JOURNAL OF UNIVERSAL SCIENCE RESEARCH

ISSN (E): 2181-4570 ResearchBib Impact Factor: 6,4 / 2023

Automated Trading Bot
Design and Implementation for Cryptocurrency Transactions

Abdallah M. Shnaino

Department Software Development, Islamic University of Gaza, Palestine,

abdallah.shnaino@gmail.com

Abstract**—**This project addresses the limitations of traditional methods of placing orders on cryptocurrency platforms. With a sophisticated bot and an innovative website, customers can make transactions easily and efficiently at any time and from anywhere. The solution makes use of the latest technology to automate the trading process, saving time and enabling clients to take advantage of market volatility. The website provides an easy-to-use interface, real-time market data, and customized trading preferences. Security is prioritized through strong encryption protocols and multiple layers of security. The project aims to revolutionize digital currency trading, and enable individuals to achieve financial success by automating the trading process.

Keywords: *Agile; Cryptocurrency; Fintech; Automation; Multithreading***.**

Introduction

This paper proposes the automation of cryptocurrency trading through the development of a bot that utilizes real-time market analysis and a collection of strategies to execute buy, sell, and hold actions on a chosen platform, such as Binance. The primary aim is to enhance user well-being by employing an event-driven architecture to interconnect and share data in the business workflow from preferred platforms until the chosen strategy is executed.

Acknowledging the necessity for a non-manual trading system, the paper implements a web-based application where users can create their own bots, select specific strategies, and define indicators for the bots to work on. The trading process runs from start to finish or until cancellation without requiring users to be constantly in front of the trading screen.

The primary objective of this paper is to demonstrate the effectiveness and necessity of automated trading processes.

Our solution revolves around the incorporation of a sophisticated trading bot and a user-friendly website interface. Each bot is distinguished according to the strategy it uses, some of which use Relative Strength Index (RSI) and others that use Exponential

ISSN (E): 2181-4570 ResearchBib Impact Factor: 6,4 / 2023

Moving Average (EMA) or even a customized strategy RSI as strategy binance as trading platform. In our project, we employed XP as our chosen agile methodology.

I. Methodology

Requirements

Requirements and specifications the main objective of this project is to create an automated trading process, enabling users to trade without the constant need to be in front of the trading screen. Users will input specific parameters, such as the type of strategy and its corresponding values, into the bot. The bot will then execute trades based on these values until the pre-defined conditions for buying and selling are met. To accomplish this, a seamless connection between the trading platform (e.g., Binance), TradingView for pre-defined indicators, and our system is crucial. Additionally, the system must address non-functional requirements, including ensuring robustness, security, and efficiency in the automated trading process.

Design

Database design

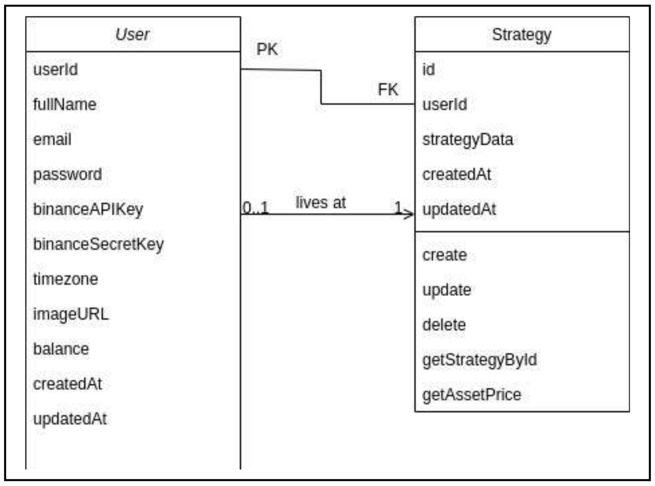


Figure 1. Database class diagram

In the given database design, there are two main entities: User and Strategy. The User entity represents individuals who can create their own User entity by providing their full name, custom password, Binance API (Application Programming Interface) key, and secret key. With their user entity, users gain access to various operations associated with the Binance platform. The Strategy entity allows users to create, update,

ISSN (E): 2181-4570 ResearchBib Impact Factor: 6,4 / 2023

and cancel individual bots for their trading strategies. Each Strategy entity contains specific data related to the strategy it represents.

