Forex Trading AI

Computer Science Coursework
2022-2023

Timon Plemenitaš

Table of Contents

Analysis

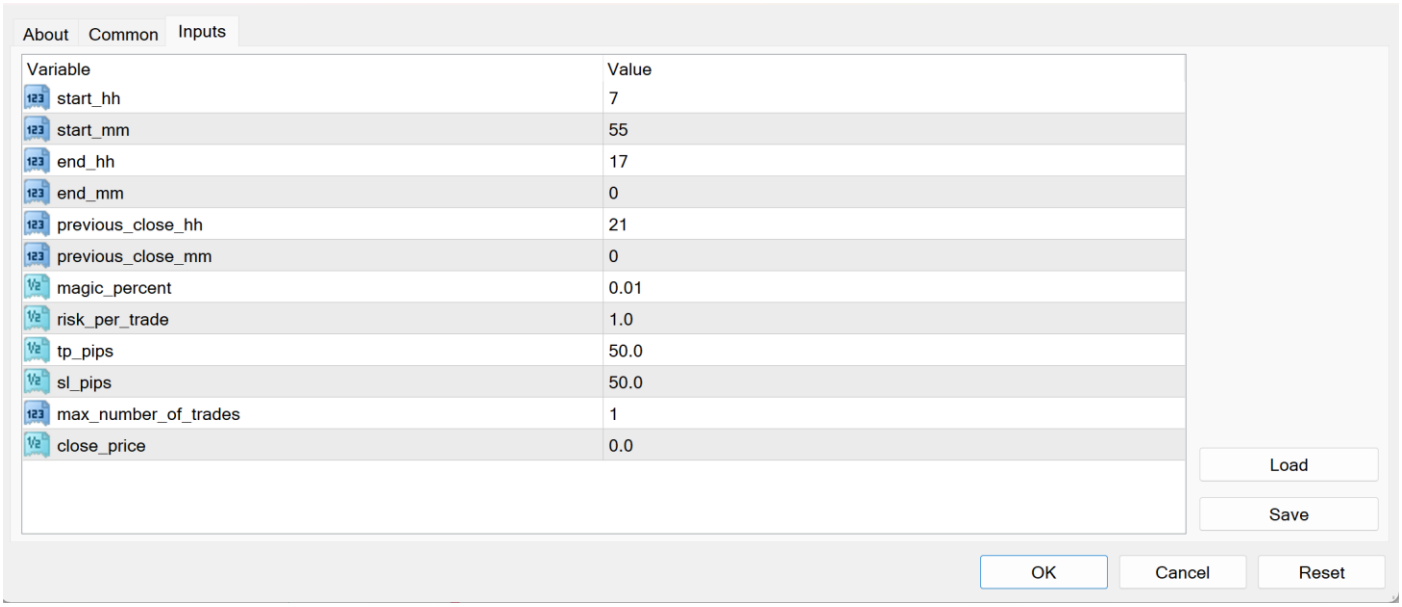
Introduction and Statement of the Problem

Trading equities, forex, cryptocurrencies, commodities, and other financial instruments have grown in popularity among the general public in recent years. However, most people who put themselves in this situation have neve1r had any experience with financial markets before, therefore presenting them with numerous challenges. Key issues include managing emotions and making informed decisions under rapidly changing market conditions, as well as the time-consuming nature of manual trading as people return to their pre-pandemic routines.

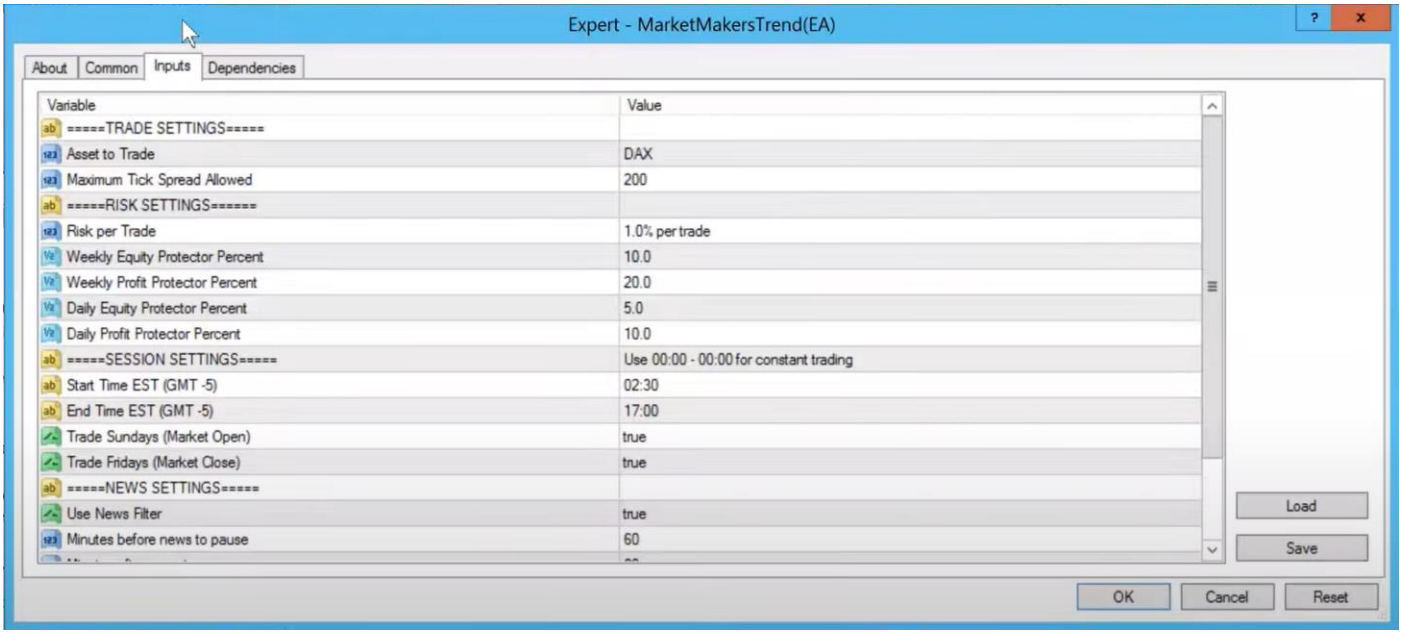
My solution to these problems is to create an automated trading software that is easy to set up and consistently generates profits through effective risk management and accurate market predictions. The solution must also address potential issues that other existing software might have, as well as incorporate any suggestions made by end users. Furthermore, the software should make market predictions based on historical data using neural networks.

Existing Systems

Current trading software solutions, such as Aurora Prime and Eaconomy, attempt to address some of these challenges created by manual trading process. However, they often fall short in crucial areas, including accurate market prediction and effective risk management. As they tend to work based on losing small and doubling down until getting it right instead of trying to predict the market with correct risk management. Furthermore, those softwares can not only be very expensive making them very inaccessible for people that are just starting with trading, but they are also often very hard to set up as they require prior knowledge of trading terms and calculations.



Eaconomy risk management and trading parameters interface.



Aurora Prime risk management and trading parameters interface.

As you can see above there are a lot of parameters that user has to set up so that the trading bot can run properly and make profit. Furthermore, while there is brief guide on how to set up these parameters on each of providers websites, it doesn't really give any in depth insight what parameters are best for what pair and how it should be used. Thus, making it very hard for new traders to understand it and optimised it. In addition to that all the trading bots I have come across including those two mentioned earlier operate through affiliate marketing making it very questionable on how those firms actually operate and what their goal is.

Users

Prospective Users

People who would benefit the most from my system are simply those who would like to be involved in financial markets but don't have the time and experience to be successful, as well as the money to spend on expensive software.

End User

To better comprehend overall user needs and expectations, I have carefully chosen my end user to be someone with a lot of experience in FX trading and software related to it. This someone will be my friend Luca Guaizza an experienced FX trader working for ACY Securities hedge found with over a decade of trading FX and other financial instruments under his belt. He has used various trading tools and software throughout his career, and I believe he has valuable insights to share and help me develop my software.

Identification of User Needs

User Interview

What are the key features you look for in an effective automated trading software?

> An effective automated trading software should have accurate market predictions, effective risk management strategies, a user-friendly interface, customisable trading parameters, and consistent profit generation capabilities.

What are some of the common limitations you've encountered with existing automated trading systems?

From my perspective, the limitations of existing systems include inaccurate market predictions, poor risk management, high costs, complicated setup processes, and the requirement of prior trading knowledge for usage, which is not a problem for me, but I am very well aware that not a lot of people have the same knowledge about FX markets as I do.

As a seasoned FX trader, what aspects of risk management do you consider to be most important for an automated trading software to incorporate?

Critical aspects of risk management to consider are setting appropriate stop-loss and take-profit levels, maintaining a favourable risk-reward ratio as well as managing position sizes.

What level of customization and user control would you prefer in an automated trading software?

A reasonable level of customization would allow users to set their preferred trading parameters, such as risk tolerance and specific trading instruments. Users should also have the ability to create an account so that their broker details are stored and have to be only entered upon sign-up.

How important is the ease of use and setup for new users who may have limited experience with trading software and financial markets?

As I mentioned earlier, ease of use and setup is crucial for new users, as they often lack the knowledge and experience to navigate complex trading tools. Therefore, I believe clear and easy instructions are vital in making software more accessible to a broader range of users, including those just starting with trading.

How important is it to have a visually appealing and user-friendly GUI for an automated trading software?

A user-friendly GUI is vital for trading software, as it enables users to quickly access and comprehend the information they need. However, for an AI-based trading tool that operates autonomously and your "customer base" is people with less or no experience, a command prompt would be sufficient as long as it offers very clear instructions.

What level of transparency and reporting would you expect from an automated trading software?

A high level of transparency and reporting is essential for users to evaluate the software's performance and make informed decisions on whether or not software is profitable. Therefore, users should have access to detailed trading logs and real-time trade monitoring.

What are some essential security features that an automated trading software should have to protect users' funds and personal information?

Security feature that I see as an essential in automated trading software is multi-factor verification as I believe in 2022, every software should have one, but unfortunately, from the experience I have, I can't say that is the case with most of the platforms.

From my analysis of other similar software, I realized you usually have to log on to the platform or a VPS in order to check the results. I believe that is very time-consuming. Therefore, I was planning to make results being sent out with an email at the end of a trading day to each user. What are your thoughts on this feature?

I think that providing users with a daily email summary of trading results is an excellent idea. This feature would save users time and offer a convenient way to stay informed about their trading performance without needing to log in to the platform or a VPS constantly. It would also help users keep track of their investments and make adjustments if needed. Overall, I believe that this feature would enhance the user experience and make the software more accessible and user-friendly.

How important is it for automated trading software to have a diverse range of supported financial instruments?

> Supporting a diverse range of financial instruments is important for automated trading software, as it allows AI to take advantage of various markets. However, your software is going to execute trades on the MT5 platform, whereas the range of financial instruments will depend on the user's broker. Thus, making you unable to influence it.

What are your expectations regarding the software's performance?

> Software should maintain consistent performance by effectively managing risk and adapting its trading strategies. While some fluctuations in performance are expected, the software should be able to minimise losses and capitalise on opportunities arising from the market.

So, what would you assess in order to evaluate the performance of an automated trading software? What key metrics would you consider?

> Performance assessment of an automated trading software should include metrics such as the win rate, risk-reward ratio, profit factor, maximum drawdown, and return on investment. Consistency in performance and the ability to minimise losses during unfavourable market conditions are also essential factors to consider.

Established User Needs and Objectives

- Accurate market predictions
- Effective risk management
- Command Prompt is fine, but instructions have to be very clear.
- Customisable trading instruments and risk.
- Multi-factor verification
- Email summaries

Acceptable Limitations

- In order for software to run properly, script has to run 24/7 therefore user would preferably have to set up a VPS to achieve the best results.
- User must download MT5 terminal from their website and enable algo traiding manually when using it for first time.
- Neural networks training and prediction might take a while depending on user’s computer capabilities.
- Not all pairs can be traded since that depends on the broker that user is using.

Objectives

1) A system that can predict, execute, and manage FOREX positions on a pair that the user wishes to trade.
2) Check that the user has an established internet connection.
    a) Display an error message if an internet connection is not established.
    b) Wait for the user to establish an internet connection.
3) The user is asked to input a number for the mode they want to use.
    a) Accept only valid inputs, no letters, just numbers within the range.
        i) If the input is not valid, display an error message.
        ii) Ask for input again until a valid input is entered.
    b) If users want to use DEMO mode.
        i) No email or broker account is needed.
        ii) The system connects to a pre-set demo account.

iii) Makes a prediction and executes trades like a real account to show the user how it works.

c) Account mode, where users can create an account or log into an existing one.
   i) The program checks that the user's input is an email; otherwise, it asks for another one.
   ii) The email is checked if an account with the same email already exists in the database.
      (1) If they do not have an account, a confirmation code is sent (using API) to this email to confirm the email.
         (a) Users have the option to change their email address.
         (b) They are asked to create a password if the code is correct.
            (i) Password must be input twice, ensuring no errors were made in the first entry.
            (ii) The user is asked to restart the password creation process if passwords do not match.
         (c) The user's IP address is collected and checked in which country the user is using the software. Then, the IP and country of usage are stored in a MySQL database
         (d) Users are asked to input broker details, which are then checked (based on connection to the terminal; if an error is returned, it is invalid), and, if not valid, then the account is not created, and the user has to input those again or start over if they input 0
         (e) Store all broker data in the database, so users do not need to enter them again next time (if valid).
      (2) If they already have an account:
         (a) Their IP address is checked.
            (i) If the IP address hasn't been used yet, a confirmation email is sent, and if the code is correct, the IP is stored in a MySQL database so that next time the user uses this IP address, they don't need to do email authorization.
         (b) Users are then asked to enter their password.
            (i) If a user enters the password incorrectly three times, recovery mode is enabled.
            (ii) Users then must verify their identity using an email verification code.
            (iii) If correct, the user can create a new password and continue using the account.
         (c) They are automatically logged into the broker account as all the details they provided upon sign-in are stored in the MySQL database.

d) Mode to show profits others have made:
   i) Users that have an account; profits are recorded and stored in a MySQL database every day at 21:00 GMT.
   ii) Using a merge sort algorithm, all those profits from accounts are sorted in descending order, and the top 5 are displayed.
   iii) Show average profit (using MySQL AVG () function.
   iv) Redirect users back to the start to choose between demo or sign-in/login mode.

4) Users are asked to choose from 3 different risk options depending on their preferences of how much they are willing to risk.
5) Users are asked to enter a trading pair name they wish to use.
   a) Five pairs that the software was optimised for are displayed.
   b) Users can enter any other forex pair that is supported by their broker.
      i) The pair is checked if the broker supports it (the pair is sent to the terminal; if an error is returned, it is invalid).
      ii) Ask users to enter a pair again if invalid.
6) Making the prediction:
   a) The prediction is going to be based on an LSTM model trained using past data of the pair set to be predicted; this data is queried using the Yahoo Finance API.
7) As markets are unpredictable and even systems that worked well in the past might not accurately predict the market in the future, there is room to reduce risk and exposure while maximising profits by developing a special profit-taking system that takes profit every few times the market moves by a certain distance, regardless of the direction.

8) At the end of the trading day (21:00 GMT), the system sends an email to the user with the profits for the day. This is needed since the program has to be run on a server; therefore, it is more convenient to get profits delivered to your email instead of having to log on to the server to check them.
   a) Along with profits, the system also sends a motivational quote (generated by an API) to make the user feel better in case the software has made a loss :)
   b) The number of trades taken and won/lost is also recorded and stored/updated in the MySQL database.
9) The software will run continuously and make predictions at the start of a new trading day (21:00 GMT) and then, based on that, enters the market. It also closes the previous day's loss/profit at the same time.
10) Host the MySQL database on a server.
11) If errors occur, the software will be stopped, trades will be closed, and the program will run again.
12) Enable users to also check trades using an app terminal on their phone if they wish (only for login mode)
13) Create an .exe file and generate an application so it can be executed on any computer with or without Python installed, and all the code can be transferred with one application/file.

Data Directories

As in-depth data dictionaries and data flow diagrams require very good knowledge of internal processes and data structures of the above-mentioned software, which are private as if they were publicly disclosed, it could potentially harm their competitive advantage in the market. Thus why, I will only provide a general overview of the data elements and data flow for a typical trading robot.

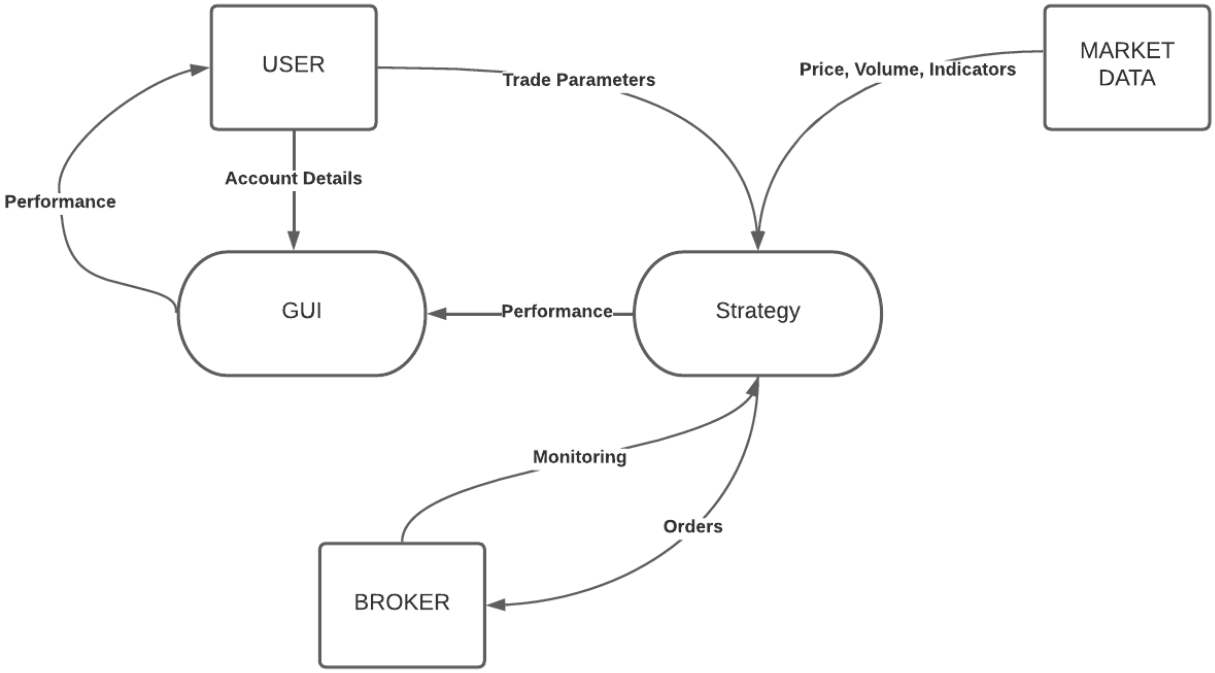
User related data

Data (element)	**Data Type**	**Description (what is it? What is it used for?)**
Account Information	String	Email, Password, Country
Broker Account Information	Integer/Float	Account number, Balance, Margin, Open positions
Trade Parameters	Integer/Float	Trade size, Risk tolerance
Performance Metrics	String/Integer/Float	Statistics (profit/loss, win rate, drawdown)

Machine related data

Data (element)	**Data Type**	**Description (what is it? What is it used for?)**
Market Data	Integer/Float	Historical data, Technical indicators, Volumes
Trading Signals	Integer/String	Used to decide and what to do and what order to use
Trade Execution	Integer/Float	Instructions sent to the broker

Data Flow



Hardware & Software

Minimum requirements:

- 1 gigahertz (GHz) or faster CPU with two or more cores (64-bit)
- 4 GB or greater of RAM
- 64 GB of secondary storage
- Stable internet connection

However, to ensure the smooth operation of the software, the user is suggested to use a system with at least 16GB of RAM, a newer GPU, and a 6-core CPU. This is because the software will use LSTM neural networks model to come up with the price predictions meaning it will require a significant amount of computing power to process data and train itself, thus having a robust hardware setup will ensure fast prediction and better market entries as the market is not stopped while prediction is being made.

In addition, the only software requirement apart from my software itself will be the MT5 terminal which the user should install before running my application.

Data Sources and Destinations

Two main data sources will be Yahoo finance API and MT5 terminal. Data from Yahoo API will be directly fed into the training model in the shape of a data frame variable to make a prediction; therefore, no storing LT is needed. Moreover, data from the MT5 terminal will be quarried and then filtered to extract key data. Furthermore, depending on the data quarried from the MT5 terminal, data is either going to be stored in the MySql database or as a local variable.

Data Volumes

The data volumes in the project are relatively small, with the exception of historical data (few MB), which is obtained through Yahoo Finance API in order to feed the neural networks. Therefore, it is not significant enough to be highlighted or considered a challenge in terms of storage or processing capacity in general with the exception of neural networks.

Design

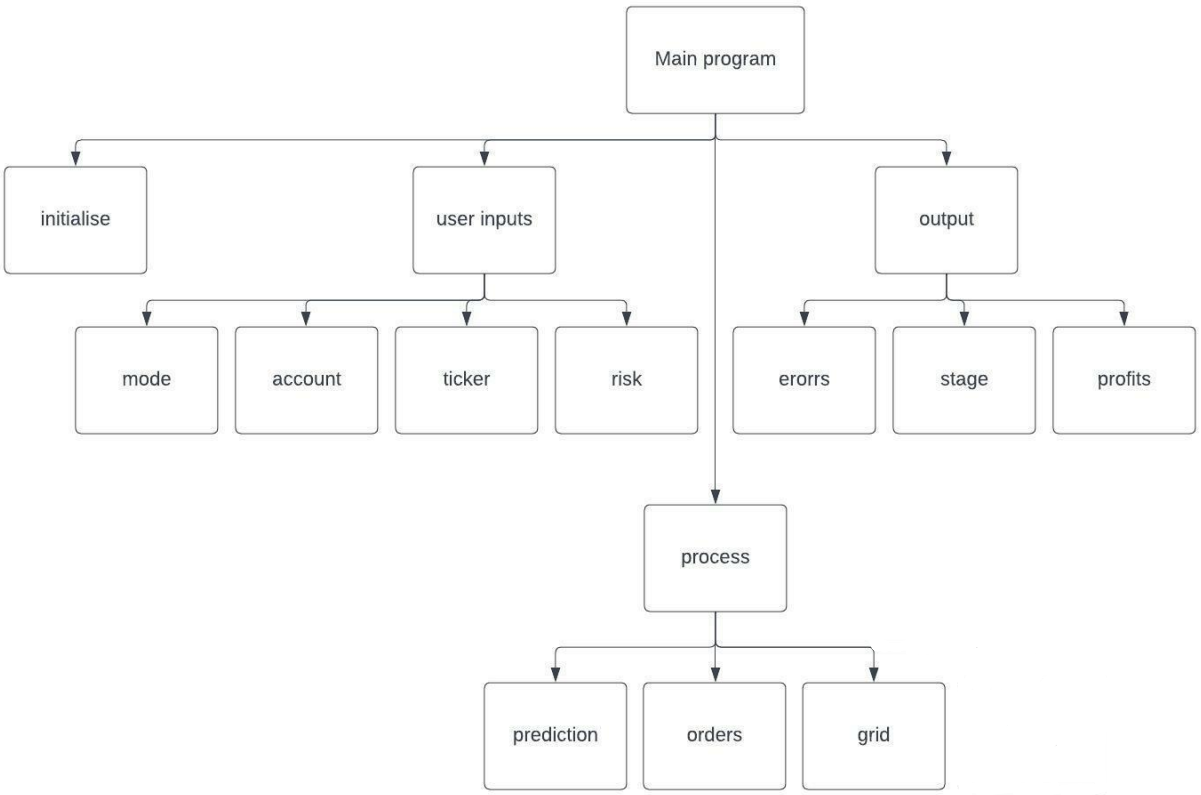
Overall System Design

This project will be split into several separate programs/algorithms, which will be combined into one main one representing the final product. This will enable me to organise my code more clearly and make tests as I go along, making my coding more efficient. In this section, I will only cover the most relevant algorithms in an abstract way for the purposes of understanding the algorithms and what they actually do.

