction.	T5	24/10 – 10/1
Design	T8	Project Upgrade.	Checking out trending related work for comparison and adding useful features to the project.	T5	31/10 – 27/11
	T9	System architecture.	Design of use-case diagrams, expanded use cases, sequence diagrams, activity diagrams, class and object diagrams.	T8	23/12 – 06/1
	T10	UI interface.	Designing the UI interface and all the screens of the website.	T9	25/12 – 09/1
	T11	Database design.	Design ER diagram with tables schema.	T9	05/1 – 08/1
	T12	Reports design.	Designing system reports for transaction and trades details.	T9	06/1 – 08/1

Figure 1 Gant Chart

2.3 Roles and Responsibilities

Both team members worked on researching about the idea of the project and its implementation in the real world. Both wrote the documentation separately and the result is the combination of the work were written in the final draft.

Algorithm type and implementation is carried out by both members as well as testing it on the data set.

Yaman Khattab was responsible of back-end of the website.

Ahmad Zalmout was responsible of front-end, database design and its schema.

2.4 Risk Assessment

Table 5 Systems' risks

Risk ID	Risk Name	Probability	Impact	Solution
R1	Lack of dataset.	High.	Reducing the performance of the trained model which leads to decrease in accuracy.	websites like “Yahoo Finance” and “Binance” were founded to provide historical data dating back 20 years.
R2	No high accuracy rates reached.	High.	Incorrect predictions resulting in loss of traders’ money.	Running multiple tests with varying training-to-testing ratio data sets. Also, testing more than one algorithm on the data. And providing warning to users that the prediction cannot be accurate 100%.
R3	DDoS attack.	Medium.	Server failures.	include DDoS protection provided by the hosting company.
R4	Invalid data entry.	Low.	False system output.	Implementation of validation rules on data entry.
R6	Unintentional button clicked.	High.	Undesirable trade occurring.	With critical buttons, implementation of confirmation message will be made.

2.5 Cost Estimation

Table 6 Total cost estimation for building the project

Name	Description	Cost Estimation
Website Domain Name	Reservation a name on the internet like “Examaple.com”.	$15 / Year
Website Hosting Server	Hosting the website on server that runs 24 hours to make it available for every user at any time and to provide the services that the website offers.	$35 / Year
Advanced plan for Hosting	Providing DDoS protection and SQL Injection Protection against attacks.	$1500 / Year
Maintenance	Check if there are any errors in the system and maintain it as fast as possible provided by the team members.	$150 / Year
Cloud Storage	Huge storage for backing up the data we have in our system because there are sensitive data which include (usernames, passwords, credit cards, phone numbers and emails) and to make the website scalable. 2 TB will be used for backing up the files by Google Cloud	$550/ Year
Total Cost Estimation		**$2100 / Year**

2.6 Project Management Tools

The tools that have been used in this project were:

- Oracle for Database
- Visual Studio Code
- PyCharm Community Edition 2020
- Jupyter notebook
- Gantt Chart Tool
- Diagrams.net Tool for building database schema.

Chapter 3
Requirements Specification

3.1 Stakeholders

In our system, users will start with 100,000 dollars in his/her e-wallet representing virtual money that can be used to place virtual trades in the system for training reasons and to test the predicated prices that the intelligent system generate.

Below is a summary of the stakeholders of the system. The table also mentions how stakeholders play a role in the system in addition to the critical level they belong to.

Table 7: Stakeholders

Stakeholders	**Roles**	**Critical level**
Admin	Manages the system and responsible for making sure the system runs smoothly and desirably as the user needs	High
Trader (Real Mode)	Uses the system to make profit from trading stocks and cryptocurrencies with his own money	High
Trader (Virtual Mode)	Uses the system to understand how stocks and cryptocurrencies work and train on virtual money to see how much profit can be made according to the system's charts and predictions	High
Banks	Export and import money according to the user's choice of trading	High

3.2 Platform Requirements

Below is a description of what the user has to have to use the system as efficiently as possible. Also, there is what the server side needs for a smooth operation.

Table 8: Server/Client requirements

Systems' sides	Requirement number	Description	Importance level
Client side	1	Internet connection	Essential
	2	Personal computer or Mobile	Essential
	3	Screen size more than 17 inches	Recommended
	4	Internet browser	Essential
	5	Bank account/E-wallet	Essential (not for training)
Server side	1	Internet connection	Essential
	2	Large storage	Essential
	3	Python support	Essential
	4	Cloud storage	Essential
	5	DDoS and SQL injection protection	Essential

3.3 Functional Requirements

And now is a summary of the functional requirements the system has and what it offers to the users.

Table 9: Functional Requirements

System Pages	Requirement ID	Requirement Name	Description	Priority
Registration Page	FR1	Sign up	Registration of new users to access the system	Essential
	FR2	Sign In	Signing in by the username and password	Essential
	FR3	Forget Password	Request to change the password through email	Essential
	FR4	Sign out	Logging out of the website	Essential
Transaction page (Real Mode)	FR5	Deposit	Add money to the wallet.	Essential
	FR6	Withdraw	Withdraw money from the wallet.	Essential
	FR7	Trade	Buy or sell shares of cryptocurrencies, currencies, or stock.	Essential
Transaction Page (Virtual Mode)	FR8	Trade	Buy or sell shares of cryptocurrencies, currencies, or stocks with the market prices but using virtual money	Essential
Home screen	FR9	The market	View graphs of cryptocurrencies, currencies, and stocks.	Essential
	FR10	Search	Search anything on the website by name	Optional
	FR11	Notification	Shows the user important information related to his/her trades.	

Portfolio Page	FR12	Portfolio	Showing the user total Equity, total profits, detailed information about the deals in progress and in past with profits/losses in every trade.	Essential
	FR13	Cancel Trade	User can cancel any trade in progress	Essential
	FR14	Update Trade	User can update any trade in progress by changing the take profit and stop loss.	Essential
Prediction Page	FR15	Predictions	User can access the intelligent system which shows the predicted prices of the market he/she interested in.	Essential

A description of the functional requirements with their input, output, constraints, and the detailed process of each are in the table below.

Table 10: Functional Requirements Description

Req. ID	Input	Output	Constraints	Process
FR1	Username, Password, Phone number, Email, user type.	An SMS and email containing a verification code will be sent to the verify the user. A notification welcoming the user to the system via email will be sent.	Username shouldn't be duplicated in the database. Password must be at least 8 character containing numbers, symbols, capital and small letters. Length of the phone number should be 10. Email address should be valid.	The user will type a unique username and password, followed by phone number email and user type whether it is trading or training. User will click on create account button which will send a SMS and email verification code to verify them that they belong to the user.
FR2	Username, Password.	User will be directed to the home page.	Check if there exist the username entered, and check if the password belongs to this username.	User enters username and password.

FR3	Username, phone number or email, and the code sent via mobile phone or email.	An SMS or email will be sent to the phone number or email address provided by the user containing a code. After the process is done, a notification will be sent to notify the password have been changed.	Username entered have to be present in the database and phone number or email have to match that username. Code will be randomly generated.	User clicks on forget password button, then the user enters his username and chooses whether to send a SMS or an email containing a code. The screen will show the user a textbox where the user has to enter that code to securely allow only the user him/herself change the password. User will type a new password with the same constraints in FR1 and the password will be updated.
FR4	Click on the sign out button.	User will be directed to the log in page.	None.	When the user clicks on sign out button, the profile will close and the page will be directed to the log in page.
FR5	Amount of money to deposit in US dollar, credit card number expiration date and CVV.	SMS and email will be sent notifying the user that funds were added to the wallet.	User has to be trader. Checks if credit card number is valid.	Trader goes to transaction page and clicks on deposit button. Then system will ask to input credit card information. From the credit card number, the system will direct the user to the designated bank withdrawal page to withdraw The wallet will be updated.

FR6	Amount of money user want to withdraw from the wallet, confirmation code.	SMS or email will be sent containing a confirmation code. A notification indicating that a withdrawal has been done.	User has to be trader. Total amount should be greater than 10 dollars in order to withdraw. Verification of code typed successfully completed.	Trader goes to transaction page and click on withdraw. Then form will pop up which asking for the amount of money the user wants to withdraw and the verification method the user wishes to choose from. It will be done either by SMS or email. Once the user is verified, the wallet is updated and the requested amount will be sent to his registered credit card.
FR7	The user chooses the preferred market to trade in (cryptocurrency, currency, or stocks) and the amount the user wishes to trade with.	Notification will be sent informing that a trade has took place.	User has to be in Real mode. User can't trade on the days where the market is closed. User need to have money in their wallet.	Trader goes to transaction page and click on the trade button. Then a list of available markets will be shown to choose which one the user wishes to put the money in.

FR8	The user chooses the preferred market to trade in (cryptocurrency, currency, or stocks) and the amount the user wishes to trade with.	Notification will be sent informing that a trade has took place.	User has to be in Virtual Mode. User can't trade on the days where the market is closed. User need to have money in their wallet.	Trainee goes to transaction page then click on the trade button. A list of available markets will be shown to choose which one the user wishes to put the money in.
FR9	None.	A number of graphs will be on the home page showing the past, current and future prices of stocks available on the system.	There will be cryptocurrencies, currencies, and stock markets of some international firms.	The user can maximize the graph to see it on a bigger scale by hovering the mouse pointer over the graph.

FR10	Enter keywords to search for.	Search results will show up.	Make sure that the keywords are valid and spilled in the right way.	The user will click on search button and will type keywords to search for. If the keyword is not valid, an error message will be displayed instead of the search results.
FR11	Click on the notification icon.	They will be shown in a small menu on the home page. As well as, every notification will be sent also to the email. Shows the user important updates to his portfolio.	Sorted by date received (newest first).	When the system predicts sudden increase or decrease it informs the user of possible sudden changes. If a sudden change actually happen the system notifies the user. It also shows all transactions happened in his portfolio like (sell, buy, withdraw, deposit). Notifications also are sent to the email.
FR12	Click on the portfolio button to be directed to the portfolio page.	User will be redirected to his/her portfolio which contains the profits, losses, total equity, time and date the deal happened or still in progress.	User must have been carrying out trading in order to show results.	User can click on any deal which either in the past or present and it will show detailed information regarding the deal like price opened, price closed, volume, or risk management if the deal still in progress.
FR13	Click on the cancel trade	Warning message will be popped up to the user to confirm closing the trade	Trade must be in progress	User will click on the cancel trade and click on yes to confirm canceling trade.

FR14	Click on the update trade and enter the changed stop loss and take profit.	Form will be popped up to the user to change the take profit and stop loss to the trade	Stop loss can't be greater or equal to the current price Take profit can't be less than or equal to the current price.	User will click on update trade and will change the take profit and stop loss.
FR15	Click on prediction page. Then, industry name.	Chart will be shown to the user indicating the predicted price for the industry chosen.	None	User will click on prediction page and choose the industry he/she interested in. Then, Chart will be shown to the user indicating the predicted price.

3.4 Non-Functional Requirements

The table below shows the non-functional requirements of the system and its description.

Table 11: Non-Functional Requirements

ID	Requirement	Description
NFR1	Security	High level of security by encrypting the database which will be hard to understand and decrypt. Using Secure Sockets Layer in the website which makes the communication between the server and user encrypted.
NFR2	Availability	The website is working 24/7 by the best hosting plan including DDoS protection that prevent the server of going down regarding many fake requests, and that will allow users to access the website from anywhere at any time
NFR3	Easy to use	System is designed very well and simple to use for any user at any age.
NFR4	Scalability	The design of our system is responsive which make it accessed by any device. Also, it will handle as much users and requests as possible.
NFR5	Performance	System is working on its highest performance by implementing the back-end and machine learning algorithms as efficiently as possible.

3.5 Other Requirements

More requirements would be related to:

1. Jordan's financial restrictions.
2. Amman stock exchange.
3. Banks restrictions on the amount and number of transactions.

Chapter 4

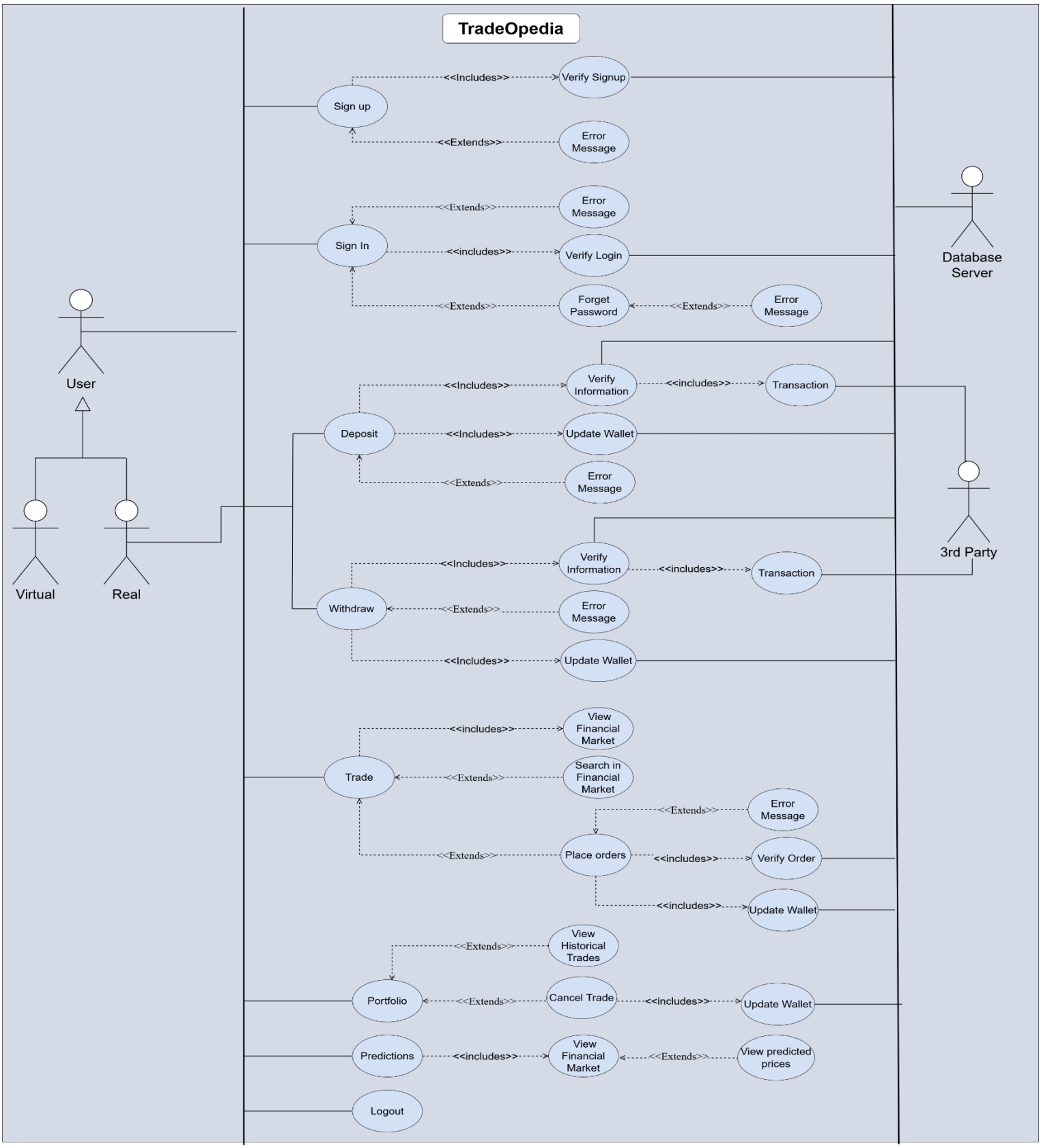
System Design

4.1 Logical Model Design

4.1.1 Use Case Diagrams

Use case diagrams shows what use cases and features each actor (user of administrator) has the privilege to use. It also shows how each use case make have other use cases which help in its functionality (includes) or adds to it (extends). Below the diagram is the expanded use case diagram which describes in details the process of each use case, success scenarios, actors involved and much more details.

Figure 2 Use case Diagram



Use case: Sign up

Actors: Users, and Database Server.

Purpose: User is able to create an account in order to log in to the system and use its features.

Overview (success scenario): User enters information like username, password, email, phone number, and choosing the type of user (trader, trainee). Then, after database server checks validation of data, an account created successfully.

Type: Primary.

Cross Reference: FR1.

Typical Course of Actions:

Actor Actions	**System Response**
1- User enters the information required.	2- System will check the validation of the data entered and the uniqueness of the username.
	3- System will send a verification code via email or SMS in order to authorize the user to create an account.
	4- Account is created successfully and the user will be redirected to login page to use the system.

Alternatives:

- If the user enters existed username, system will pop up an error message asking the user to change it.

- If the user re-entered a password which is not the same as the original one, system will pop up an error message to make sure that the password entered twice should be the same.

- If the user entered non valid email, system will pop up an error message asking the user to enter a valid email.

Use case: Sign in.

Actors: Users, and Database Server.

Purpose: logging the user into the system.

Overview (success scenario): User enters the username and password. After the system checks the validity of the information, the user will be redirected to the home page.

Type: Primary.

Cross References: FR2, FR3.

Typical Course of Actions:

Actor Actions	**System Response**
1- User enters the username and the password.	2- System will check the validation of the data entered.
	3- System will authorize the user to login into the website if only the username and password are correct.
	4- System will redirect the user to the home page.
5- User has the option to change the password.	6- System will ask the user to enter the user's email or phone number.
	7- System will check the existence of the email entered or the phone number.

	8- System will send a verification code to the email entered or to the phone number.
9- User enters the verification code.	10- System will check if the code typed is correct and authorize the user to be redirected to the reset password page.
11- User will enter a new password.	
12- User will re-enter a new password.	13- System will check if both passwords entered by the user are identical.
	14- System will update the password for the username.
	15- System will redirect the user to login page.

Alternatives:

- If the user enters username that does not exist in the database, system will pop up an error message to enter an existing email or create a new account if user does not have an account.
- If the user enters a wrong password for the username entered, system will pop up an error message to retype the password.
- If the user entered a wrong email or phone number after clicking *forget password*, system will pop up an error message to enter an existing email or phone number.
- If one of the passwords entered in the reset password page is not the same to the other one, system will pop an error message to make sure both passwords are matching.

Use Case: Deposit

Actors: Trader users.

Purpose: Depositing money to the wallet on the website to start trading.

Overview (success scenario): User enters the type of transaction like: PayPal, Mastercard, and Visa card. Then user will fill the form with the information required like: Cardholder's name, Card number, CVV, Expiration date, and amount of money. After the system receives the amount entered, the money will be added to the wallet.

Type: Primary.

Cross Reference: FR5.

Typical Course of Actions:

Actor Actions	**System Response**
1- User will choose the type of transaction and fill the form with the information required.	2- System will check the validity of the information by the following: check if credit card number is valid, and check if expiration date is not yet met.
	3- System will send the information to a 3rd party website who is responsible for financial transactions.
	4- System will receive the money from the bank account and system will update user's wallet.

Alternatives:

- If any of the entered data was invalid like (Cardholder's name, card number, CVV, and expiration date) system will pop up an error message to reenter the information correctly.

- If the 3rd party could not withdraw the money, system will pop up an error message indicating that the money could not be withdrawn from the user's account.

Use Case: Withdraw.

Actors: Trader users.

Purpose: Withdraw money from the wallet on the website to the bank account of the user.

Overview (success scenario): User enters the type of transaction like: PayPal, Mastercard, and Visa card. Then user will fill the form with the information required like: Cardholder's name, Card number, CVV, Expiration date, and amount of money. After the system sends the amount entered to the bank, the money will be removed from the wallet.

Type: Primary.

Cross Reference: FR6.

Typical Course of Actions:

Actor Actions	**System Response**
1- User will choose the type of transaction and fill the form with the information required.	2- System will check the amount of money specified to be withdrawn is less than or equal to the total money of user's wallet.
	3- System will send the information to a 3rd party website who is responsible for financial transactions.
	4- System will send the money to the bank account and system will update user's wallet.

Alternatives:

- If any of the entered data was invalid like (Cardholder's name, card number, CVV, and expiration date) system will pop up an error message to reenter the information correctly.

- If the 3rd party could not deposit the money, system will pop up an error message indicating that the money could not be deposited to the user's account.

Use Case: Trade.

Actors: Users.

Purpose: user will view and search in financial market and choose what to trade in.

Overview (success scenario): user will choose a stock, currency, or cryptocurrency to start trading. user enters an amount of money to invest but have to be less than or equal to the total money in the wallet. Taking the profit and stopping the loss is automatically set to 50% of the amount of money specified by user and it can be changed by the user. Then money will be withdrawn from the user and trade will be placed successfully.

Type: Primary.

Cross Reference: F7, F8, F9.

Typical Course of Actions:

Actor Actions	**System Response**
1- **User chooses what to invest in.**	
2- **User then chooses amount of money and the stop loss and take profit of the trade.**	
3- **User can disable the stop loss and take profit feature.**	
	4- System will check validity of the information entered.
	5- System will place an order if the information entered are correct.
	6- System will withdraw from wallet the amount of money specified for a trade.

	7- System will cancel the trade if user enabled automatic trade stop and the take profit or stop losses reached its limit.
	8- After system automatically stops the trade, it will transfer the amount of money earned or lost in the trade to the wallet.

Alternatives:

- If the amount of money specified for the trade is greater than the total money in the wallet then, system will pop up an error message indicating that trade cannot be completed.
- If the amount of money specified for the trade is not greater than or equal to the minimum money specified for every trade, system will pop up an error message shows that trade cannot be completed.
- Stopping losses shouldn't exceed half of the money specified for a trade. Otherwise, system will pop up an error message shows that trade cannot be completed.

Use Case: Portfolio.

Actors: Users, Database Server.

Purpose: user will view and search in the historical and current trades by showing profits and losses. Also, user can cancel the current trades and check the total profits or losses of his/her overall trades.

Overview (success scenario): user will search in the historical trades and check for every trade how much he/she gained or lost. User can cancel any current trades or update take profit or stop loss.

Type: Primary.

Cross Reference: F10.

Typical Course of Actions:

Actor Actions	**System Response**
1- **User can view the historical trades and check the profits or losses he/she made and also the overall trades.**	
2- **User is able to cancel any current trade or update the stop losses and take profits.**	3- System will cancel the trade if user clicked cancel trade or if the stop losses or take profits exceeds the percentage specified by user.
4- **User can update take profit and stop loss.**	5- System will update the take profit or stop losses as the user requested.