To summarize:

- The User entity stores user information, including their full name, custom password, and Binance API keys.
- Users can perform operations associated with Binance using their user entity.
- The Strategy entity enables users to create, update, and cancel trading bots.
- Each Strategy entity holds data specific to the corresponding trading strategy.

This database design facilitates the management of user information, Binance API access, and trading strategy entities, enabling users to effectively create, update, and cancel their trading bots as per their requirements.

1) Software Architecture

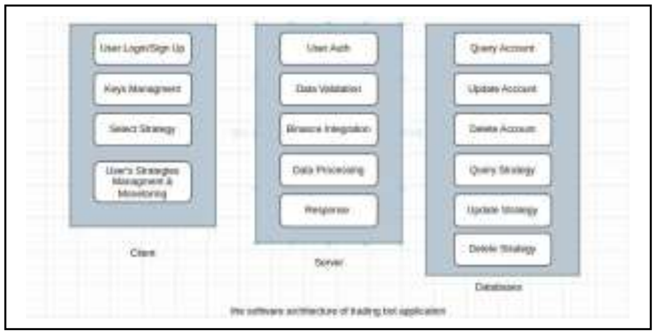


Figure 2. The software architecture of trading bot application

2) Software Design Pattern

In our development process, we have employed the MVC (Model-View-Controller) architectural pattern. MVC provides a structured approach for organizing and implementing our software application. The Model represents the data and business logic, the View handles the user interface and presentation, and the Controller acts as the intermediary between the Model and the View, managing the flow of data and user interactions [1].

ISSN (E): 2181-4570 ResearchBib Impact Factor: 6,4 / 2023

By adopting the MVC pattern, we have achieved separation of concerns, allowing for better code organization, modularity, and maintainability. The Model encapsulates the data and core functionality, ensuring data integrity and consistency. The View focuses on rendering the user interface, providing an intuitive and visually appealing experience. The Controller coordinates the communication between the Model and View, facilitating user interactions and business logic execution.

This architecture has facilitated collaboration among team members, as different components can be developed independently and integrated seamlessly. It has also enhanced the scalability of our system, enabling easy modification or expansion of specific components without affecting the entire application.

Overall, the utilization of the MVC pattern in our development process has significantly contributed to the efficiency, flexibility, and robustness of our software solution.

1) Information System Analysis of trading bot application

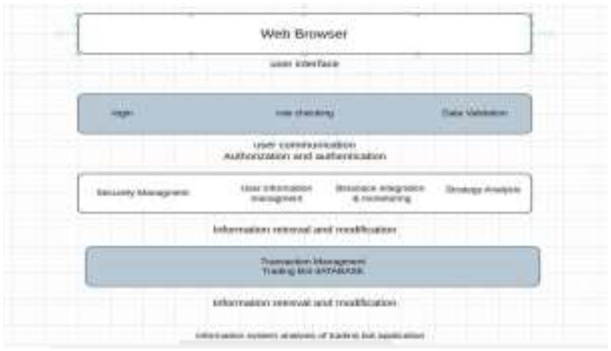

Web Browser
user interface
login
role checking
Data Validation
user communication
Authorization and authentication
Security Managment
User information managment
Binance integration & monitoring
Strategy Analysis
Information retrieval and modification
Transaction Management
Trading bot DATABASE
Information retrieval and modification
Information system analysis of trading bot application

Figure 3. Information system analysis of trading bot application

2) Strategy Architecture and System Context and Interaction

- **Strategy architecture**

In the realm of strategy architecture, we outline the functioning of a strategy. The data flow initiates from the client side, triggering the strategy's operation, which involves interacting with the database and Binance. The strategy subsequently produces outcomes based on this process.

- **System context and interaction**

Our system operates simultaneously with Binance for executing trading operations and with TradingView for analyzing the market using real-time data.

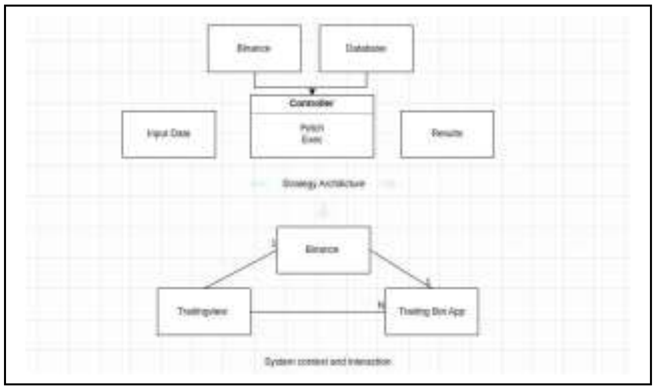


Figure 4. Strategy architecture and system context and interaction

II. RSI Strategy Worker

The RSI (Relative Strength Index**)** is displayed as an oscillator (a line graph) on a scale of zero to 100. The indicator was developed by J. Welles Wilder Jr [2].