File Structure

- Brain.py
- CreatingDataBase.py
- Grid.py
- Orders.py
- SendProfits.py
- UserLogIn.py
- Main.py

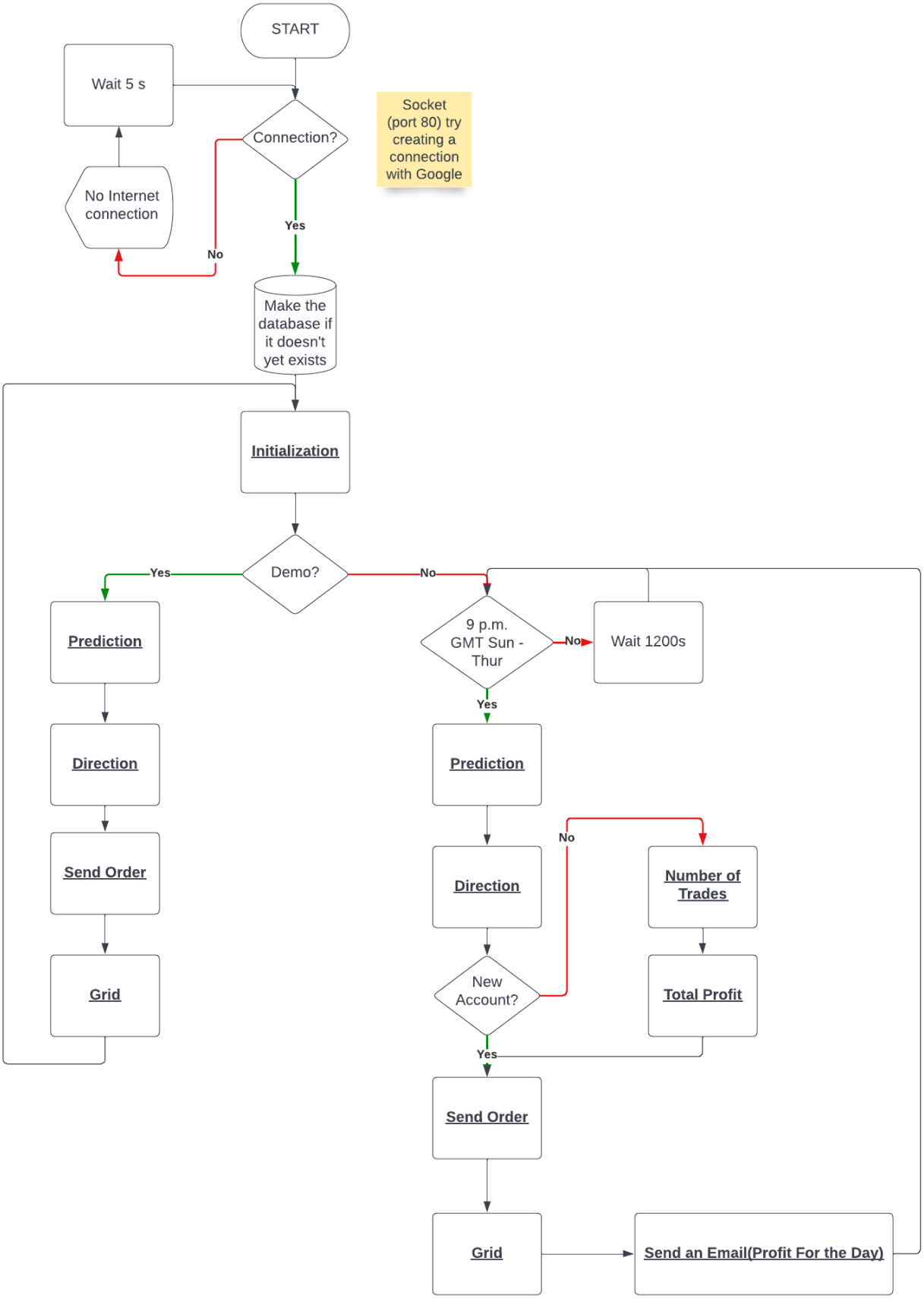
Overall Structure Diagram



The basic idea behind the software is that users can operate it in two different modes (demo mode and account mode), or it can simply display the profits that other users have made in the past. When users choose the mode they want to run, they are either prompted to create an account and input all the necessary details or to enter their email and password if they have already created an account. Next, they are asked to select the pair they want to trade. Afterwards, they can choose their preferred level of risk. Risk settings are presented as conservative, moderate, and aggressive, enabling even those without trading knowledge to use the software. Once this is done, predictions are made using neural networks, and trades are executed and managed. The demo mode functions similarly but with less accurate predictions. If errors occur, a message and error code will be displayed.

Algorithms

General System Flowchart (Main)

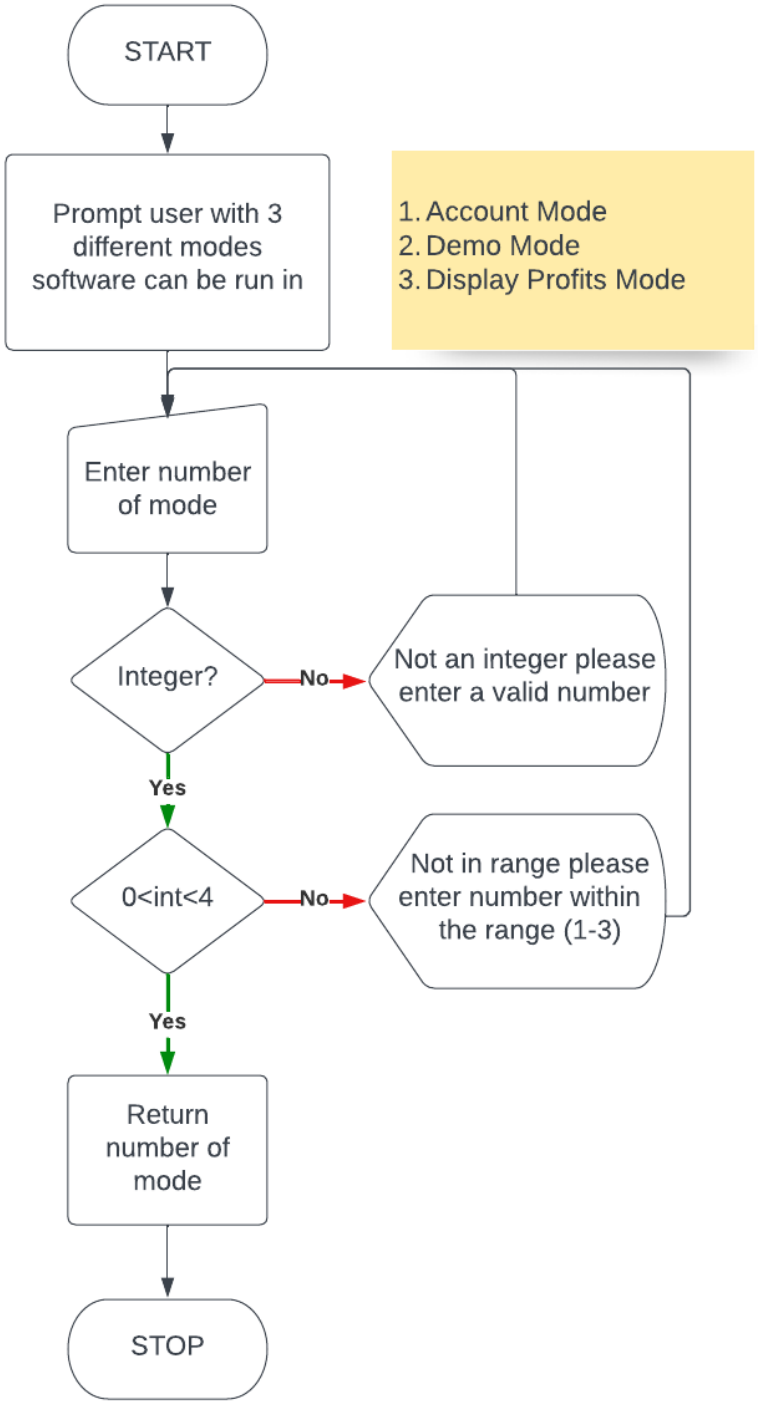


The general system flowchart illustrates the main idea behind the program and how it should function using information-hiding abstraction. First, the program checks if the computer has an established internet connection, as it cannot run without one. This is done by attempting to connect with Google; if there is no connection, an error message is displayed (every 5 seconds) until the user establishes an internet connection. Once connected, a database is created if it does not already exist. After initialization, the program proceeds to either demo mode or account mode.

In demo mode, the program starts by making a prediction and determining the direction. It then sends an order and manages it with the grid function. If a trade is closed, a new prediction is made immediately. In account mode, the program first checks if the time is right to enter the market (9 p.m. GMT) since this is when the daily candle closes, and the prediction is most accurate. The program then runs the prediction and determines the direction. However, before opening a trade (if it is not the first trade on the account), the results (number of trades, total profit) are recorded in the database. The position is then opened and managed by the grid. When the trade is closed, the user receives an email detailing the profits made by the AI that day. The bold and underlined processes in the flowcharts are explained in greater detail throughout this section to enable readers to understand the ideas more clearly and comprehensively.

User LogIn

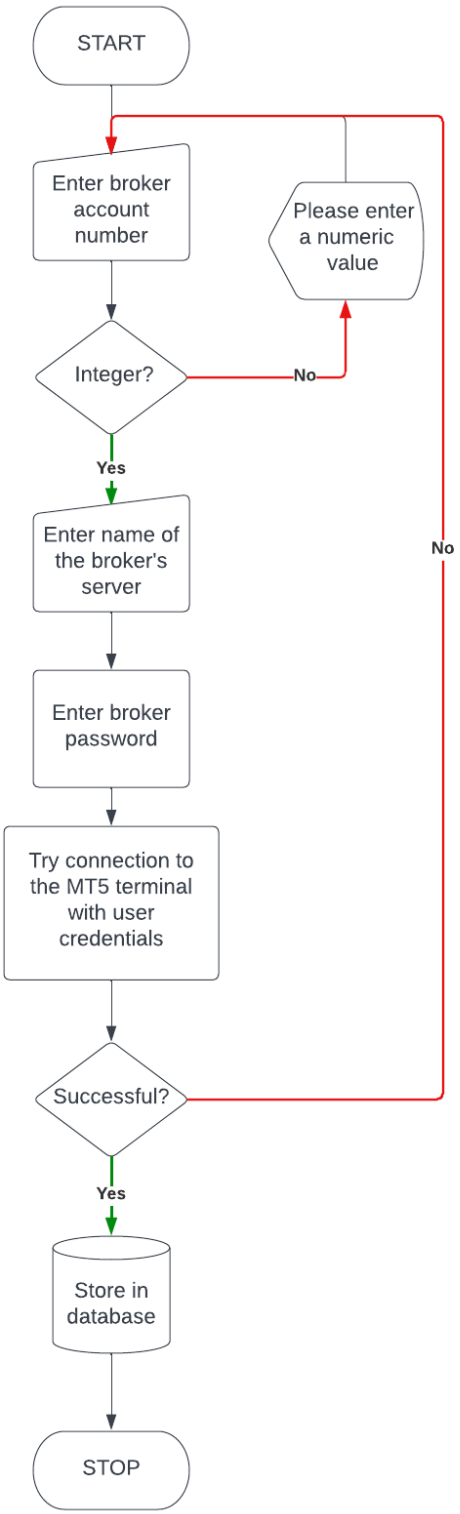
Mode



```
FUNCTION Mode()
    OUTPUT 1. Account
    OUTPUT 2. DEMO
    OUTPUT 3. Show Profits
    Mode ← INPUT Enter a Number

    FUNCTION ValidateMODEInput(Mode)
        TRY
            ModeINT ← CONVERT_TO_INTEGER()
            IF ModeINT < 1 OR ModeINT > 3 THEN
                WHILE ModeINT < 1 OR ModeINT > 3 DO
                    Mode ← INPUT Valid Number
                    ValidateMODEInput(Mode)
                END WHILE
            ELSE
                ModeToUse ← Mode
            END IF
        EXCEPT ValueError THEN
            Mode ← INPUT Numeric value
            ValidateMODEInput(Mode)

    END FUNCTION

    Mode ← ValidateMODEInput(Mode)
    RETURN Mode

END FUNCTION
```

Mode algorithm prompts the user to enter a number to select the mode in which they want to run the software. To ensure that the user enters a valid input, the algorithm first checks if the input is a number. If not, it displays a message stating that the input should be a valid number. Additionally, the algorithm verifies that the input is within range; otherwise, an error message is displayed. The function then returns the valid user input..

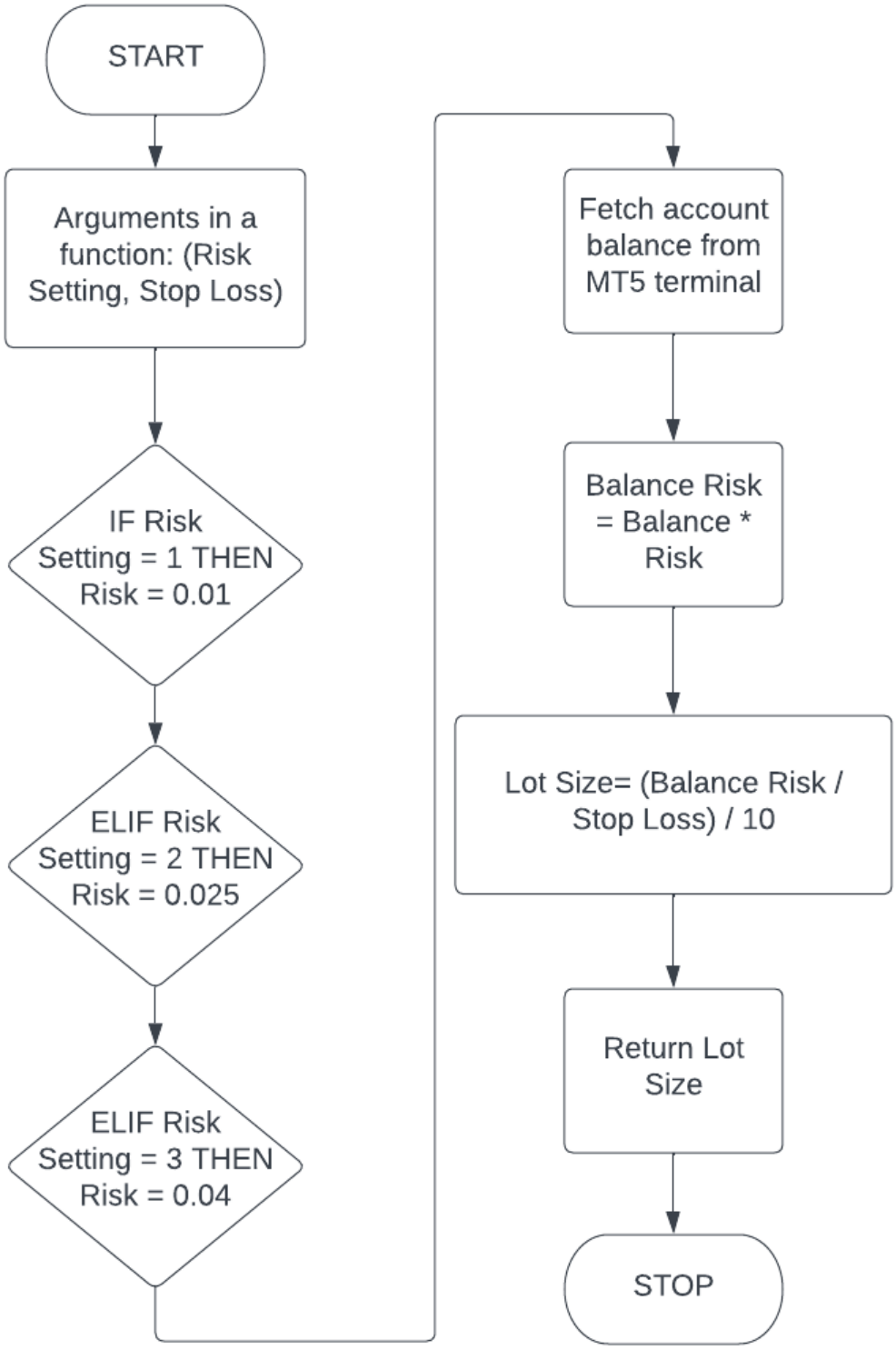
Broker Data


START
Enter broker account number
Please enter a numeric value
Integer?
No
Yes
Enter name of the broker's server
No
Enter broker password
Try connection to the MT5 terminal with user credentials
Successful?
Yes
Store in database
STOP

```
FUNCTION BrokerData()
    AccountNumber ← INPUT Broker account number

    FUNCTION ValidateAccountNumber(EnterAccountNumber):
        TRY
            EnterAccountNumberINT← CONVERT_TO_INTEGER()
            RETURN EnterAccountNumberINT

        EXCEPT ValueError
            AccountNumber← INPUT Numeric value
            ValidateAccountNumber(AccountNumber)
    END FUNCTION

    FinalAccountNumber← ValidateAccountNumber(AccountNumber)
    IF FinalAccountNumber = 0 THEN
        BACK to START

    ELSE
        Server← INPUT Broker server
        PasswordBroker← INPUT Broker password

        IF LogInMT5 not Succesfull THEN
            BrokerData()
        END IF
    END IF

END FUNCTION
```

This algorithm prompts the user to enter their broker account details to create an account and so it can then be automatically connected to the account when used again. The user is first asked to enter their account number, and if it is a number, they can enter server details and a password. These inputs are then passed to the terminal to connect to the broker. If the user's credentials are incorrect, the connection will be unsuccessful, and the user will be asked to enter them again. However, if the credentials are correct, they are stored for use the next time the user logs in.

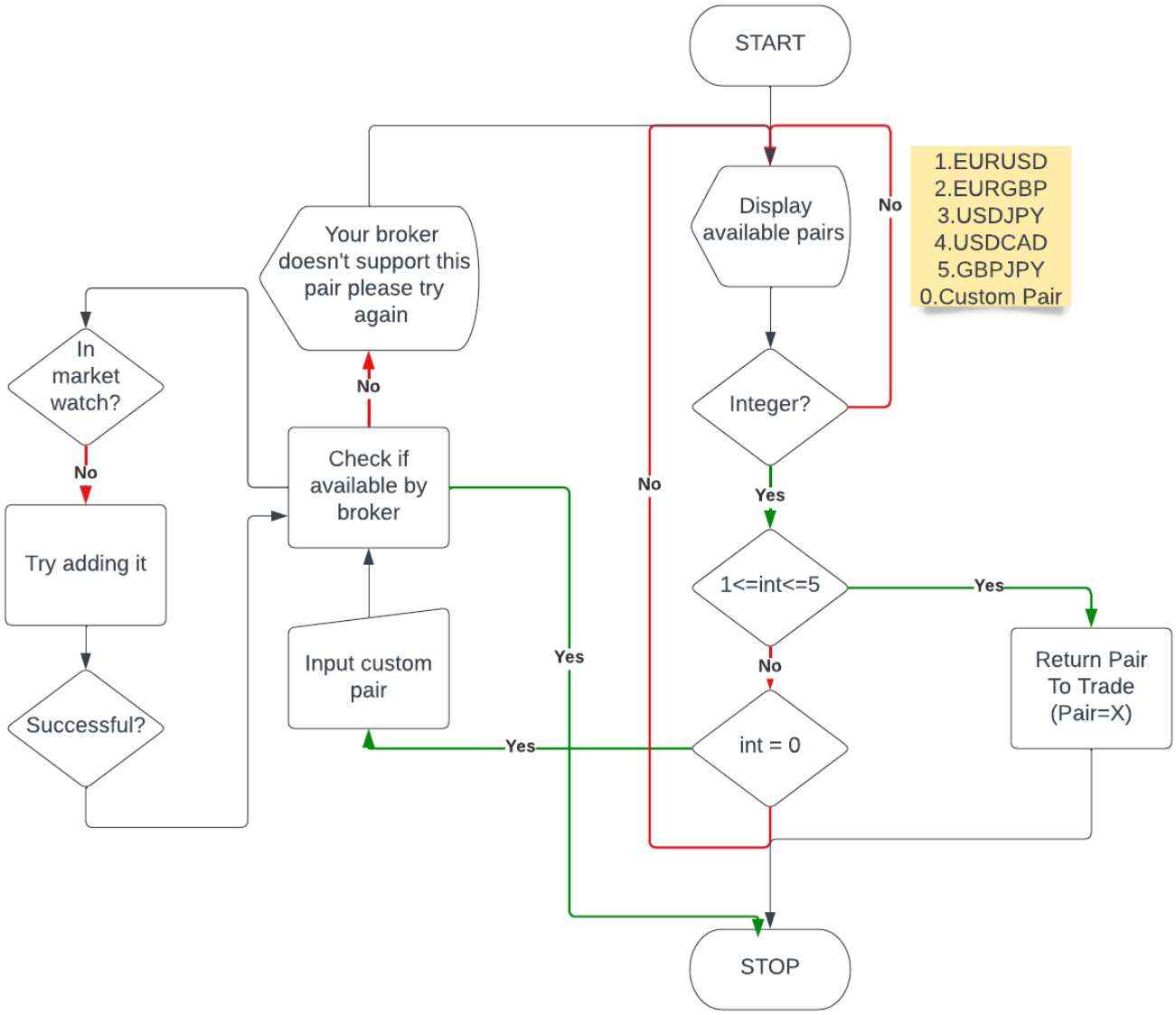
Calculating Lot Size



```
FUNCTION CalculatingLotSize(RiskSetting, SL)
    AccountInfo ← FROM MT5
    IF AccountInfo ≠ None THEN
        AccountInfoDictionary ← Balance from AccountInfo
        Balance ← CONVERT to Data Frame
        Balance ← CONVERT_TO_INTEGER() and round
    END IF

    RiskSetting ← CONVERT_TO_INTEGER()
    Risk ← 0
    IF RiskSetting = 1 THEN
        Risk ← 0.01
    ELSEIF RiskSetting = 2 THEN
        Risk ← 0.025
    ELSEIF RiskSetting = 3 THEN
        Risk ← 0.4
    END IF

    BalanceRisk ← Balance * Risk
    Lotsize ← BalanceRisk / SL
    Lotsize ← Lotsize / 10
    Lotsize ← round to 2.d.p

    RETURN Lotsize
END FUNCTION
```

The lot size is calculated based on the risk setting, where each risk setting represents a certain percentage of the account balance. To calculate the lot size, the account balance is fetched from the terminal and multiplied by the risk to determine how much of the account the user is willing to risk. This number is then divided by the stop loss and 10 to obtain the correct lot size to be used when opening a position.

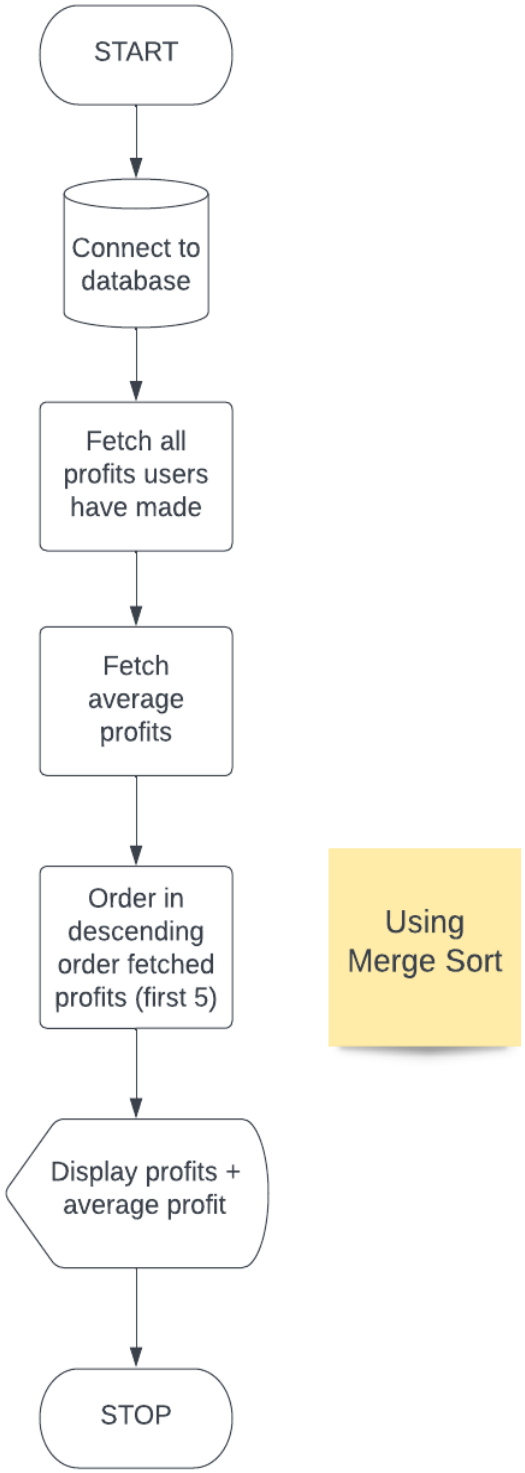
What To Trade



```
FUNCTION WhatToTrade()
    OUTPUT Pair 1
    OUTPUT Pair 2
    OUTPUT Pair 3
    OUTPUT Pair 4
    OUTPUT Pair 5
    OUTPUT 0 for other

    PairToTrade ← INPUT Number of Pair

    FUNCTION ValidatePairToTrade(PairToTrade)
        PairToTradeINT ← CONVERT_TO_INTEGER()
        IF PairToTradeINT = 0 THEN

            FUNCTION CustomPairToTrade()
                CustomPair ← INPUT FOREX Pair
                TRY
                    SymbolInfo ← get symbol info from MT5
                    IF SymbolInfo is None THEN
                        OUTPUT CustomPair, not found
                EXCEPT
                    OUTPUT Not Supported
                    WhatToTradeInput()

                RETURN CustomPair
            END FUNCTION

            PairToTradeFinal ← CustomPairToTrade()

        ELSE IF PairToTradeINT < 1 OR PairToTradeINT > 5 THEN
            WHILE PairToTradeINT < 1 OR PairToTradeINT > 5 DO
                PairToTrade ← INPUT  Number between 1 and 5, or  0
                ValidatePairToTrade(PairToTrade)
            END WHILE
        ELSE
            PairToTradeFinal ← PairToTrade
        END IF
        RETURN PairToTradeFinal
    END FUNCTION

    ValidatePairToTrade(PairToTrade)
    RETURN PairToTradeFinal
END FUNCTION
```

This algorithm asks the user to choose a symbol (pair) to trade. Users can select from five pairs that the AI was optimised for or choose option 0 to enter a custom symbol. If the custom symbol the user wants to trade isn't already in the market watch (recommended by the broker), the algorithm attempts to add it. If it cannot be added, that means the broker does not support that pair, and a message is displayed informing the user while prompting them to enter another pair. This process also ensures that only valid forms of input can be entered.