Alternatives:

- An error message will pop up if the user tries to update the percentage of take profits or stop losses greater than 50% for the money specified for the trade.

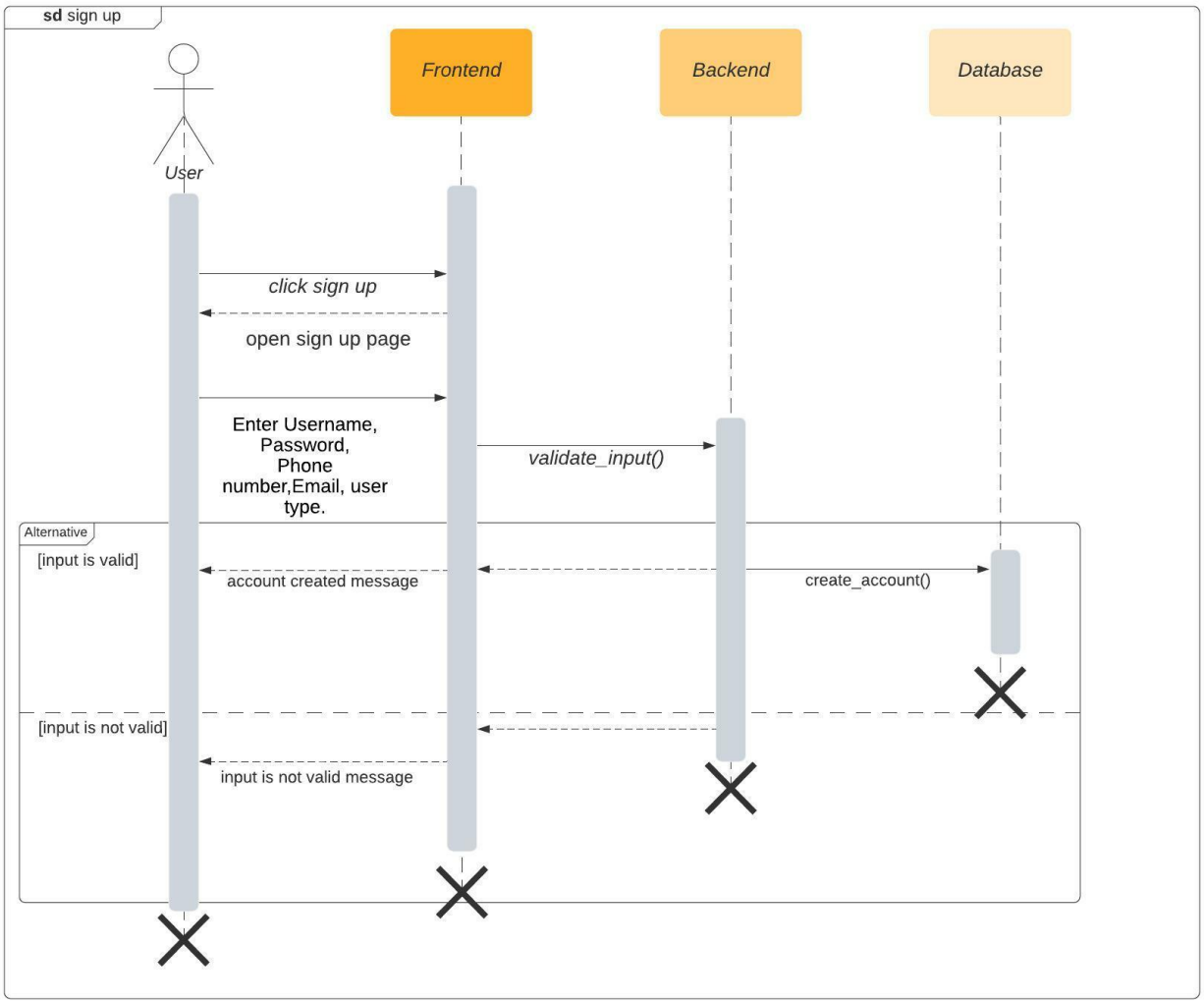
4.1.2 Sequence Diagrams

This diagram shows the interaction of an actor (a user or an administrator) with the objects of the system in a time sequence. Each arrow pointing to the right indicate that certain message or a command is sent from the actor to the object, and the arrow pointing to the left indicated the response (if present) from the object back to the previous object or the actor.

Sign-up sequence diagram:

The figure below shows the sign-up process. It starts when the user wants to create an account on the system. The user clicks on create an account, then the system opens the sign-up page. The user enters the required information to create the account. Backend object checks the validity of the information; and when it is valid the account is created, stored in the database, and the user is informed via email and SMS. But if the information is not valid, the backend object displays to the user that the account was not created because information was not compatible with the system requirements.

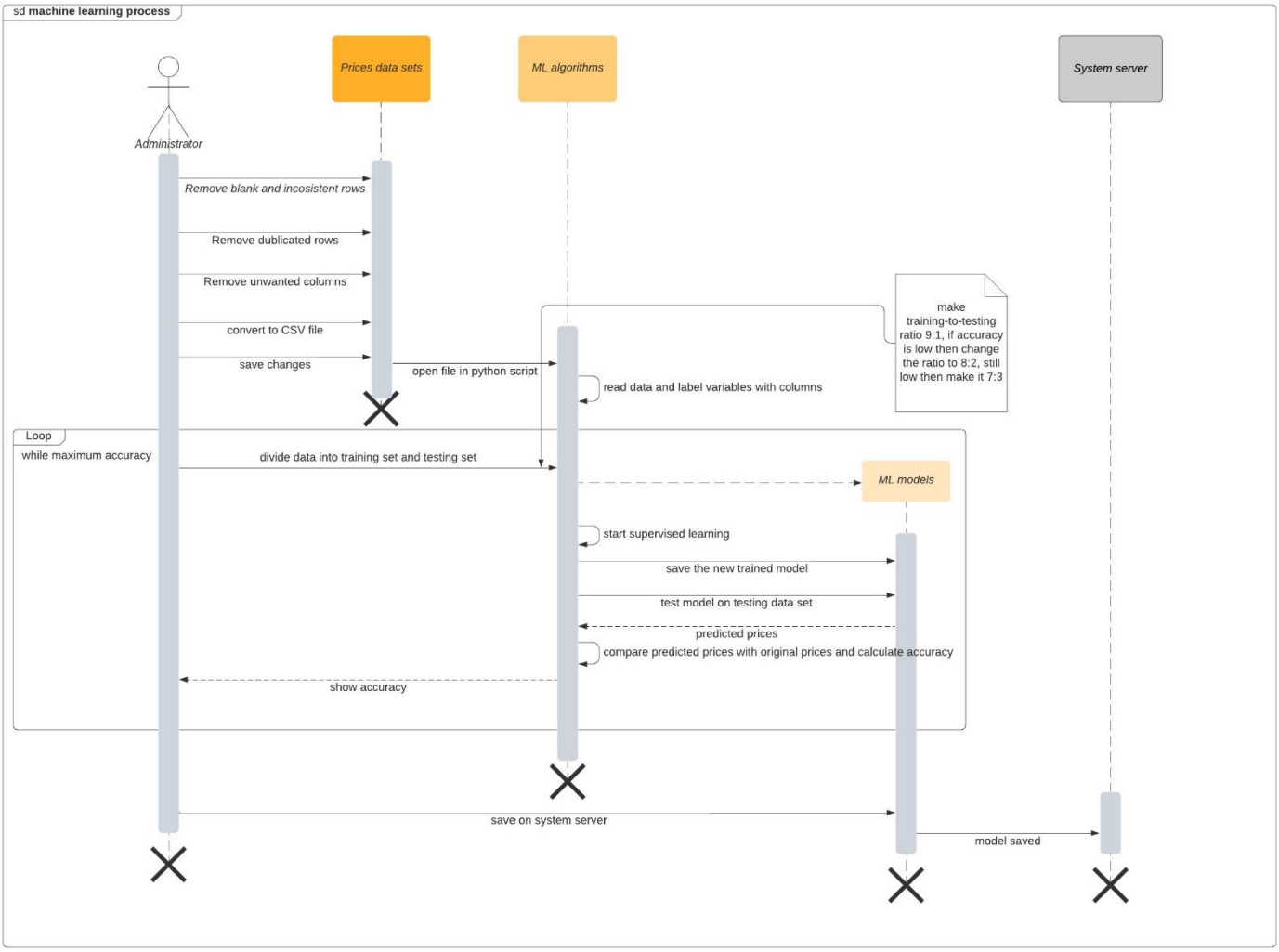
Figure 3 Sign up Sequence Diagram



Machine learning sequence diagram:

The figure below explains the machine learning process and how predicted prices are created. It starts by carrying out data filtering on the data sets so that ML algorithm can work and train itself on them. Then data sets are split into 2 parts, the first is for training the algorithm and the second is for testing. After that, a supervised learning mechanism is carried out. After training, testing phase starts by predicting prices of times that was not a part of the training set. Then a comparison between the predicted and the actual data from the testing set is done. This is called the accuracy, the higher the accuracy the better because it means that prediction is precise and reliable. If the accuracy is low, the administrator changes the training-to-testing ratio and repeats the training once again until a high accuracy is achieved, and if this did not help the algorithm is changed.

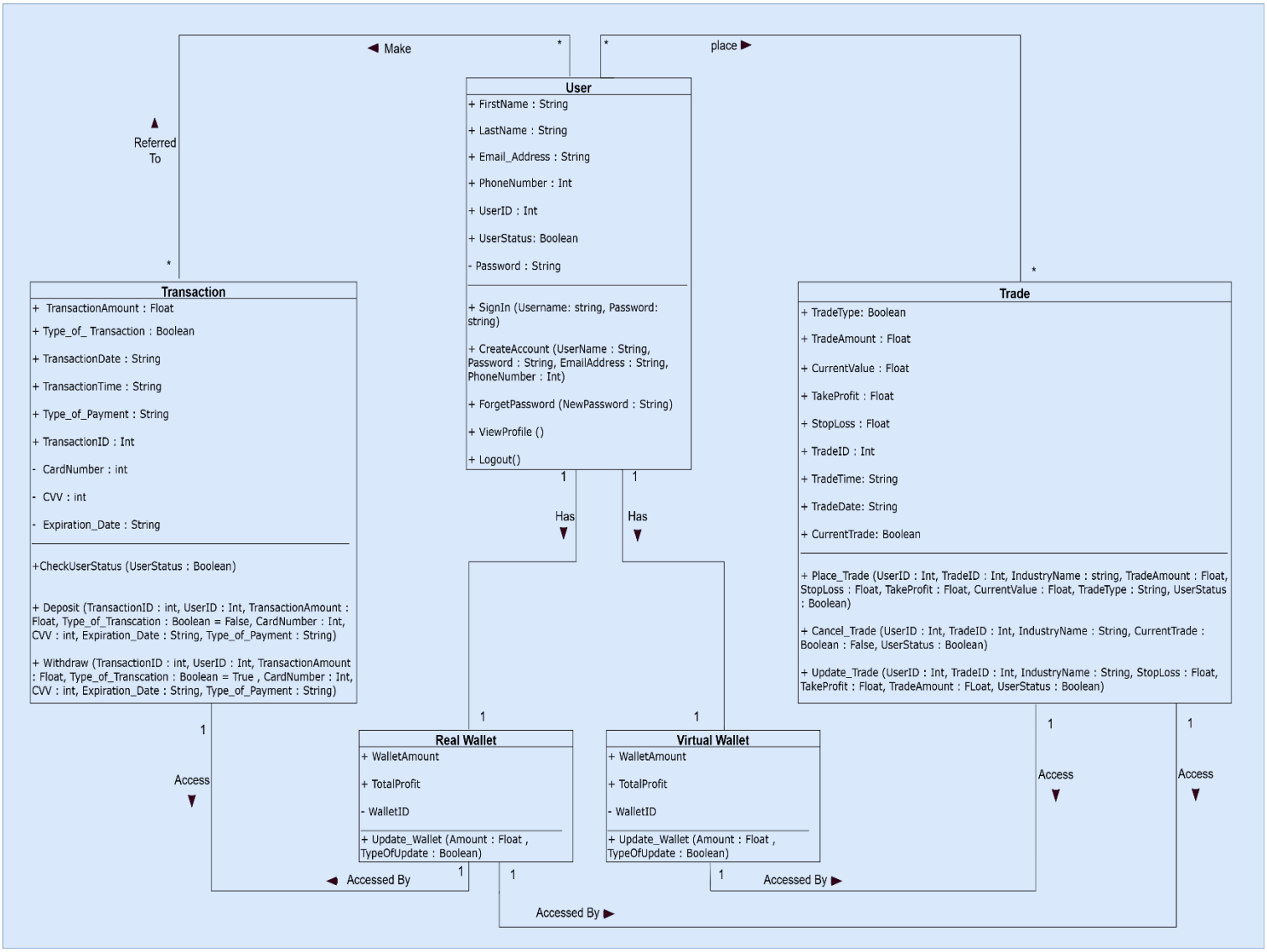
Figure 4 machine learning process



4.1.3 Class Diagram

As an object-oriented approach was selected to represent the system, a class diagram was necessary to represent the different classes the system have. It shows how each class is connected to the other at the same time showing what each class require from attributes and methods.

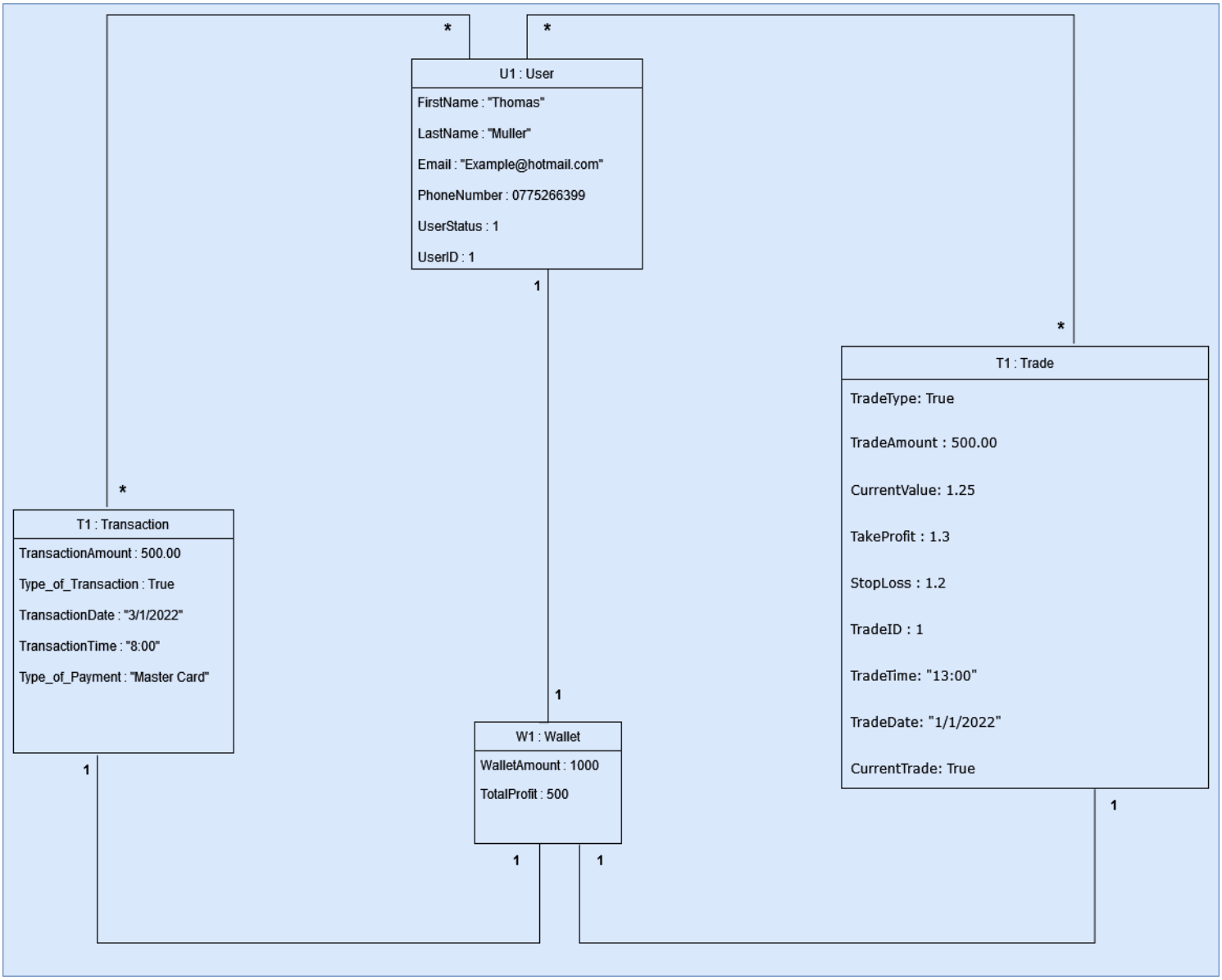
Figure 5 System Class Diagram



4.1.4 Object Diagram

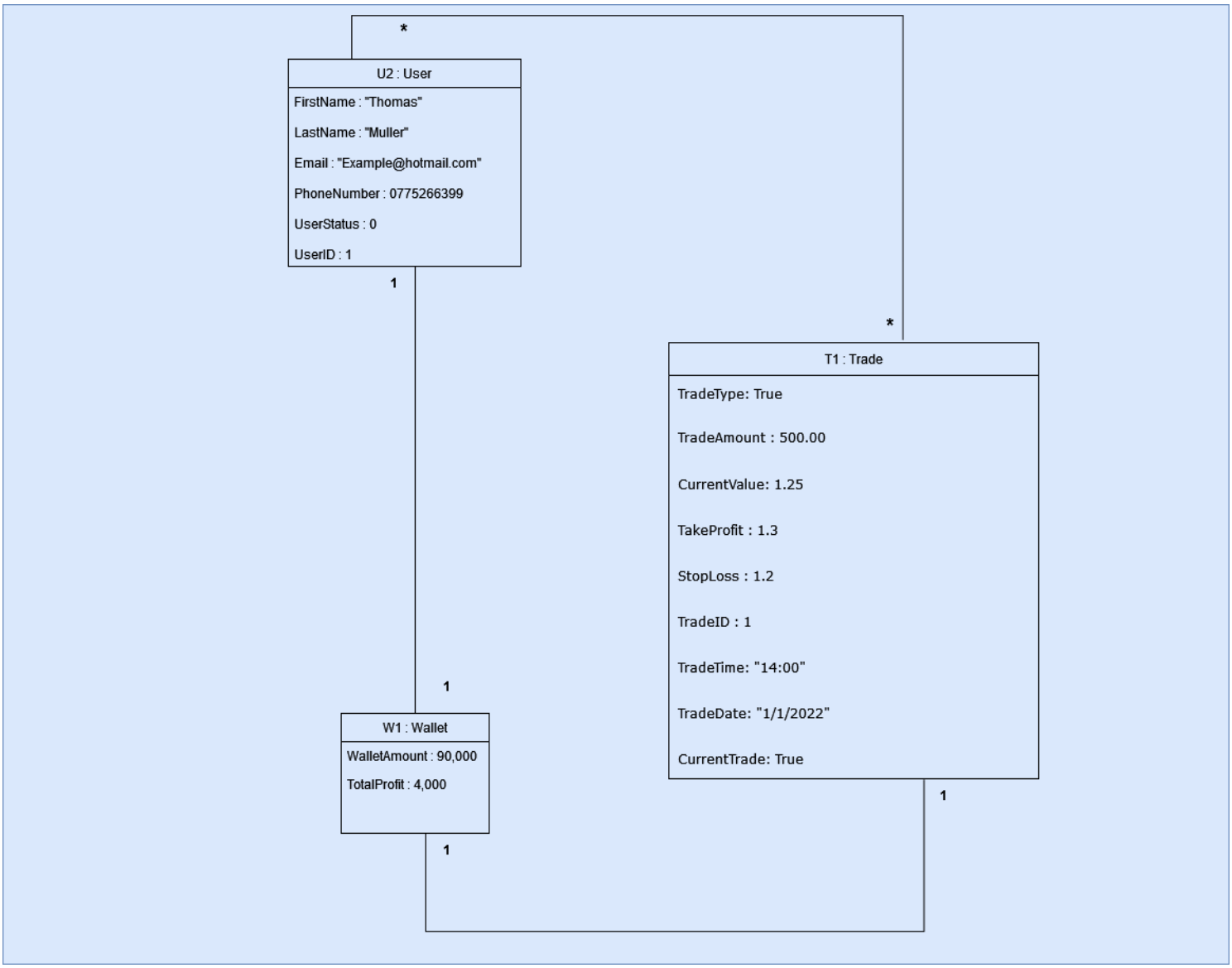
An object diagram shows an instance of the class diagram. Below is an instance of a user that carried out an actual trade and used the type of wallet for the actual trading process.

Figure 6 Actual trading account Object Diagram



The object diagram below shows the same user who carried out a virtual trading process that used the type of wallet for the virtual trading process.

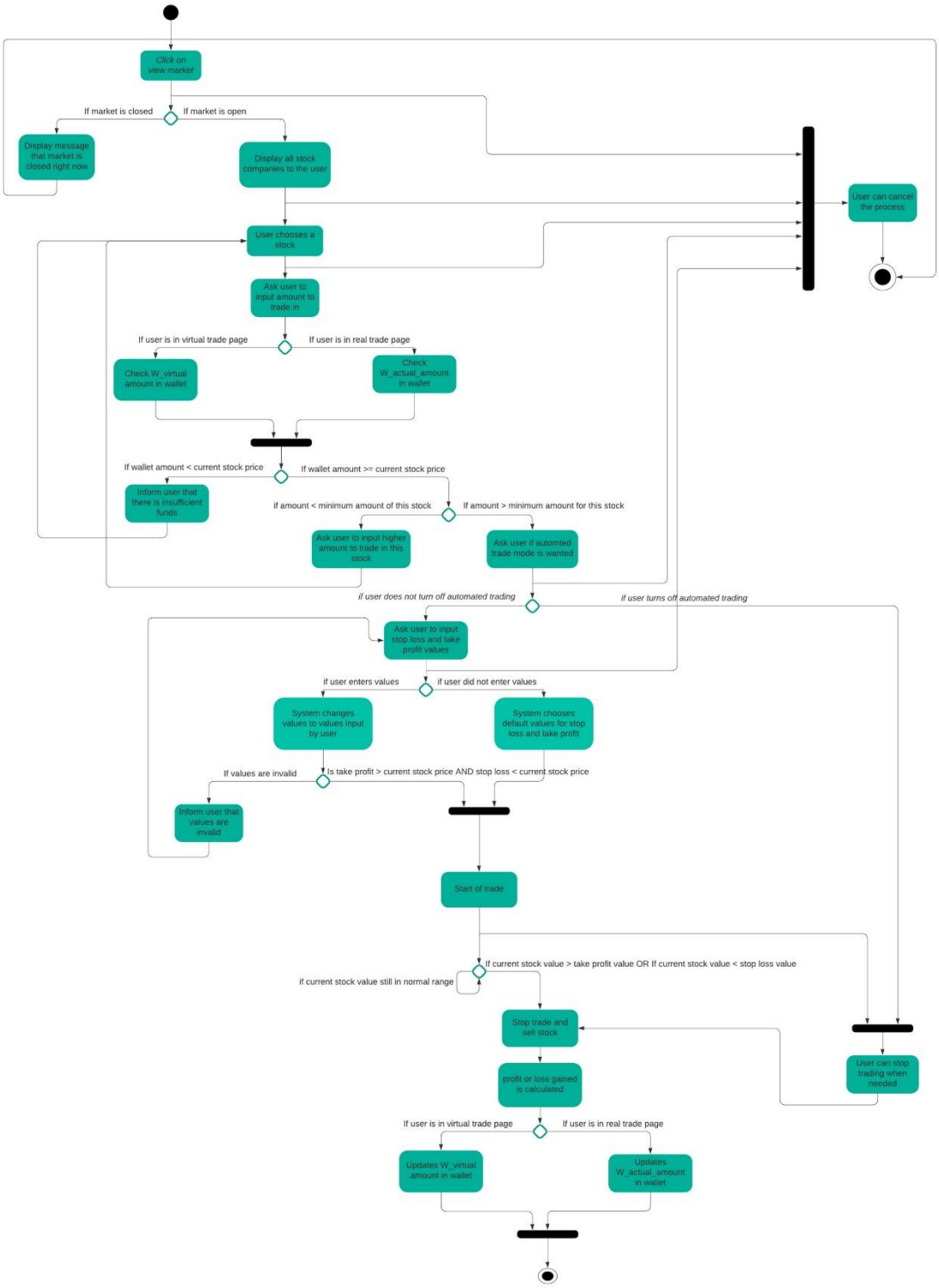
Figure 7 Virtual trading account Object Diagram



4.1.5 Activity Diagram

This diagram shows the trading process and its details.

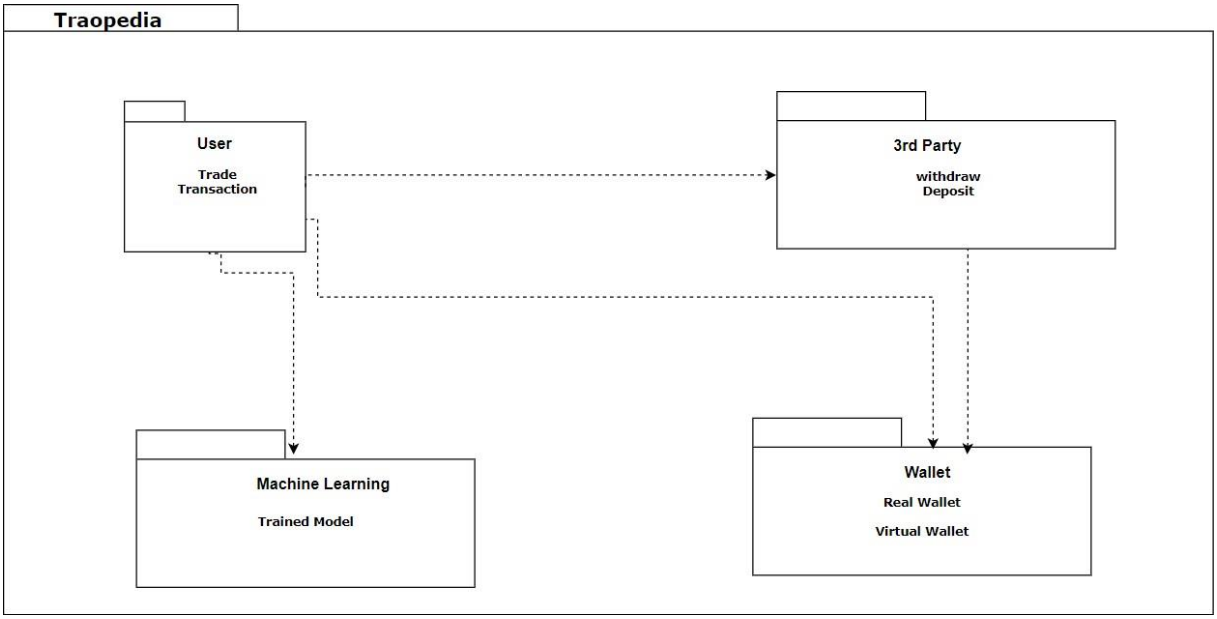
Figure 8 Activity Diagram



4.1.6 Package Diagram

Package Diagram simplify and organize class diagram of the system.

Figure 9 Package Diagram



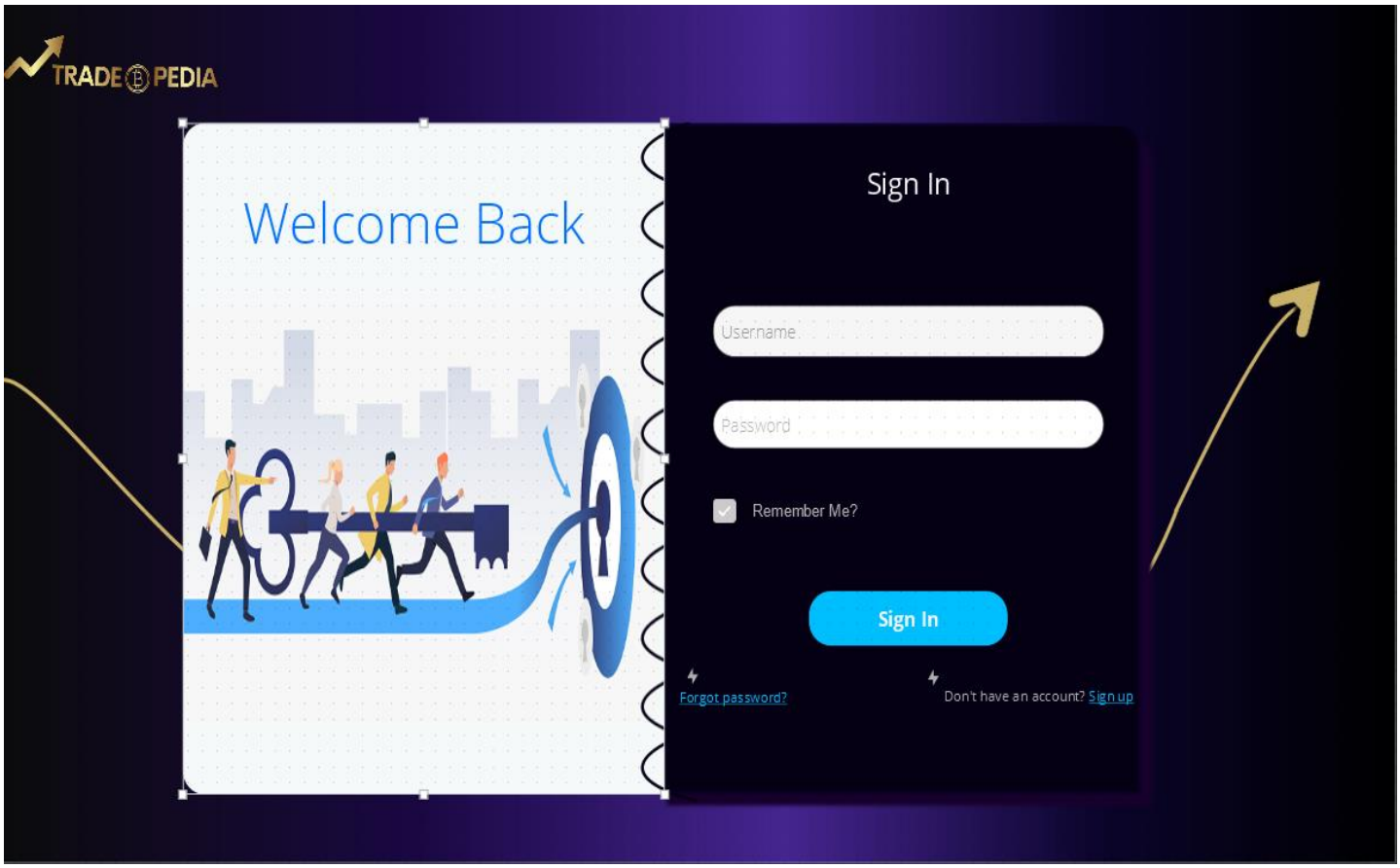
4.2 Physical Model Design

In this section, we explain and show our system in the real world and how it will actually turn out to be. All of the screens that the user will see is designed and present below. Along with the design of the database that will store all the data that comes in and out of the system.

4.2.1 UI Design

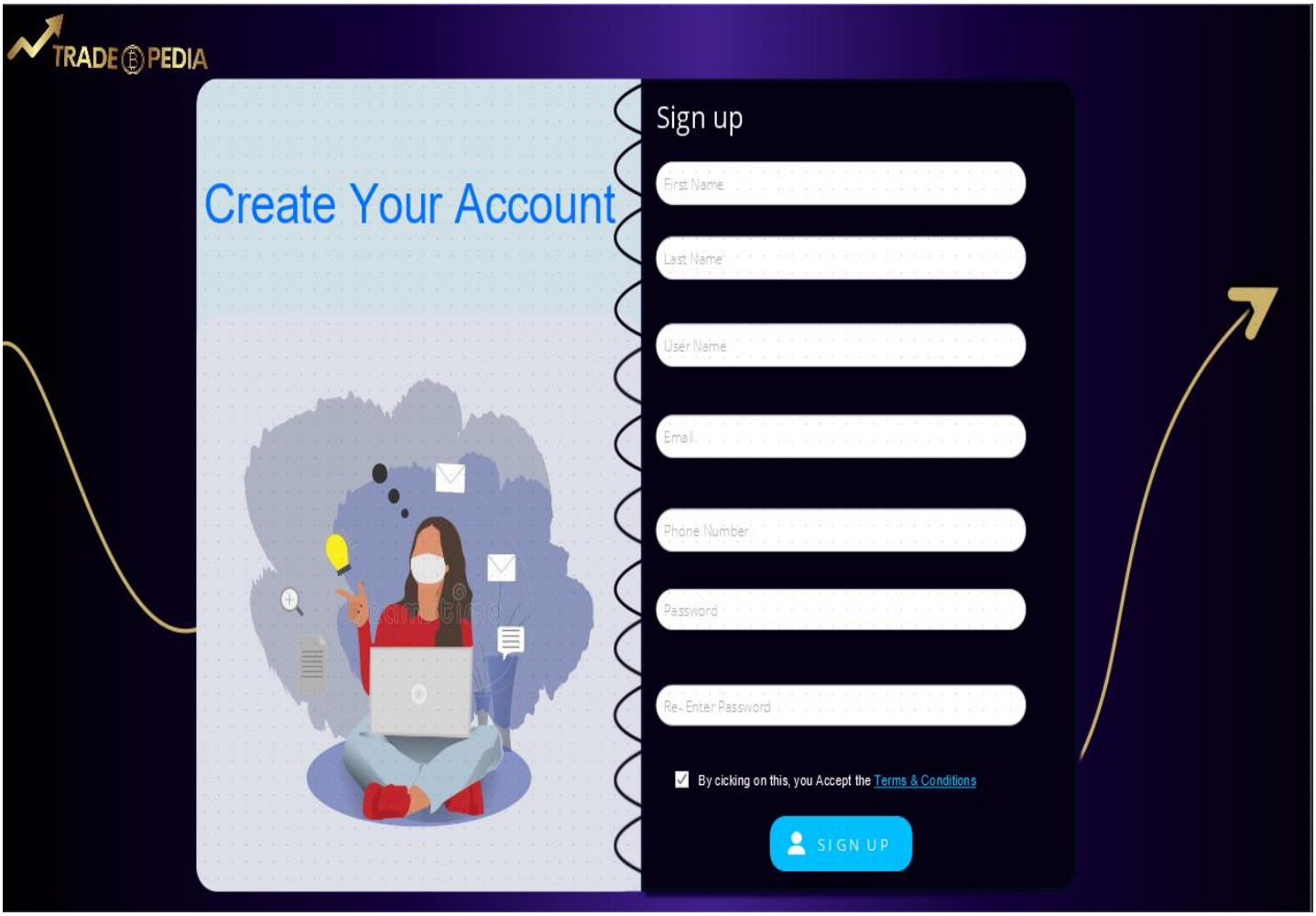
We begin by showing the first screen the user will see when the website is open.

Figure 10 Sign in Page



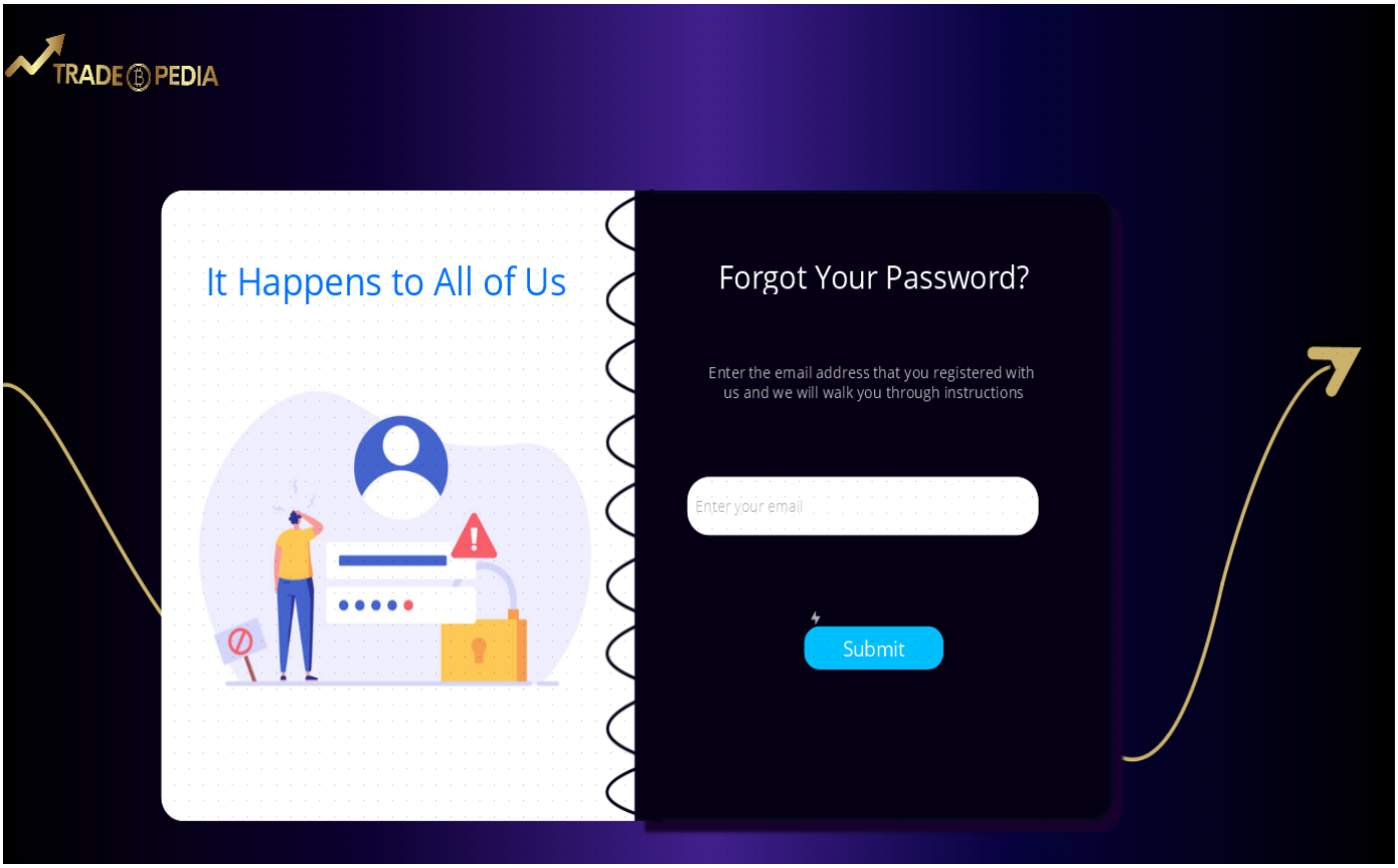
By clicking on sign up which exists in the right bottom of the sign in page, the user will be directed to sign up page.

Figure 11 Sign up Page



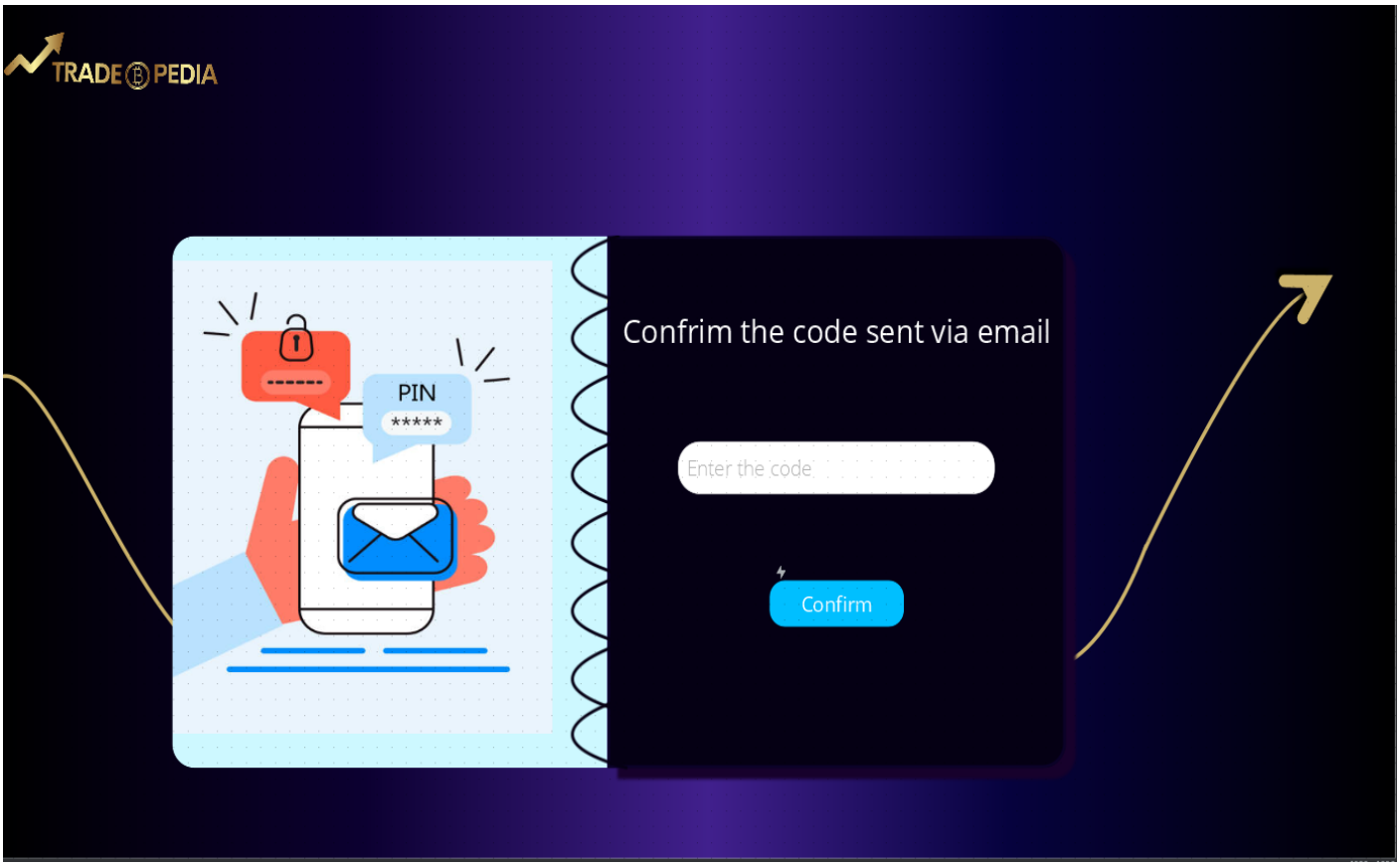
By Clicking on Forget Password which exists in the left bottom of the sign in page, user will be directed to forget password page.