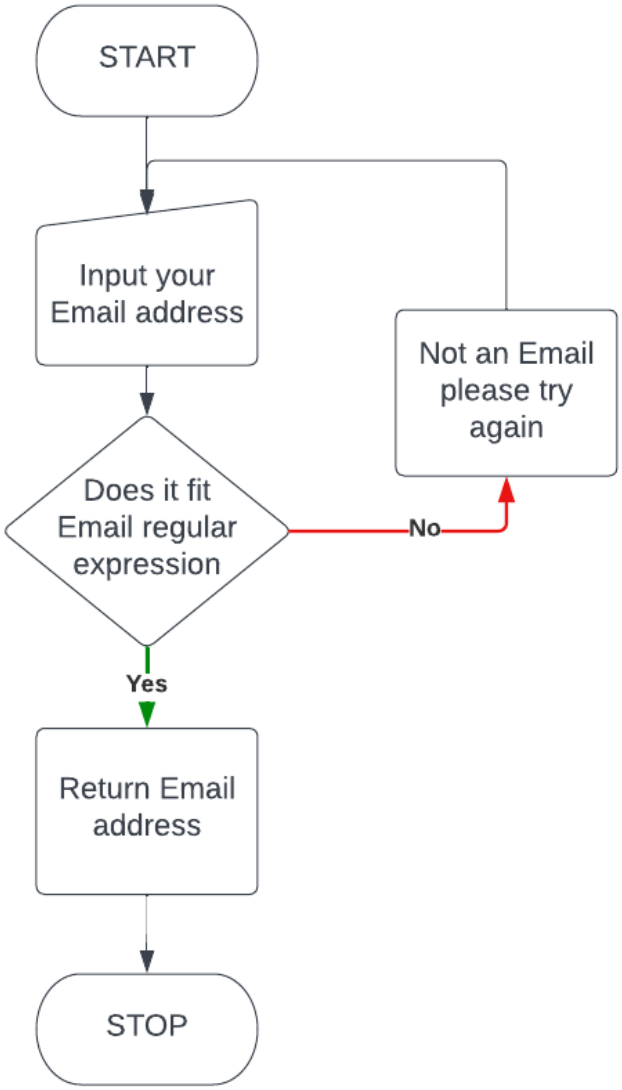
Show Profits



This algorithm connects to the database to fetch all profits users have made in the past (only real accounts are recorded). It processes the data using a merge sort algorithm to sort and display the five largest profits. It also calculates the average of all users' profits for display purposes. The reason for using the merge sort algorithm is that, despite extensive research on the sorting algorithm used by MySQL when fetching data by order, I couldn’t find it. Therefore, as a precaution, I implemented merge sort algorithm in case the number of users increases to the extent that it would affect the processing times for displaying profits.

```
FUNCTION ShowProfits()
    CONNECT to the DataBase
    AllProfitsTuple  ← FETCH All profits from the database
    AverageProfit ← FETCH Avrage profit from the database

    OUTPUT AverageProfit

    AllProfitsTuple ← CONVERT into list

    FUNCTION MergeSort(ListToSort)
        IF LENGTH(ListToSort) > 1 THEN
            Middle ← LENGTH(ListToSort) // 2
            LeftSide ← ListToSort[:Middle]
            RightSide ← ListToSort[Middle:]
            MergeSort (LeftSide)
            MergeSort (RightSide)
            t ← 0
            f ← 0
            m ← 0

            WHILE t < LENGTH(LeftSide) AND f < LENGTH(RightSide) DO
                IF LeftSide[t] <= RightSide[f] THEN
                    ListToSort[m] ← LeftSide[t]
                    t ← t + 1
                ELSE
                    ListToSort[m] ← RightSide[f]
                    f ← f + 1
                END IF
                m ← m + 1
            END WHILE

            WHILE t < LENGTH(LeftSide) DO
                ListToSort[m] ← LeftSide[t]
                t ← t + 1
                m ← m + 1
            END WHILE

            WHILE f < LENGTH(RightSide) DO
                ListToSort[m] ← RightSide[f]
                f ← f + 1
                m ← m + 1
            END WHILE
        END IF
    END FUNCTION

    MergeSort (AllProfitsTuple)

    ProfitsToShow ← 5
    WHILE ProfitsToShow > 0 DO
        OUTPUT Profit, ProfitsList[last]
        POP(last)
        ProfitsToShow ← ProfitsToShow - 1
    END WHILE
END FUNCTION
```

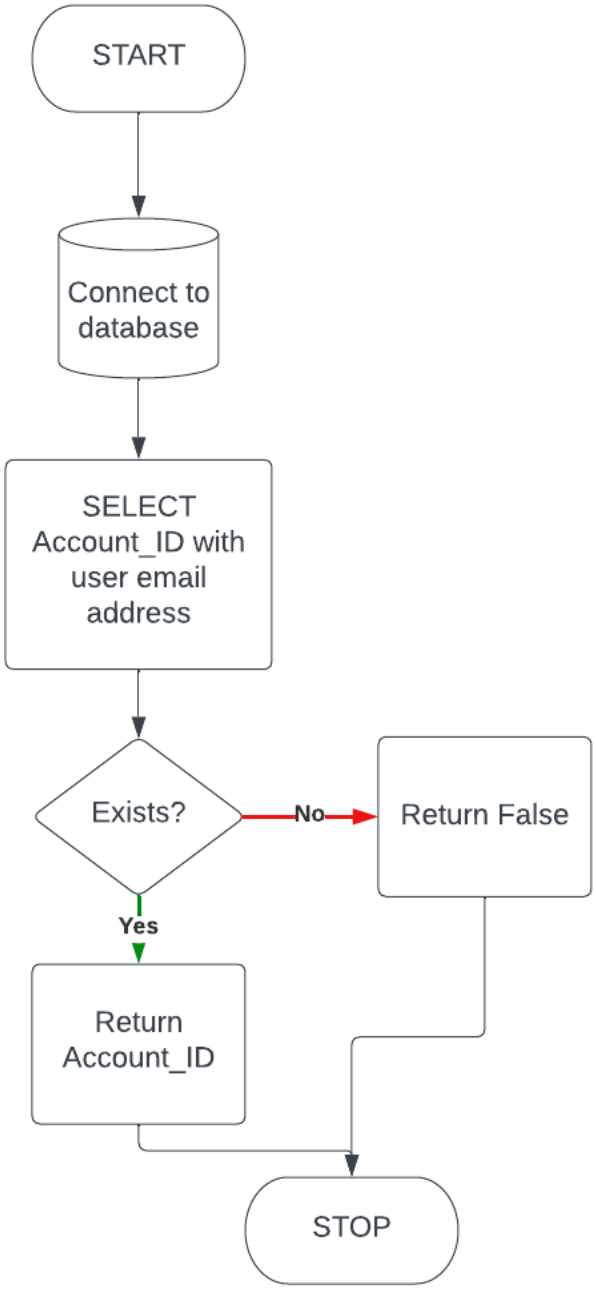
Email Validation



This algorithm checks if the input entered is an email. If not, it prompts the user to enter their email address again. This is done using regular expressions.

```
FUNCTION EmailValidation()
    Email ← INPUT Email address
        FUNCTION Check(EmailToCheck)
            EmailTemplate ← regular expression for correct email address
            IF Email MATCHES_PATTERN THEN
                RETURN EmailToCheck
            ELSE
                OUTPUT Invalid Email
                Email ← INPUT()
                RETURN Check(Email)
            END IF
        END FUNCTION

    Email ← Check(Email)

    RETURN Email
END FUNCTION
```

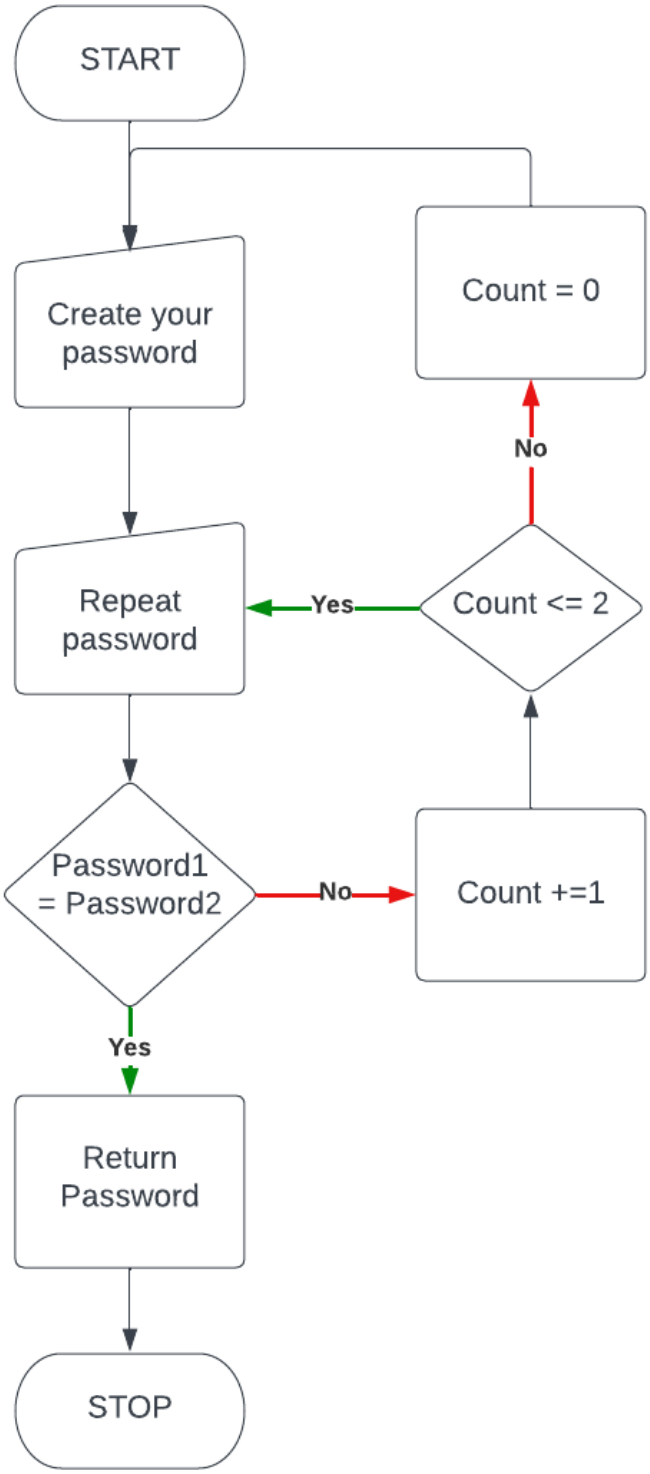
SignIn/SignUp



Initially, the algorithm connects to the database and attempts to select the email address entered by the user. If it does not exist, 'false' is returned (as an error occurs if there is nothing to fetch), then allowing the user to create an account. Otherwise, the account ID is returned.

```
FUNCTION SignUpSingIn(EmailToCheck):
    CONNECT to the DataBase
    Emails ← FETCH All emails that match email provided
    TRY
        IDNumber ← from Emails get ID number for it
        RETURN IDNumber
    EXCEPT IndexError
        RETURN False
END FUNCTION
```

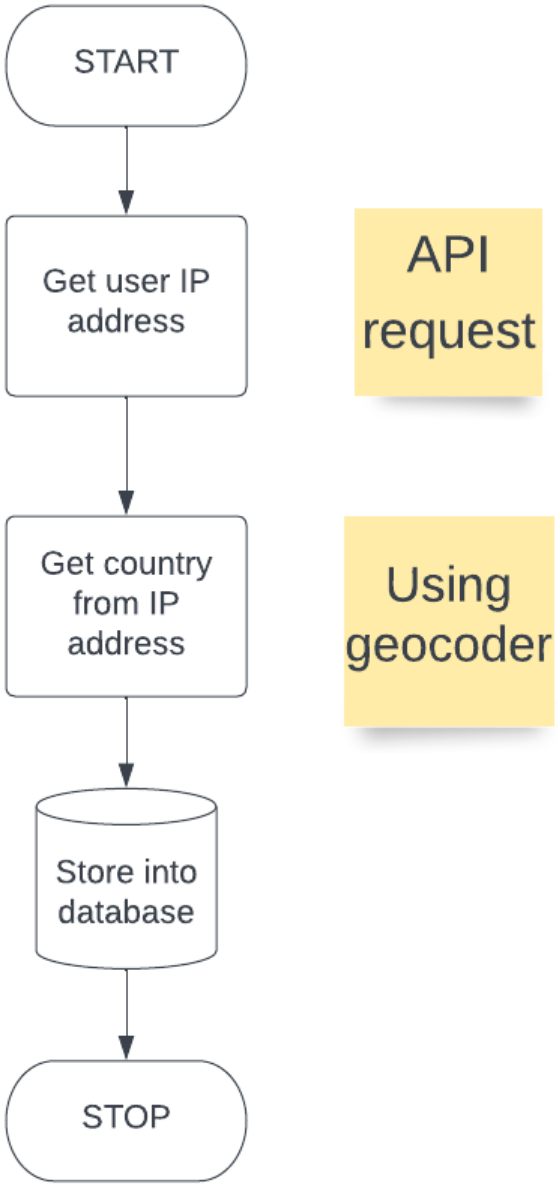
User Password Creation



This algorithm guides the user in creating a password. The user is first prompted to enter a password and then asked to repeat it correctly. If the user fails to do so after three attempts, they will be prompted to create a new password. Once the password is repeated correctly (confirmed), it is returned to be stored in the database along with other account data.

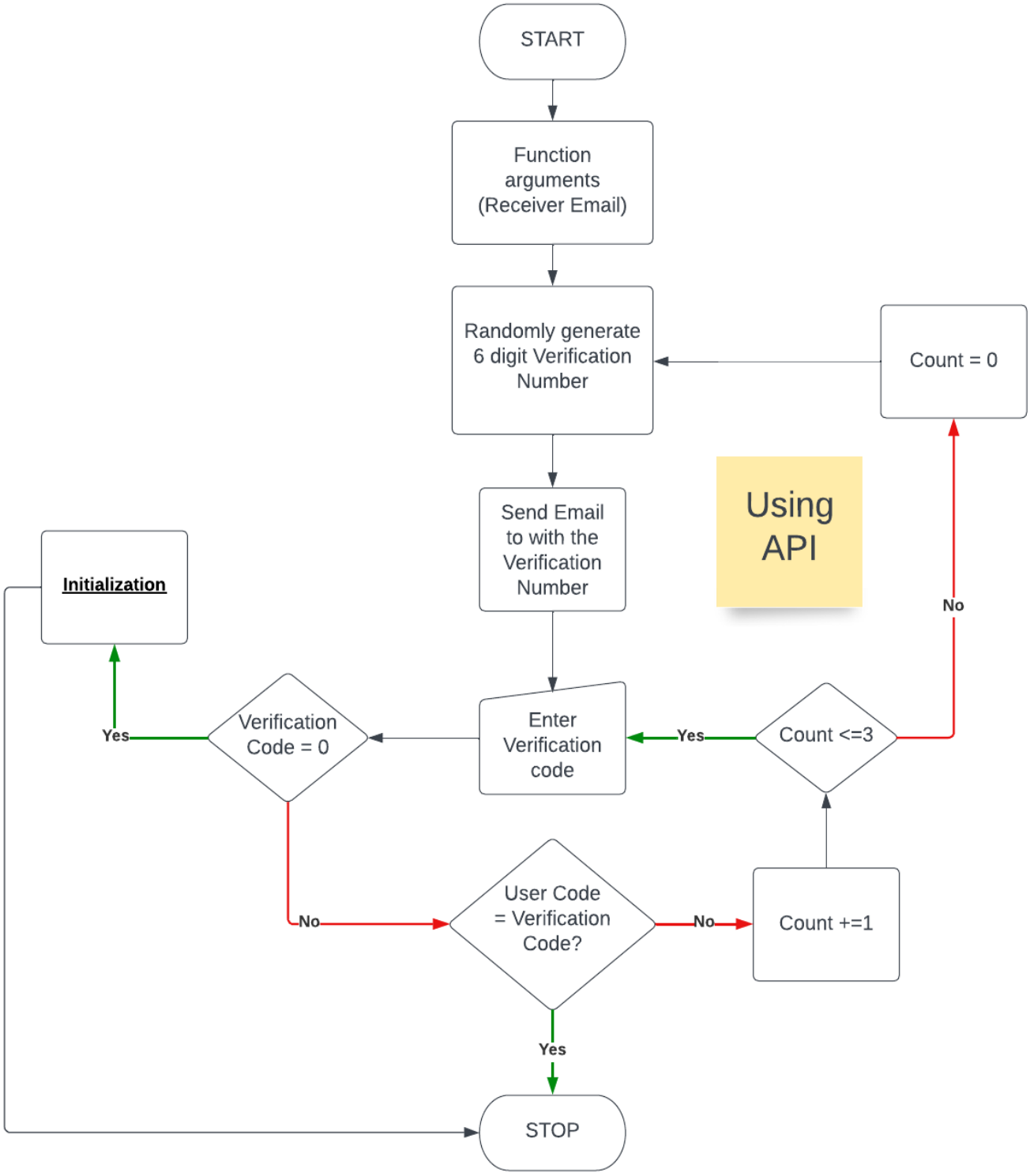
```
FUNCTION UserPasswordCreation()
    PasswordTrys ← 2
    UserPasswordInput1 ← INPUT Create password
    UserPasswordInput2 ← INPUT Password again

    IF UserPasswordInput1 ≠ UserPasswordInput2 THEN
        WHILE PasswordTrys > 0 AND UserPasswordInput1 ≠ UserPasswordInput2 DO
            PasswordTrys ← PasswordTrys - 1
            OUTPUT No match , PasswordTrys
            UserPasswordInput2 ← INPUT Repeat password

            IF PasswordTrys = 0 AND UserPasswordInput1 ≠ UserPasswordInput2 THEN
                OUTPUT Failed try again
                UserPasswordCreate()

            ELSE IF UserPasswordInput1 = UserPasswordInput2 THEN
                OUTPUT Saved
                RETURN UserPasswordInput1
            END IF
        END WHILE
    ELSE:
        OUTPUT Saved
        RETURN UserPasswordInput1
    END IF
END FUNCTION
```

Get IP Address



In this algorithm, the user's IP address is acquired through an API request that returns an IPv4 or IPv6 address. This address is then passed to a geocoder, which determines the user's country (e.g., UK, USA, France, etc.). This information about the user is then stored in the database.

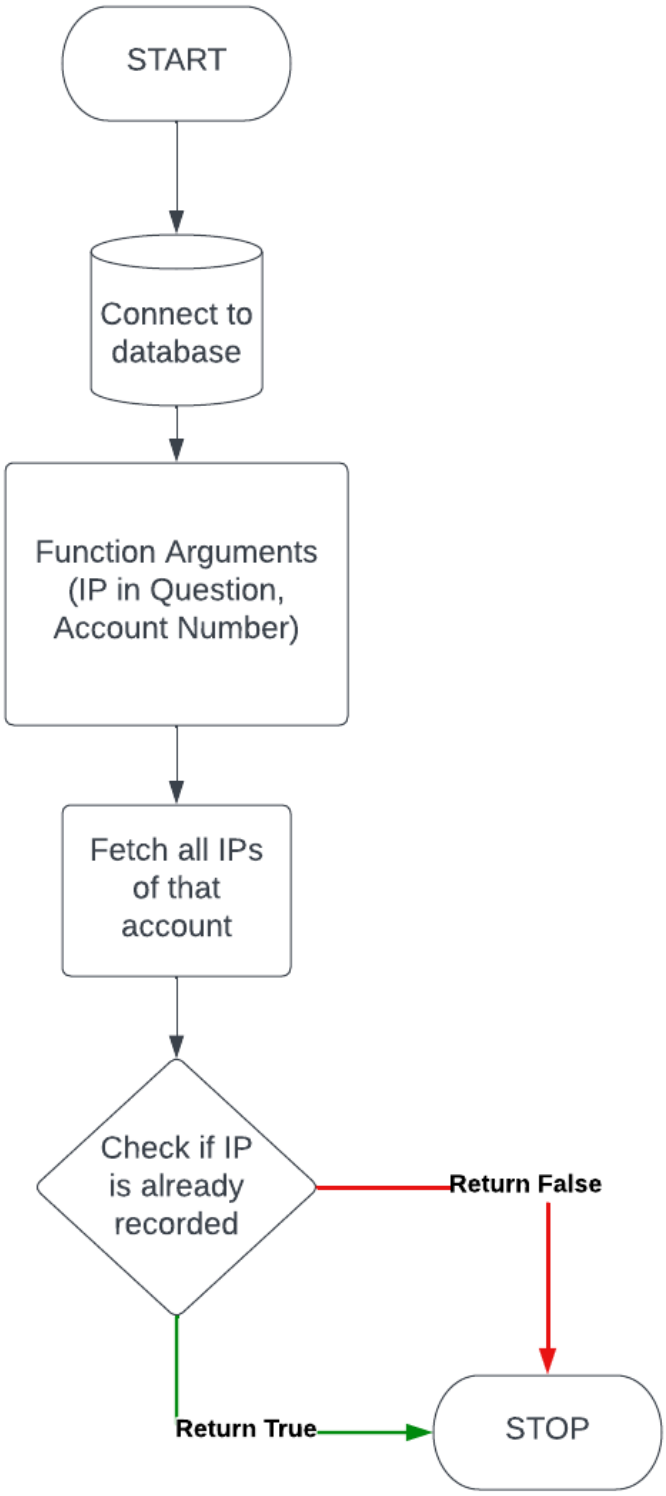
Email Verification



The main purpose of this algorithm is to create a verification code that is sent to the user's email to confirm their email address and identity. This is achieved by generating a random 6-digit number and sending it to the user's email address through an API. If the code is entered incorrectly more than three times, a new email with a new code will be sent. Moreover, if the user enters 0 as the verification code, the software will return to the start, prompting the user to choose the mode in which they want to run the program.

```
FUNCTION EmailVerification(ReceiverEmail)
    Code ← RANDOM_NUMBER_BETWEEN(100000 and 999999)
    send an email with the code
    UserTrys ← 3
    UserCode ← INPUT Verification code

    TRY
        UserCode ← CONVERT_TO_INTEGER(UserCode)
        IF UserCode = 0 THEN
            BACK TO START
        ELSE
            IF UserCode = Code THEN
                OUTPUT Finished
            ELSE
                WHILE Code ≠ UserCode AND UserTrys > 0 DO
                    UserCode ← INPUT Try again
                    UserTrys ← UserTrys - 1
                END WHILE

                IF Code = UserCode THEN
                    OUTPUT Finished
                ELSE
                    OUTPUT Send again
                    EmailCodeVerification(ReceiverEmail)
                END IF
            END IF
    EXCEPT
        OUTPUT Send again
        EmailCodeVerification(ReceiverEmail)
END FUNCTION
```

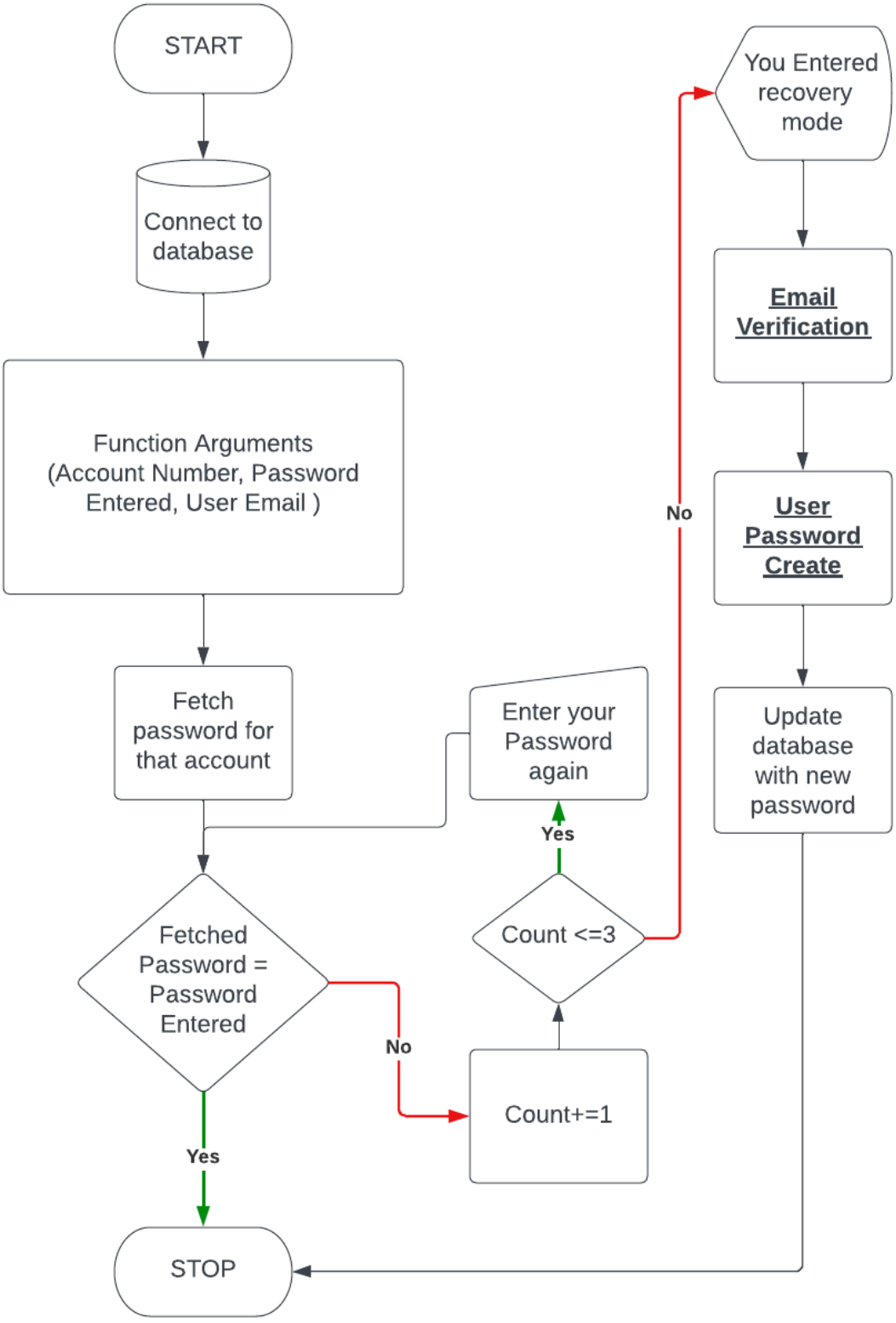
Check If IP Trusted



This algorithm aim is to check if the user has already logged in from the IP address they are currently using. This is achieved by storing all the user's IP address upon every successful login. The algorithm fetches all IP addresses the user has used since opening an account and checks if the current IP address has been used. If it has, 'true' is returned, and the user does not need to perform email verification to confirm the IP address.

```
FUNCTION CheckIfIPTrusted(AccountNumber, IPInQuestion)
    CONNECT to the database
    MainIP ← FETCH Main IP

    FUNCTION INOntherIps(AccountNumber, IPInQuestion)
        AllIps ← FETCH ALL IPs ever used with the account
        TRY
            FOR i IN AllIps DO
                IF AllIps[0][1] = AccountNumber THEN
                    RETURN True
                END IF
                AllIps ← AllIps[1:]
            END FOR
        EXCEPT
            RETURN False
    END FUNCTION

    IF INOntherIps(AccountNumber, IPInQuestion) = True THEN
        RETURN True
    ELSE
        RETURN False
    END IF
END FUNCTION
```