Figure 12 Forget password Page



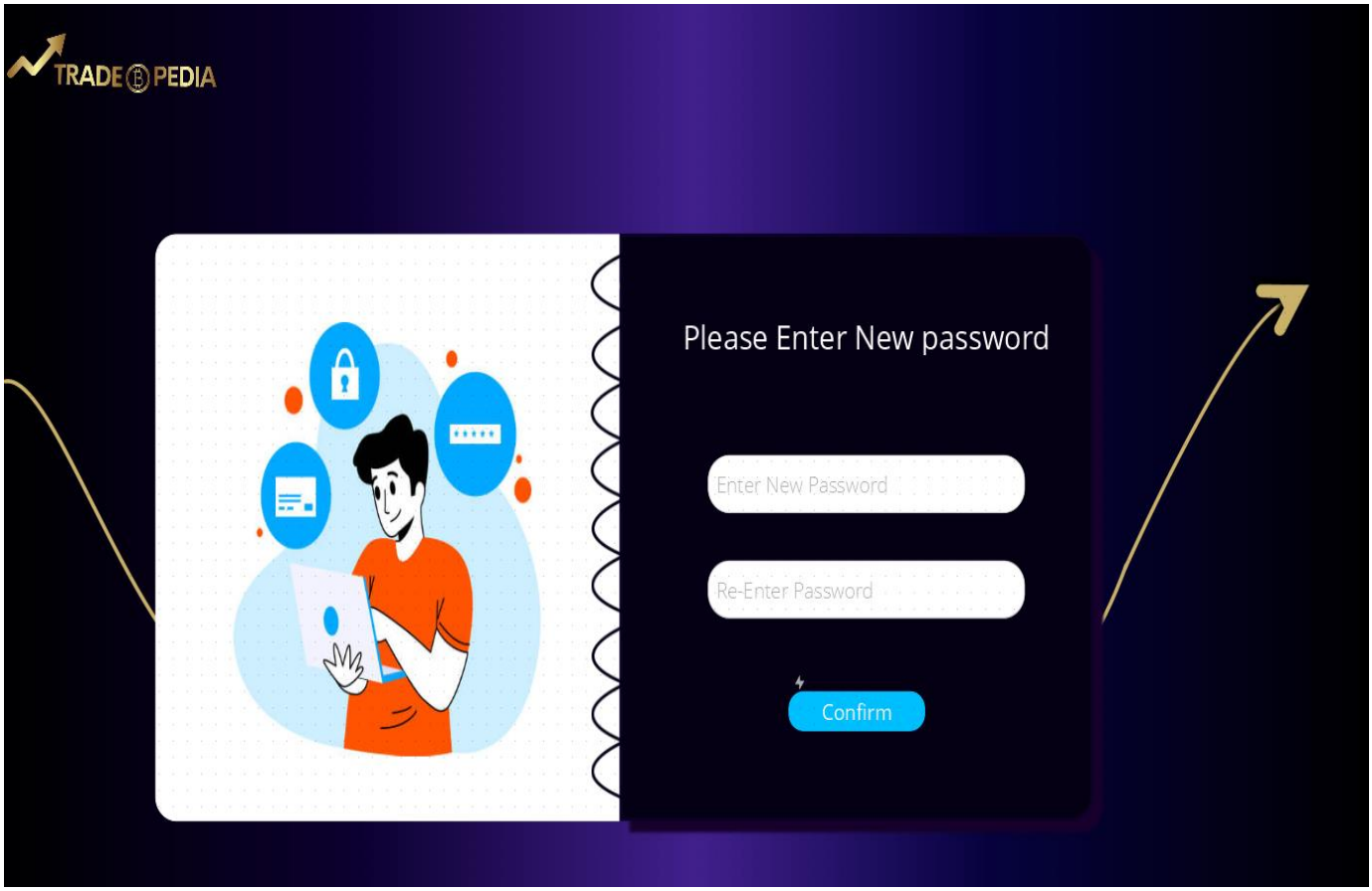
During the forget password process, a verification code will be sent to the same email or phone number used to register on the website. And this is the screen the user will see.

Figure 13 Code verification Page



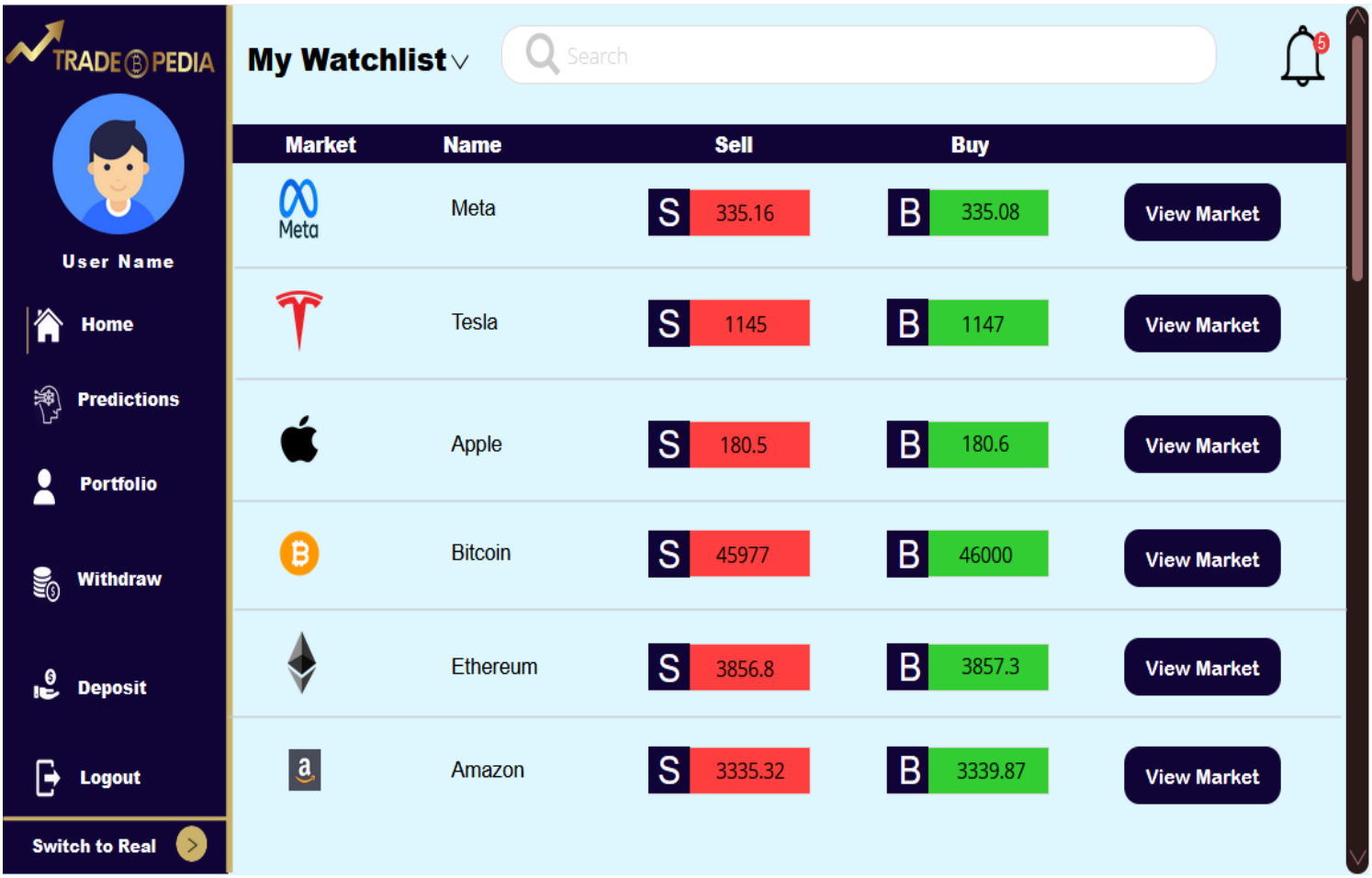
After confirming the code, user will be redirected to another page contains a form to reset the password.

Figure 14 Reset password Page



This is the home page of the website after the user logs in the account.

Figure 15 Home Page

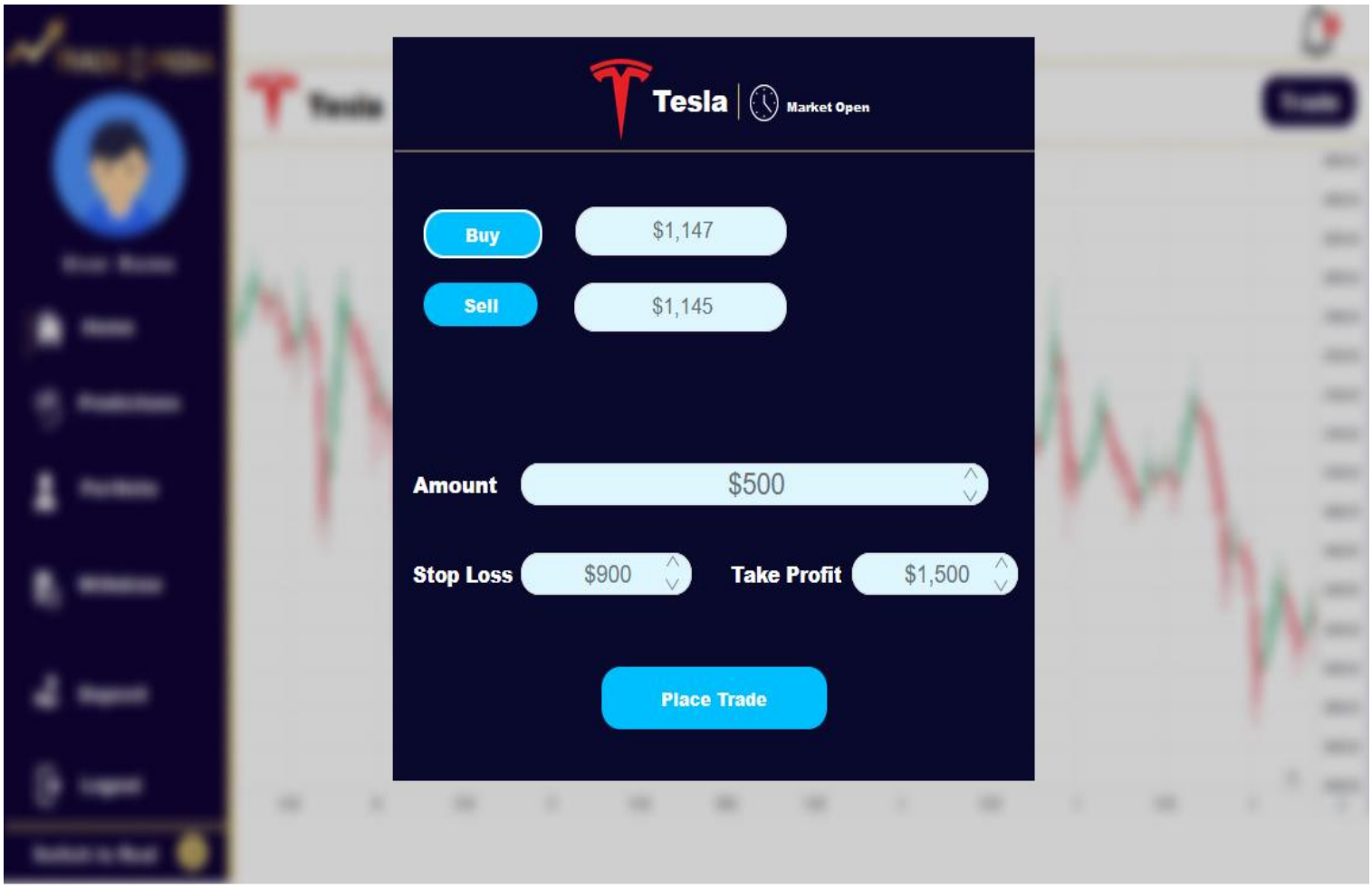


This page will open when the user clicks on View Market in the Home Page.

Figure 16 Market Page

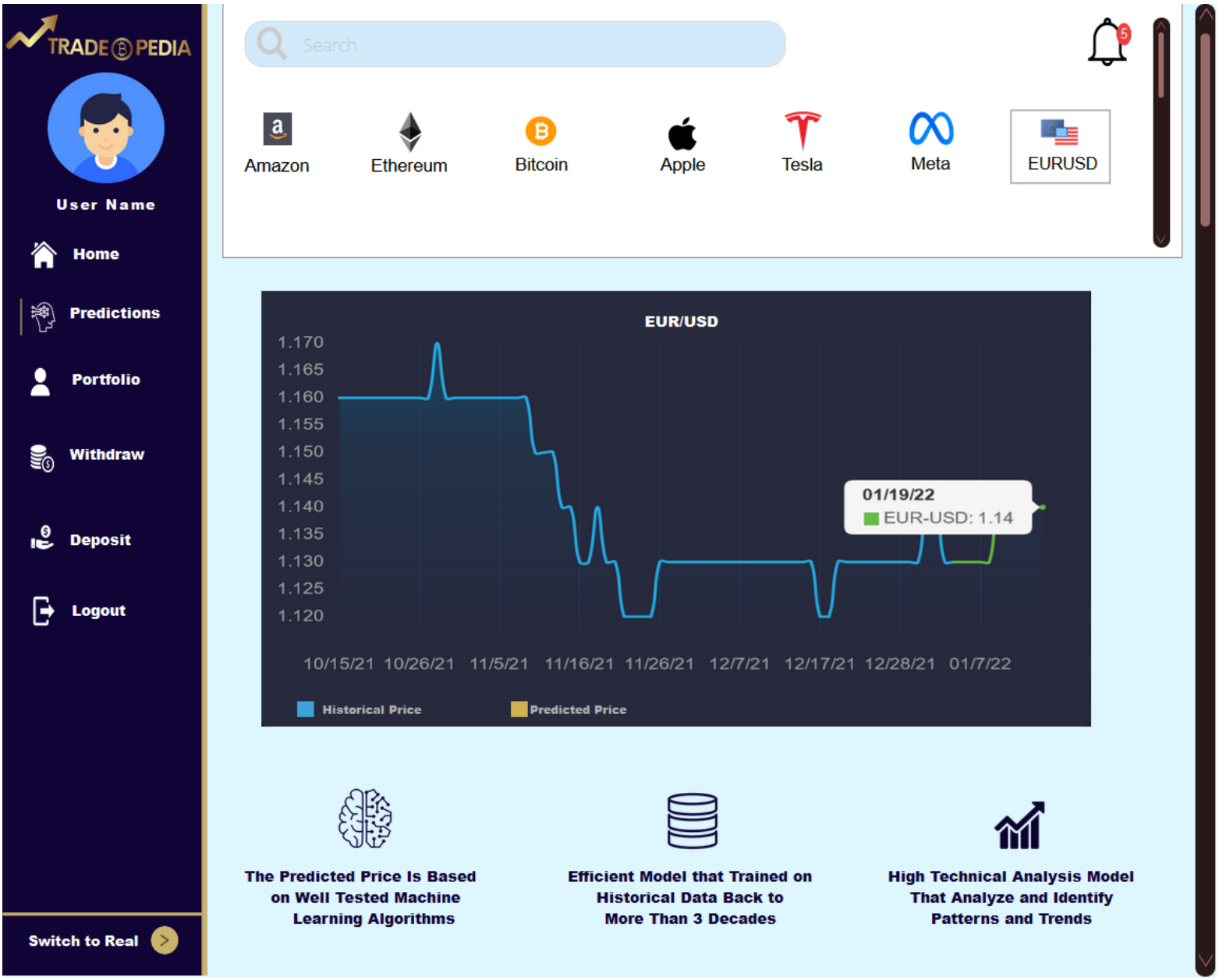
Trade Page will be opened the moment user clicks on Trade.

Figure 17 Trade Page



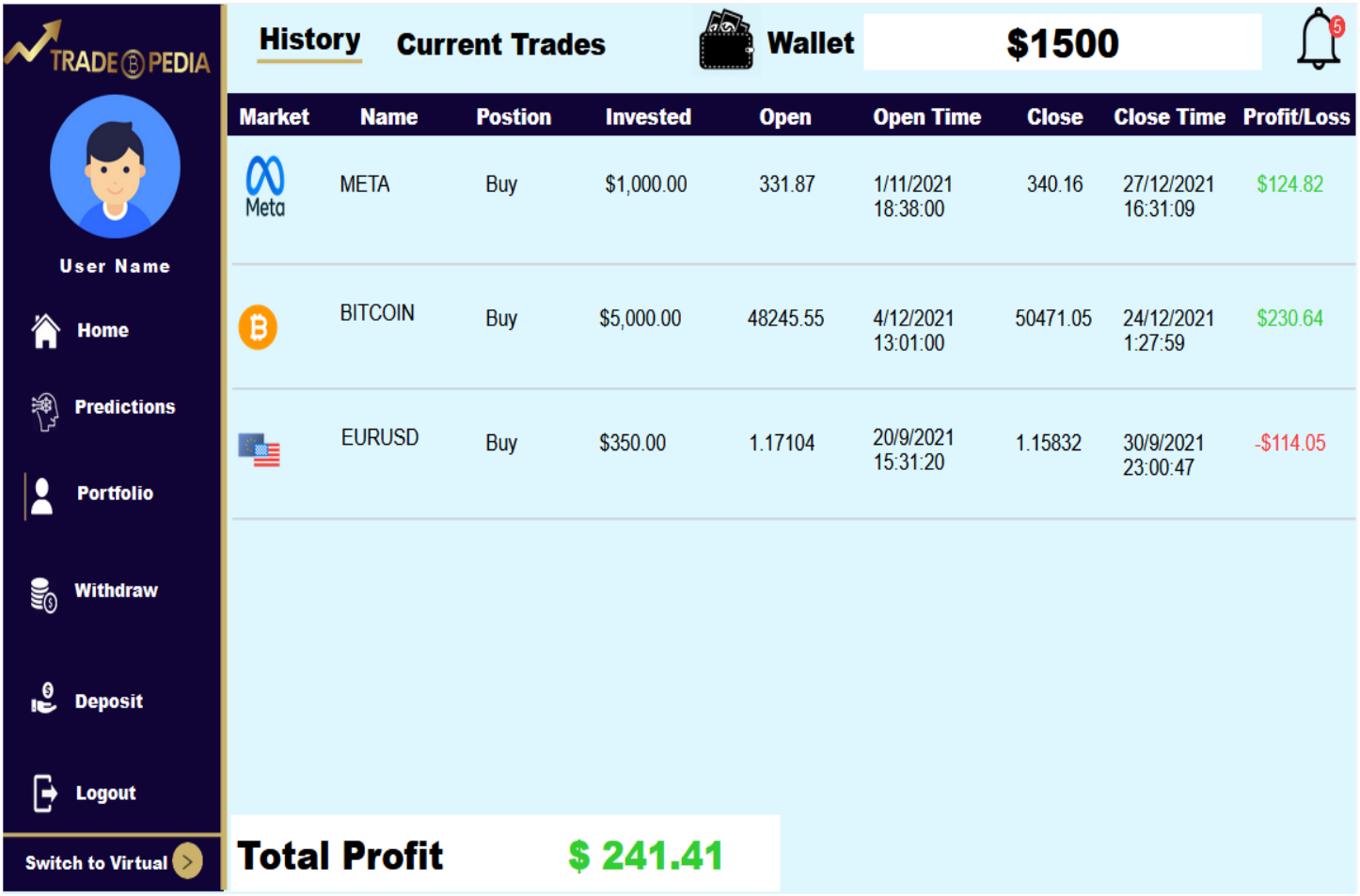
This page will open when the user clicks on the predictions button.

Figure 18 Predictions Page



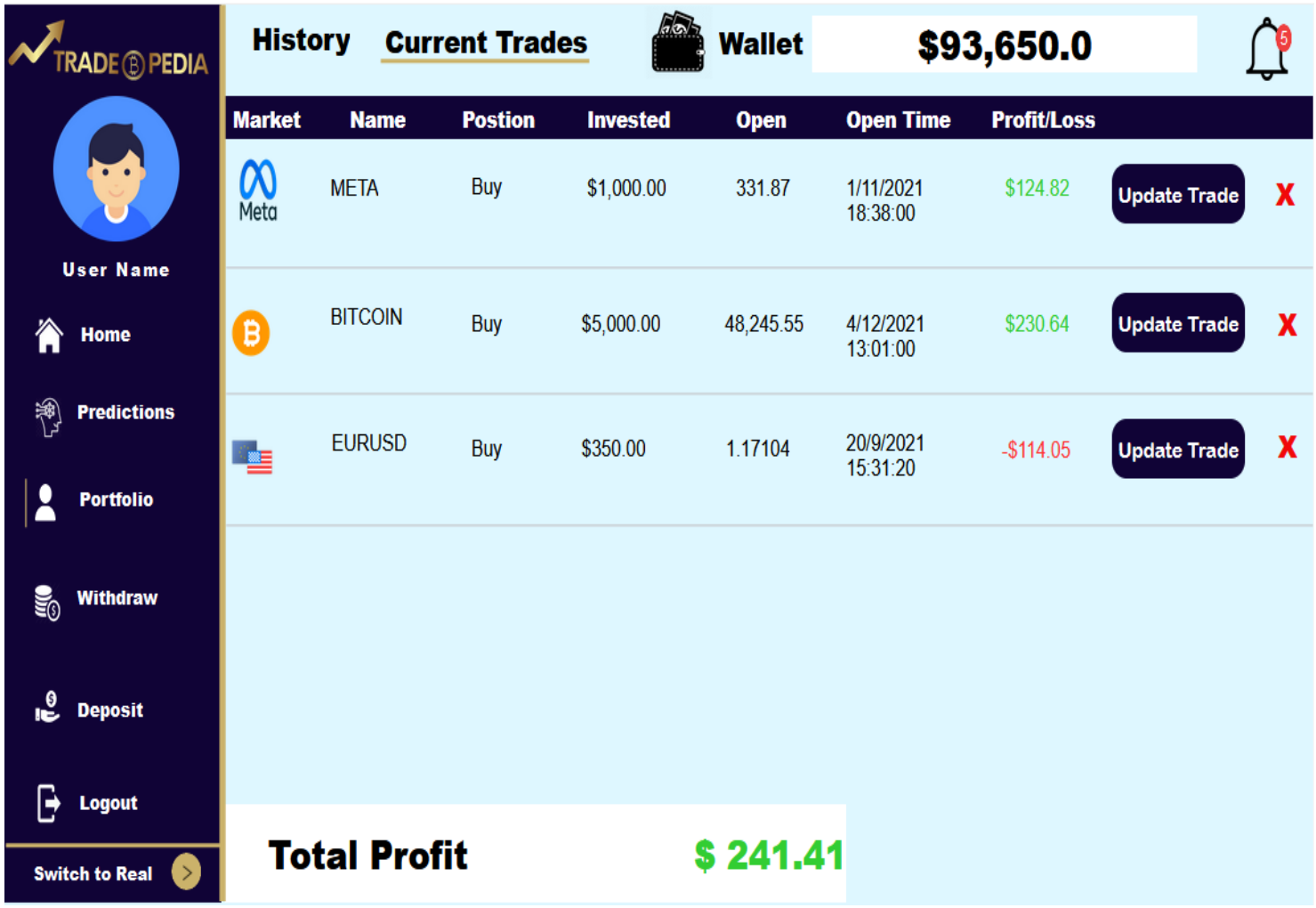
The portfolio page showing the historical trades will show up when clicking portfolio.