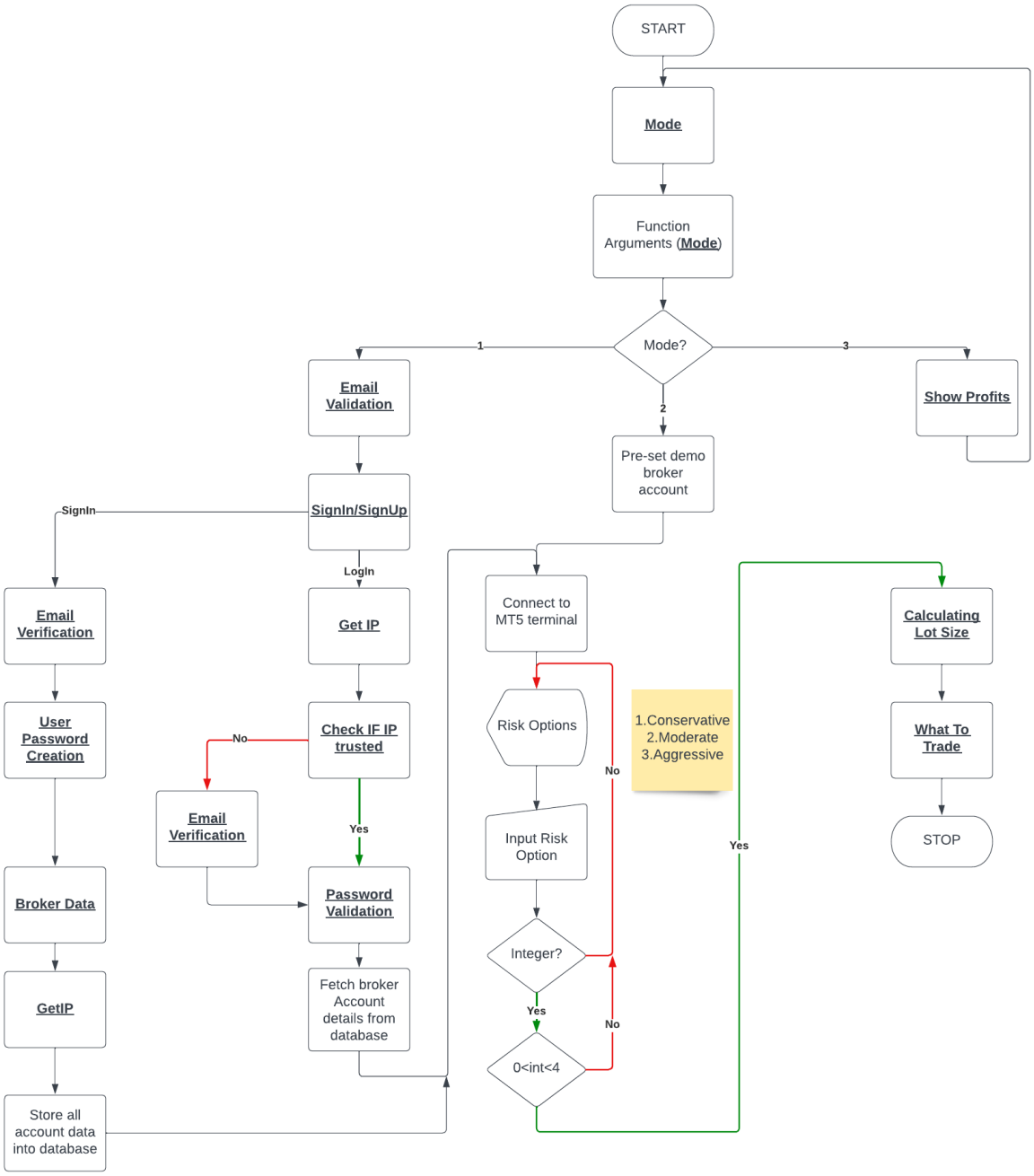
Password Validation



The purpose of this algorithm is to verify if the password entered by the user is correct. It connects to the database and fetches the password associated with the matching account ID (based on the email). If the fetched password and the one entered by the user don't match, the user has three attempts before entering recovery mode, which requires email verification and password reset. If the passwords match, the user is signed in with the credentials provided during sign-up.

```
FUNCTION PasswordValidation(AccountNumber, PasswordEnteredByUser, UserEMail)
    PasswordEnteredByUser ← CONVERT_TO_STRING(UserCode)

    CONNECT to the database

    PasswordDataBase ← FETCH Password where account number is matched
    NumberOfTrys ← 3

        WHILE PasswordEnteredByUser ≠ PasswordDataBase AND NumberOfTrys > 0 DO
            PasswordEnteredByUser ← INPUT Incorrect, try again
            NumberOfTrys ← NumberOfTrys - 1
        END WHILE

        IF NumberOfTrys = 0 AND (PasswordEnteredByUser ≠ PasswordDataBase) THEN
            OUTPUT Recovery mode
            Verify over email code
            Create new password
            UPDATE(NewUserPassword) into the database
        END IF

    RETURN True

END FUNCTION
```

Initialization



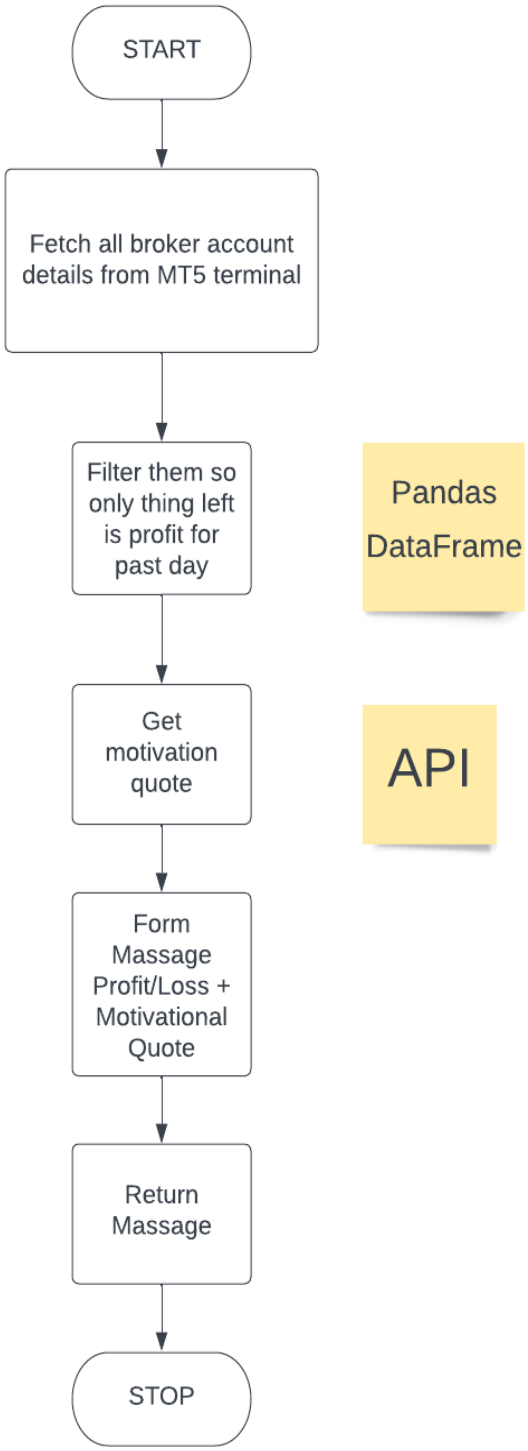
The primary objective of this algorithm is to integrate all the algorithms related to sign-in and sign-up into one comprehensive algorithm. It achieves this by first determining the mode (1/2/3) in which the user wants to run the program and then executing the appropriate steps. If mode 1 (sign-in/sign-up) is chosen, the email is validated, and it is determined whether the user is new or existing. If the user is new, email verification, broker data entry, and password creation are required, and the information is stored in the database for faster sign-in next time. If the user is not new, the IP address is fetched and checked for previous use. If the IP address hasn't been used before, the user must go through email verification and enter their password. Afterwards, account details are fetched from the database to ensure sign-in into the terminal. The user is prompted to choose a risk option, which must be a number within the specified range; otherwise, the user has to enter it again. Once this is completed, the lot size is calculated, and the user is asked to select the trading pair they want to trade. Mode 2 (demo) follows a similar process, but it proceeds directly to prompt the user to choose a risk option. Mode 3 (show profits) simply displays profits and prompts the user to choose a mode once again.

```
FUNCTION Initialization(ModeForAccount):
        NewAccount ← False
        Get Broker data

        IF ModeForAccount = 2 THEN
            Demo Account Details
            Demo ← True

        ELSE IF  ModeForAccount = 3 THEN
            display other profirs
            BACK to START

        ELSE IF ModeForAccount = 1 THEN
            AccountEmail← prompt email entry
            AccountID← find email in the database
            Demo ← False

            IF LogInSingIn(AccountEmail) = False THEN
                NewAccount ← True
                verify Emial
                get broker account data
                get IP
                get Location of IP

            ELSE
                IpTocheck ← get ip
                IpInDataBase ← Check if IN DataBase BOOL

                IF IpInDataBase = False THEN
                    OUTPUT Email Verification
                    verify Email

                PasswordForAccount← INPUT Account Password
                Validate Password

                IF IpInDataBase = False THEN
                    Store new IP in the DataBase

                FETCH Broker account details from the database
                END IF
            END IF
        END IF

        IF LogInMT5 not Succesfull THEN
            BACK to START
        END IF
        IF NewAccount = True THEN
            Store all the data (broker details, IP, password, Email, country) into the
database

        ELSE
            OUTPUT yey
        END IF
```

```
        AccountInfo ← fetch account info from MT5
        IF AccountInfo ≠ None THEN
            AccountBalance ← FETCH(account balance from AccountInfo)
            OUTPUT(AccountBalance)
            OUTPUT 1. Conservative
            OUTPUT 2. Moderate
            OUTPUT 3. Aggressive
            RiskSetting ← INPUT Risk option
            FUNCTION ValidateRisk(RiskSetting)
                TRY
                    RiskSettingINT← CONVERT_TO_INTEGER()
                    IF RiskSettingINT < 1 or RiskSettingINT > 3 THEN
                        WHILE RiskSettingINT < 1 or RiskSettingINT > 3 THEN
                            RiskSetting ← INPUT Valid Number
                            ValidateRisk(RiskSetting)
                        END WHILE
                    ELSE
                        RiskSettingToUse ← RiskSetting
                    END IF

                EXCEPT ValueError
                    RiskSetting ← INPUT Numeric value
                    ValidateRisk(RiskSetting)
            END FUNCTION
            ValidateRisk(RiskSetting)

        RETURN RiskSettingToUse, AccountEmail, AccountID, Demo, NewAccount
END FUNCTION
```

Send Profits

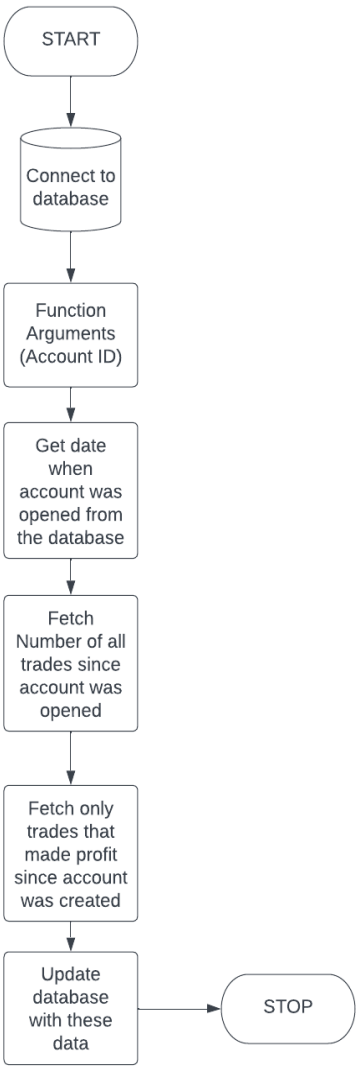
Profit For the Day



First, a message is generated using a motivational quote, and the profit made that day. This is achieved by fetching a motivational quote through an API request. Additionally, daily profit is fetched from the MT5 terminal by first obtaining all account data, which is then filtered and reshaped to calculate the day's profit. The message is then composed, combining the quote and daily profit, and sent using the send email function with the appropriate subject and content.

```
FUNCTION ProfitForDay()
    Quote ← get quote from an API
    Calculate time frame
    deals ← get deals from MT5

    IF deals = None THEN
        OUTPUT Nothing
    ELSE IF len(deals) > 0 THEN
        df ← CONVERT deals to DataFrame
        DayProfit ← filter and fetch profit in correct time frame
        IF DayProfit <= 0 THEN
            DayProfitMassage ← Profit massage with quote and profit
        ELSE
            DayProfitMassage ← Loss massage with quote and profit
        END IF
        RETURN DayProfitMassage
    END IF

END FUNCTION
```

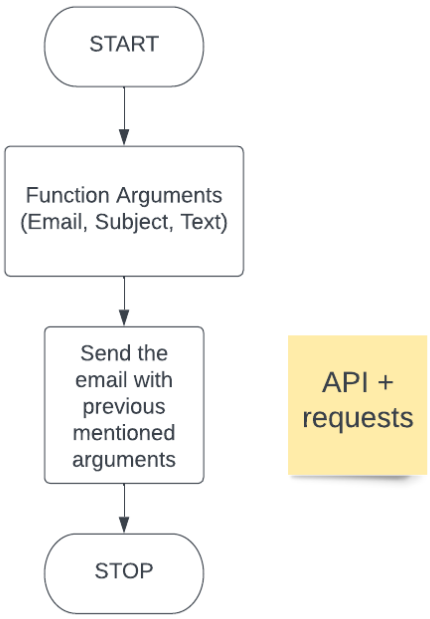
Number of Trades



This algorithm retrieves and records the number of trades taken and the number of profitable trades by the user. This is achieved by first connecting to the MT5 terminal, where trade and account data are fetched and filtered to obtain data for the correct time period. This data is then updated in the database.

```
FUNCTION NumberOfTrades(AccountID)
    OpenDate ← FETCH (Account open date)
    deals ← get deals from MT5 from OpenDate
    IF deals = None THEN
        OUTPUT Nothing
    ELSE IF len(deals) > 0 THEN
        df ← CONVERT deals to DataFrame
        NumberOfTrades ← Get number of trades
        NumberOfGoodTrades ← filter trades
        UPDATE the database with new info
    END IF
END FUNCTION
```

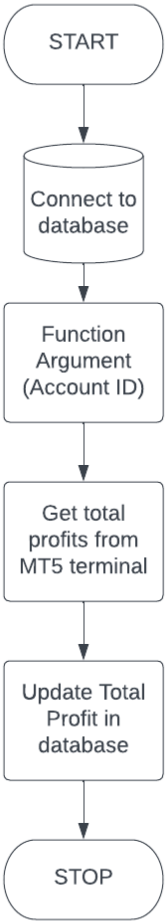
Send an Email



This algorithm sends an email to a specific email address with a custom subject and text, achieved through requests and an email-sending API.

```
FUNCTION SendAnEmail(Email,Subject,Text)
    TRY
        request ← Form request with function arguments
        send that request to an API to send an email
    EXCEPT
        OUTPUT Couldn't send an email
END FUNCTION
```

Total Profit

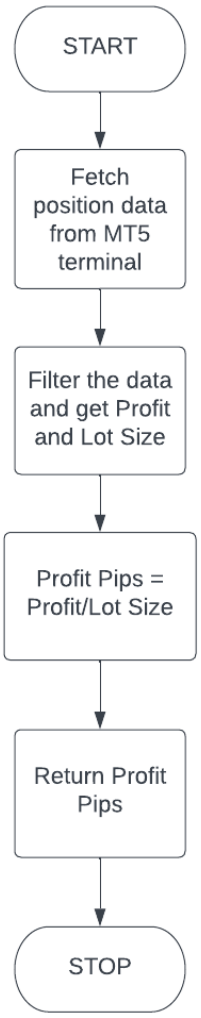


The flowchart above demonstrates how the total profits made by the user are recorded in the database. First, account data is fetched from the MT5 terminal and filtered to obtain profit. This number is then updated in the database.

```
FUNCTION TotalProfit(AccountID)
    Info ← get account info from MT5 starting from OpenDate
    TotalProfit ← filter info
    IF Info = None THEN
        OUTPUT Nothing
    ELSE
        UPDATE the database with total profit
    END IF
END FUNCTION
```

Orders

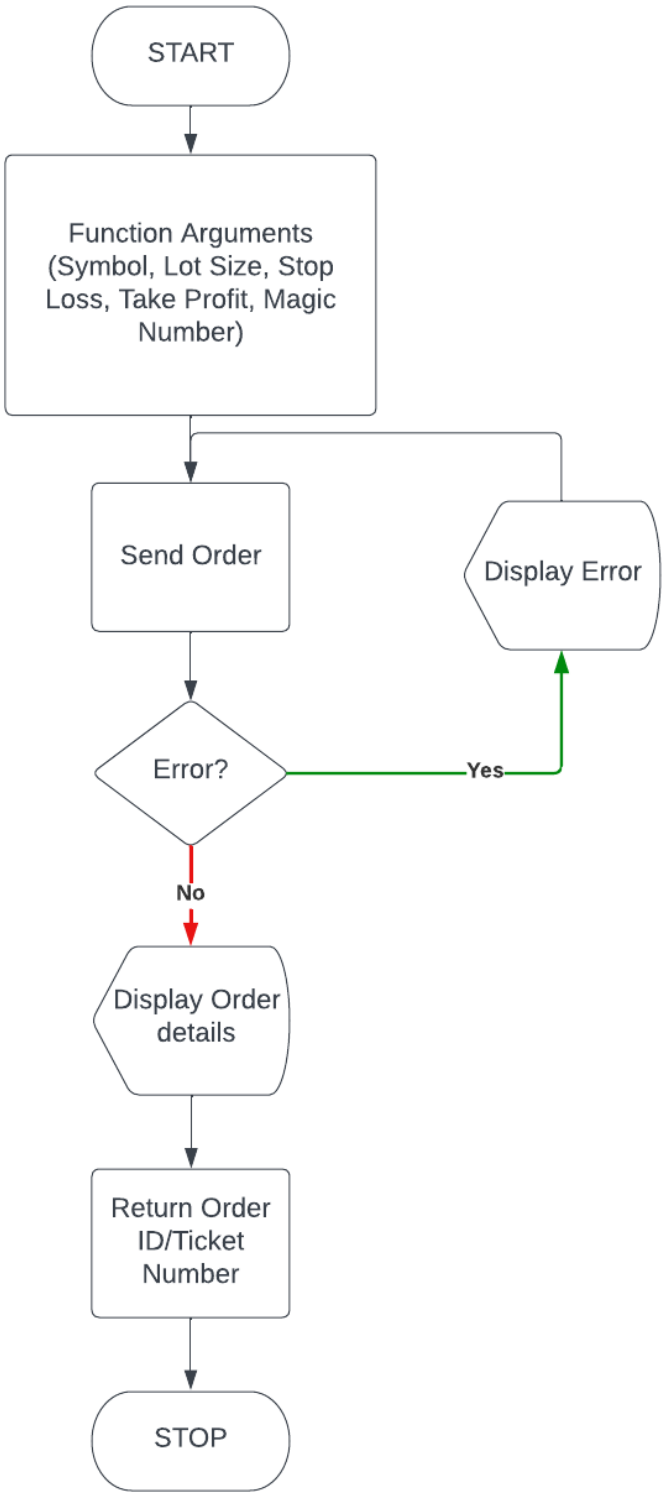
Profit in Pips



Initially, the algorithm fetches data about the specific position. Since this data is quite messy and contains a lot of information, it is first converted into pandas data frame to facilitate easier extraction of key information. After extracting the key information (current lot size and nominal profit), calculations are performed to convert it into pips, which are then returned for display when the trade is running.

```
FUNCTION CheckProfitPips(Symbol)
    Positions ← Check for Positions on symbol
    IF Positions = None THEN
        OUTPUT Nothing
    ELSE
        df= CONVERT positios into DataFrame
        LotSize= get LotSize from df
        Profit= get Prfoit from df
        LotSize= LotSize* 10
        PipsProfit= Profit/LotSize
        PipsProfit= round(PipsProfit,1)
        LotSize= round to 2 d.p.
        RETURN PipsProfit, LotSize
    END IF
END FUNCTION
```

Send Order

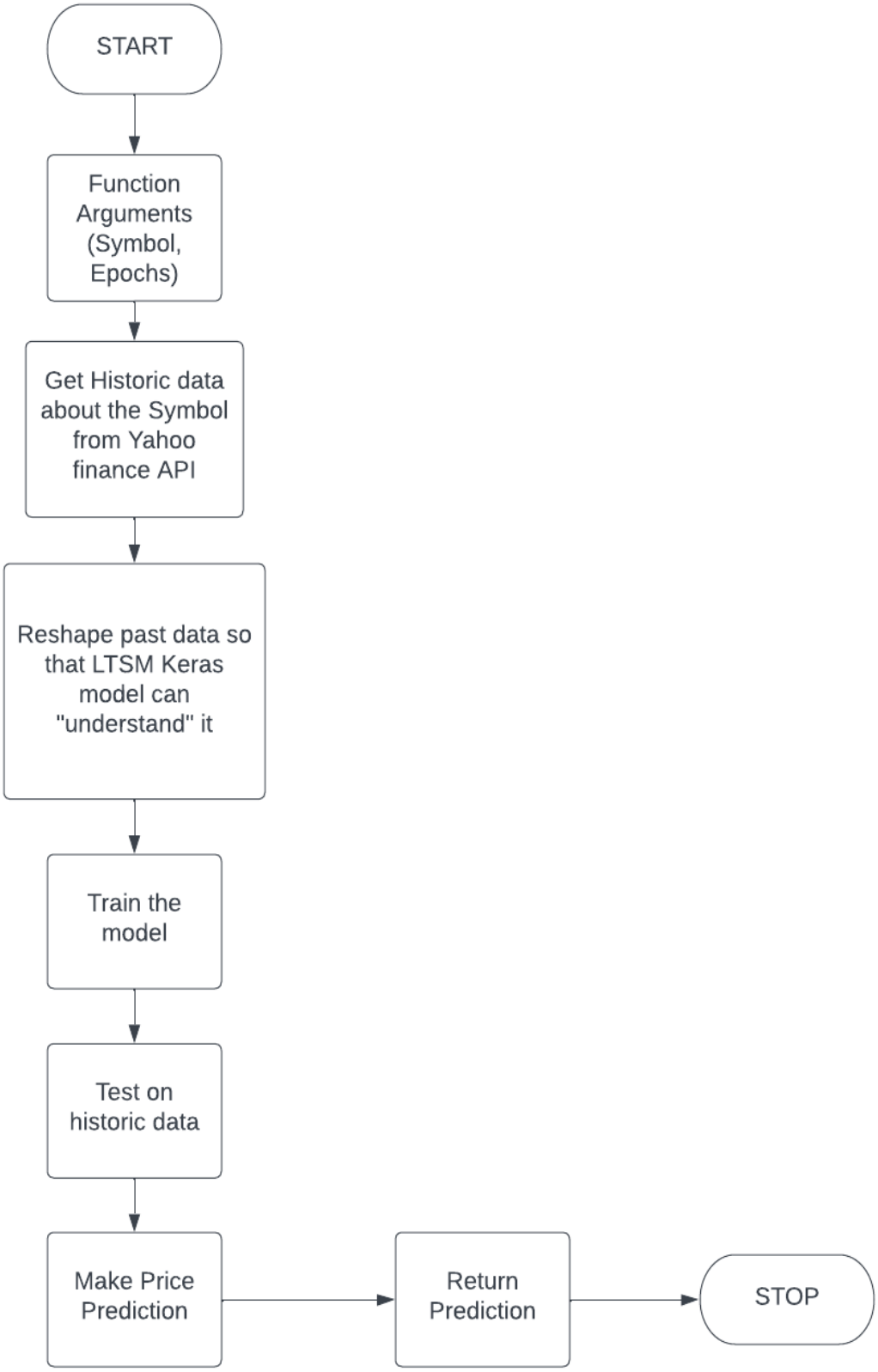


Parameters such as lot size, stop loss, take profit, symbol, and magic number are initially passed into a function to open a trade. If the order cannot be opened, the program retries until an error indicating too many requests occurs, which means the broker's spread is too large to open a trade at the correct price. It is worth noting that there are two functions: one for opening a sell and one for opening a buy, each with slightly different request structures for the terminal. Additionally, there are two other functions, based on the same principles, that close sell and buy positions by a specific amount. This implementation is necessary for grid trading, where small trade portions must be closed at different times.

```
FUNCTION openOrder(Symbol,LotSize,SL,TP,MagicNumber):
    Preapere Symbol
    point ← get point info for symbol
    price ← get ask price info for symbol / bid price for sell
    deviation ← 30

    Form request
    Order ← Send request

    IF Order = Error THEN
        OUTPUT Failed
        openOrder(Symbol,LotSize,SL,TP,MagicNumber)
    END IF

    OUTPUT Position data
    RETURN Order
END FUNCTION
```