Figure 19 History Portfolio Page



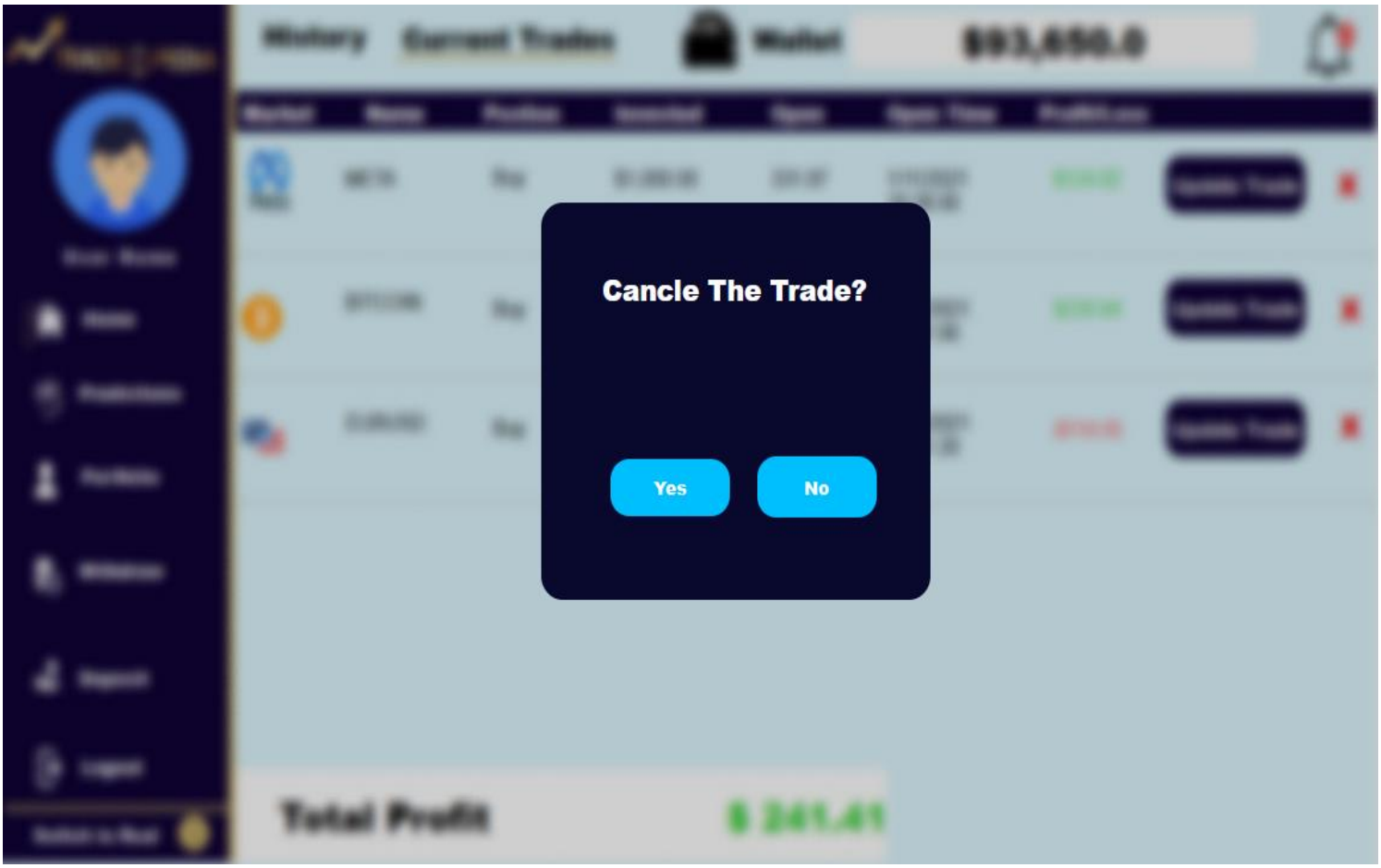
But if the user selected the current trades in the portfolio page, this page will be shown.

Figure 20 Current Portfolio Page



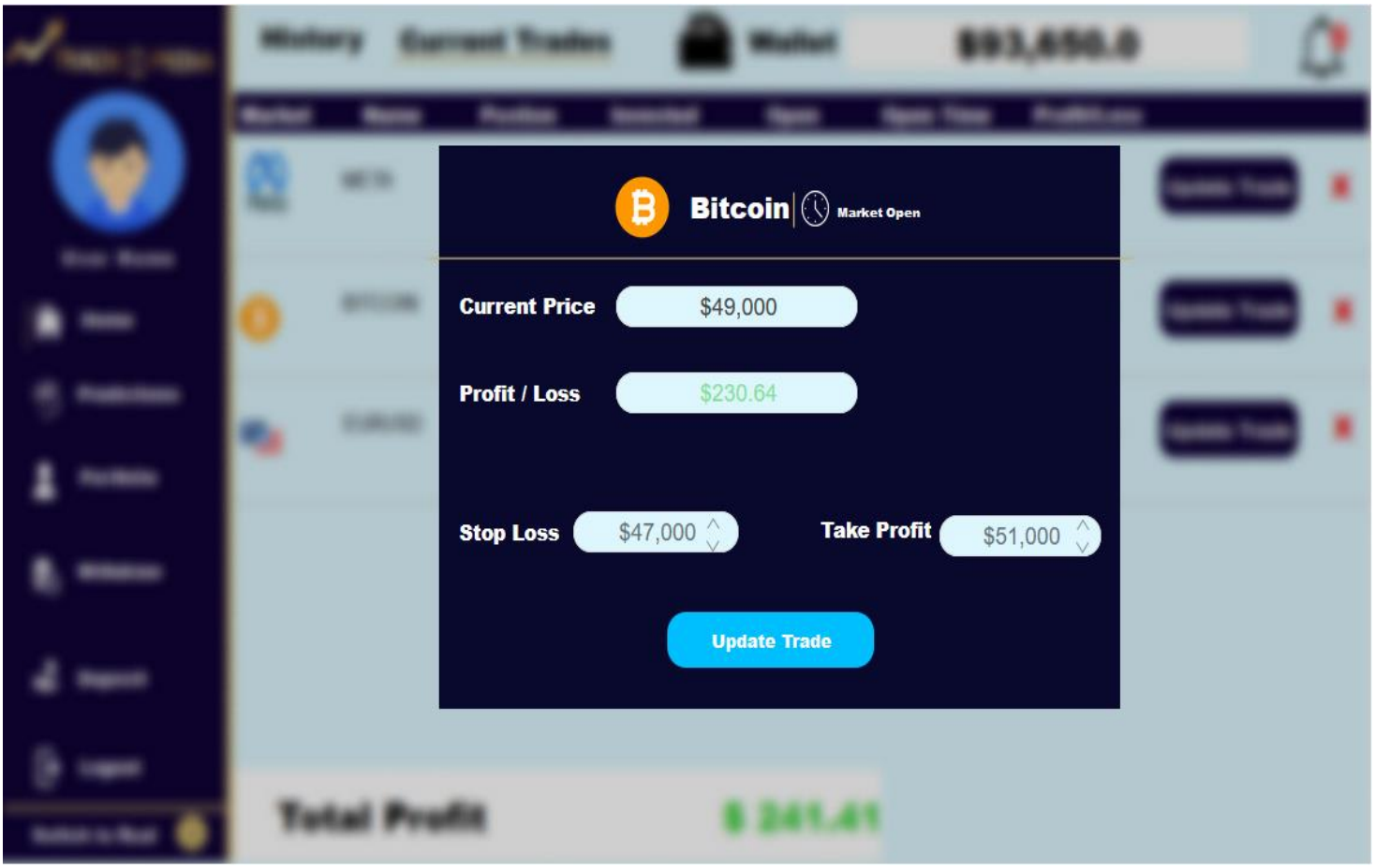
If the user wishes to cancel the trade a confirmation pop up will be displayed like the one below.

Figure 21 Cancel trade confirmation Page



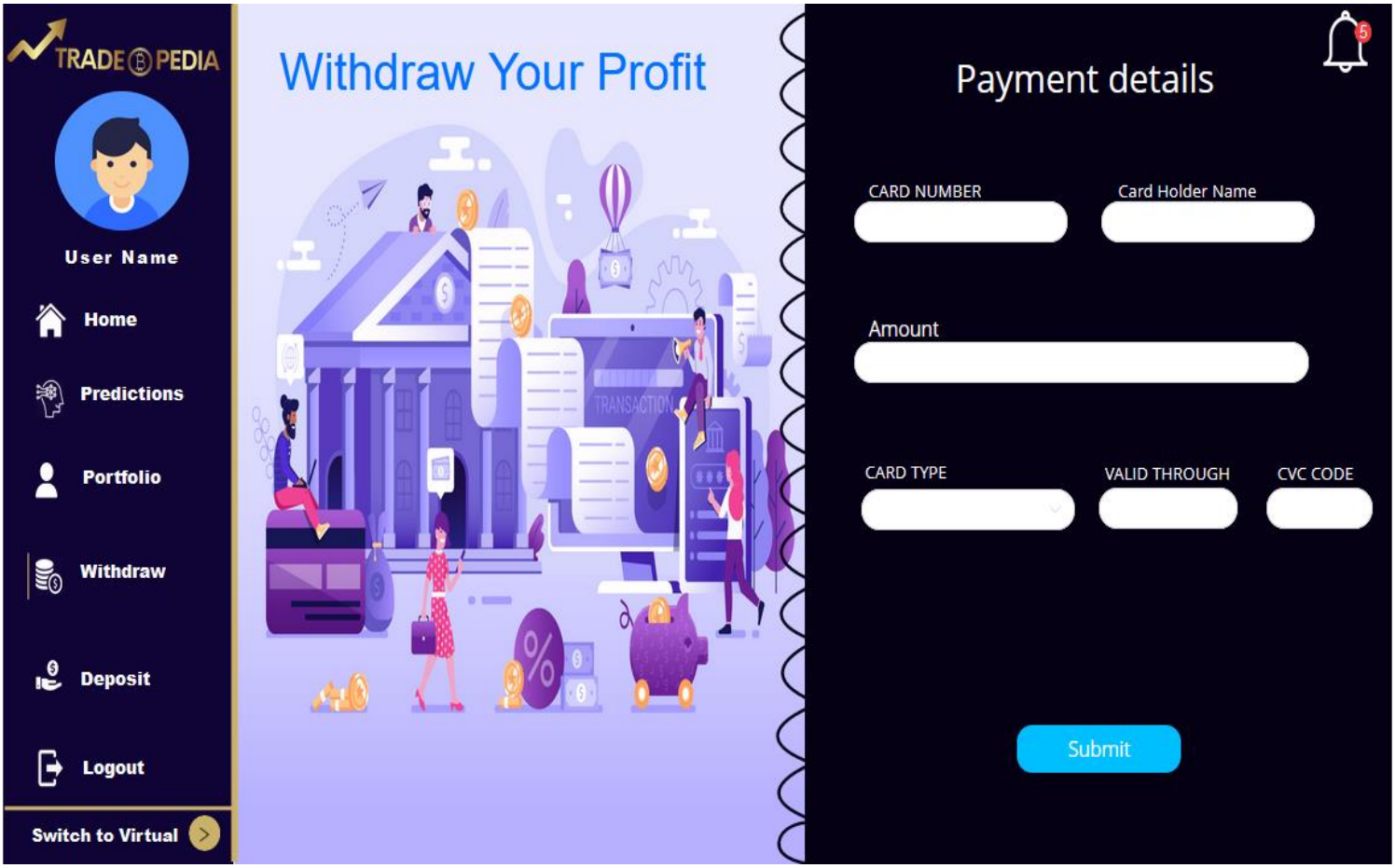
When updating the trade, this screen will be displayed.

Figure 22 Update trade Page



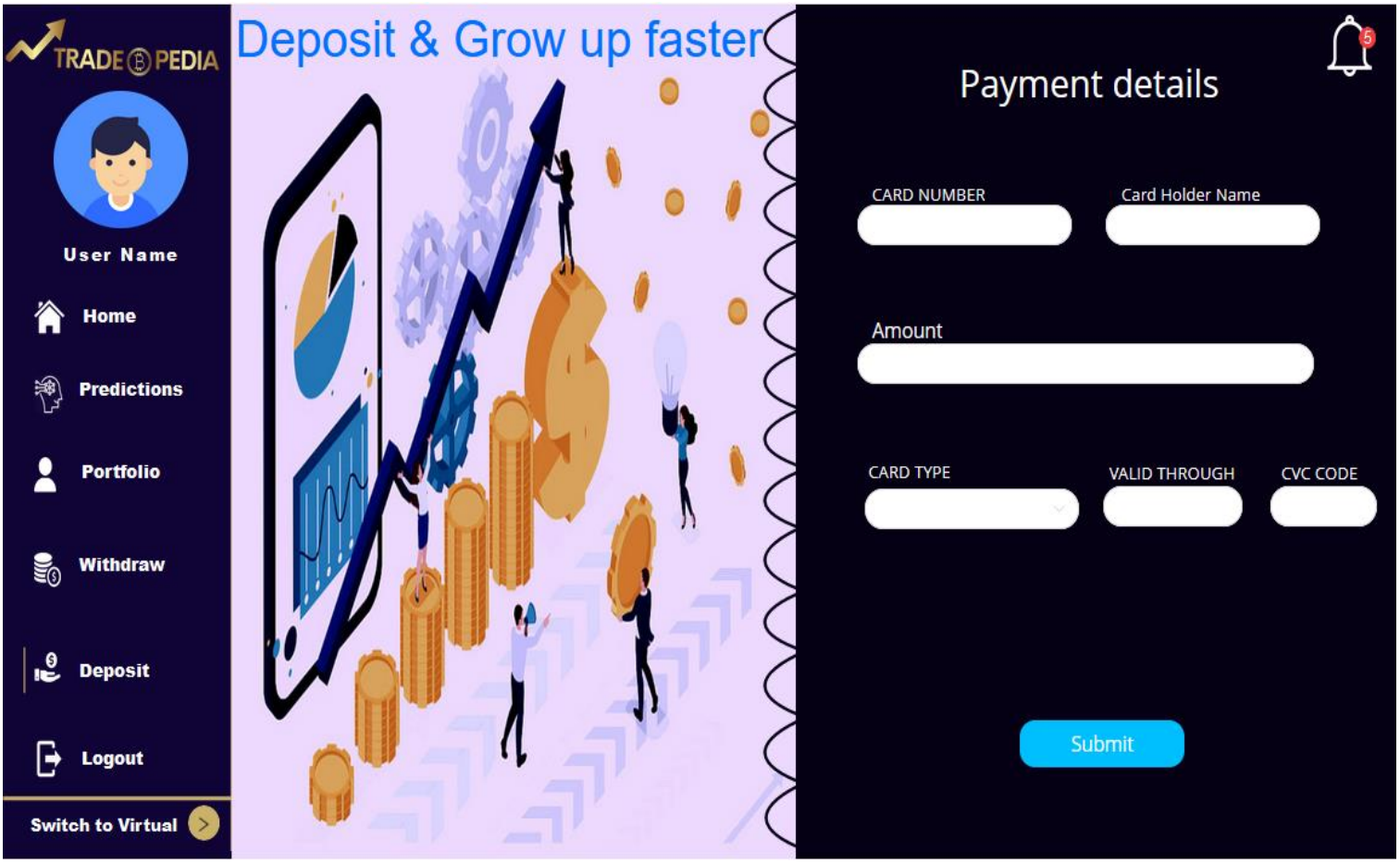
User will be directed to the Withdraw page if he/she clicks on the deposit section in home page.

Figure 23 Withdraw Page



User will be directed to the deposit page if he/she clicks on the deposit section in home page.

Figure 24 Deposit Page



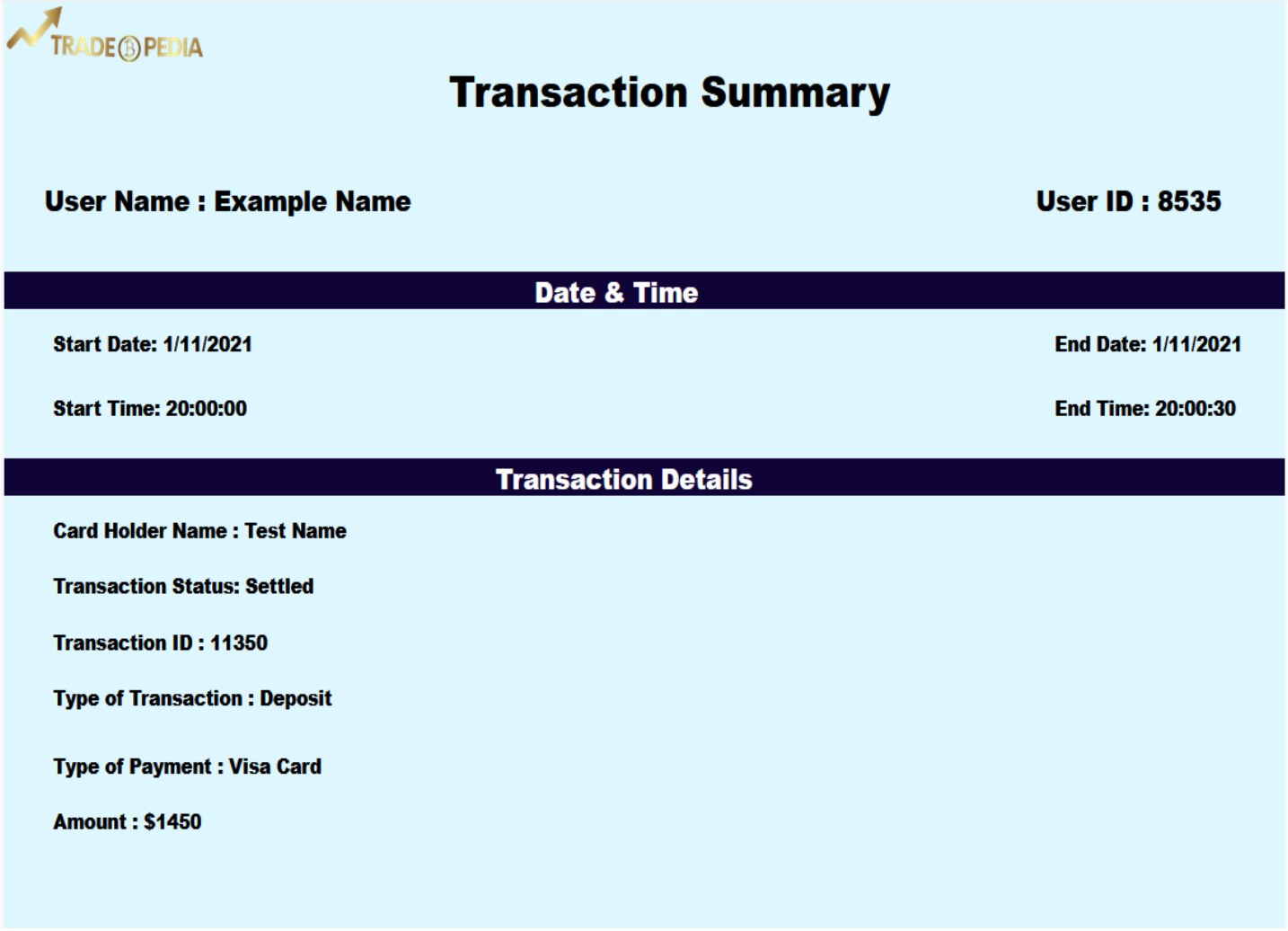
4.2.2 Reports Design

4.2.2.1 Trade Report

Figure 25 trade Report

4.2.2.2 Transaction Report

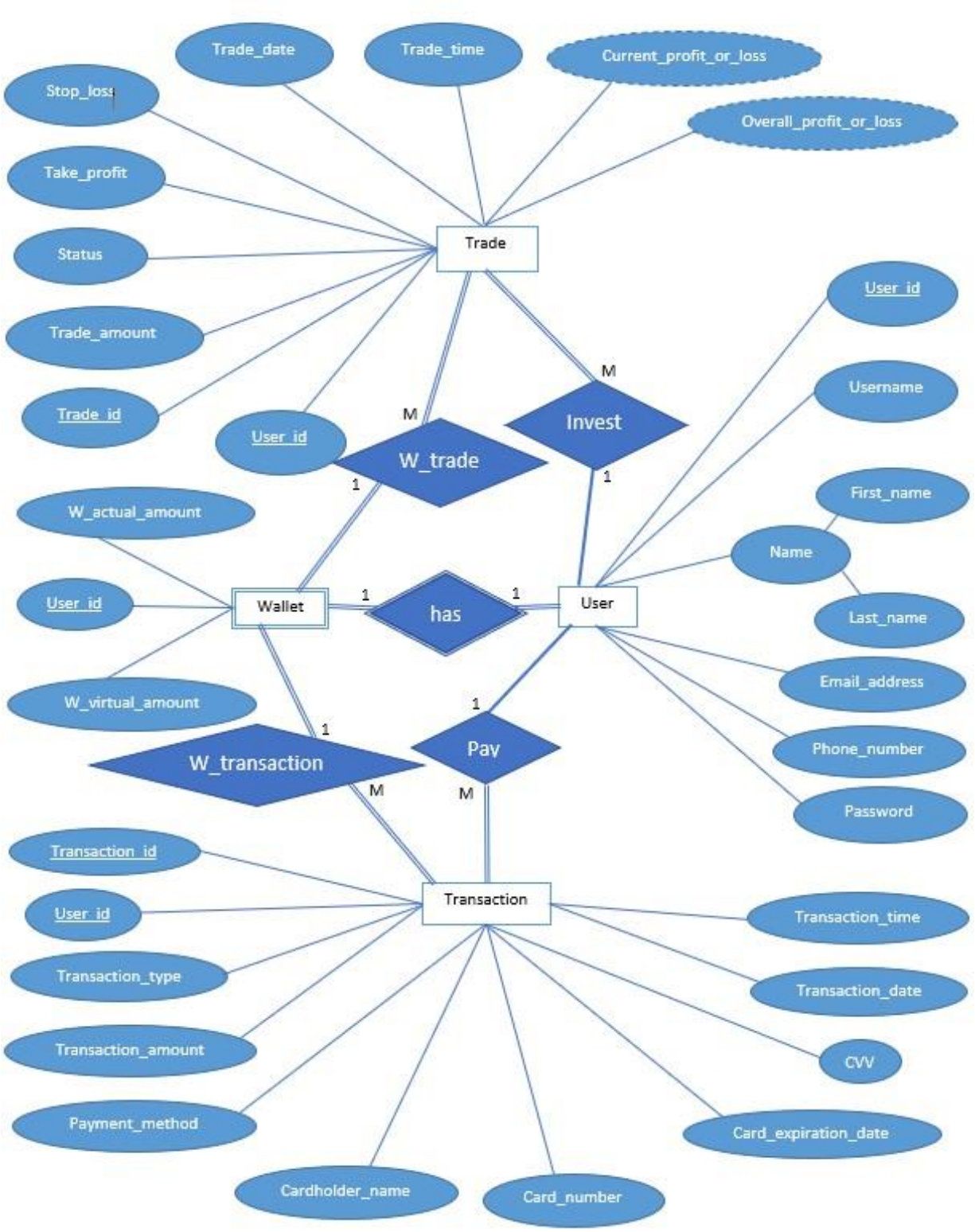
Figure 26 Transaction summery



4.2.2 Database design

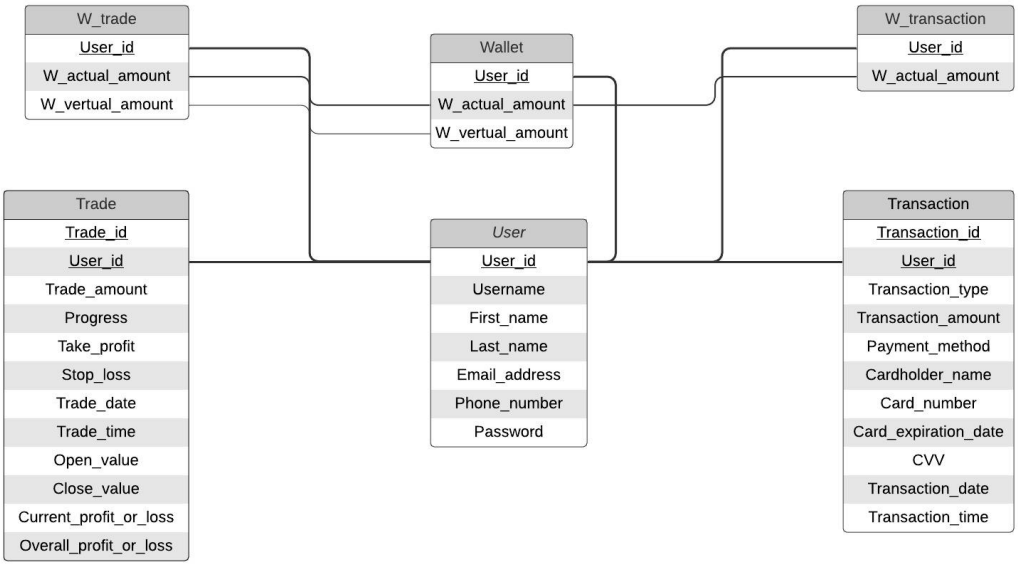
E-R Diagram:

Figure 27 E-R diagram



Database schema:

Figure 28 Database schema



Chapter 5
Implementation

5.1 Overview

As our project directed to investors and trades, a website was a better choice rather than a mobile application. As a user, handling trades, money, and charts especially with live data is fairly difficult to do using a mobile application. For that matter, a website was built for better user experience and less strain caused on the eyes of the user trying to read numbers of a huge chart from a mobile phone's screen.

Furthermore, the predictive feature of the website needed machine learning models to be implemented. For that matter, python was the first and last option that was available due to its support to varies modules (libraries) that aided the development process. Consequently, a python web framework had to be used as other frameworks cannot read python files (the trained models).

5.1.1 Operating system

Some modules were not compatible with Windows OS and so Ubuntu Linux OS had to be used. VMware Workstation Pro was used as the virtual machine that ran Ubuntu, as well as a virtual environment that the website was implemented on for better security and safer option compared to running the website on the main OS environment.

5.1.2 Front-end

As for the front-end, the mostly used languages were used as they did the job pretty good. And so, HTML, CSS, and JavaScript were used to create the design of the website.

5.1.3 Back-end

Large number of python frameworks were found, and so a small study was carried out by the team members using the internet and a number of experts in the web development field. The study concluded in choosing Django framework the one to use, due to its ease of use, properly structured documentation, and easily available sources to benefit from.

5.2 Tools

5.2.1 Software

Visual studio Code was the most used software in the development due to its compatibility with multiple programming languages at once as well as the ability to use different interpreters in the same time without downloading an entire software. As a result, Jupyter interpreter was added as an extension to the VS code to work with machine learning part. The rest of the python implementation was on the default interpreter in VS code. However, one of the team members used PyCharm to carry our ML testing as it took less resources from the computer and helped in speeding up the learning execution time.

5.2.2 Libraries

The table below represents a detailed description of the technologies and modules used in the project.

Table 12 Libraries used

	Module name	Module description
ML modules	pandas	Includes built-in functions that work well with datasets; displaying, cleaning, manipulating, and analyzing them.
	numpy	Popular for complex mathematical operations and deals with arrays much more efficiently that regular arrays.
	sklearn	Well-known library that has various ML algorithms implemented inside it.
	yfinance	API from Yahoo Finance website that allows python to download market data from the same website.
	matplotlib	For drawing plots in python
	seaborn	Another plotting library in python that is already based on matplotlib but much more efficient in its functions.
	statsmodels	For statistical calculations and test.
	plotly	Plotting library that produces interactive visualizations.
	keras	It provides functionality and interface for artificial neural networks.
Back-end modules	channels	A Django library that allows the framework to handle web sockets and handles connections in either synchronous or asynchronous methods.
	django	The main Django library the provides all necessary functionalities for the framework.

	curses.ascii	Provides services to deal with ASCII characters.
	multiprocessing	Provides the support of multiprocessing and concurrency in doing certain tasks.
	celery	It's a task queue that supports asynchronous jobs. Extremely beneficial for real time operations.
	requests	It simplifies HTTP requests implementations
	asgiref	A standard that allows async functions to work with sync functions.
	bs4	Used for web scrapping
	aiohttp	
	asyncio	Used to write concurrent functions when dealing with task queues.
	csv	Used to read and write csv files.
	json	Needed to handle JSON files.
	logging	Used to deal with event logging system.
	threading	Used to carry out threading functionality in python.
	webbrowser	Used to deal with browsers
	base.routing	Allows routing between webpages in the framework.
	pathlib	Used to access files using their local system paths.
	os	Offers functionality related to the OS

5.3 Development

Since the project is a trading website, it was necessary to put a live chart representing the fluctuations of the stock. For that matter, we needed a way to keep our server to check for the latest price a specific stock has reached and plot it. And as we are using python and not Node-JS, it was extremely challenging.

In Node-JS, the server is working constantly with constant requests without the need for an external event to trigger it. Conversely, Django waits for the user to refresh the page in order to make a new request and return the new price. And obviously this is not a practical way to deal with the financial market so we had to figure out a solution.

After months of searching, we were faced with one way but could not figure out whether it is the most practical one. Task scheduling was the answer, it worked by setting a time interval of 10 seconds to do a repeated task. The task was of course to request from the API the latest prices. Although, the API doesn't return the latest price; it returns historical prices with the newest being late by a whole day. For that matter we used web scrapping to obtain the most recent price.

Scheduling the task of returning the price from the website used to take 5 seconds for each stock! And this was a disaster for the loading time of the website. Luckily, this was solved using a separate server with separate port. This port contains a queue of tasks in which web scrapping the yahoo finance to obtain a price for many industries all at once. Basically, requesting the task asynchronously. This made the load time for obtaining the whole data for all the requested industries 5 seconds!

5.4 Code Snippets

In order to do all the project and most of the mentioned requirements in the previous chapters, almost 4892 lines of code had to be written. And some of the most important functionalities are shown below.

5.4.1 Website implementation

Below is the code for the explained solution of task scheduling.

```
import os
from celery import Celery
from django.conf import settings
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'realtime.settings')

app = Celery('realtime')

app.config_from_object('django.conf:settings', namespace='CELERY')

app.autodiscover_tasks(settings.INSTALLED_APPS)

app.conf.beat_schedule = {
    'get_joke_3s': {
        'task': 'base.tasks.get_joke',
        'schedule': 10.0
    },

    'Profit/Loss': {
        'task': 'base.tasks.checkPrice',
        'schedule': 10.0
    }
}


@app.task(bind=True)
def debug_task(self):
    print(f'Request: {self.request!r}')
```

Figure 29 Task scheduling code

Trading system could not be done without separate profiles for different users to use. Consequently, below is the code for implementing the log-in page in Django.

```
def loginPage(request):
    if request.user.is_authenticated:
        return redirect("home")
    else:
        if request.method == "POST":
            username = request.POST.get("username")
            password = request.POST.get("password")

            try:
                user = User.objects.get(username=username)
            except:
                messages.error(request, "Invalid Username or password")


            user = authenticate(request, username=username, password=password)
            if user is not None:
                login(request, user)
                return redirect("home")
            else:
                messages.error(request, "Invalid Username or password")

        return render(request,"base/login.html")
```

Figure 30 Login page code

If there is a new user to the website, the user can create an account. Below is the implementation for the sign-up page.

```
def signupform(reque is_authenticated: Any
    if request.user.is_authenticated:
        return redirect("home")
    else:
        form = CreateUserForm()
        userWallet = Walletform()
        if request.method == "POST":
            form = CreateUserForm(request.POST)
            if form.is_valid():
                form.save()
                user = form.save()

                profile = userWallet.save(commit=False)
                profile.user = user
                profile.save()
                xuser = form.cleaned_data.get("username")
                messages.success(request, "Account was created for " + xuser)
                return redirect("login")

        context = {"form": form}
        return render(request, "base/signup.html", context)
```

Figure 31 Sign-up page code

After signing the user will be directed to the home page. Below is the snippet for its code.

```
@login_required(login_url="login")
def home(request):
    info1 = Notification.objects.all()
    info1 = info1.filter(Nuser = request.user)
    y1 = []
    y2 = []
    y3 = []
    y4 = []
    user = request.user
    industry = Industry.objects.all()

    for xx in industry:

        if str(xx.typet) == "cryptocurrencies":
            y1.append(xx)

        if str(xx.typet) == "stocks":
            y3.append(xx)
        if str(xx.typet) == "commodities":
            y4.append(xx)


    context = {"user": user, "cryptocurrencies": y1, "stocks":y3, "commodities":y4, "info1":info1}


    if request.method == 'POST':
        if 'q' in request.POST:
            y1 = []
            y3 = []
            y4 = []
            q = request.POST['q']
            print(q)
            industry = Industry.objects.filter(industry_name__icontains= str(q))
            user = request.user
            for xx in industry:
```

Figure 32 Home page code part 1

```
            if str(xx.typet) == "cryptocurrencies":
                y1.append(xx)

            if str(xx.typet) == "stocks":
                y3.append(xx)
            if str(xx.typet) == "commodities":
                y4.append(xx)


        context = {"user": user, "cryptocurrencies": y1, "stocks":y3, "commodities":y4}
        return render(request,"base/home.html", context)

    if request.method == 'POST':

        # CHECK wlist duplicated!
        wlist = Industry.objects.all()
        for qq in wlist:
            xx = request.POST.get(str(qq))
            if xx != None and "remove" in request.POST:
                wlist = Wishlist.objects.get(id_W = request.user)
                wlist.industry_id.remove(Industry.objects.get(industry_name = str(xx)))
                wlist.save()
            elif xx != None and "add" in request.POST:
                wlist = Wishlist.objects.get(id_W = request.user)
                wlist.industry_id.add(Industry.objects.get(industry_name = str(xx)))
                wlist.save()

    return render(request,"base/home.html", context)
```

Figure 33 Home page part 2

In the side menu, there is a link to enter the watch list page. This page contains all the industries the user wishes to add in order to track their prices individually and to receive notifications about their status every now and then. So, below is the code for the watchlist page.

```
@login_required(login_url="login")
def watchlist(request):

    wl = []
    wlist = Wishlist.objects.get(id_W = request.user)
    xy = wlist.industry_id.all()
    for qq  in xy:
        xx = request.POST.get(str(qq))
        if xx != None:
                wlist = Wishlist.objects.get(id_W = request.user)
                wlist.industry_id.remove(Industry.objects.get(industry_name = str(xx)))
                wlist.save()
                return redirect("watchlist")
        wl.append(qq)
    context = {"wishlist":wl}
    return render(request, "base/watchlist.html", context)
```

Figure 34 Watchlist page code

Another link in the side menu is to open a portfolio page. This page shows all the trades the user has made either still open or closed trades. The table inside the page indicates the industry traded in, the amount invested, the date, an option to change the take profit or stop loss, and a button to stop the trade. The code below is the implementation for this page.

```
@login_required(login_url="login")
def Portfolio(request):
    current_trades = Trade.objects.all()
    uwallet = Wallet.objects.get(user = request.user)
    hh = []
    tamount = 0.0
    for x in current_trades:
        xx = str(x).split("-")
        in1 = Industry.objects.get(industry_name = xx[1])
        curr1 = current_trades.filter(Tuser = request.user , indus_name = in1, id = x.id)
        for h in curr1:
            hh.append(h)
            if request.method == 'POST':
                tamount = h.trade_amount
                price1 = Industry.objects.get(industry_name = h.indus_name)
                if str(h.indus_name) in request.POST:
                    history = HistoryTrades(Tuser = h.Tuser, indus_name = h.indus_name, close_time = timezone.now(), open_value = h.open
                    history.save()
                    uwallet.amount = float(uwallet.amount) + float(h.trade_amount) + float(h.current_profit_loss)
                    uwallet.save()
                    h.delete()
                    return redirect("Portfolio")
                if "edit" in request.POST:
                    tp = request.POST.get("take_profit")
                    sl = request.POST.get("stop_loss")
                    if(float(sl) <= tamount/2):
                        h.take_profit = float(tp)
                        h.stop_loss = float(sl) * float(-1)
                        h.save()
                    else:
                        messages.error(request, "stop loss is greater than Half of the ammount!!")

                    return redirect("Portfolio")
```

Figure 35 Portfolio page code part 1

```
user = request.user
wallet = Wallet.objects.get(user=user.id)
context = {"user": user, "wallet": wallet, "data": hh}
return render(request, "base/portfolio.html", context)
```

Figure 36 Portfolio page code part 2

When the user choses an industry and clicks on view chart, a live chart of the stock price will appear. Below the chart there is a form that the user fills if a trade wished to be opened. Below is the code for trade function.

```
@login_required(login_url="login")
def trade(request, name):


    Date_req = date.today() + timedelta(days=1)
    pathRequest = str(request.path)
    pathRequest = pathRequest.replace("/trade/", "")
    pathRequest = pathRequest.replace("/", "")

    index = ["BTC-USD", "ETH-USD" , "TSLA", "FB", "NFLX", "NVDA","AMZN","CL=F", "MGC=F", "NG=F"]
    name = ["BITCOIN", "ETHEREUM", "TESLA", "FACEBOOK", "NETFLIX", "NVIDIA", "AMAZON", "OIL","GOLD","NATGAS"]
    if pathRequest not in name:
        return redirect("home")

    indexx = name.index(pathRequest)

    upt_data = yf.download(index[indexx], start="2022-04-27", end=Date_req)
    sf = upt_data["Close"]
    fd = pd.DataFrame({"Date": sf.index, "Values": sf.values})

    pathh = "Historical_Prices/" + str(pathRequest) + ".csv"
    file = open(pathh)
    csvreader = csv.reader(file)
    rows = []
    cpt = 0
    for row in csvreader:
        if row[0] != "Date":
            element1 = datetime.datetime.strptime(row[0], "%Y-%m-%d")
            timestamp1 = datetime.datetime.timestamp(element1)
            rows.append(str(int(timestamp1)) + ";")
            rows.append(str(row[1]) + ";")
            rows.append(str(row[2]) + ";")
            rows.append(str(row[3]) + ";")
            rows.append(str(row[4]) + ";")
            cpt += 5
```

Figure 37 Trade code part 1

```
    file.close()

    updated_prices = []
    for i in range(0, len(fd["Date"])):
        yy = str(fd["Date"][i]).removesuffix(" 00:00:00")
        element = datetime.datetime.strptime(yy, "%Y-%m-%d")
        timestamp = datetime.datetime.timestamp(element)
        updated_prices.append(str(int(timestamp)) + ";")
        updated_prices.append(str(upt_data["Open"][i]) + ";")
        updated_prices.append(str(upt_data["High"][i]) + ";")
        updated_prices.append(str(upt_data["Low"][i]) + ";")
        updated_prices.append(str(upt_data["Close"][i]) + ";")

    arr = rows + updated_prices
    context = {"data": arr, "name": pathRequest}

    if request.method == 'POST':

        form = TradeForm(request.POST, request.FILES)
        xind = Industry.objects.get(industry_name=pathRequest)
        uwallet = Wallet.objects.get(user = request.user)
        currentPrice = xind.price

        if float(uwallet.amount) - float(request.POST.get("trade_amount")) >= 0.0 and float(request.POST.get("trade_amount")) > 0 and

            uwallet.amount = float(uwallet.amount) - float(request.POST.get("trade_amount"))
            uwallet.save()
            if form.is_valid():
                instance = form.save(commit=False)
                instance.trade_amount = float(request.POST.get("trade_amount"))
                instance.Tuser = request.user
                instance.indus_name = xind
                instance.open_value = float(currentPrice)
                instance.units = float(request.POST.get("trade_amount")) / float(currentPrice)
                instance.TypeTrade = request.POST.get("TradeType")
                instance.status = "active"
```

Figure 38 Trade code part 2

```
                    instance.stop_loss = (float(request.POST.get("trade_amount")) / float(2) ) * float(-1)
                else:
                    instance.stop_loss = float(request.POST.get("stop_loss")) * float(-1)

                if int(request.POST.get("take_profit")) == int(0):
                    instance.take_profit = (float(request.POST.get("trade_amount")) / float(2) )
                else:
                    instance.take_profit = float(request.POST.get("take_profit"))


                instance.current_profit_loss = float(0.0)
                instance.save()
                messages.error(request, str(pathRequest) + " Trade is Placed Successfully!")
                info = Notification(Nuser = request.user, message = str(pathRequest) + " Trade Placed Successfully!", messagetype = "n
                info.save()

                context = {"data": arr, "name": pathRequest}
                return render(request, "base/tradeform.html", context)

            else:

                messages.error(request, "form is not valid!")
                return render(request, "base/tradeform.html", context)

        else:
            if float(uwallet.amount) - float(request.POST.get("trade_amount")) < 0.0:
                messages.error(request, "The trade amount exceedes the money you have in the wallet!")

            elif float(request.POST.get("trade_amount")) < 100:
                messages.error(request, "Trade Amount Must Be at least 100$!")
            elif float(request.POST.get("stop_loss")) > float(request.POST.get("trade_amount")) / 2:
                messages.error(request, "Maximum Stop Loss is " + str(float(request.POST.get("trade_amount")) / 2))

            return render(request, "base/tradeform.html", context)


    return render(request, "base/tradeform.html", context)
```

Figure 39 Trade code part 3

5.4.2 Machine learning implementation

Two machine learning algorithms were implemented to give the most accurate result of the stock price. K nearest neighbor (KNN) and long short-term memory (LSTM) were used and tested on each stock the website has. Also, the hyper parameters were tuned to see which gives the best accuracies.

Below is the implementation for KNN algorithm.

```
df = df.values
X, y = df[:, :-1], df[:, -1]

X_train, X_test, yC_train, yC_test = train_test_split(X, y, test_size=0.3, random_state=1)

scaler = MinMaxScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```

modelC = KNeighborsRegressor(n_neighbors=5, weights='uniform', algorithm='auto', metric_params=None, n_jobs=-1 )

reg = modelC.fit(X_train, yC_train)

pred_labels_tr = modelC.predict(X_train)

pred_labels_te = modelC.predict(X_test)
```

```

print("")
print('****************** KNN Regression ******************')
print('Effective Metric: ', reg.effective_metric_)
print('Effective Metric Params: ', reg.effective_metric_params_)
print('No. of Samples Fit: ', reg.n_samples_fit_)
print("")
scoreR_te = modelC.score(X_test, yC_test)
print('Test Accuracy Score: ', scoreR_te)
scoreR_tr = modelC.score(X_train, yC_train)
print('Training Accuracy Score: ', scoreR_tr)
```

Figure 40 KNN code

LSTM is considered a deep learning algorithm and it has been tested and gave very promising results. Below is its implementation.

```
x_train, y_train = np.array(x_train), np.array(y_train)

# Reshape the data
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))

x_train.shape

model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape=(x_train.shape[1], 1)))
model.add(Dropout(0.2))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))

model.compile(optimizer='adam', loss='mean_squared_error')

model.fit(x_train, y_train, batch_size=10, epochs=1)


test_data = scaled_data[training_data_len - 60:, :]
x_test = []
y_test = dataset[training_data_len:, :]
for i in range(60, len(test_data)):
    x_test.append(test_data[i - 60:i, 0])

x_test = np.array(x_test)

x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))

# Get the models predicted price values
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
print(predictions)
# Get the root mean squared error (RMSE)
rmse = np.sqrt(np.mean(((predictions - y_test) ** 2)))
print("this is the error " + str(rmse))
```

Figure 41 LSTM code

In addition, a statistical prediction algorithm was tested. Autoregressive integrated moving average (ARIMA) was tested on the prices and also gave good results, and below is its implementation.

```
from statsmodels.tsa.stattools import adfuller
adfuller(log_df)

# check with ADF Test for stationarity

y_diff= log_df.diff().dropna()
y_diff_diff= log_df.diff().diff().dropna()

print('p-value zero-diff: ', adfuller(log_df)[1])
print('p-value first-diff: ', adfuller(y_diff)[1])
print('p-value second-diff: ', adfuller(y_diff_diff)[1])


from pmdarima.arima.utils import ndiffs

d = ndiffs(log_df)
d #1 diff will be enough!

from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf

#figure setup
fig = plt.figure(figsize=(20,10))
ax1 = fig.add_subplot(2,1,1)
ax1.set_title('1st Order Differencing')
ax2 = fig.add_subplot(2,2,3)
ax3 = fig.add_subplot(2,2,4)



q = 1
p = 1

from statsmodels.tsa.arima.model import ARIMA
```

Figure 42 Arima code part 1

```
y= log_df.Close #just to be precise

arima = ARIMA(y, order=(p,d,q))
arima = arima.fit()

import pmdarima as pm
smodel = pm.auto_arima(y,
                       start_p =0, max_p = 5,
                       start_d = 0, max_d = 5,
                       start_q = 0, max_q = 5,
                       seasonal = False,
                       trace = True)


train_size=0.7 #70% of data

index = round(train_size*log_df.shape[0])

y_train = log_df.iloc[:index] #first 70% rows for training set
y_test = log_df.iloc[index:] #last 30% rows for test set
n_train=len(y_train)

best_order = smodel.order # best (p,d,q)
y_train.shape, y_test.shape
y_train

plt.figure(figsize=(14,6))
plt.plot(ARIMA(y_train,order=best_order).fit().get_forecast(10).predicted_mean)
plt.plot(ARIMA(y_train,order=best_order).fit().get_forecast(10).conf_int())


arima_df = ARIMA(y_train,order=best_order).fit().get_forecast(10).conf_int()
arima_df['pred'] = ARIMA(y_train,order=best_order).fit().get_forecast(10).predicted_mean
arima_df
```

Figure 43 ARIMA code part 2

```python
def walk_forward_validation(n_train, test):
    # create dataframe to store the outcome
    result = pd.DataFrame(columns=['forecast', 'lower_interval', 'upper_interval'])
    # predict one point at a time
    for i in range(len(test)):
        # define train set
        train_ = log_df.iloc[:n_train+i, :].copy()
        # train the model
        arima = ARIMA(endog = train_, order=best_order).fit(method_kwargs={"warn_convergence": False})
        # get the forecast
        results = arima.get_forecast(1, alpha=0.05)
        # central
        result.loc[i, 'forecast'] = results.predicted_mean[0]
        # lower interval
        result.loc[i, 'lower_interval'] = results.conf_int().iloc[0, 0]
        # upper interval
        result.loc[i, 'upper_interval'] = results.conf_int().iloc[0, 1]
    # join with test dataframe
    result.index=test.index
    result = result.apply(pd.to_numeric)
    return test.join(result)

result = walk_forward_validation(n_train, y_test)
print("i am here!")

# baseline = tomorrows prediction is todays price
result['base'] = result['Close'].shift()
result['base'].iloc[0] = y_train.Close[-1]
result.apply(lambda x: np.exp(x).astype('float16')).tail(5) #exp for visualization only
```

Figure 44 ARIMA code part 3

5.5 Implemented features

Large efforts were put into creating the project as efficient as possible. And it was focused that all the function requirements to be implemented. Thankfully, all requirements were done except for one which was due to limited time frame. For more details, please refer to the table below.