Brain

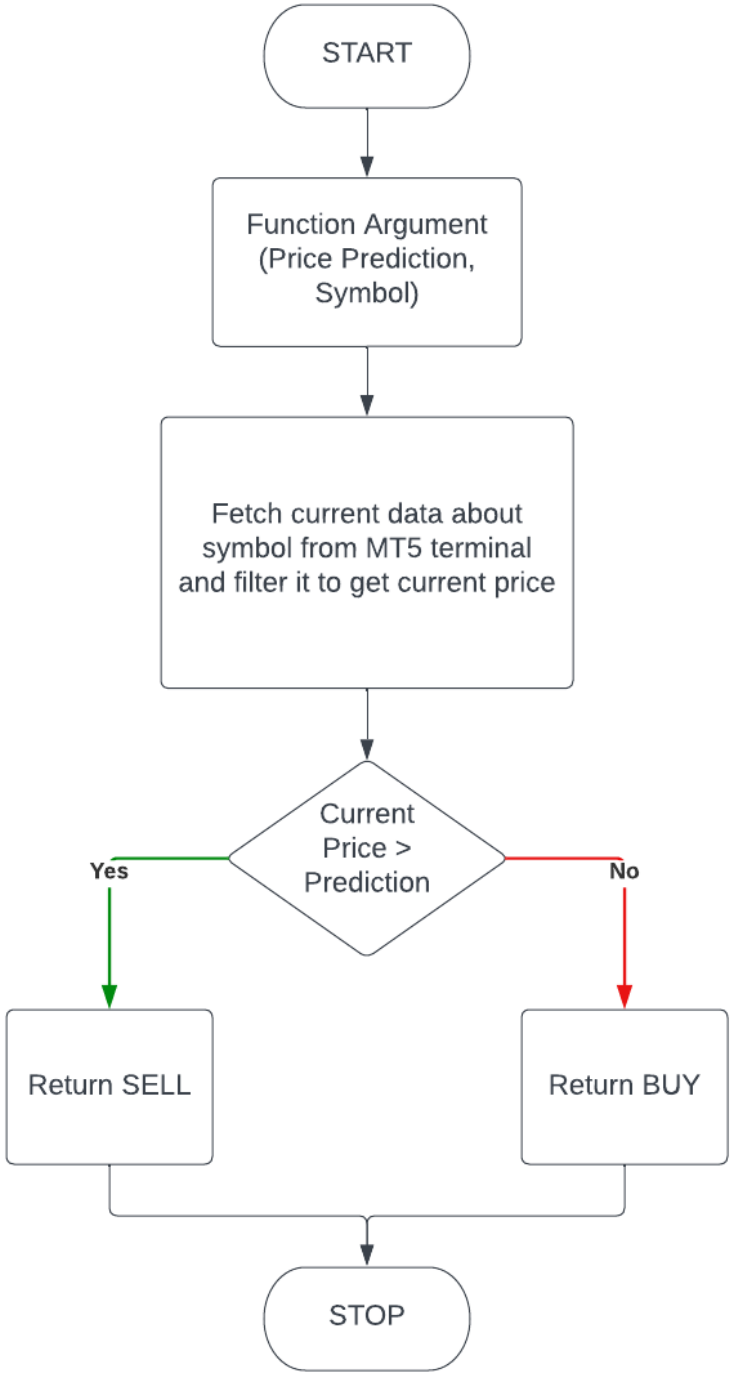
Prediction



Firstly, data is fetched from the Yahoo Finance API, which is then "reshaped" so that the Keras sequential model can understand and make an LTSM prediction. This is done by initially training the model on past data (2009-2022). After the training is complete, a test on historical data is performed to obtain a rough estimate of accuracy. Finally, the prediction is made, and the predicted price is returned.

```
FUNCTION TheBrain(TickerForPrediction,Epochs)
    Fetch historic data (since 2009) from Yahoo finace API
    Format data
    DaysToBasePredictionsOn ← 15
    FineseSettings (Epoch,Optimizer,BatchSize)
    Train the Model
    Test the Model
    Prediction ← Make Prediction for DaysToBasePredictionsOn
    RETURN Prediction
END FUNCTION
```

Direction



When making a prediction, only the price is returned. This function fetches the current price of the symbol from the terminal and compares it with the predicted price generated by the neural network. It then returns a buy or a sell signal, informing the program whether to open a buy or sell position.

```
FUNCTION Direction(PriceFromBrain,Symbol)
    CurrentPrice ← Get current price from MT5
    IF PriceFromBrain > CurrentPrice THEN
        RETURN BUY
    ELSE
        RETURN SELL
    END IF
END FUNCTION
```

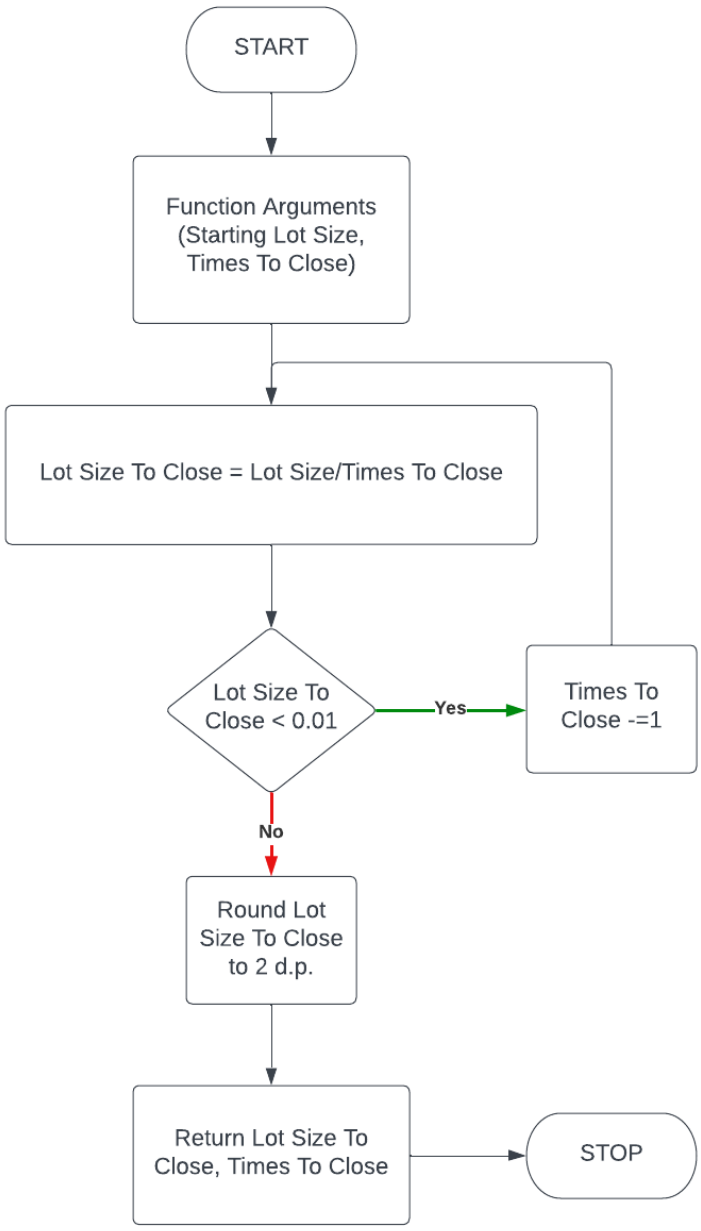
Grid

Firstly, the function determines whether to close a buy or sell position based on the type (buy or sell) passed into it. Afterwards, the algorithm checks if the "times to close" parameter is greater than 0 and then assesses the profit in pips. If the pips in profit are larger than the interval, a portion of the trade is closed. Subsequently, the interval is adjusted, and "times to close" is reduced by 1. If the trade hits stop loss or take profit before all portions of the trade are closed, or the lot size becomes too small (<0.01), or "times to close" reaches 0, the function returns False. After this either a new prediction will be made in demo mode or the software will wait for a new time of entry on a real account.

```
FUNCTION Grid(Ticket,Symbol,Type,TimesToClose,Interval,LotSize,MultiDirectional)
    TimesToClose ← Make adjustments to lot size and times to close
    NegativeInterval← 0 - Interval
    StartingInterval← Interval
    IF Type = BUY THEN
        WHILE True THEN
            TRY
                ProfitPips ← Calculate profit pips
                IF lot size to small OR end of the workingweek THEN
                    Close All Positions
                    BREAK
                ELSE
                    TRY
                        IF ProfitPips >= Interval THEN
                            TimesToClose -← 1
                            NegativeInterval ← NegativeInterval + StartingInterval
                            Interval ← Interval + StartingInterval
                            Close buy partially
                        ELSE IF ProfitPips <= NegativeInterval THEN
                            TimesToClose -← 1
                            NegativeInterval← NegativeInterval - StartingInterval
                            Interval ← Interval - StartingInterval
                            Close buy partially
                        END IF
                    EXCEPT
                        RETURN False
                END IF
            EXCEPT TypeError
                RETURN False
        END WHILE
        RETURN False

    ELSE IF Type = SELL THEN
        WHILE True THEN
            TRY
                ProfitPips ← Calculate profit pips
                IF lot size to small OR end of week THEN
                    Close All Positions
                    BREAK
                ELSE
                    TRY
                        IF ProfitPips >= Interval THEN
                            TimesToClose -← 1
                            NegativeInterval ← NegativeInterval + StartingInterval
                            Interval ← Interval + StartingInterval
```

```
                        Close sell partially
                    ELSE IF ProfitPips <= NegativeInterval THEN
                        TimesToClose -← 1
                        NegativeInterval ← NegativeInterval - StartingInterval
                        Interval ← Interval - StartingInterval
                        Close sell partially
                    END IF
                EXCEPT
                    RETURN False
            END IF
        EXCEPT TypeError
            RETURN False
    END WHILE
    RETURN False
END FUNCTION
```

Grid Adjustment Lot Size

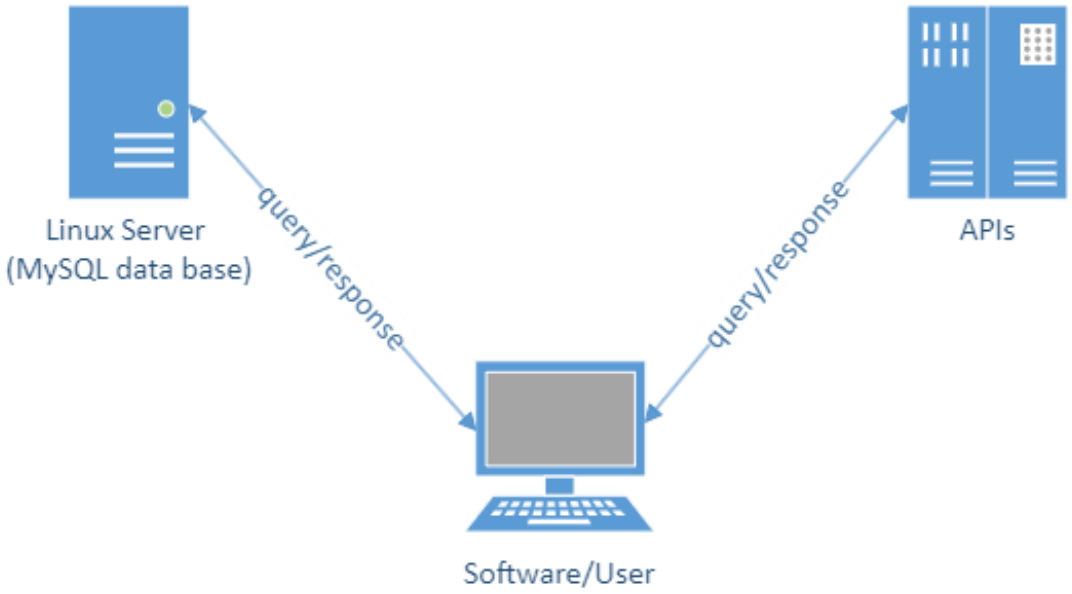


Since the initial lot size calculation is based on a single trade, it must be recalculated based on the desired number of times a trade should be closed while running. This must be done so that the lot size to close is not smaller than 0.01, as brokers do not support smaller lot sizes. The loop decreases "times to close" and recalculates the lot size to close until it is greater than 0.01. This value is then rounded to two decimal places, and both the lot size to close and "times to close" are returned. These values are then passed into the grid function to close profits accordingly.

```
FUNCTION GridAdjustClose(StartingLotSize,TimesToClose)
    LotSizeToClose ← StartingLotSize/TimesToClose
    IF LotSizeToClose < 0.01 THEN
        WHILE LotSizeToClose < 0.01 THEN
            TimesToClose ← TimesToClose -1
            LotSizeToClose ← StartingLotSize/TimesToClose
        END WHILE
    END IF
    RETURN TimesToClose
END FUNCTION
```

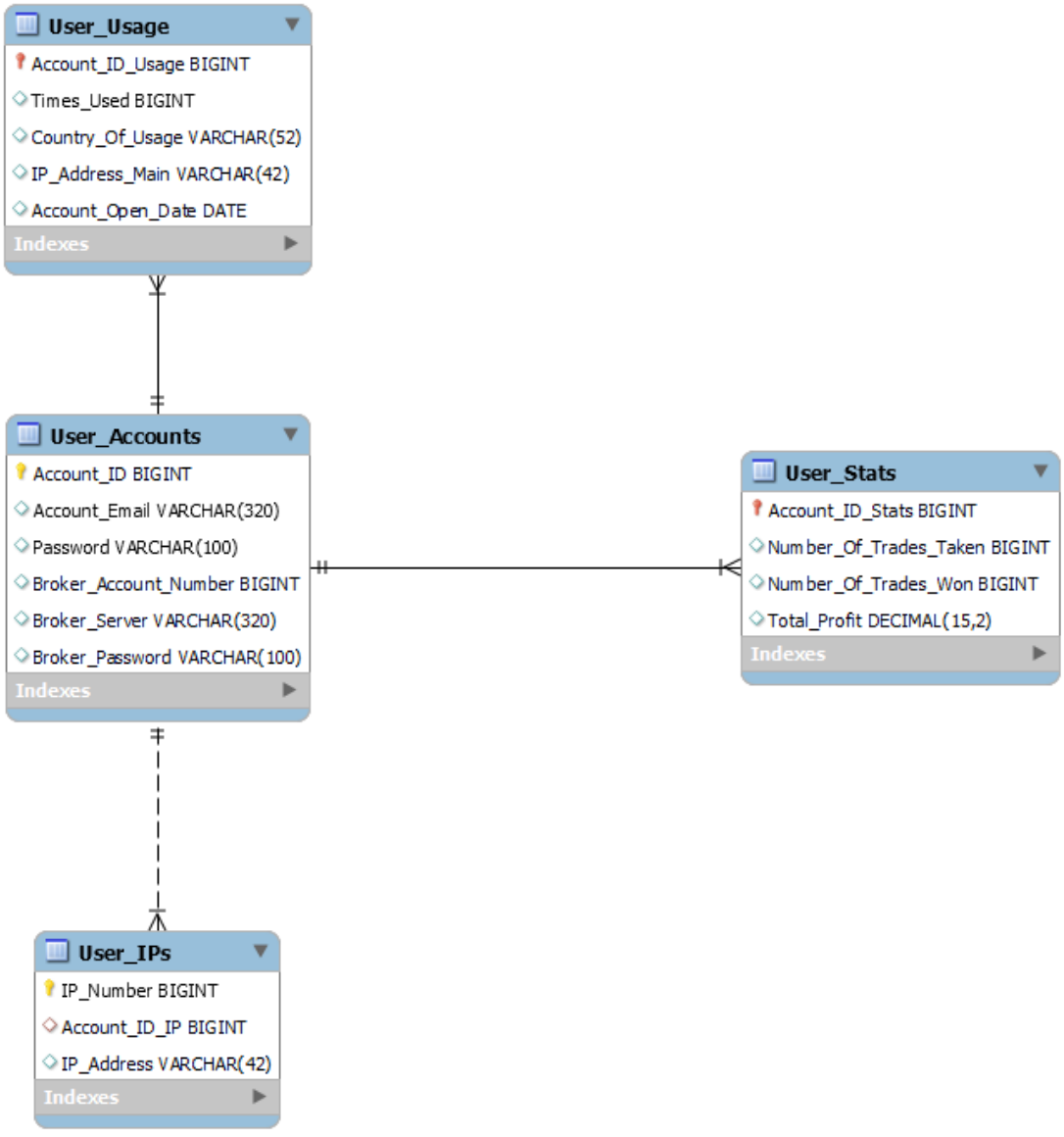
Database structure

Client Server Diagram



I have chosen to use a MySQL database to manage and store essential data. MySQL database will be hosted on a secure Linux server, ensuring optimal performance and reliability for very little cost.

Relational Database design



SQL Design for Key Functions

Updates total profit for a specific user.

```
UPDATE User_Stats
SET Total_Profit = '%s'
WHERE Account_ID_Stats IN (
  SELECT Account_ID
  FROM User_Accounts
  WHERE Account_ID = %s
);
```

Adds usage record for the user upon opening an account.

```
INSERT INTO User_Usage (Account_ID_Usage, Country_Of_Usage, IP_Address_Main,
Account_Open_Date)
VALUES (%s, %s, %s, %s);
```

Retrieves the user's account open date.

```
SELECT Account_Open_Date
FROM User_Usage
WHERE Account_ID_Usage IN (
  SELECT Account_ID
  FROM User_Accounts
  WHERE Account_ID = %s
);
```

Updates the trade count for the user.

```
UPDATE User_Stats
SET Number_Of_Trades_Taken = '%s'
WHERE Account_ID_Stats IN (
  SELECT Account_ID
  FROM User_Accounts
  WHERE Account_ID = %s
);
```

Updates trade wins for the user.

```
UPDATE User_Stats
SET Number_Of_Trades_Won = '%s'
WHERE Account_ID_Stats IN (
  SELECT Account_ID
  FROM User_Accounts
  WHERE Account_ID = %s
);
```

Adds new user IP record.

```
INSERT INTO User_IPs (Account_ID_IP, IP_Address)
VALUES (%s, %s);
```

Gets the user's broker information.

```
SELECT ua.Broker_Account_Number, ua.Broker_Server, ua.Broker_Password
FROM User_Accounts ua
WHERE ua.Account_ID = %s;
```

Increments how many times user used the software.

```
UPDATE User_Usage
SET Times_Used = Times_Used + 1
WHERE Account_ID_Usage IN (
  SELECT ua.Account_ID
  FROM User_Accounts ua
  WHERE ua.Account_ID = %s
);
```

Adds a new user account upon sign-up.

```
INSERT INTO User_Accounts (Account_Email, Password, Broker_Account_Number, Broker_Server,
Broker_Password)
VALUES (%s, %s, %s, %s, %s);
```

Gets the latest user account ID.

```
SELECT *
FROM User_Accounts
WHERE Account_ID = (
  SELECT MAX(Account_ID)
  FROM User_Accounts
);
```

Adds new user stats record.

```
INSERT INTO User_Stats (Account_ID_Stats)
VALUES (%s);
```

Retrieves all total profits.

```
SELECT Total_Profit
FROM User_Stats;
```

Retrieves the average of total profit.

```
SELECT AVG(Total_Profit)
FROM User_Stats;
```

Finds and retrieve users by email.

```
SELECT *
FROM User_Accounts
WHERE Account_Email LIKE %s;
```

Gets the main IP address for the user.

```
SELECT IP_Address_Main
FROM User_Usage
WHERE Account_ID_Usage = %s;
```

Finds if IP is already recorder in the databse as trusted by that user.

```
SELECT *
FROM User_IPs
WHERE IP_Address LIKE %s;
```

Gets the user's password.

```
SELECT Password
FROM User_Accounts
WHERE Account_ID = %s;
```

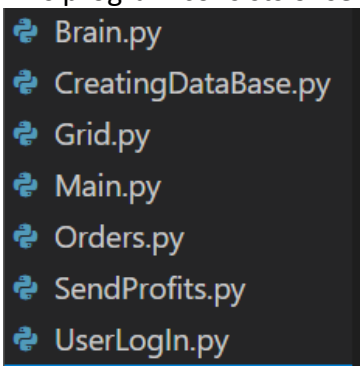
Updates the user's password.

```
UPDATE User_Accounts
SET Password = '%s'
WHERE Account_ID = %s;
```

Technical Solution

Breakdown of the program

This program consists of several files:



Main

```
from SendProfits import SendProfits
from datetime import datetime
from UserLogIn import LogIn
from Brain import BigBrain
from Orders import Orders
from Grid import Grid
import socket
import time

# Objective 2.
# Try's to connect to Google.com if failed that means computer doesn't have internet
connection
def ActiveWiFi():
    try:
        socket.create_connection(("Google.com",80))
        return True
    except OSError:
        return False

def Main():
    try:
        # Objective 2.b.
        # Asks user to establish internet connection if they don't have one
        while ActiveWiFi() == False:
            # Objective 2.a.
            print("Please establish an internet connection in order to continue... ")
            time.sleep(5)
            ActiveWiFi()

        # Database is created if it doesn't yet exists
        from CreatingDataBase import MakeDatabase
        MakeDatabase()

        # Initialization and choosing mode in which user wants to run the AI
        RiskSetting, AccountEmail, AccountID, Demo, NewAccount,
AccountNumber,Server,PasswordBroker = LogIn.InitializeMT5(LogIn.AccountMode())
        # User inputs pair they want to trade
```

```
        TickerForBrain= LogIn.PairForTraiding(AccountNumber,Server,PasswordBroker)
        # Ticker/pair name is modified so MT5 terminal as well as yahoo finance API can
understand it
        Ticker= LogIn.SymbolMod(TickerForBrain)

        def Traiding(RiskSetting, AccountEmail, AccountID, Demo, NewAccount,
AccountNumber,Server,PasswordBroker):
            DateTime= datetime.now()
            # Makes sure AI is not making a prediction when market is closed
            if DateTime.weekday() != (5 or 6):
                if Demo == False:
                    # Makes sure that prediction is made at 21 GMT after daily candle for
previous day has closed
                    if time.gmtime().tm_hour == 21 and time.gmtime().tm_min > 59:
                        # Calculates lot size of position based on risk setting selected
by user
                        LootSize = LogIn.CalculatingLotSize(RiskSetting,70)
                        if NewAccount == False:
                            # Objective 3.d.i.
                            # Gets total profit of the account and updates the database
                            SendProfits.TotalProfit(AccountID)
                            # Objective 8.
                            # Sends an email to the user if they have made any profit or
loss
                            SendProfits.SendAnEmail(AccountEmail,"Results for
Today",SendProfits.ProfitForDay())
                            # Objective 8.b.
                            # Gets total number of trades taken by user and updates the
database with it
                            SendProfits.NumberOfTrades(AccountID)

                        # Makes the price prediction
                        PricePrediction= BigBrain.TheBrain(TickerForBrain,3)
                        # Gets price prediction and turns it into a direction
                        direction= BigBrain.Direction(PricePrediction,Ticker)
                        # Opens BUY/SELL and starts to close down profit/loss depending on
settings of the "Grid"
                        if direction == "BUY":
                            Ticket= Orders.openBUY(Ticker,LootSize,70,100,0)
                            Grid(Ticket,Ticker,direction,100,1,LootSize,False)
                            Traiding(RiskSetting, AccountEmail, AccountID, Demo,
NewAccount, AccountNumber,Server,PasswordBroker)

                        elif direction == "SELL":
                            Ticket= Orders.openSELL(Ticker,LootSize,70,100,0)
                            Grid(Ticket,Ticker,direction,100,1,LootSize,False)
                            Traiding(RiskSetting, AccountEmail, AccountID, Demo,
NewAccount, AccountNumber,Server,PasswordBroker)

                    # Waits for right time to enter if all trades have been closed already
by hitting TP/SL
                    print("Waiting for time to Enter... ")
                    time.sleep(120)
```

```
                    Traiding(RiskSetting, AccountEmail, AccountID, Demo, NewAccount,
AccountNumber,Server,PasswordBroker)

                # In demo mode it runs the prediction instantly so user doesn't need to
wait for right time (21 GMT) in order to see how AI works
                # However it makes less accurate prediction as less training is involved
to ensure faster prediction
                elif Demo == True:
                    # Calculates lot size of position based on risk setting selected by
user
                    LootSize = LogIn.CalculatingLotSize(RiskSetting,70)
                    # Makes the price prediction
                    PricePrediction= BigBrain.TheBrain(TickerForBrain,3)
                    direction= BigBrain.Direction(PricePrediction,Ticker)

                    if direction == "BUY":
                        Ticket= Orders.openBUY(Ticker,LootSize,70,100,0)
                        print(Ticker,LootSize)
                        Grid(Ticket,Ticker,direction,100,0.1,LootSize,True)
                        ## Recursive algorithm (skill A)
                        Traiding(RiskSetting, AccountEmail, AccountID, Demo, NewAccount,
AccountNumber,Server,PasswordBroker)

                    elif direction == "SELL":
                        print(Ticker,LootSize)
                        Ticket= Orders.openSELL(Ticker,LootSize,70,100,0)
                        Grid(Ticket,Ticker,direction,100,0.1,LootSize,True)
                        ## Recursive algorithm (skill A)
                        Traiding(RiskSetting, AccountEmail, AccountID, Demo, NewAccount,
AccountNumber,Server,PasswordBroker)
                    Traiding(RiskSetting, AccountEmail, AccountID, Demo, NewAccount,
AccountNumber,Server,PasswordBroker)

            # If market is closed it waits and checks again if it has opened yet
            else:
                print("Waiting for market to open!")
                time.sleep(14400)
                ## Recursive algorithm (skill A)
                Traiding(RiskSetting, AccountEmail, AccountID, Demo, NewAccount,
AccountNumber,Server,PasswordBroker)

        ## Recursive algorithm (skill A)
        Traiding(RiskSetting, AccountEmail, AccountID, Demo, NewAccount,
AccountNumber,Server,PasswordBroker)

        # Objective 10.
    except:
        # If there is an error AI is stopped
        print("Sorry, seems like we bumped into a problem!")
        if ActiveWiFi() == False:
            print("One of the problems that we run into might be your internet connection
so please connect to the internet in order to continue... ")
            while ActiveWiFi() != True:
                print("Please establish an internet connection in order to continue... ")
```

```
                time.sleep(5)
                ActiveWiFi()
            Main()
        try:
            # Try's closing all positions
            Orders.CloseAllPositions()
            print("Don't worry, we have closed all trades for you")

        except:
            print("We couldn't close trades for you so if you have any open we suggest you
to close them manually")

        print("We couldn't locate the problem so please check back in few minutes or
restart the AI.")
        #Sends Email with AccountID so problem can be located and addressed if needed
        SendProfits.SendAnEmail("pythontradingbot0@gmail.com","User Error",AccountID)
        time.sleep(60)
        ## Recursive algorithm (skill A)
        Main()




if __name__ == "__main__":
    Main()
```

Creating Data Base

```
import mysql.connector

## Complex Client-Server model (skill A)
# MySQL databse server is run on a Linux machine that I have set up for the purpose of
this project
# Connection to database server
DataBase= mysql.connector.connect(
    host="139.162.199.24",
    user="Timko",
    password="Plemenitas1."
    )
mycursor= DataBase.cursor()

## 4 interlinked tables (skill A)
# Creates the Database called ForexTrader and tables User_Accounts and User_Stats thet are
inside if they don't yet exists
def MakeDatabase():
    global DataBase
    global mycursor
    mycursor.execute("CREATE DATABASE IF NOT EXISTS ForexTrader")
    DataBase.commit()
    mycursor.execute("CREATE TABLE IF NOT EXISTS ForexTrader.User_Accounts (Account_ID
bigint UNSIGNED AUTO_INCREMENT, PRIMARY KEY(Account_ID),Account_Email VARCHAR(320),
```

```
Password VARCHAR(100),Broker_Account_Number BIGINT UNSIGNED, Broker_Server VARCHAR(320),
Broker_Password VARCHAR(100))")
    DataBase.commit()
    mycursor.execute("CREATE TABLE IF NOT EXISTS ForexTrader.User_Stats (Account_ID_Stats
bigint UNSIGNED PRIMARY KEY, FOREIGN KEY(Account_ID_Stats) REFERENCES
User_Accounts(Account_ID), Number_Of_Trades_Taken BIGINT DEFAULT 0, Number_Of_Trades_Won
BIGINT DEFAULT 0, Total_Profit decimal(15,2) DEFAULT 0.0)")
    DataBase.commit()
    mycursor.execute("CREATE TABLE IF NOT EXISTS ForexTrader.User_Usage(Account_ID_Usage
bigint UNSIGNED PRIMARY KEY, FOREIGN KEY(Account_ID_Usage) REFERENCES
User_Accounts(Account_ID), Times_Used BIGINT DEFAULT 0, Country_Of_Usage VARCHAR(52),
IP_Address_Main VARCHAR(42), Account_Open_Date DATE)")
    DataBase.commit()
    mycursor.execute("CREATE TABLE IF NOT EXISTS ForexTrader.User_IPs(IP_Number bigint
UNSIGNED AUTO_INCREMENT, PRIMARY KEY(IP_Number), Account_ID_IP bigint UNSIGNED, FOREIGN
KEY(Account_ID_IP) REFERENCES User_Accounts(Account_ID), IP_Address VARCHAR(42))")
    DataBase.commit()
```

User LogIn

```
from SendProfits import SendProfits
from datetime import date
import MetaTrader5 as mt5
import mysql.connector
import pandas as pd
import requests
import geocoder
import random
import re


class LogIn ():
    # Makes user choose in which mode they would like to run the AI
    # OBJECTIVE 3)
    @staticmethod
    def AccountMode():
        print("\nWELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED
OPTIONS IN ORDER TO CONTINUE:")
        print("1. Connect to your Account OR Create one (Please note that in order to use
this function you have to have your own Broker Account)")
        print("2. One time DEMO use")
        print("3. Show Profits other users have made using this AI\n")
        Mode= input("Please enter a Number for the MODE you want to use: ")
        # Objective 3)a) Makes sure that user input is valid
        def ValidateMODEInput(Mode):
            try:
                ModeINT= int(Mode)
                if ModeINT < 1 or ModeINT > 3:
                    # Objective 3)a)i)
                    Mode = input("Please enter a valid Number in order to choose your
Mode: ")
                    # Objective 3)a)ii)
```

```
                    ## Recursive algorithm (skill A)
                    Mode= ValidateMODEInput(Mode)
                else:
                    return Mode

            except ValueError:
                # Objective 3)a)i)
                Mode = input("Please enter a Numeric value: ")
                # Objective 3)a)ii)
                ## Recursive algorithm (skill A)
                Mode= ValidateMODEInput(Mode)
            return Mode

        Mode= ValidateMODEInput(Mode)
        return int(Mode)

    # Connects to MT5 terminal based on Account mode input
    @staticmethod
    def InitializeMT5(ModeForAccount):
        NewAccount = False

        # Collects broker data to be stored and associatide with a user account
        def BrokerData():
            AccountNumber = input("Please enter number of your Broker account or press 0
to start again: ")

            # Evaluates so that AccountNumber is a number
            def ValidateAccountNumber(AccountNumber):
                try:
                    AccountNumber= int(AccountNumber)
                    return AccountNumber

                except ValueError:
                    AccountNumber= input("Please enter a numeric value for your Account
number: ")
                    ## Recursive algorithm (skill A)
                    AccountNumber= ValidateAccountNumber(AccountNumber)
                return AccountNumber

            # Gets Broker details
            AccountNumber= ValidateAccountNumber(AccountNumber)
            if AccountNumber == 0:
                ## Recursive algorithm (skill A)
                LogIn.InitializeMT5(LogIn.AccountMode())

            else:
                Server= input("Please Enter name of the server your Broker is connected
to: ")
                PasswordBroker= input("Please Enter your Broker password: ")

                # Objective 3.c.ii.1)d)
                if not mt5.initialize(login= AccountNumber, server= Server,password=
PasswordBroker):
                    print(" Initialization failed, error code =",mt5.last_error())
```

```
                print(" Sorry your Broker details are Invalid please try again.")
                ## Recursive algorithm (skill A)
                AccountNumber, Server, PasswordBroker= BrokerData()

            return AccountNumber, Server, PasswordBroker

    # Runs account in demo mode without need to log/sign in
    # Connection to Demo Broker account
    # Objective 3.b.ii.
    if ModeForAccount == 2:
        AccountNumber= 61784120
        Server= "MetaQuotes-Demo"
        PasswordBroker= "acl3knnt"
        AccountEmail = None
        AccountID = None
        Demo = True

    # Displays profits others have made
    # Objective 3.d.
    elif ModeForAccount == 3:
        LogIn.ShowOtherProfits()
        # Objective 3.d.iv.
        ## Recursive algorithm (skill A)
        LogIn.InitializeMT5(LogIn.AccountMode())


    # Objective 3.c.
    elif ModeForAccount == 1:
        AccountEmail= LogIn.EmailInput()
        AccountID= LogIn.LogInSingIn(AccountEmail)
        Demo = False

        # Check if user needs new account or if they are an existing user
        if AccountID == False:
            NewAccount = True
            # Objective 3.c.ii.1)
            LogIn.EmailCodeVerification(AccountEmail)
            # Objective 3.c.ii.1)b)
            PasswordToSave= LogIn.UserPasswordCreate()
            # Objective 3.c.ii.1)d)
            AccountNumber, Server, PasswordBroker = BrokerData()
            # Objective 3.c.ii.1)c)
            NewIp= LogIn.GetIp()
            NewLocation= LogIn.GetLocation(NewIp)

        else:
            IpTocheck= LogIn.GetIp()
            # Objective 3.c.ii.2)a)
            IpInDataBase= LogIn.DataBaseIP(AccountID,IpTocheck)

            # Check if IP is in database if not then it has to be confirmed#
            # Objective 3.c.ii.2)a)i)
            if IpInDataBase == False:
```

```
                    print("We can't recognise your IP so you will have to confirm that
this is you by Email Verification.")
                    LogIn.EmailCodeVerification(AccountEmail)

                # Objective 3.c.ii.2)b)
                PasswordForAccount= str(input("Please Enter your account Password: "))
                LogIn.PasswordValidation(AccountID,PasswordForAccount,AccountEmail)

                # Connection to database
                DataBase= mysql.connector.connect(
                host="139.162.199.24",
                user="Timko",
                password="Plemenitas1.",
                database= "ForexTrader"
                )
                mycursor= DataBase.cursor()
                AccountIDList= []
                AccountIDList.append(AccountID)
                if IpInDataBase == False:
                    mycursor.execute("INSERT INTO User_IPs(Account_ID_IP,IP_Address)
VALUES(%s,%s)",(AccountID,IpTocheck))
                    DataBase.commit()

                # Gets all the important info. from database about account that has
already been created so that it can be logged into mt5 terminal
                # Objective 3.c.ii.2)c)
                ## Cross table parameterised SQL (skill A)
                mycursor.execute("""
                                SELECT
                                    ua.Broker_Account_Number,
                                    ua.Broker_Server,
                                    ua.Broker_Password
                                FROM User_Accounts ua
                                WHERE ua.Account_ID = %s
                                """, (AccountIDList))
                Result = mycursor.fetchone()
                FinalAccountNumber = int(Result[0])
                Server = str(Result[1])
                PasswordBroker = str(Result[2])

                ## Cross table parameterised SQL (skill A)
                mycursor.execute("""
                                UPDATE User_Usage uu
                                SET uu.Times_Used = uu.Times_Used + 1
                                WHERE uu.Account_ID_Usage IN
                                (
                                    SELECT ua.Account_ID
                                    FROM User_Accounts ua
                                    WHERE ua.Account_ID = %s
                                )
                                """, (AccountIDList))
                DataBase.commit()
```

```
        if not mt5.initialize(login= AccountNumber, server= Server,password=
PasswordBroker):
            print(" Initialization failed, error code =",mt5.last_error())
            print(" Sorry something went wrong please try again.")
            LogIn.InitializeMT5(LogIn.AccountMode())

        # Connection to database
        DataBase= mysql.connector.connect(
        host="139.162.199.24",
        user="Timko",
        password="Plemenitas1.",
        database= "ForexTrader"
        )
        mycursor= DataBase.cursor()

        # Stores account details if new account has been created
        # Objective 3.c.ii.1)e)
        if NewAccount == True:
                mycursor.execute("INSERT INTO User_Accounts(Account_Email,Password,
Broker_Account_Number, Broker_Server, Broker_Password)
VALUES(%s,%s,%s,%s,%s)",(AccountEmail,PasswordToSave,FinalAccountNumber,Server,PasswordBro
ker))
                ## Aggregate SQL function (skill A)
                mycursor.execute("SELECT * FROM User_Accounts WHERE Account_ID=(SELECT
MAX(Account_ID) FROM User_Accounts)")
                LastID= (mycursor.fetchone()[0])
                Date= date.today()
                mycursor.execute("INSERT INTO User_Usage(Account_ID_Usage,
Country_Of_Usage, IP_Address_Main, Account_Open_Date)
VALUES(%s,%s,%s,%s)",(LastID,NewLocation,NewIp,Date))
                LastIDList= []
                LastIDList.append(LastID)
                mycursor.execute("INSERT INTO User_Stats(Account_ID_Stats)
VALUES(%s)",(LastIDList))
                DataBase.commit()
                print("\n Your Account has been Created, YEY!")

        else:
            print("\nYEY, you are IN!")

        # Objective 4.
        AccountInfo=mt5.account_info()
        if AccountInfo!=None:
            AccountInfoDictionary = mt5.account_info()._asdict()
            Balance=pd.DataFrame(list(AccountInfoDictionary.items()),columns=["property","
value"])
            Balance= float(round(Balance.iloc[10,1],2))
            print("\nYour Account Balance: ${}".format(Balance))
            print("Please choose one of below listed risk options in order to continue:")
            print("1. Conservative")
            print("2. Moderate")
            print("3. Aggressive")
            RiskSetting= input("Please enter a number for risk option you want to choose:
")
```

```
            def ValidateRisk(RiskSetting):
                try:
                    RiskSettingINT= int(RiskSetting)
                    if RiskSettingINT < 1 or RiskSettingINT > 3:
                        RiskSetting = input("Please enter a valid Number in order to
choose your Mode: ")
                        ## Recursive algorithm (skill A)
                        RiskSetting= ValidateRisk(RiskSetting)

                    else:
                        return RiskSetting

                except ValueError:
                    RiskSetting = input("Please enter a Numeric value: ")
                    ## Recursive algorithm (skill A)
                    RiskSetting= ValidateRisk(RiskSetting)

                return RiskSetting

            RiskSetting= ValidateRisk(RiskSetting)

        return RiskSetting, AccountEmail, AccountID, Demo, NewAccount,
AccountNumber,Server,PasswordBroker

    # Objective 4.
    # Calculates how much money is user willing to lose based on their risk setting and
account balance
    @staticmethod
    def CalculatingLotSize(RiskSetting,SL):
            AccountInfo=mt5.account_info()
            if AccountInfo!=None:
                ## Parsing complex data type (skill A)
                AccountInfoDictionary = mt5.account_info()._asdict()
                Balance=pd.DataFrame(list(AccountInfoDictionary.items()),columns=["propert
y","value"])
                Balance= int(round(Balance.iloc[10,1]))

            RiskSetting = int(RiskSetting)
            Risk = 0
            if RiskSetting == 1:
                Risk = 0.01
            elif RiskSetting == 2:
                Risk = 0.025
            elif RiskSetting == 3:
                Risk = 0.4

            BalanceRisk= Balance * Risk
            Lotsize= BalanceRisk / SL
            Lotsize= Lotsize / 10
            Lotsize= round(Lotsize,2)

            return Lotsize
```

```
    # Makes user input what they want to trade
    # Objective 5.
    @staticmethod
    def WhatToTradeInput(AccountNumber,Server,PasswordBroker):
        print("\nPlease choose one of options bellow or enter your own pair to trade.")
        print("Please note that this AI has been optimised for pairs listed bellow
therfore it produces the best results traiding those")
        # Objective 5.a.
        print("1. EURUSD")
        print("2. EURGBP")
        print("3. USDJPY")
        print("4. USDCAD")
        print("5. GBPJPY")
        print("Press 0 to Enter any other FOREX pair")

        PairToTrade= input("Please enter a number in order to choose which pair to trade:
")
        # Makes sure that pair is entered correctly as well as avaliable to trade
        def ValiadatePairToTrade(PairToTrade):

            try:
                PairToTradeINT= int(PairToTrade)
                if PairToTradeINT == 0:
                    # Objective 5.b.
                    def CustomPairToTrade():
                        CustomPair= input("Please enter a FOREX pair eg. EURGBP: ")

                        # Objective 5.b.i.
                        try:
                            mt5.initialize(login= AccountNumber, server= Server,password=
PasswordBroker)
                            symbol_info = mt5.symbol_info(CustomPair)
                            if symbol_info is None:
                                print("")
                                print(CustomPair, "not found, can not call order_check()")
                                print(CustomPair, "is not visible, trying to switch on")
                                if not mt5.symbol_select(CustomPair,True):
                                    print("symbol_select({}}) failed, exit",CustomPair)
                                    mt5.shutdown()
                                    quit()

                        except:
                            # Objective 5.b.ii.
                            print("Your broker doesn't support that pair please try
agian")
                            ## Recursive algorithm (skill A)
                            CustomPair=
LogIn.WhatToTradeInput(AccountNumber,Server,PasswordBroker)

                        print("Custom Pair",CustomPair)
                        return CustomPair

                    PairToTrade= CustomPairToTrade()
```

```
                else:
                    if PairToTradeINT < 1 or PairToTradeINT > 5:
                        PairToTrade = input("Please enter number between 1 and 5 or press
0 for entering a custom FOREX pair: ")
                        ## Recursive algorithm (skill A)
                        PairToTrade= ValiadatePairToTrade(PairToTrade)

                    else:
                        return PairToTrade

            except ValueError:
                PairToTrade = input("Please enter a numeric value between 1 and 5 or
preess 0 for a custom FOREX pair: ")
                ## Recursive algorithm (skill A)
                PairToTrade= ValiadatePairToTrade(PairToTrade)

            return PairToTrade

        PairToTrade= ValiadatePairToTrade(PairToTrade)
        print("")
        return PairToTrade

    # Returns a pair chosen with numbers in WhatToTradeInput so that prediction can be
based on
    @staticmethod
    def PairForTraiding(AccountNumber,Server,PasswordBroker):
        TraidingPair= LogIn.WhatToTradeInput(AccountNumber,Server,PasswordBroker)
        try:
            TraidingPair=int(TraidingPair)
            if TraidingPair == 1:
                return "EURUSD=X"
            if TraidingPair ==2:
                return "EURGBP=X"
            if TraidingPair == 3:
                return "USDJPY=X"
            if TraidingPair == 4:
                return "USDCAD=X"
            if TraidingPair == 5:
                return "GBPJPY=X"

        except ValueError:
            TraidingPair = TraidingPair+"=X"
            return TraidingPair

    # Shows how much profit have most succesful users made
    @staticmethod
    def ShowOtherProfits():
        # Connection to database
        DataBase= mysql.connector.connect(
        host="139.162.199.24",
        user="Timko",
        password="Plemenitas1.",
        database= "ForexTrader"
        )
```

```
        mycursor= DataBase.cursor()

        mycursor.execute("SELECT Total_Profit FROM User_Stats")
        AllProfitsTuple= mycursor.fetchall()
        ## Aggregate SQL function (skill A)
        mycursor.execute("SELECT AVG(Total_Profit) FROM User_Stats")
        AvgProfits= (mycursor.fetchone()[0])

        # Objective 3.d.iii.
        print("\nAvrage Profit of our Users in past 30 days
was:",round(float(AvgProfits),1),"Dollars")

        # Transforming a tuple to a list so that it can be sorted
        AllProfitsTuple= list(AllProfitsTuple)
        ProfitsList= []
        for SingleProfit in AllProfitsTuple:
            SingleProfit= str(SingleProfit)
            SingleProfit= SingleProfit[10:-4]
            SingleProfit = float(SingleProfit)
            ProfitsList.append(SingleProfit)

        # Merge sort algorithm to sort profits so that user can see how much other user
have made in past in order
        ## Merge sort (skill A)
        # Objective 3.d.ii.
        @staticmethod
        def SortProfits(ListToSort):
            if len(ListToSort) > 1:
                Middle = len(ListToSort) // 2
                LeftSide = ListToSort[:Middle]
                RightSide = ListToSort[Middle:]
                SortProfits(LeftSide)
                SortProfits(RightSide)
                t = 0
                f = 0
                m = 0

                while t < len(LeftSide) and f < len(RightSide):
                    if LeftSide[t] <= RightSide[f]:
                        ListToSort[m] = LeftSide[t]
                        t += 1

                    else:
                        ListToSort[m] = RightSide[f]
                        f += 1
                    m += 1

                while t < len(LeftSide):
                    ListToSort[m] = LeftSide[t]
                    t += 1
                    m += 1

                while f < len(RightSide):
                    ListToSort[m]=RightSide[f]
```

```
                    f += 1
                    m += 1


        SortProfits(ProfitsList)
        ProfitsToShow= 5
        print("While most succesful Users have Made: ")
        while ProfitsToShow > 0:
            print("$",ProfitsList[-1])
            ProfitsList.pop(-1)
            ProfitsToShow -=1
        print("In the Past 30 Days!")

    # Makes medication to the pair entered by user so mt5 can understand it
    @staticmethod
    def SymbolMod(Symbol):
        size= len(Symbol)
        if Symbol[-2:] == "=X":
            Symbol= Symbol[:size-2]
            return Symbol
        else:
            return Symbol

    # Checks if Email/Account already exists in database so that it know if user wants to
create account or just log in
    # Objective 3.c.ii.
    @staticmethod
    def LogInSingIn(EmailToCheck):
        # Connection to database
        DataBase= mysql.connector.connect(
        host="139.162.199.24",
        user="Timko",
        password="Plemenitas1.",
        database= "ForexTrader"
        )
        mycursor= DataBase.cursor()
        EmailToCheckList= []
        EmailToCheckList.append(EmailToCheck)
        # Fetch row if an Email already exits
        mycursor.execute("SELECT * FROM User_Accounts WHERE Account_Email LIKE %s
",(EmailToCheckList))
        EmailsTuple= mycursor.fetchall()
        # Returns Account ID if account already exits otherwise returns false so user can
create account
        try:
            IDNumber= (EmailsTuple[0][0])
            return IDNumber

        except IndexError:
            return False
```

```
    # Makes user enter their Email address
    # Objective 3.c.i.
    @staticmethod
    def EmailInput():
        Email= input("Please enter your Email address: ")
        # Checks so that Email entered is an Email
        def Check(EmailToCheck):
            ## Pattern matching regular expression (skill A)?
            EmailTemplate = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
            if(re.fullmatch(EmailTemplate, EmailToCheck)):
                Email = EmailToCheck
                return Email

            else:
                Email= input("Sorry your Email address seems to be Invalid please try
again: ")
                ## Recursive algorithm (skill A)
                Email= Check(Email)
            return Email

        Email= Check(Email)
        return Email

    # Asks user to create a password
    # Objective 3.c.ii.1)b)
    @staticmethod
    def UserPasswordCreate():
        PasswordTrys= 2
        UserPasswordInput1 = str(input("Please create your Password: "))
        UserPasswordInput2 = str(input("Please enter your Password again just make sure it
is correct:) "))

        # Objective 3.c.ii.1)b)i)
        if UserPasswordInput1 != UserPasswordInput2:
            while PasswordTrys > 0 and UserPasswordInput1 != UserPasswordInput2:
                PasswordTrys -= 1
                print("Your Passwords don't match please try again you
have",PasswordTrys,"try(s) left")
                UserPasswordInput2 = str(input("Please enter your Password again here: "))

                if PasswordTrys == 0 and UserPasswordInput1 != UserPasswordInput2:
                    print("You failed too many times Please try creating your password
again ")
                    # Objective 3.c.ii.1)b)ii)
                    ## Recursive algorithm (skill A)
                    LogIn.UserPasswordCreate()

                elif UserPasswordInput1 == UserPasswordInput2:
                    print("Thank You! Your Password has been saved.")
                    return UserPasswordInput1

        else:
            print("Thank You! Your Password has been saved.")
            return UserPasswordInput1
```

```
    # Gets IP address of user using API
    ## Use of web serviced API (skill A)
    @staticmethod
    def GetIp():
        print("IP")
        Response = requests.get('https://api64.ipify.org?format=json').json()
        print("IP over")
        return Response["ip"]

    # Gets location from IP address
    @staticmethod
    def GetLocation(IP):
        IP= geocoder.ip(IP)
        return IP.country

    # Generates and sends verification code via user Email address
    # Objective 3.c.ii.1)
    @staticmethod
    def EmailCodeVerification(ReceiverEmail):
        Code= random.randint(100000,999999)
        Massage= "Verification code: {}".format(Code)
        SendProfits.SendAnEmail(ReceiverEmail,"Email Verification",Massage)

        UserTrys= 3
        print("Code might slide into SPAM so please check there as well!")
        UserCode= input("Please Enter Verification code you recived in an Email or Press 0
to enter Email again: ")

        try:
            UserCode= int(UserCode)
            if UserCode == 0:
                # Objective 3.c.ii.1)a)
                LogIn.InitializeMT5(LogIn.AccountMode())

            else:
                if UserCode == Code:
                    return print(" YEY, Verification finished!")

                else:
                    while Code != UserCode and UserTrys > 0:
                        UserCode= int(input("Upps this isn't the right code Please try
Again: "))
                        UserTrys -=1

                    if Code == UserCode:
                        return print(" Verification finished")

                    else:
                        print("Your Email will be sent again")
                        ## Recursive algorithm (skill A)
                        LogIn.EmailCodeVerification(ReceiverEmail)
```

```
        except ValueError:
            ## Recursive algorithm (skill A)
            print("Your Email will be sent again")
            LogIn.EmailCodeVerification(ReceiverEmail)

    # Checks if IP that user is logging in with is trusted
    # Objective 3.c.ii.2)a)
    @staticmethod
    def DataBaseIP(AccountNumber,IPInQuestion):
        AccountNumberList= []
        AccountNumberList.append(AccountNumber)
        # Connection to database
        DataBase= mysql.connector.connect(
        host="139.162.199.24",
        user="Timko",
        password="Plemenitas1.",
        database= "ForexTrader"
        )
        mycursor= DataBase.cursor()
        mycursor.execute("SELECT IP_Address_Main FROM User_Usage WHERE Account_ID_Usage =
%s ",(AccountNumberList))
        IPAddressMain= mycursor.fetchall()
        IPAddressMain= IPAddressMain[0][0]

        def INOntherIps(AccountNumber,IPInQuestion):
            IPInQuestionList= []
            IPInQuestionList.append(IPInQuestion)
            mycursor.execute("SELECT * FROM User_IPs WHERE IP_Address LIKE
%s",(IPInQuestionList))
            AllIps= mycursor.fetchall()

            try:
                for i in AllIps:
                    if AllIps[0][1]== AccountNumber:
                        return True
                    AllIps= AllIps[1:]

            except IndexError:
                return False


        if IPInQuestion == IPAddressMain or INOntherIps(AccountNumber,IPInQuestion) ==
True:
            return True

        else:
            return False


    # Checks if password entered by user with account is correct for his account and if
not correct it can be changed by Email verification
    # Objective 3.c.ii.2)a)
    @staticmethod
    def PasswordValidation(AccountNumber,PasswordEnteredByUser,UserEMail):
```

```
        PasswordEnteredByUser=str(PasswordEnteredByUser)
        AccountNumberList= []
        AccountNumberList.append(AccountNumber)
        # Connection to database
        DataBase= mysql.connector.connect(
        host="139.162.199.24",
        user="Timko",
        password="Plemenitas1.",
        database= "ForexTrader"
        )
        mycursor= DataBase.cursor()
        # Fetch password
        mycursor.execute("SELECT Password FROM User_Accounts WHERE Account_ID =%s
",(AccountNumberList))
        PasswordDataBase= str(mycursor.fetchone()[0])
        NumberOfTrys= 3
        PasswordEnteredByUserMySql=  """'"""+ PasswordEnteredByUser + """'"""

        if (PasswordEnteredByUser or PasswordEnteredByUserMySql) == PasswordDataBase:
            print("")
            return True

        else:
            while ((PasswordEnteredByUser and PasswordEnteredByUserMySql) !=
PasswordDataBase) and NumberOfTrys > 0:
                PasswordEnteredByUser = str(input("Password was incorect please try again
you have {} trys left: ".format(NumberOfTrys)))
                PasswordEnteredByUserMySql=  """'"""+ PasswordEnteredByUser + """'"""
                if (PasswordEnteredByUser or PasswordEnteredByUserMySql) ==
PasswordDataBase:
                    break
                NumberOfTrys -=1

            # After 3 failed attempts user is asked to change password via Email
verification
            # Objective 3.c.ii.2)b)i)
            if NumberOfTrys == 0 and ((PasswordEnteredByUser and
PasswordEnteredByUserMySql) != PasswordDataBase):
                PasswordDataBaseList= []
                PasswordDataBaseList.append(PasswordDataBase)
                print("\nYou have now Enetered recovery mode")
                # Objective 3.c.ii.2)b)ii)
                LogIn.EmailCodeVerification(UserEMail)
                # Objective 3.c.ii.2)b)iii)
                NewUserPassword= LogIn.UserPasswordCreate()
                NewUserPasswordList=[]
                NewUserPasswordList.append(NewUserPassword)
                mycursor.execute("""UPDATE User_Accounts SET Password = "%s" WHERE
Account_ID = %s """,(NewUserPassword,AccountNumberList[-1]))
                DataBase.commit()

            print("")
            return True
```