Table 13 Implementation Status of requirements

System Pages	Requirement ID	Requirement Name	Implementation Status
Registration Page	FR1	Sign up	Implemented
	FR2	Sign In	Implemented
	FR3	Forget Password	Not Implemented
	FR4	Sign out	Implemented
Transaction page (Real Mode)	FR5	Deposit	Not Implemented
	FR6	Withdraw	Not Implemented
	FR7	Trade	Not Implemented
Transaction Page (Virtual Mode)	FR8	Trade	Implemented
Home screen	FR9	The market	Implemented
	FR10	Search	Implemented
	FR11	Notification	Implemented
Portfolio Page	FR12	Portfolio	Implemented
	FR13	Cancel Trade	Implemented
	FR14	Update Trade	Implemented
Prediction Page	FR15	Predictions	Implemented

Chapter 6
Testing

6.1 Testing Approach

Testing a project is essential and requires a lot of effort; for that matter, we carried out the testing throughout the implementation phase. And once we finished the whole website, we were already done with the testing phase.

6.1.1 Functional testing

The functional testing was carried out as follow:

1) Unit testing: there were no functions or features implemented unless it was tested individually. We made sure that each function works before implementing it to the whole system. For example: after making the live chart, we had to make sure that it is in fact live and actually displays correct prices.

2) Integration testing: after making sure each function is working separately, we started adding each function one at a time. The reason is that we weren't sure all the functions will work together and so we wanted every problem to appear individually so we could know from where it came from. For example: after adding the live chart for every industry, we made sure that the chart displays the prices for the specific industry the user is clicking on and not a different one.

3) Regression testing: after adding most of the features and ran the website, we noticed some changes that had to be made. And so, we carried out regression testing for making sure the operation is running smoothly and with minimum delays; for example, when bringing the live data from the API, it used to take 5 seconds for each industry. However, we felt that this is not acceptable; so, we utilized a server containing a queue to obtain the data from the API of all industries all at once.

4) System testing: when all the different modules were put together, a system testing was carried. A general overview of the website was necessary to check everything is running smoothly and efficiently. For example, the redirection of pages, the main theme colors, and a unified side bar that is present in all pages of the website all were tested to work efficiently.

6.1.2 Nonfunctional testing

Other aspects were considered when testing the website other than the functional features. Some of the main nonfunctional testing points are mentioned below:

1) Security: it is our most important aspect we had to consider as the website deals with money and stocks; hence, we emphasized large efforts on implementing all the security measures we could. To name a few:

    a. We made sure that the URL contains the minimum amount of information by hiding and data or requests sent on the channel between the user and the server.

    b. All the forms in the website are protected against cross-site request forgery (CSRF) attacks.

    c. And of course, there is no need to mention that passwords in the data base is encrypted. In fact, they are encrypted using hashing function from Django where it encrypts each password differently even if two passwords are the same, they do not match the same hashed values.

2) Efficiency: the speed of requests when fetching prices from the API was tested and modified more than once, resulting in minimum delays to achieve a very efficient data retrieval.

3) Usability: as our goal is to allow more of the general public to get into the financial market, we ensured the design to match all age groups, gender, and even people with certain disabilities. To achieve this, we followed WCAG 2.0 Level AA standards and implemented most of its requirements.

6.2 Testing Tools

- Visual studio code debugging mode.
- WCAG 2.0 Level AA standards.
- Different operating systems.
- Different physical devices.

6.3 Tested Features

A summary of the tested features that we caried out is in the table below:

Table 14 Tested features

System Pages	Requirement ID	Requirement Name	Functionality tested	Test Results
Registration Page	FR1	Sign up	- Essential and unique username. - Password must be at least 8 characters, containing letters and numbers, and not related to the username.	Success

	FR2	Sign In	- Username must exist in the database. - Password must be owned by the username entered.	Success
	FR3	Forget Password	It was not tested and implemented because multi-factor authentication requires a server which we don't have.	Fail
	FR4	Sign out	Must redirect to login screen.	Success
Transaction page (Real Mode)	FR5	Deposit	It was not tested because it was not implemented in the system.	Fail
	FR6	Withdraw	It was not tested because it was not implemented in the system.	Fail
	FR7	Trade	It was not tested because it was not implemented in the system.	Fail
Transaction Page (Virtual Mode)	FR8	Trade	- Input value must be smaller than wallet. - Stop loss must not be less than invested amount. - Shows notification when trade is placed or stopped. - Unique for each user.	Success
Home screen	FR9	The market	- Show chart of selected industry. - Show the newest stock price in the market. - Show percentage drop or increase of the stock. - Show chart of moving average.	Success
	FR10	Search	Shows results even if user typed industry name not correctly 100%.	Success
	FR11	Notification	Every trade done or closed it shows a notification.	Success
Portfolio Page	FR12	Portfolio	- Shows table of running trade. - Shows table of past closed trades.	Success
	FR13	Cancel Trade	A confirmation button must be clicked.	Success
	FR14	Update Trade	- A form appears when button is clicked. - Input value must be smaller than wallet. - Stop loss must not be less than invested amount.	Success

Prediction Page	FR15	Predictions	Shows a chart showing previous price and near future prices with maximum accuracies.	Success

Chapter 7
Conclusions and Future Work

The financial market is vast and filled with numbers, charts, indicator, and a lot of titles. Our aim is to eliminate the complexity as much as possible so that more people would invest and gain profits. In order to do that, a huge amount of work should be done and a lot of research. And as students who had other subjects to study and the limited time and experience, we have we didn't manage to implement all the ideas we initially hoped to create.

As a result, there is a number of features that if they were properly added, we guarantee that the project would turn into profitable company located in the Silicon Valley and having friends in the wall street.

One of the most important features would be fundamental analysis of news, websites, and even tweets using NLP. This will drastically benefit the prediction process and give much more accurate numbers.

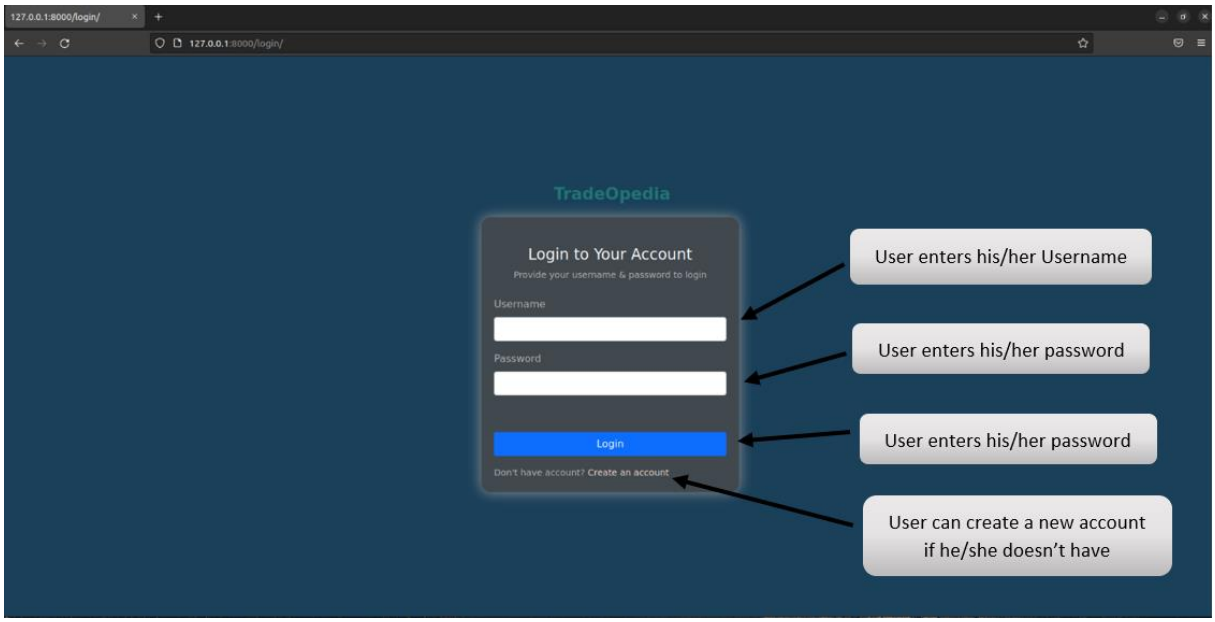
Another very important feature is recommendation messages. After studying the stock very deeply and having high confidence in predicting its direction, the website would send the user a message warning him about the change that would happen after specific time.

Although there exist much less complicated features that can be added, due to the limited time and experience we had we were not been able to. Summarized below is a list of the features we believe they can benefit the user experience greatly:

- Optimized mobile app.
- Send notifications to the user's mobile phone.
- Provide explanation page where the user can read about all the technical terms in the market.