Brain

```
from sklearn.preprocessing import MinMaxScaler
from keras.layers import Dense, Dropout, LSTM
from keras.models import Sequential
import matplotlib.pyplot as plt
from datetime import timedelta
import MetaTrader5 as mt5
import datetime as dt
import yfinance as yf
import pandas as pd
import numpy as np

## Advanced Matrix Operations and LTSM neural networks (skill A)
class BigBrain():
    # Objective 6.
    @staticmethod
    def TheBrain(TickerForPrediction,Epochs):

        # Loading historic data from Yahoo finance API from 1.1.2009 to today -30 days
        Start= dt.datetime(2009,1,1)
        End= dt.date.today()-timedelta(days=30)

        # Objective 6.a.
        print("Downloading past data")
        ## Web service API (skill A)
        APIData= yf.download(tickers = TickerForPrediction, start = Start ,end = End )
        print("")

        # Simplifying/formatting data so that python can understand it
        Scaler= MinMaxScaler(feature_range=(0,1))
        ScaledData= Scaler.fit_transform(APIData["Close"].values.reshape(-1,1))

        DaysToBasePredictionsOn= 15

        # Prepare the training data.
        x_train= []
        y_train= []

        for i in range(DaysToBasePredictionsOn, len(ScaledData)):
            x_train.append(ScaledData[i-DaysToBasePredictionsOn:i,0])
            y_train.append(ScaledData[i,0])


        # Convert the training data to NumPy arrays.
        x_train, y_train= np.array(x_train), np.array(y_train)
        x_train= np.reshape(x_train,(x_train.shape[0],x_train.shape[1],1))

        # Define and train the LSTM model
        Model= Sequential()

        Model.add(LSTM(units=150, return_sequences= True,
input_shape=(x_train.shape[1],1)))
        Model.add(Dropout(0.2))
```

```
        Model.add(LSTM(units=150))
        Model.add(Dropout(0.2))

        Model.add(Dense(units=1))

        # Finese settings
        Model.compile(optimizer="adam",loss="mean_squared_error")
        Model.fit(x_train,y_train, epochs=Epochs, batch_size=64)

        # Prepare the test data.
        TestStart= dt.datetime(2022,1,1)
        TestEnd= dt.date.today()-timedelta(days=15)

        TestData= yf.download(tickers = TickerForPrediction, start = TestStart ,end =
TestEnd )


        TotalDataSet= pd.concat((APIData["Close"], TestData["Close"]), axis= 0)

        # Prepare the inputs for the model
        ModelInputs = TotalDataSet[len(TotalDataSet)- len(TestData)-
DaysToBasePredictionsOn:].values
        ModelInputs = ModelInputs.reshape(-1,1)
        ModelInputs= Scaler.transform(ModelInputs)

        # Making predictions on test data
        x_test= []
        for x in range(DaysToBasePredictionsOn,len(ModelInputs)):
            x_test.append(ModelInputs[x- DaysToBasePredictionsOn:x, 0])

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

        PricePrediction= Model.predict(x_test)
        PricePrediction= Scaler.inverse_transform(PricePrediction)

        # Actual prediction
        RealData= [ModelInputs[len(ModelInputs)+1 -
DaysToBasePredictionsOn:len(ModelInputs+1),0]]
        RealData= np.array(RealData)
        RealData= np.reshape(RealData,(RealData.shape[0], RealData.shape[1],1))

        Prediction= Model.predict(RealData)
        Prediction= Scaler.inverse_transform(Prediction)
        return Prediction



    # Gives either a sell or buy signal based on current and predicted price
    @staticmethod
    def Direction(PriceFromBrain,Symbol):

        # Trys selecting symbol (checks if Broker supports it)
```

```
        Selected=mt5.symbol_select(Symbol,True)
        if not Selected:
            print("Failed to select ",Symbol)
            print("Your Broker does not support this pair Please run the AI again with
another pair")

        # Gets current price from MT5
        LastTick=mt5.symbol_info_tick(Symbol)
        CurrentPrice= float((list(LastTick)[1]))


        # Compares current and predicted price
        # If predicted price is higher then BUY is returned otherwise SELL
        if PriceFromBrain > CurrentPrice:
            return "BUY"
        else:
            return "SELL"
```

Grid

```
from Orders import Orders
from datetime import datetime
import time

# Makes sure that position is not to small to be closed
# It is adjusted if it is to small
def GridAdjustClose(StartingLotSize,TimesToClose):
    LotSizeToClose= StartingLotSize/TimesToClose
    if LotSizeToClose < 0.01:
        print("Lot Size to small! Adjusting")
        while LotSizeToClose < 0.01:
            TimesToClose = TimesToClose -1
            LotSizeToClose= StartingLotSize/TimesToClose

    LotSizeToClose= round(LotSizeToClose,2)
    return TimesToClose, LotSizeToClose

# Objective 7
# It is closing small parts of a trade no matter the direction market is moving returns
false if trade has been closed
def Grid(Ticket,Symbol,Type,TimesToClose,Interval,LotSize,MultiDirectional):
    DateTime= datetime.now()
    TimesToClose, LotSizeToClose = GridAdjustClose(LotSize,TimesToClose)
    NegativeInterval= 0 - Interval
    StartingInterval= Interval
    print("")
    # Closing no matter in profit or loss
    if MultiDirectional == True:
        if Type == "BUY":
            while True:
                try:
                    # Calculates and shows profit running in pips
                    ProfitPips= (Orders.CheckProfitPips(Symbol)[0])
```

```
                    print("Profit in Pips",ProfitPips,end= '\r')

                    # Checks if trade lotsize is big enough to be closed if not closes
whole trade(s)
                    # Closes all trades at GMT 21 so new prediction can be made
                    if ((Orders.CheckProfitPips(Symbol)[1]) < LotSizeToClose) or
TimesToClose <= 0 or time.gmtime().tm_hour == 21 or (DateTime.weekday() == 5 and
time.gmtime().tm_hour == 19):
                        Orders.CloseAllPositions()
                        print("All trades Closed")
                        break

                    else:
                        try:
                            # Closing small parts of the trade if interval in pips is
bigger than the interval set
                            if ProfitPips >= Interval:
                                TimesToClose -= 1
                                NegativeInterval = NegativeInterval + StartingInterval
                                Interval= Interval + StartingInterval
                                Orders.closeBUY(Symbol,LotSizeToClose,Ticket,1)

                            elif ProfitPips <= NegativeInterval:
                                TimesToClose -= 1
                                NegativeInterval= NegativeInterval - StartingInterval
                                Interval = Interval - StartingInterval
                                Orders.closeBUY(Symbol,LotSizeToClose,Ticket,1)
                        except:
                            return False
                except TypeError:
                    return False
            return False


        elif Type == "SELL":
            while True:
                try:

                    # Calculates and shows profit running in pips
                    ProfitPips= (Orders.CheckProfitPips(Symbol)[0])
                    print("Profit in Pips",ProfitPips,end= '\r')

                    if ((Orders.CheckProfitPips(Symbol)[1]) < LotSizeToClose) or
TimesToClose <= 0 or time.gmtime().tm_hour == 21 or (DateTime.weekday() == 5 and
time.gmtime().tm_hour == 19):
                        Orders.CloseAllPositions()
                        print("All trades Closed")
                        break

                    else:
                        try:
                            #  Closing small parts of the trade if interval in pips is
bigger than the interval set
                            if ProfitPips >= Interval:
```

```
                            TimesToClose -= 1
                            NegativeInterval = NegativeInterval + StartingInterval
                            Interval= Interval + StartingInterval
                            Orders.closeSELL(Symbol,LotSizeToClose,Ticket,1)

                        elif ProfitPips <= NegativeInterval:
                            TimesToClose -= 1
                            NegativeInterval= NegativeInterval - StartingInterval
                            Interval = Interval - StartingInterval
                            Orders.closeSELL(Symbol,LotSizeToClose,Ticket,1)
                    except:
                        return False
                except TypeError:
                    return False
            return False



    # Closing only when in profit
    elif MultiDirectional == False:
        if Type == "BUY":
            while True:
                try:
                    # Calculates and shows profit running in pips
                    ProfitPips= (Orders.CheckProfitPips(Symbol)[0])
                    print("Profit in Pips",ProfitPips,end= '\r')

                    if (Orders.CheckProfitPips(Symbol)[1]) < LotSizeToClose or
TimesToClose <= 0 or time.gmtime().tm_hour == 21 or (DateTime.weekday() == 5 and
time.gmtime().tm_hour == 19):
                        Orders.CloseAllPositions()
                        print("All trades Closed")
                        break

                    else:
                        try:
                            #  Closing small parts of the trade if interval in pips is
bigger than the interval set
                            if ProfitPips >= Interval:
                                TimesToClose -= 1
                                NegativeInterval = NegativeInterval + StartingInterval
                                Interval= Interval + StartingInterval
                                Orders.closeBUY(Symbol,LotSizeToClose,Ticket,1)

                            elif ProfitPips > 0 and ProfitPips <= NegativeInterval:
                                TimesToClose -= 1
                                NegativeInterval = NegativeInterval - StartingInterval
                                Interval= Interval - StartingInterval
                                Orders.closeBUY(Symbol,LotSizeToClose,Ticket,1)

                        except:
                            return False
                except TypeError:
                    return False
```

```
            return False

        elif Type == "SELL":
            while True:
                try:
                    # Calculates and shows profit running in pips
                    ProfitPips= (Orders.CheckProfitPips(Symbol)[0])
                    print("Profit in Pips",ProfitPips,end= '\r')


                    if (Orders.CheckProfitPips(Symbol)[1]) < LotSizeToClose or
TimesToClose <= 0 or time.gmtime().tm_hour == 21 or (DateTime.weekday() == 5 and
time.gmtime().tm_hour == 19):
                        Orders.CloseAllPositions()
                        print("All trades Closed")
                        break

                    else:
                        try:
                            # Closing small parts of the trade if interval in pips is
bigger than the interval set
                            if ProfitPips >= Interval:
                                TimesToClose -= 1
                                NegativeInterval = NegativeInterval + StartingInterval
                                Interval= Interval + StartingInterval
                                Orders.closeSELL(Symbol,LotSizeToClose,Ticket,1)

                            elif ProfitPips > 0 and ProfitPips <= NegativeInterval:
                                TimesToClose -= 1
                                NegativeInterval= NegativeInterval - StartingInterval
                                Interval = Interval - StartingInterval
                                Orders.closeSELL(Symbol,LotSizeToClose,Ticket,1)
                        except:
                            return False
                except TypeError:
                    return False
            return False
```

Orders

```
import MetaTrader5 as mt5
import pandas as pd
import time


class Orders():
    # Checks whether or not symbol is avaliable to trade on mt5 platform
    @staticmethod
    def PreapereSymbol(SymbolToPreapere):
        symbol_info = mt5.symbol_info(SymbolToPreapere)
        if symbol_info is None:
            print(SymbolToPreapere, "not found, can not call order_check()")
            mt5.shutdown()

        if not symbol_info.visible:
            print(SymbolToPreapere, "is not visible, trying to switch on")
            if not mt5.symbol_select(SymbolToPreapere,True):
                print("symbol_select({}}) failed, exit",SymbolToPreapere)
                mt5.shutdown()
                quit()

    # Opens a buy order
    @staticmethod
    def openBUY(Symbol,LotSize,SL,TP,MagicNumber):

        Orders.PreapereSymbol(Symbol)

        # Get symbol info
        point = mt5.symbol_info(Symbol).point
        price = mt5.symbol_info_tick(Symbol).ask
        deviation = 30

        request = {
        "action": mt5.TRADE_ACTION_DEAL,
        "symbol": Symbol,
        "volume": LotSize,
        "type": mt5.ORDER_TYPE_BUY,
        "price": price,
        "sl": price - SL * point * 10,
        "tp": price + TP * point * 10,
        "deviation": deviation,
        "magic": MagicNumber,
        "comment": "Python script open",
        "type_time": mt5.ORDER_TIME_DAY,
        "type_filling": mt5.ORDER_FILLING_RETURN
        }

        # Sends the order
        Order= mt5.order_send(request)

        # Check if trade was opened if not order is sent again
        if Order.retcode != mt5.TRADE_RETCODE_DONE:
```

```
            print(" Order has failed, retcode={}".format(Order.retcode))
            ## Recursive algorithm (skill A)
            Orders.openBUY(Symbol,LotSize,SL,TP,MagicNumber)

        result_dict=Order._asdict()
        print("BUY Trade Opened")
        for field in result_dict.keys():
            print("   {}={}".format(field,result_dict[field]))

        return Order

    # Opens a sell order
    @staticmethod
    def openSELL(Symbol,LotSize,SL,TP,MagicNumber):

        Orders.PreapereSymbol(Symbol)

        # Get symbol info
        point = mt5.symbol_info(Symbol).point
        price = mt5.symbol_info_tick(Symbol).bid
        deviation = 30

        request={
        "action": mt5.TRADE_ACTION_DEAL,
        "symbol": Symbol,
        "volume": LotSize,
        "type": mt5.ORDER_TYPE_SELL,
        "price": price,
        "sl": price + SL * point* 10,
        "tp": price - TP * point * 10,
        "deviation": deviation,
        "magic": MagicNumber,
        "comment": "Python script open",
        "type_time": mt5.ORDER_TIME_DAY,
        "type_filling": mt5.ORDER_FILLING_RETURN
        }
        # Sends the order
        Order= mt5.order_send(request)

        # Check if trade was opened if not order is sent again
        if Order.retcode != mt5.TRADE_RETCODE_DONE:
            print(" Order has failed, retcode={}".format(Order.retcode))
            ## Recursive algorithm (skill A)
            Orders.openSELL(Symbol,LotSize,SL,TP,MagicNumber)

        result_dict=Order._asdict()
        print("SELL Trade Opened ")
        for field in result_dict.keys():
            print("   {}={}".format(field,result_dict[field]))

        return Order

    # Closes buy position
    @staticmethod
```

```
    def closeBUY(Symbol,LotToClose,Position,MagicNumber):

        # Get symbol info
        position_id=Position.order
        price=mt5.symbol_info_tick(Symbol).bid
        deviation=30

        request={
            "action": mt5.TRADE_ACTION_DEAL,
            "symbol": Symbol,
            "volume": LotToClose,
            "type": mt5.ORDER_TYPE_SELL,
            "position": position_id,
            "price": price,
            "deviation": deviation,
            "magic": MagicNumber,
            "comment": "Python script close",
            "type_time": mt5.ORDER_TIME_DAY,
            "type_filling": mt5.ORDER_FILLING_RETURN,
        }

        # Sends the order
        Resultclose=mt5.order_send(request)

        # Check if trade was closed if not order is sent again
        if Resultclose.retcode != mt5.TRADE_RETCODE_DONE:
            print(" Order has failed, retcode={}".format( Resultclose.retcode))
            ## Recursive algorithm (skill A)
            Orders.closeBUY(Symbol,LotToClose,Position,MagicNumber)

    # Closes sell position
    @staticmethod
    def closeSELL(Symbol,LotToClose,Position,MagicNumber):

        # Get symbol info
        position_id=Position.order
        price=mt5.symbol_info_tick(Symbol).ask
        deviation=30

        request={
            "action": mt5.TRADE_ACTION_DEAL,
            "symbol": Symbol,
            "volume": LotToClose,
            "type": mt5.ORDER_TYPE_BUY,
            "position": position_id,
            "price": price,
            "deviation": deviation,
            "magic": MagicNumber,
            "comment": "Python script close",
            "type_time": mt5.ORDER_TIME_DAY,
            "type_filling": mt5.ORDER_FILLING_RETURN,
        }

        # Sends the order
```

```
        ResultClose=mt5.order_send(request)

        # Check if trade was closed if not order is sent again
        if ResultClose.retcode != mt5.TRADE_RETCODE_DONE:
            print(" Order has failed, retcode={}".format(ResultClose.retcode))
            ## Recursive algorithm (skill A)
            Orders.closeSELL(Symbol,LotToClose,Position,MagicNumber)

    # Checks how many open trades there are
    @staticmethod
    def TotalPositions():
        positions_total=mt5.positions_total()
        if positions_total>0:
            print("Total positions=",positions_total)
        else:
            print("Positions not found")

    # Checks profit by a symbol
    @staticmethod
    def CheckProfitPips(Symbol):

        # Checks for positions
        position=mt5.positions_get(symbol= Symbol)
        if position==None:
            print("No positions, error code={}".format(mt5.last_error()))

        elif len(position)>0:
            ## Parsing complex data type (skill A)
            df=pd.DataFrame(list(position),columns=position[0]._asdict().keys())
            df.drop(["ticket","time","type","magic","identifier","reason","price_open","sl
","tp","price_current","swap","symbol","comment",'time_update', 'time_msc',
'time_update_msc', 'external_id'], axis=1, inplace=True)
            Profit = df.astype({'profit':'float'})
            LotSize= df.astype({'volume':'float'})
            LotSize= LotSize.iloc[0,0]
            Profit= Profit.iloc[0,1]
            LotSize= LotSize* 10
            PipsProfit= Profit/LotSize
            PipsProfit= round(PipsProfit,1)
            LotSize= float(round(LotSize/10,2))
            return PipsProfit, LotSize

    # Closes all open positions no matter the symbol
    @staticmethod
    def CloseAllPositions():
        Positions=mt5.positions_get()
        # Checks for positions
        if Positions==None:
            print("No positions, error code={}".format(mt5.last_error()))
            return None
```

```
        else:
            def ClosePosition(Ticket):
                Tick= mt5.symbol_info_tick(Ticket.symbol)
                request={
                "action": mt5.TRADE_ACTION_DEAL,
                "symbol": Ticket.symbol,
                "volume": Ticket.volume,
                "type": mt5.ORDER_TYPE_BUY if Ticket.type == 1 else mt5.ORDER_TYPE_SELL,
                "position": Ticket.ticket,
                "price": Tick.ask if Ticket.type == 1 else Tick.bid,
                "deviation":30,
                "magic": Ticket.magic,
                "comment": "Python script close",
                "type_time": mt5.ORDER_TIME_GTC,
                "type_filling": mt5.ORDER_FILLING_IOC,
                }

                # Sends the order
                ResultClose=mt5.order_send(request)

                # Check if trade was closed if not order is sent again
                if ResultClose.retcode != mt5.TRADE_RETCODE_DONE:
                    print(" Order has failed, retcode={}".format(ResultClose.retcode))
                    ## Recursive algorithm (skill A)
                    ClosePosition(Ticket)
                return ResultClose

            for Position in Positions:
                ## Recursive algorithm (skill A)
                ClosePosition(Position)
```

Send Profits

```
from datetime import timedelta
from datetime import datetime
from datetime import date
import MetaTrader5 as mt5
import mysql.connector
import pandas as pd
import requests

class SendProfits():
    ## Web service API (skill A)
    # Using an API to send an Email
    @staticmethod
    def SendAnEmail(Email,Subject,Text):
        Url = "https://rapidprod-sendgrid-v1.p.rapidapi.com/mail/send"

        Payload = {
            "personalizations": [
                {
                    "to": [{"email": Email}],
                    "subject": Subject
```

```
                }
            ],
            "from": {"email": "pythontradingbot0@gmail.com"},
            "content": [
                {
                    "type": "text/plain",
                    "value": Text
                }
            ]
        }
        Headers = {
            "content-type": "application/json",
            "X-RapidAPI-Key": "13f2253b2fmsh012cbdafb66d357p141c2bjsn9e1abc4893d7",
            "X-RapidAPI-Host": "rapidprod-sendgrid-v1.p.rapidapi.com"
        }

        # Sends Email
        requests.request("POST", Url, json=Payload, headers=Headers)

    # Objective 3.d.i.
    # Gets total profit/loss account has and updates Database
    @staticmethod
    def TotalProfit(AccountID):
        ## Parsing complex data type (skill A)
        account_info_dict = mt5.account_info()._asdict()
        df=pd.DataFrame(list(account_info_dict.items()),columns=['property','value'])
        Totalprofit= float(df.iloc[12][1])
        # Connection to database
        DataBase= mysql.connector.connect(
        host="139.162.199.24",
        user="Timko",
        password="Plemenitas1.",
        database= "ForexTrader"
        )
        ## Cross table parameterised SQL (skill A)
        mycursor= DataBase.cursor()
        mycursor.execute("""UPDATE User_Stats SET Total_Profit="%s" WHERE Account_ID_Stats
IN (SELECT Account_ID FROM User_Accounts WHERE Account_ID= %s)
""",(Totalprofit,AccountID))
        DataBase.commit()

    # Objective 8.
    ## Web service API (skill A)
    # It usees API to get a quote of the day which is then sent in an email containing
profit/loss to the user
    @staticmethod
    def ProfitForDay():
        url = "https://motivational-quotes1.p.rapidapi.com/motivation"

        payload = {
            "key1": "value",
            "key2": "value"
        }
        headers = {
```

```
            "content-type": "application/json",
            "X-RapidAPI-Key": "13f2253b2fmsh012cbdafb66d357p141c2bjsn9e1abc4893d7",
            "X-RapidAPI-Host": "motivational-quotes1.p.rapidapi.com"
        }

        response = requests.request("POST", url, json=payload, headers=headers)

        Quote= str(response.text)
        ## Parsing complex data type (skill A)
        # Gets profit for the current trading day
        MidnightDate= str(date.today() + timedelta(days= 2))
        MidnightDate= tuple(MidnightDate.split("-"))
        DateTimeNow=
datetime(int(MidnightDate[0]),int(MidnightDate[1]),int(MidnightDate[2]))
        MidnightDate= str(date.today())
        MidnightDate= tuple(MidnightDate.split("-"))
        Yestarday=
datetime(int(MidnightDate[0]),int(MidnightDate[1]),int(MidnightDate[2]))
        deals = mt5.history_deals_get(Yestarday, DateTimeNow)
        if deals == None:
            print("No deals, error code={}".format(mt5.last_error()))
        elif len(deals) > 0:
            df=pd.DataFrame(list(deals),columns=deals[0]._asdict().keys())
            df['time'] = pd.to_datetime(df['time'], unit='s')
            df.drop(["ticket","type","magic","reason","time","symbol","swap","comment",'ti
me_msc', 'external_id',"commission","position_id","order","entry","volume","price","fee"],
axis=1, inplace=True)
            DayProfit= round(float(df.sum(axis = 0)),2)
            if DayProfit <= 0:
                DayProfitMassage = "Hi there! \nI have made You a LOSS of {}$ Today! Don't
worry though, you still get your Quote of the day: \n{} \nI Hope that at least that makes
you Feel Better :)".format(DayProfit,Quote)
            else:
                DayProfitMassage= "YEY! \nI have made You a PROFIT of {} Today! To make
you feel even Happier :) Here is your Quote of the day: \n{} ".format(DayProfit,Quote)
            return DayProfitMassage

    # Objective 8.b.
    # Gets total number of trades user has taken with that account and updated the
Database
    @staticmethod
    def NumberOfTrades(AccountID):
        AccountIDList= []
        AccountIDList.append(AccountID)
        # Connection to database
        DataBase= mysql.connector.connect(
        host="139.162.199.24",
        user="Timko",
        password="Plemenitas1.",
        database= "ForexTrader"
        )
        mycursor= DataBase.cursor()
        ## Cross table parameterised SQL (skill A)
```

```
        mycursor.execute("SELECT Account_Open_Date FROM User_Usage WHERE Account_ID_Usage
IN (SELECT Account_ID FROM User_Accounts WHERE Account_ID= %s)",(AccountIDList))
        StartDateDataBase= str(mycursor.fetchone()[0])
        StartDateDataBase= tuple(StartDateDataBase.split("-"))
        StartDate =
datetime(int(StartDateDataBase[0]),int(StartDateDataBase[1]),int(StartDateDataBase[2]))
        MidnightDate= str(date.today() + timedelta(days= 2))
        MidnightDate= tuple(MidnightDate.split("-"))
        DateTimeNow=
datetime(int(MidnightDate[0]),int(MidnightDate[1]),int(MidnightDate[2]))

        deals = mt5.history_deals_get(StartDate, DateTimeNow)
        if deals == None:
            print("No deals, error code={}".format(mt5.last_error()))
        elif len(deals) > 0:
            ## Parsing complex data type (skill A)
            df=pd.DataFrame(list(deals),columns=deals[0]._asdict().keys())
            df['time'] = pd.to_datetime(df['time'], unit='s')
            df.drop(["ticket","type","magic","reason","time","symbol","swap","comment",'ti
me_msc', 'external_id',"commission","position_id","order","entry","volume","price","fee"],
axis=1, inplace=True)
            NumberOfTrades= int(round((int(df.shape[0])) / 2))
            ## Cross table parameterised SQL (skill A)
            mycursor.execute("""UPDATE User_Stats SET Number_Of_Trades_Taken = "%s" WHERE
Account_ID_Stats IN (SELECT Account_ID FROM User_Accounts WHERE Account_ID= %s)
""",(NumberOfTrades,AccountIDList[-1]))
            NumberOfGoodTrades= int((df>0).sum().sum())
            mycursor.execute("""UPDATE User_Stats SET Number_Of_Trades_Won = "%s" WHERE
Account_ID_Stats IN (SELECT Account_ID FROM User_Accounts WHERE Account_ID= %s)
""",(NumberOfGoodTrades,AccountIDList[-1]))
            DataBase.commit()
```

Testing

The purpose of this testing is to ensure that the software is functioning effectively and meets all the specified objectives. My plan is to test it against each of my initially set objectives as well as making sure that the software is "robust" and user-friendly as expected. All that will either be covered in the testing table with screenshots or in the testing video.

Testing Table

Test ID	Description	Test Data	Expected Outcome	Test Result	Cross Reference
01	Checking if only valid user input is accepted.	Erroneous	Prompting for correct input.	Pass	Test 1 Screenshot
02	Checking if selecting mode 1 prompts user to create account or sign into account.	Typical	User is asked to continue by entering their email address.	Pass	Test 2 Screenshot
03	Checking if selecting mode 2 logs user into terminal using demo credentials and opens trading terminal window.	Typical	Acknowledgement of successful log in and account balance is displayed, trading terminal window is also shown.	Pass	Test 3 Screenshot
04	Checking if selecting mode 3 shows profits others have made and after that ask user to select mode again.	Typical	Average profit of all users is shown as well as 5 biggest profits per user in past 30 days.	Pass	Test 4 Screenshot
05	Checking that user can't run program without internet connection.	Erroneous	Massage (every 5s) telling user that they have to establish internet connection until they do so.	Pass	Test 5 Screenshot
06	Checking that only a valid email can be entered when asked in mode 1.	Erroneous	Displays massage telling user that their email is invalid and asks for a valid email again.	Pass	Test 6 Screenshot
07	Checking if user gets an email with verification code to create an account.	Typical	An email with verification code and massage when verification has been successful.	Pass	Test 7 Screenshot
08	Checking that user can enter email again if 0 is pressed trough verification process.	Typical	User is moved back to the start to choose mode.	Pass	Test 8 Screenshot
09	Checking that only valid email verification code is accepted.	Erroneous	Tell user that code is invalid until valid one is entered.	Pass	Test 9 Screenshot
10	Checking that user has to input same password twice when creating an account.	Erroneous	If passwords don't match user can try again 2 times after that	Pass	Test 10 Screenshot

			they are asked to create new password.		
11	Checking if only number can be entered as a broker account number.	Erroneous	If input is not a number user is asked to enter account number again.	Pass	Test 11 Screenshot
12	Checking that only valid broker account can be connected.	Erroneous	If broker account is not valid error code and massage are displayed furthermore user is asked to re-enter broker details.	Pass	Test 12 Screenshot
13	Checking that user can only enter valid input when choosing risk setting.	Erroneous	Asks for the risk setting to be entered again until valid input is entered.	Pass	Test 13 Screenshot
14	Checking that user can only enter valid input when choosing which pair to trade.	Erroneous	Asks for number of the pair to be entered again until valid input is entered.	Pass	Test 14 Screenshot
15	Checking that when user chooses to enter their own pair that is not listed as optimised option input is in right form as well as broker supports the pair.	Erroneous	If pair is not supported by the broker error massage is displayed and user is asked to choose another pair.	Pass	Test 15 Screenshot
16	Checking that when account has already been created user has to confirm IP address if it hasn't been used with account yet and then enter password.	Typical	User must do email verification and then is asked to enter password.	Pass	Test 16 Screenshot
17	Checking that if IP address is already associated with an account, then email verification is not needed.	Typical	User is prompted to enter password.	Pass	Test 17 Screenshot
18	If user fails to enter password correctly 3 times, then it has to do email verification and change password.	Typical	User is prompted to do email verification and then go trough password creation again like they did when they created the account.	Pass	Test 18 Screenshot
19	Checking that it sends an email with profit/loss for the day as well as motivational quote.	Typical	Email with profit/loss and motivational quote.	Pass	Test 19 Screenshot
20	Trades can be monitored using MT5 terminal phone app	Typical	User can see open trades on the phone.	Pass	Test 20 Screenshot
21	Checking that database is updated every time user logs into account	Typical	Database is updated.	Pass	Test 21 Screenshot

22	Checking that user details are stored once account has been created	Typical	Details are the same as the ones user entered.	Pass	Test 22 Screenshot

Screenshots

Test 01

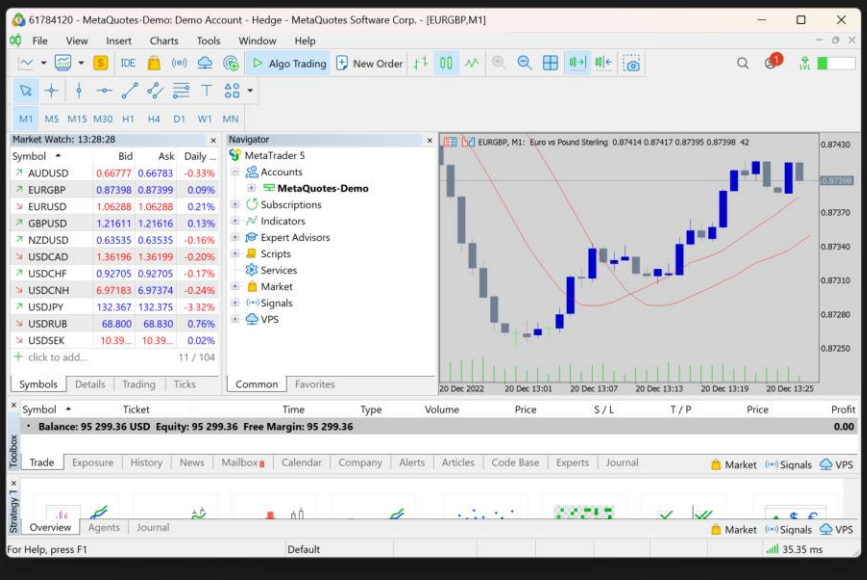
```
WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use: Banana
Please enter a Numeric value: 123456
Please enter a valid Number in order to choose your Mode: apple
Please enter a Numeric value: 1
Please enter your Email address:
```

Test 02

```
WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use: 1
Please enter your Email address:
```

Test 03

```
WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use: 2

YEY, you are IN!

Your Account Balance: $95299.36
Please choose one of below listed risk options in order to continue:
1. Conservative
2. Moderate
3. Agressive
Please enter a number for risk option you want to choose:
```



Test 04

```
WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use: 3

Avrage Profit of our Users in past 30 days was: 113.6 Dollars
While most succesful Users have Made:
$ 459.7
$ 118.87
$ 75.43
$ 20.31
$ 11.2
In the Past 30 Days!

WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use:
```

Test 05

```
Please establish an internet connection in order to continue...
Please establish an internet connection in order to continue...
Please establish an internet connection in order to continue...
Please establish an internet connection in order to continue...
Please establish an internet connection in order to continue...
Please establish an internet connection in order to continue...
Please establish an internet connection in order to continue...
Please establish an internet connection in order to continue...
Please establish an internet connection in order to continue...
Please establish an internet connection in order to continue...

WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use:
```

Test 06

```
WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use: 1
Please enter your Email address: banana
Sorry your Email address seems to be Invalid please try again: 12345
Sorry your Email address seems to be Invalid please try again: banana@apple.12345
Sorry your Email address seems to be Invalid please try again: kiyehog242@lubde.com
Code might slide into SPAM so please check there as well!
Please Enter Verification code you recived in an Email or Press 0 to enter Email again:
```

Test 07

```
WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use: 1
Please enter your Email address: codoliv162@nazyno.com
Code might slide into SPAM so please check there as well!
Please Enter Verification code you recived in an Email or Press 0 to enter Email again: 210188
 YEY, Verification finished!
Please create your Password:
```

pythontradingbot0@gmail.com

Subject: Email Verification

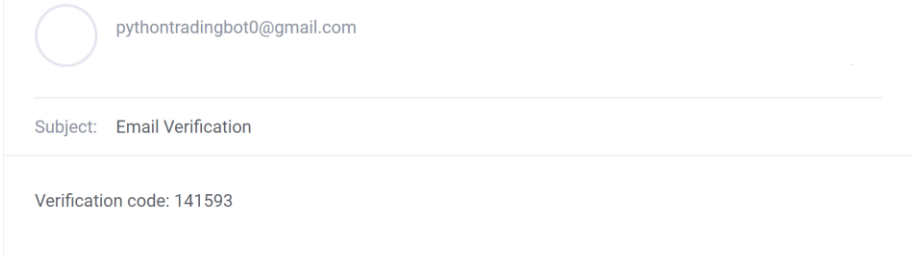
Verification code: 210188

Test 08

```
WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use: 1
Please enter your Email address: melal80032@randrai.com
Code might slide into SPAM so please check there as well!
Please Enter Verification code you recived in an Email or Press 0 to enter Email again: 0

WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use: 1
Please enter your Email address: sefibam403@paxven.com
Code might slide into SPAM so please check there as well!
Please Enter Verification code you recived in an Email or Press 0 to enter Email again:
Your Email will be sent again
Code might slide into SPAM so please check there as well!
Please Enter Verification code you recived in an Email or Press 0 to enter Email again: 306131
 YEY, Verification finished!
Please create your Password:
```

Test 09

```
WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use: 1
Please enter your Email address: hexak95264@randrai.com
Code might slide into SPAM so please check there as well!
Please Enter Verification code you recived in an Email or Press 0 to enter Email again: banana
Your Email will be sent again
Code might slide into SPAM so please check there as well!
Please Enter Verification code you recived in an Email or Press 0 to enter Email again: 23
Upps this isn't the right code Please try Again: 32432454532
Upps this isn't the right code Please try Again: 141593
 Verification finished
Please create your Password:
```



Test 10

```
WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use: 1
Please enter your Email address: sefibam403@paxven.com
Code might slide into SPAM so please check there as well!
Please Enter Verification code you recived in an Email or Press 0 to enter Email again: 370454
 YEY, Verification finished!
Please create your Password: Banana123
Please enter your Password again just make sure it is correct:) Apple123
Your Passwords don't match please try again you have 1 try(s) left
Please enter your Password again here: Apple123
Your Passwords don't match please try again you have 0 try(s) left
Please enter your Password again here: 123456
You failed too many times Please try creating your password again
Please create your Password: Apple1234.
Please enter your Password again just make sure it is correct:) Banana12
Your Passwords don't match please try again you have 1 try(s) left
Please enter your Password again here: Apple1234.
Thank You! Your Password has been saved.
```

Test 11

```
WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use: 1
Please enter your Email address: sefibam403@paxven.com
Code might slide into SPAM so please check there as well!
Please Enter Verification code you recived in an Email or Press 0 to enter Email again: 390391
 YEY, Verification finished!
Please create your Password: Banana123
Please enter your Password again just make sure it is correct:) Banana123
Thank You! Your Password has been saved.
Please enter number of your Broker account or press 0 to start again: banana
Please enter a numeric value for your Account number: 61784120
```

Test 12

```
WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use: 1
Please enter your Email address: sefibam403@paxven.com
Code might slide into SPAM so please check there as well!
Please Enter Verification code you recived in an Email or Press 0 to enter Email again: 390391
 YEY, Verification finished!
Please create your Password: Banana123
Please enter your Password again just make sure it is correct:) Banana123
Thank You! Your Password has been saved.
Please enter number of your Broker account or press 0 to start again: banana
Please enter a numeric value for your Account number: 61784120
Please Enter name of the server your Broker is connected to: MetaQuotes-Demo
Please Enter your Broker password:
 Initialization failed, error code = (-2, 'Invalid "login" argument')
 Sorry your Broker details are Invalid please try again.
Please enter number of your Broker account or press 0 to start again: 61784120
Please Enter name of the server your Broker is connected to: MetaQuotes-Demo
Please Enter your Broker password: acl3knnt

 Your Account has been Created, YEY!
```

Test 13

```
WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use: 2

YEY, you are IN!

Your Account Balance: $95299.36
Please choose one of below listed risk options in order to continue:
1. Conservative
2. Moderate
3. Agressive
Please enter a number for risk option you want to choose: Banana
Please enter a Numeric value: 123456
Please enter a valid Number in order to choose your Mode: 2

Please choose one of options bellow or enter your own pair to trade.
```

Test 14

```
WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use: 2

YEY, you are IN!

Your Account Balance: $95299.36
Please choose one of below listed risk options in order to continue:
1. Conservative
2. Moderate
3. Agressive
Please enter a number for risk option you want to choose: 2

Please choose one of options bellow or enter your own pair to trade.
Please note that this AI has been optimised for pairs listed bellow therfore it produces the best results traiding those
1. EURUSD
2. EURGBP
3. USDJPY
4. USDCAD
5. GBPJPY
Press 0 to Enter any other FOREX pair
Please enter a number in order to choose which pair to trade: Banana
Please enter a numeric value between 1 and 5 or preess 0 for a custom FOREX pair: 12345
Please enter number between 1 and 5 or press 0 for entering a custom FOREX pair: 2

Downloading past data
[**********100%**********]  1 of 1 completed
```

Test 15

```
WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Agressive
Please enter a number for risk option you want to choose: 3

Please choose one of options bellow or enter your own pair to trade.
Please note that this AI has been optimised for pairs listed bellow therfore it produces the best results traiding those
1. EURUSD
2. EURGBP
3. USDJPY
4. USDCAD
5. GBPJPY
Press 0 to Enter any other FOREX pair
Please enter a number in order to choose which pair to trade: 0
Please enter a FOREX pair eg. EURGBP: BTCUSD

BTCUSD not found, can not call order_check()
BTCUSD is not visible, trying to switch on
symbol_select({}) failed, exit BTCUSD
Your broker doesn't support that pair please try agian

Please choose one of options bellow or enter your own pair to trade.
Please note that this AI has been optimised for pairs listed bellow therfore it produces the best results traiding those
1. EURUSD
2. EURGBP
3. USDJPY
4. USDCAD
5. GBPJPY
Press 0 to Enter any other FOREX pair
Please enter a number in order to choose which pair to trade: 0
Please enter a FOREX pair eg. EURGBP: EURAUD


Downloading past data
[**********100%**********]  1 of 1 completed
```

Test 16

```
WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use: 1
Please enter your Email address: timon.plemenitas@gmail.com
We can't recognise your IP so you will have to confirm that this is you by Email Verification.
Code might slide into SPAM so please check there as well!
Please Enter Verification code you recived in an Email or Press 0 to enter Email again: 471052
 YEY, Verification finished!
Please Enter your account Password: Plemenitas1


YEY, you are IN!
```

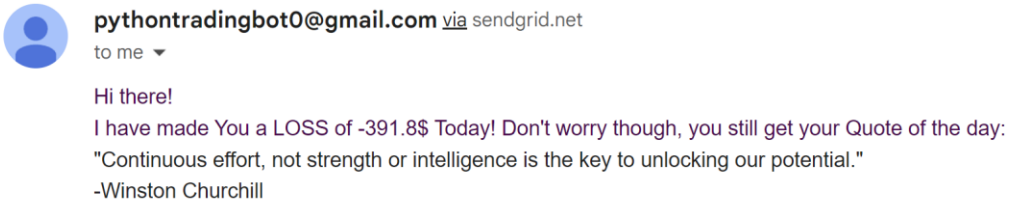
Test 17

```
WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use: 1
Please enter your Email address: timon.plemenitas@gmail.com
Please Enter your account Password: Plemenitas1


YEY, you are IN!
```

Test 18

```
WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use: 1
Please enter your Email address: timon.plemenitas@gmail.com
Please Enter your account Password: Banana123
Password was incorect please try again you have 3 trys left: Apple123
Password was incorect please try again you have 2 trys left: 12345
Password was incorect please try again you have 1 trys left: Apple

You have now Enetered recovery mode
Code might slide into SPAM so please check there as well!
Please Enter Verification code you recived in an Email or Press 0 to enter Email again: 243487
 YEY, Verification finished!
Please create your Password: Banana123.
Please enter your Password again just make sure it is correct:)
Your Passwords don't match please try again you have 1 try(s) left
Please enter your Password again here: Banana123.
Thank You! Your Password has been saved.


YEY, you are IN!
```

Test 19

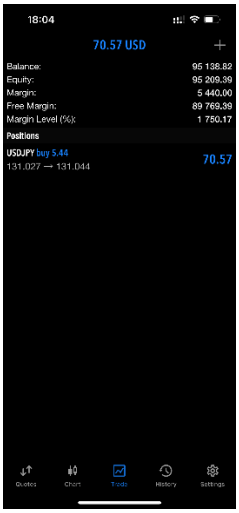


pythontradingbot0@gmail.com via sendgrid.net
to me

Hi there!
I have made You a LOSS of -391.8$ Today! Don't worry though, you still get your Quote of the day:
"Continuous effort, not strength or intelligence is the key to unlocking our potential."
-Winston Churchill

Test 20

Phone terminal



Computer terminal

Test 21

	Account_ID_Stats	Number_Of_Trades_Taken	Number_Of_Trades_Won	Total_Profit
	1	32	24	120.43
	2	3	1	20.31
	3	14	6	11.20
	4	58	36	459.70
	5	5	3	-4.16

	Account_ID_Stats	Number_Of_Trades_Taken	Number_Of_Trades_Won	Total_Profit
▶	1	33	24	118.87
	2	3	1	20.31
	3	14	6	11.20
	4	58	36	459.70
	5	5	3	-4.16

Test 22

```
WELCOME TO AUTOMATED FOREX TRADING AI PLEASE CHOOSE ONE OF BELOW LISTED OPTIONS IN ORDER TO CONTINUE:
1. Connect to your Account OR Create one (Please note that in order to use this function you have to have your own Broker Account)
2. One time DEMO use
3. Show Profits other users have made using this AI

Please enter a Number for the MODE you want to use: 1
Please enter your Email address: sefibam403@paxven.com
Code might slide into SPAM so please check there as well!
Please Enter Verification code you recived in an Email or Press 0 to enter Email again: 390391
 YEY, Verification finished!
Please create your Password: Banana123
Please enter your Password again just make sure it is correct:) Banana123
Thank You! Your Password has been saved.
Please enter number of your Broker account or press 0 to start again: banana
Please enter a numeric value for your Account number: 61784120
Please Enter name of the server your Broker is connected to: MetaQuotes-Demo
Please Enter your Broker password:
 Initialization failed, error code = (-2, 'Invalid "login" argument')
 Sorry your Broker details are Invalid please try again.
Please enter number of your Broker account or press 0 to start again: 61784120
Please Enter name of the server your Broker is connected to: MetaQuotes-Demo
Please Enter your Broker password: acl3knnt

 Your Account has been Created, YEY!
```

database

Account_ID	Account_Email	Password	Broker_Account_Number	Broker_Server	Broker_Password
1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
4	timon.plemenitas@gmail.com	Plemenitas1	61784120	MetaQuotes-Demo	acl3knnt
5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
6	sefibam403@paxven.com	Banana123	61784120	MetaQuotes-Demo	acl3knnt

Testing Video

https://www.dropbox.com/s/n3d7v7vo70mq97c/Computer%20Science%20project%20testing%20video.mkv?dl=0

Evaluation

Objective analysis

Initially, set idea of an AI that predicts the forex market and then automatically executes trades on the market has been met with all key requirements that were set at the start of the project. With all the aspects of the program considered, I believe the program has met the needs of the end user. As the final product has achieved the fundamental aspect of an automated trading software as well as some additional features enabling better user experience, e.g., emailing profits to users.

I will analyse each objective and evaluate how well it has been met:

Objective	Comment	Met?
1	System opens and then using grid closes/manages positions	Fully
2a	Massage is displayed letting user know they don't have an internet connection established	Fully
2b	It checks every 5 seconds if user has established an internet connection until user does so	Fully
3a	It only excepts valid inputs if invalid is entered error massage is displayed and user is asked to enter valid input	Fully
3b	If user want to use demo mode no broker account is needed as there is one pre set and it executes trades to demonstrate user how it works	Fully
3c	User can create account when done that they don't need to re enter all their details as stored in database furthermore it also handles email verification, passwords and IPs of users no matter if they are creating account or they already have one	Fully
3d	It shows profits others have made by firstly recording them every time user makes a prediction	Fully
4	They have 3 option (conservative, moderate, aggressive) each risking different amount (percentage) of an account balance	Fully
5a	It displays 5 optimised pairs and it tells users that those are optimised	Fully
5b	If user wants to enter custom pair then they have to press 0 and if it is available by their broker it is going to be traded otherwise user will be prompted to enter another pair	Fully
6	Historic data (prices, volume) are queried from yahoo finance API and then used to train model, test and make a prediction	Fully
7	After trade has been opened program starts to close small parts of the trade in profit in order to reduce risk and exposure thus maximise profit	Fully
8	Email with the profits for the day is sent as well as motivation quote which is queried from an API furthermore it also records all trades and winning trades in the database	Fully
9	It makes predictions everyday at approximately same time after daily candle has closed	Fully
10	It is hosted on a Linux server	Fully
11	If error occurs error massage is displayed and email is sent to me with user's account ID so I can potentially help them solve the problem. Furthermore all trades are closed unless error is loss of internet connection	Fully
12	Yes they can do that using MT5 app and sign in credentials on their phone	Fully
13	It can be run on any computer with good enough hardware to run the prediction in reasonable time however python is not required furthermore it is only one application (500MB) to run in order to get it running.	Fully

User feedback Interview

Now that you've had a chance to test the software, how accurate were the AI's market predictions compared to your expectations?

> The market predictions were quite impressive and exceeded my initial expectations. While I believe no system can be perfect, it managed to provide consistently accurate predictions.

How effective were the risk management strategies implemented in the software?

> The risk management strategies were effective in protecting my capital during testing. Especially the position sizing management (Grid) helped minimise losses and increase profits.

How user-friendly did you find the software's interface? Were there any features or aspects that you found particularly useful or difficult to navigate?

> Instructions were very clear and easy to understand, and I believe even people with less or no experience would be easily able to operate it. However, I believe you should make sure user knows that they can also check profits at any time using the MT5 terminal on their phone, as I believe your target group of inexperienced users might not know this function exists without telling them.

Did the daily email summary of trading results meet your expectations in terms of convenience and the information provided?

> The daily email summary was an excellent feature that provided me with a convenient way to stay informed about my trading performance. The information provided in the emails was very clear and was often able to put me in a better mood with the motivation quote attached.

Were there any features or aspects of the software that you felt could be improved or added to enhance its functionality?

> While the software was generally impressive, there is always room for improvement. I believe that integration with additional trading platforms, such as Binance, could further enhance its functionality and appeal to a wider range of users.

Summarised User Feedback

End user has found the AI's market predictions impressive and quite accurate. Risk management strategies, particularly position sizing management (grid), effectively protected capital, minimising losses and increasing profits. Furthermore, the software's interface was user-friendly and accessible, but could better inform users about mobile MT5 terminal functionality. In addition to that daily email summaries were not only convenient and informative but also appreciated. The user has also suggested integrating additional trading platforms, to enhance functionality. Overall, the software performed well and exceeded expectations in eyes of the end user.

Further Improvements

- At this point, when users request a prediction, they must wait for a significant amount of time (between 15-40 minutes, depending on the hardware) for the model to generate a prediction. This delay occurs because the trained model is not stored after it has been trained. Ideally, both the test model and predictions would be saved in a database for each pair, making the process faster and, in some cases, nearly instantaneous. For example, if one user runs a EUR/USD prediction in the morning, the prediction could be saved in the database as a float. Consequently, other users who want to trade the same pair that day would not need to run the prediction on their computers; instead, the price prediction would be fetched from the database, enabling faster predictions.

- To achieve more accurate predictions, I also plan to expand the data inputs for the neural networks beyond just price action. By incorporating indicators such as EMA, RSI, and others, as well as TP, SL, lot size, and grid profit-taking points, therefore the AI will be able to adjust these variables based on market conditions and the trading pair, resulting in improved accuracy.

- Allowing users to link multiple broker accounts to a single email login would make it easier for them to reduce fees and spreads across different brokers and pairs, increasing convenience. Currently, users must have two separate accounts with two different email addresses to manage multiple broker accounts.

- By enabling the AI to trade multiple pairs simultaneously, the system becomes less vulnerable to sudden market changes and potential losses on a single pair. For example, having one losing trade on one pair but two winning trades on other pairs puts the user in a stronger position, making their results more consistent. The probability of the AI being wrong three times in a row is lower (12.5%) than being wrong once, where the worst-case scenario probability is 50%.

- Integrating additional trading platforms would provide users with the ability to trade various instruments, such as cryptocurrencies, which are rarely supported by FX brokers, thus the MT5 terminal.