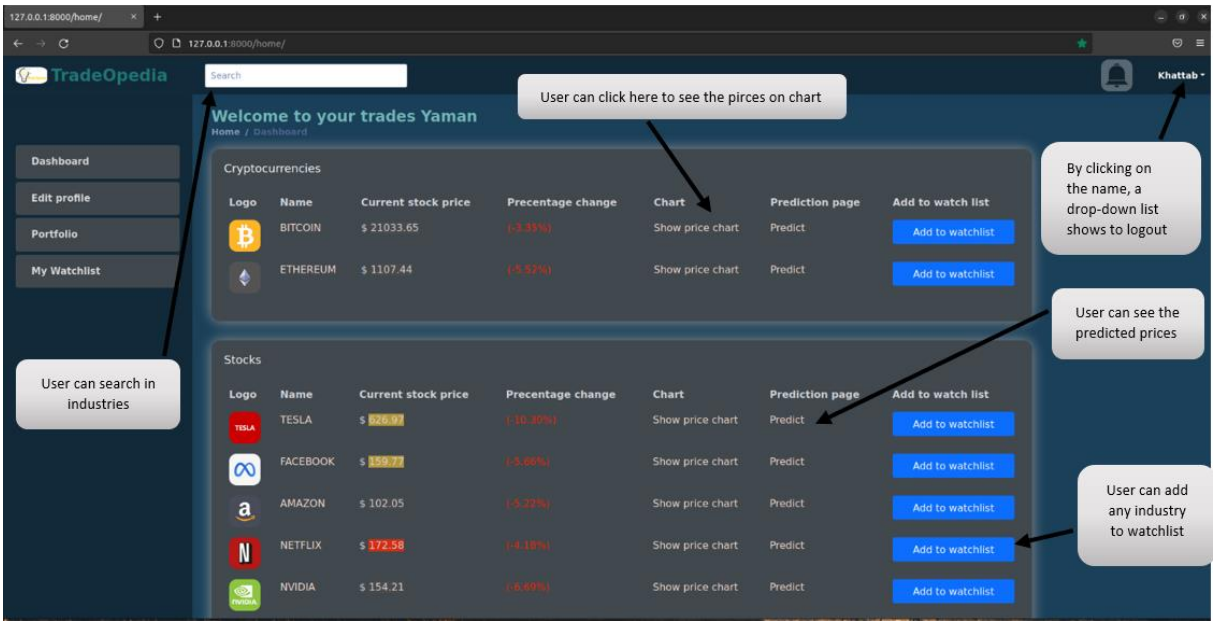
Appendix A
Users' Manual

- Log in page



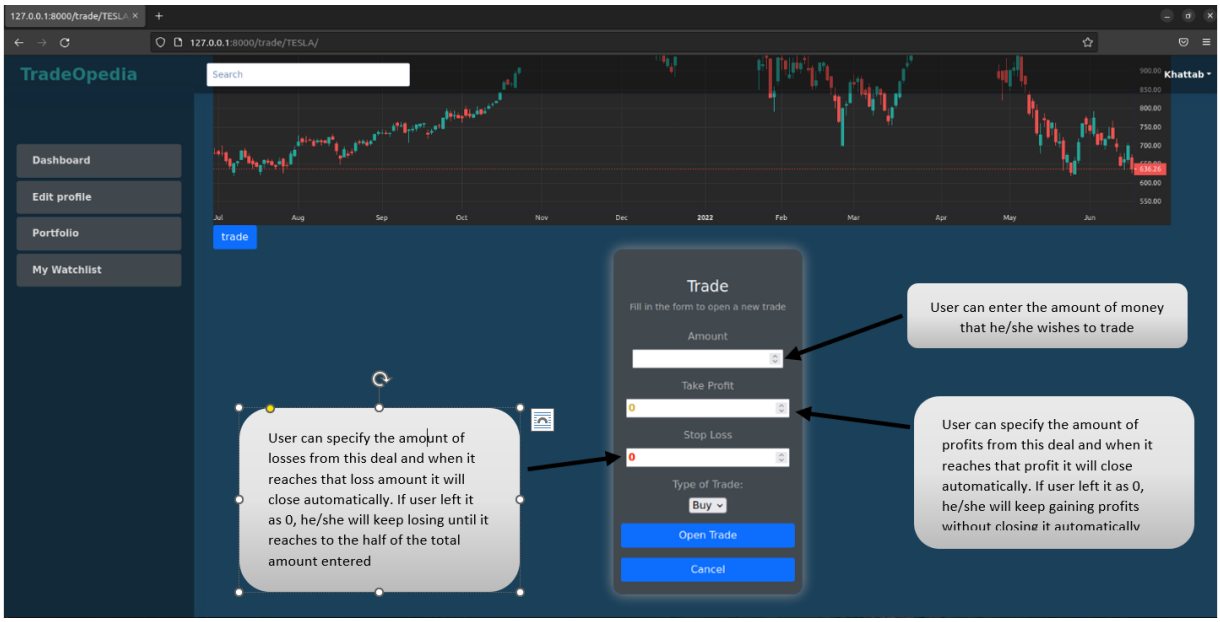
- Sign up page

- Home page

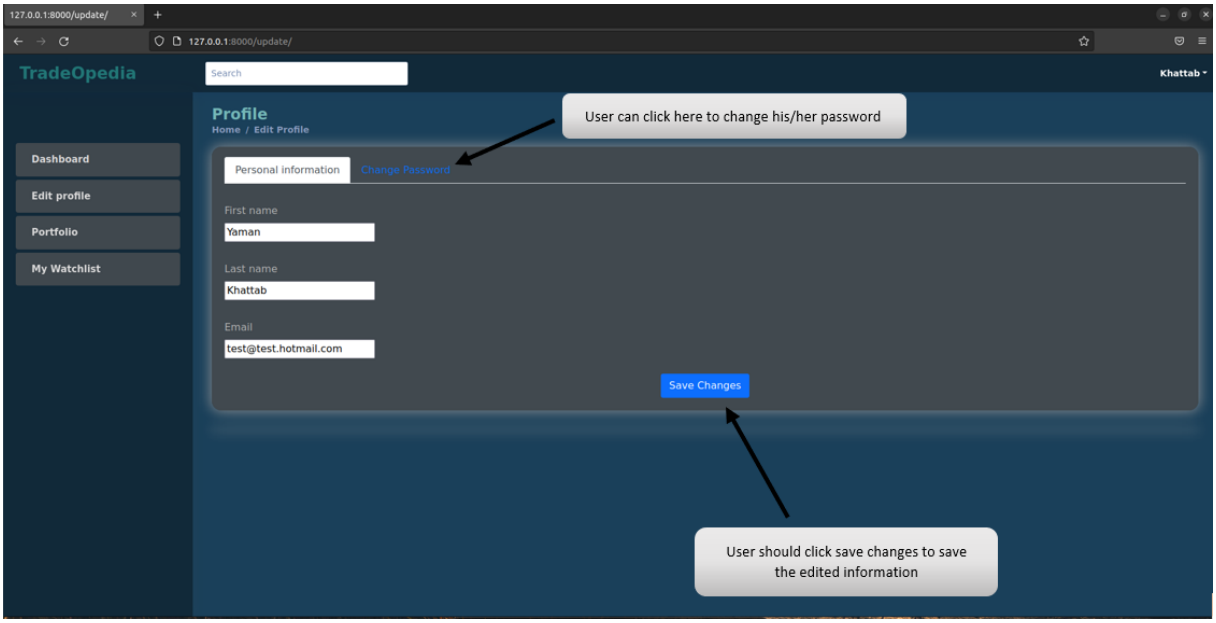


- Show chart Page

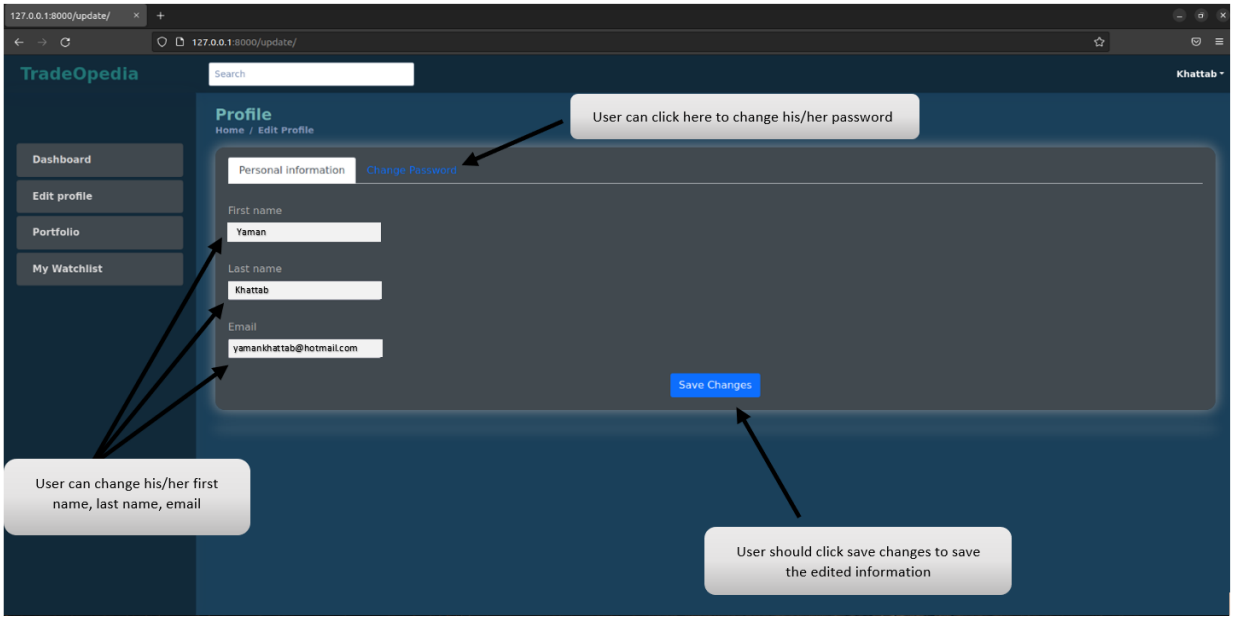
- Trade Page



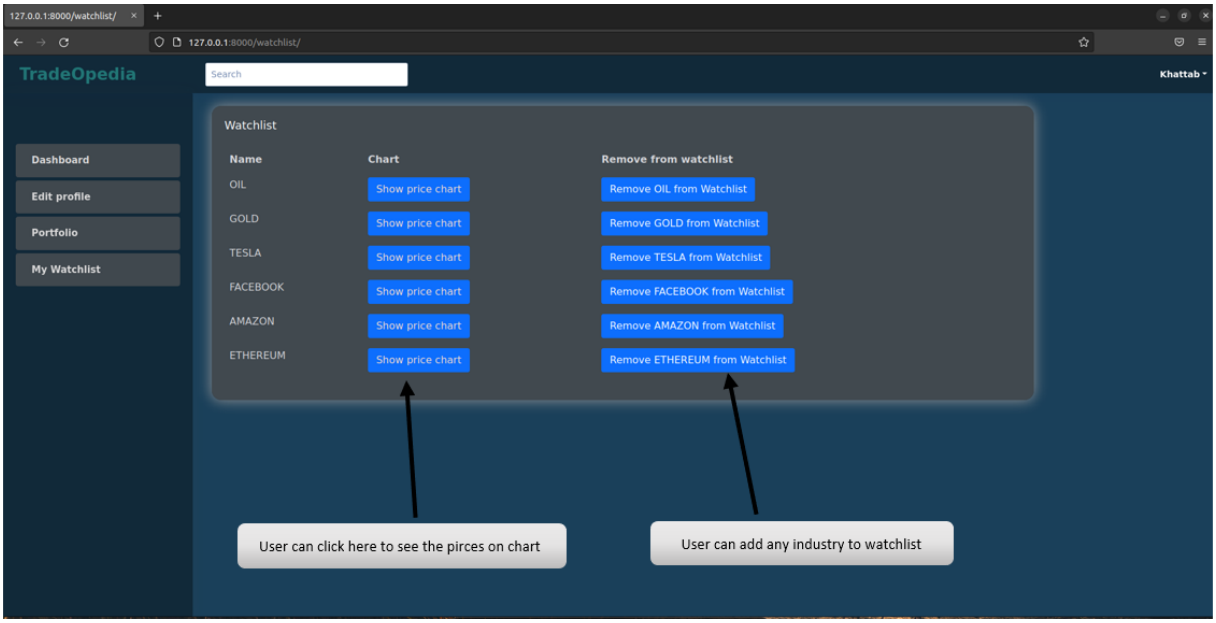
- Edit profile Page



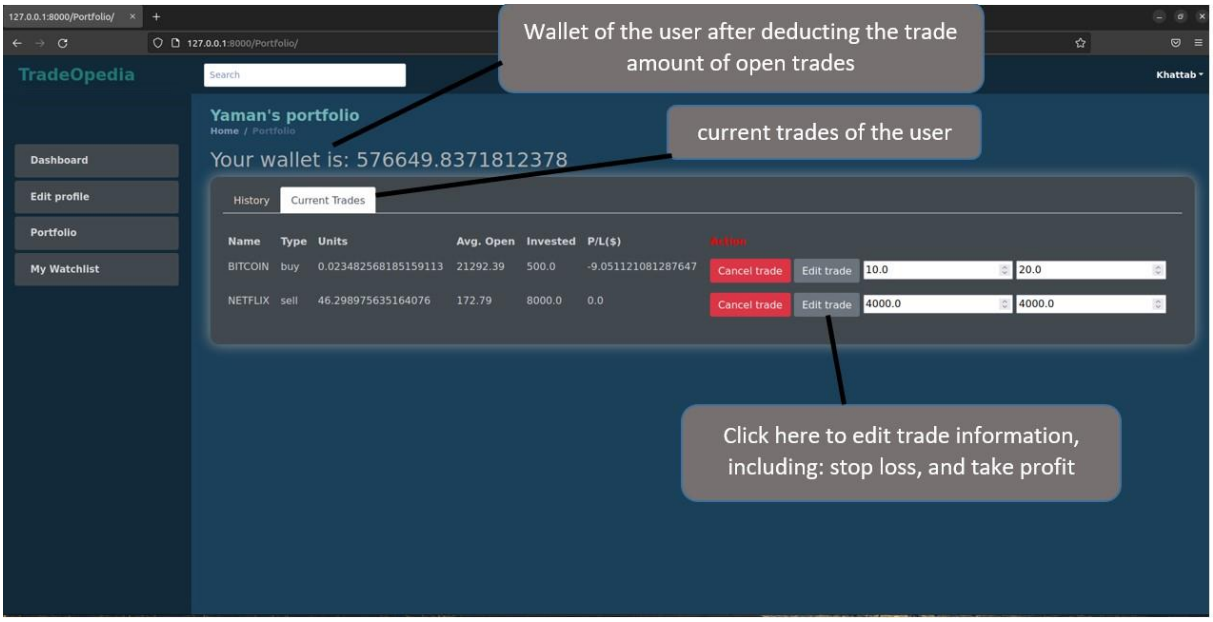
- Change password



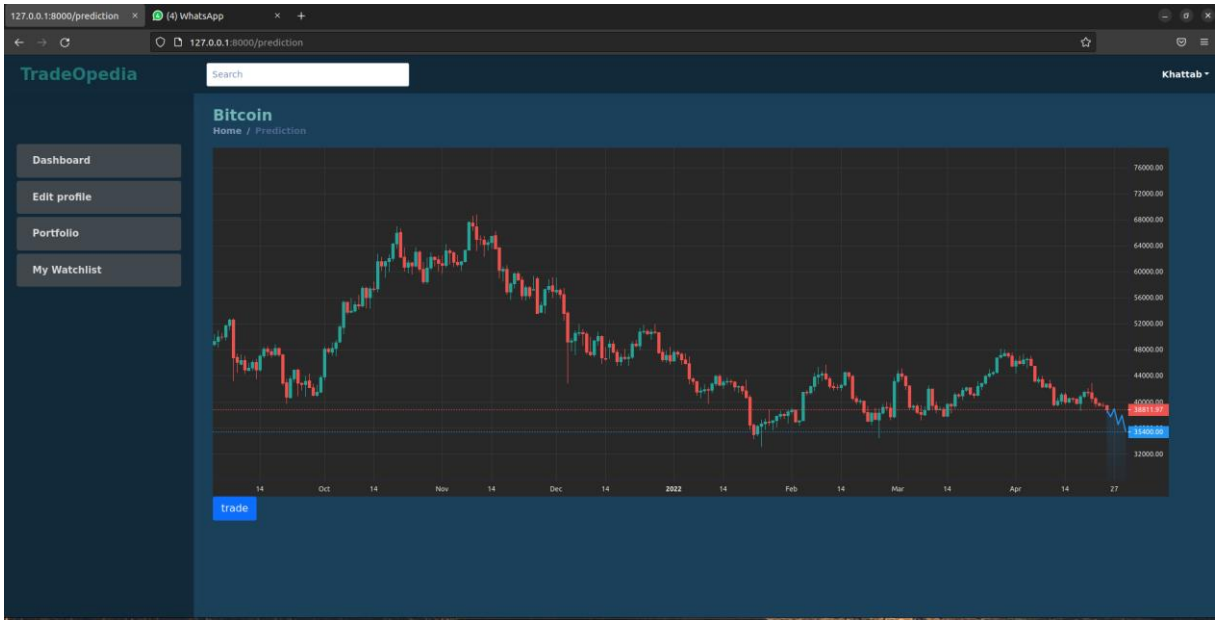
- Watchlist Page



- Profile



- Prediction



Appendix B
Document Changes

Our expectations were to make a website as real as possible by adding real money in real wallet. However, we found that adding those two require a lot of confirmation from the government sectors and banks to make secure transactions. In fact, it is considered as a whole project to work on that we are hoping to finish it in the future.

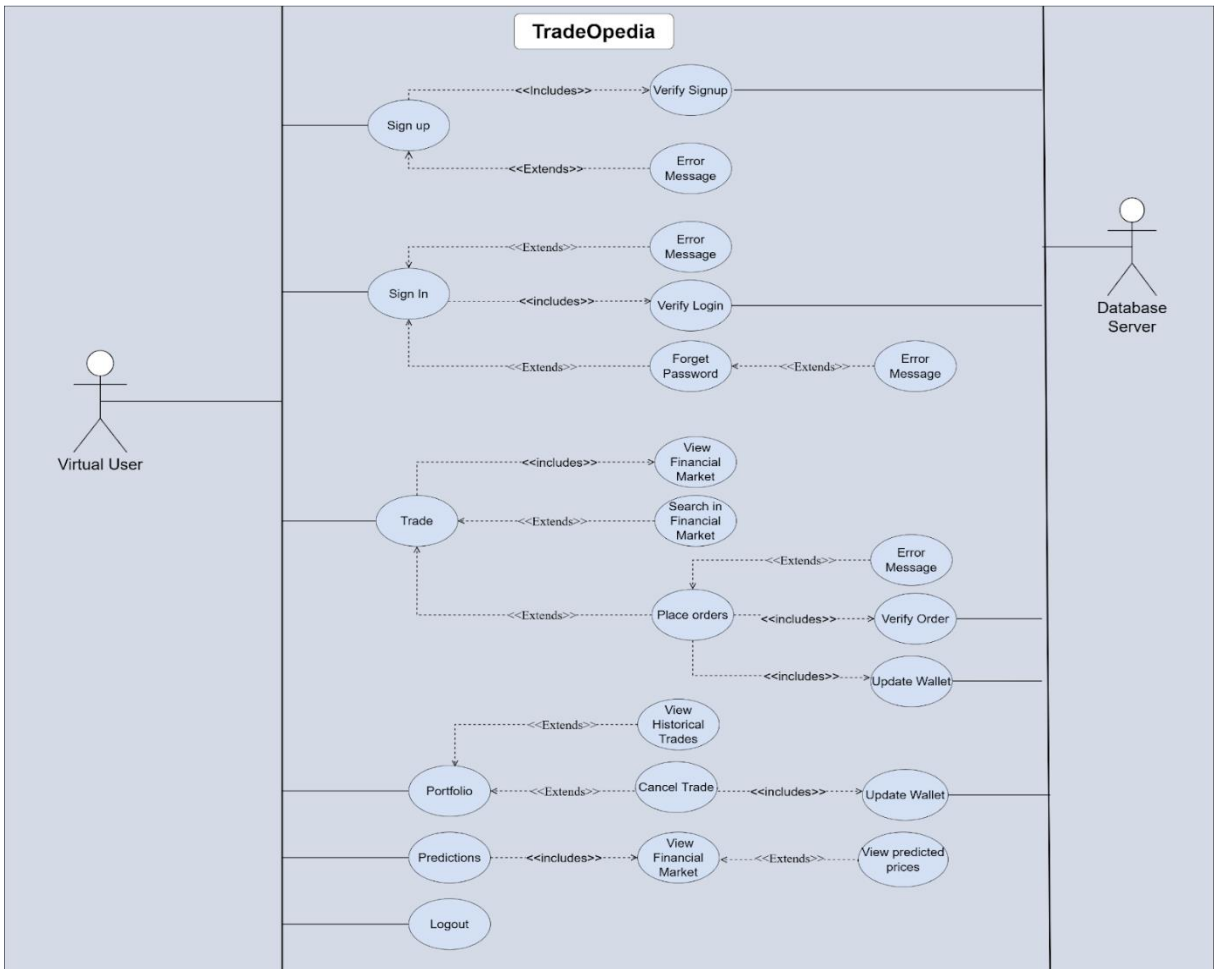
The changes are summarized in the table below as follow:

Change ID	Change	Type of Change	Reasons
C1	Real Wallet	Removed	It helped us focus on the project because implementing real wallet will require a lot of confirmations from government sectors like Amman Stock exchange.
C2	Deposit into Real Wallet	Removed	Due to the complication of the process by taking confirmation from banks. Also, it is no longer needed because the Real Wallet was removed.

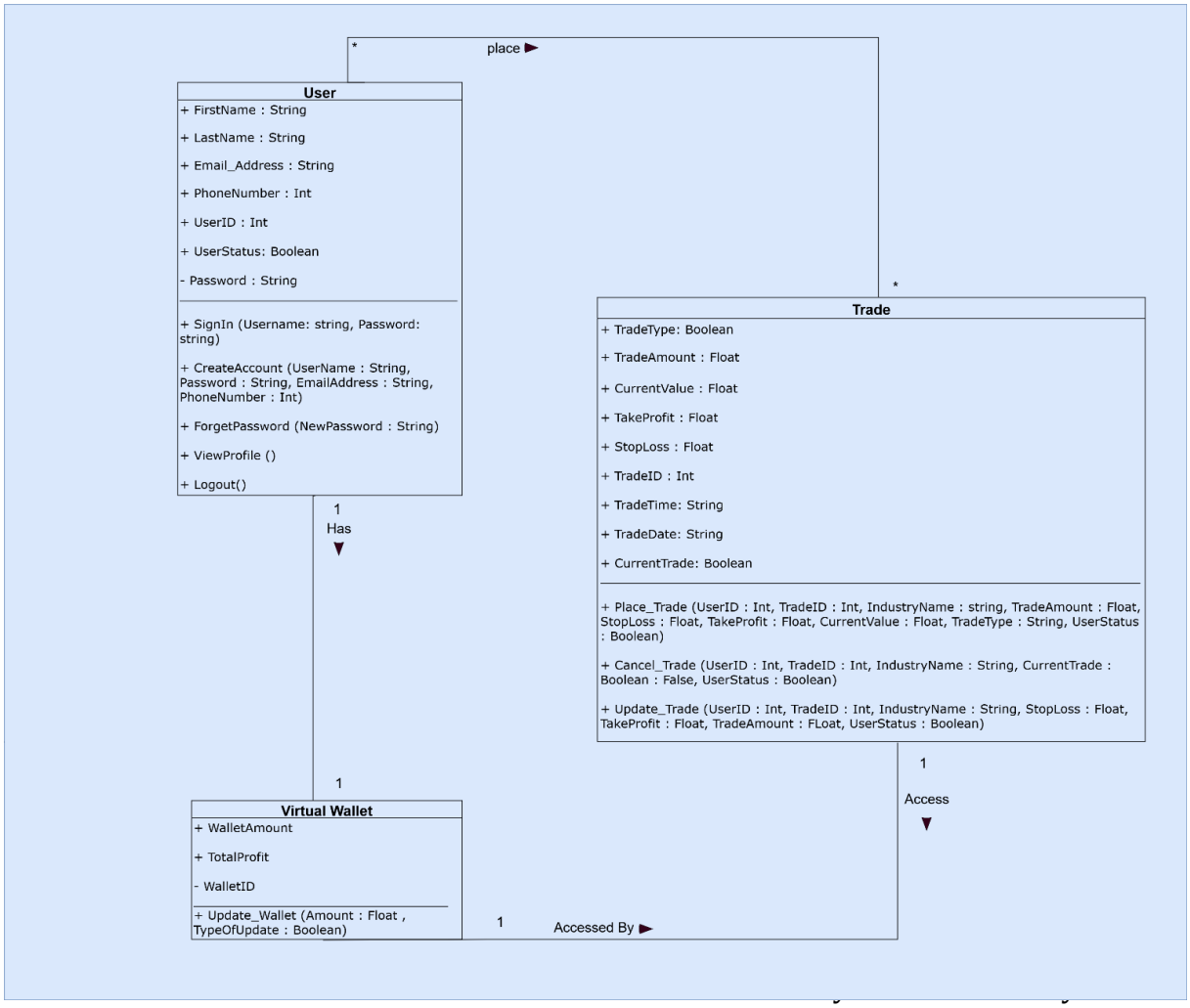
C3	Withdraw from Real Wallet	Removed	Due to the complication of the process by taking confirmation from banks. Also, it is no longer needed because the Real Wallet was removed.
C4	Forget Password	Removed	Because it requires a server to send a verification email or SMS code which we don't have.
C5	Change password	Added	User can change his/her password at any time.

According to the above changes, some Diagrams are changed as follow:

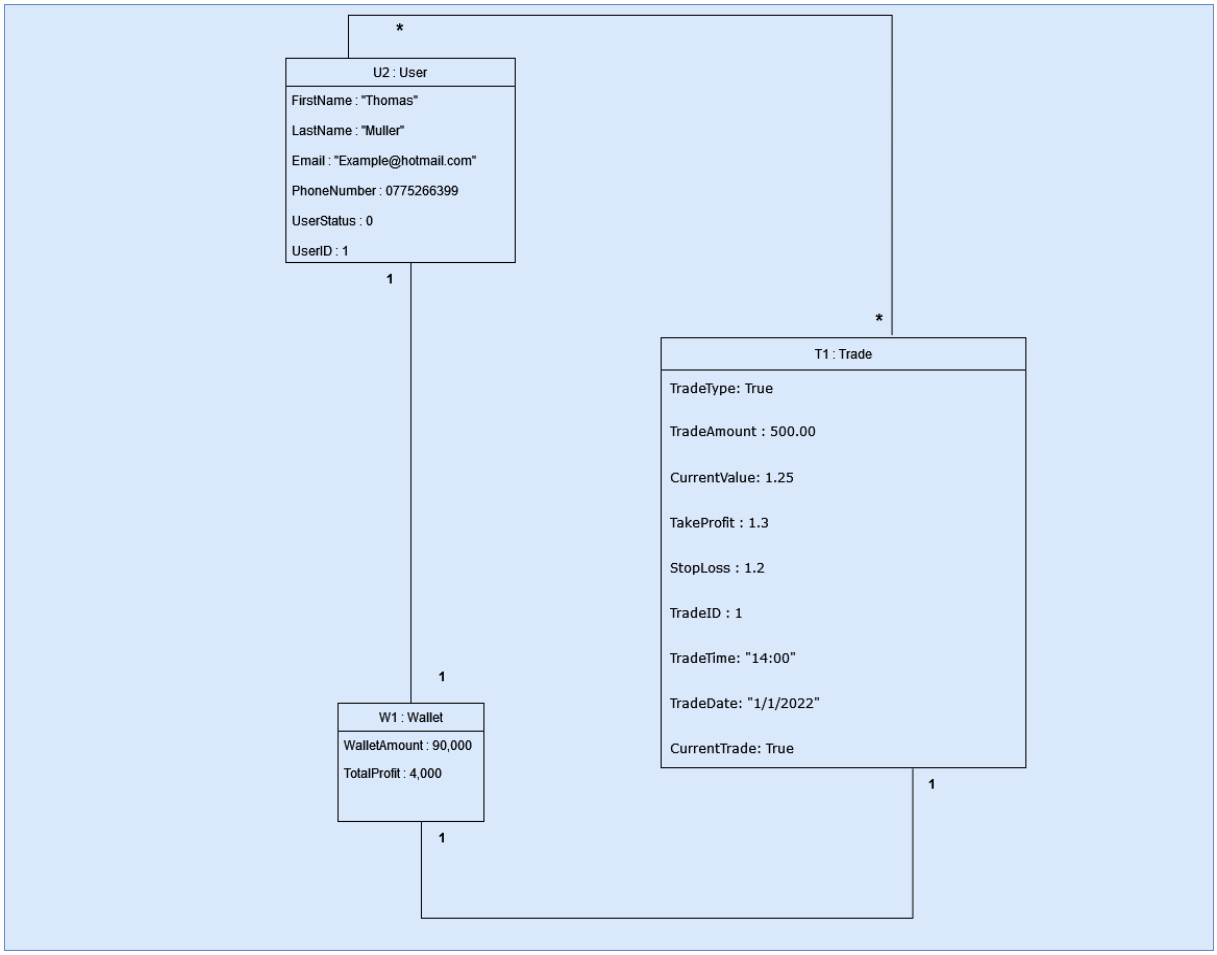
- Use Case Diagram



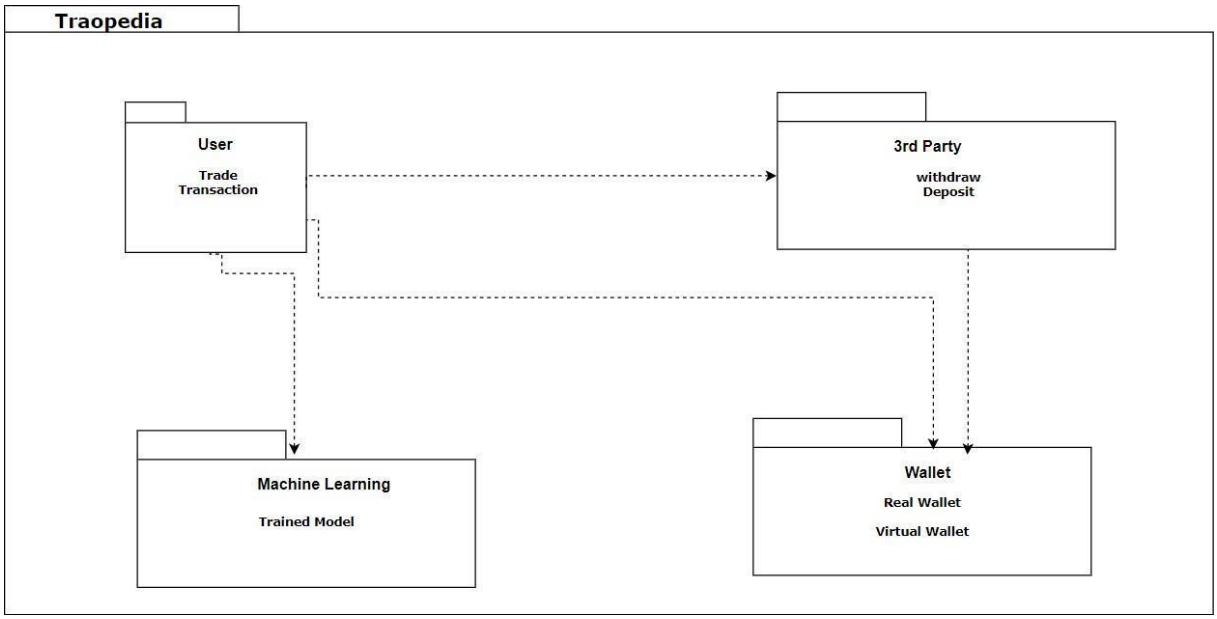
- Class Diagram



- Object Diagram



- Package Diagram



Appendix C
Code Documentation

To run the code, Linux operating system must be installed because there are some libraries we used like Celery are only compatible with Linux operating system.

The whole code is in one file named as “TradeOpedia” that has virtual environment which indicates Python and other important file. The code could be run on Visual Studio by accessing the virtual environment using the command “source venv/bin/activate” without the quotations.

After that, access the real time file and type “python manage runserver” without the quotations to run the local server.

To run the real time chart and prices these commands must by written on separate terminals and each terminal should be ran in virtual environment by typing the following commands:

- Redis-server
- Celery -A realtime beat -l info
- Celery -A realtime worker -l info

References

Geannopulos, T. (2021, 11 28). *TT® PLATFORM*. Retrieved from trading technologies website: https://www.tradingtechnologies.com/trading/tt-platform/

Iac, D. \. (2021, 11 28). *InvestoPedia*. Retrieved from https://www.investopedia.com/best-online-brokers-4587872

unknown. (2021, 11 28). *Kavout*. Retrieved from https://www.kavout.com/who-we-are/