The relative strength index (RSI) is a momentum indicator used in technical analysis. RSI measures the speed and magnitude of a security's recent price changes to evaluate overvalued or undervalued conditions in the price of that security [3].

Understanding the RSI Indicator

The RSI is a momentum oscillator that ranges from 0 to 100. The RSI formula calculates a value that oscillates between these extreme levels, providing insights into the asset's current price strength or weakness.

RS = average gain / average loss (1)

RSI = 100 – 100/(1+RS) (2)

The average gain or loss used in this calculation is the average percentage gain or loss during a look-back period. The formula uses a positive value for the average loss.

Managing the Creation of RSI Strategy Bots

This passage provides a synopsis of the approach to take in response to a user's request to create a bot that utilizes the RSI strategy. The bot in question would implement the RSI strategy and incorporate user-defined custom indicators,

ISSN (E): 2181-4570 ResearchBib Impact Factor: 6,4 / 2023

encompassing elements such as trading pairs, trade amounts, stop loss, and take profit parameters.

```
async function postCreateStrategy(req, res, next) {
    const {
        userId
    } = req.body;
    const strategyData = Object.assign(req.body, {
        monetor: [],
    });

    const strategy = await postCreate(userId, JSON.stringify(strategyData));
    const {
        binanceAPIKey,
        binanceSecretKey
    } = await findById(userId);
    let binanceClient = initClient(binanceAPIKey, binanceSecretKey);
    // To ensure our ability to execute actions on the Binance platform.
    const clientPing = await binanceClient.ping()
    if (!clientPing) {
        return res.status(400).json({
            "message": "connection refused with this client"
        });
    }
    const {
        ammount,
        pairs,
        numberOfTrades,
    } = JSON.parse(strategy.strategyData);
    try {
        await isAvailableAmmount(binanceClient, ammount, pairs, numberOfTrades)
    } catch (error) {
        return res.status(400).json({
            "message": error.message
        });
    }
    const strategyId = strategy.dataValues.id;
    // start working in a thread
    runRsiIndecator({
        keys: {
            binanceAPIKey,
            binanceSecretKey
        },
        strategyId,
    }).then(e => {
        throw new Error(e.message)
    }).catch(e => {
        console.log(' message ', e.message)
    })
    return res.status(200).json({
        "strategy": strategy
    });
}
```

Figure 5. RSI trade bot creation

RSI Strategy Automation: Dynamic Monitoring and Real-time Execution

This code has been designed to consistently oversee the strategy's implementation, continuously interfacing with the database for any updates. Should an update not be present in the database, the bot maintains vigilance over the current

ISSN (E): 2181-4570 ResearchBib Impact Factor: 6,4 / 2023

market price and cross-references it with the stored database details. Upon identifying a match, it promptly initiates a direct selling order.

Upon detecting an update, the bot promptly refreshes all relevant in-memory data. The code diligently monitors the RSI indicator until a suitable condition arises. Once a match is confirmed, the bot seamlessly carries out orders on the Binance platform.

Employing user-provided indicators, this strategy computes potential gains and losses. If the market price coincides with the predetermined profit or loss value, an immediate market order is executed.

```
async function run() {

    const {
        keys,
        strategyId,
    } = workerData;
    let binanceClient = initClient(keys.binanceAPIKey, keys.binanceSecretKey)
    let strategy = await findStrategyById(strategyId)
    let userId = strategy.dataValues.userId
    let strategyData = JSON.parse(strategy.strategyData)
    let numberOfTrades = strategyData.numberOfTrades;
    async function startRsi(strategyData, strategyId, updateStrategy) {

        try {
            let pairs = strategyData.pairs
            for (let pair of pairs) {
                const {
                    input
                } = await getDetachSourceFromOHLCV('binance', pair, strategyData.timeFrame, false) // true if you
want to get future market
                var atrData = await rsi(strategyData.period, strategyData.inputSource, input)
                let rsiValue = Number.parseInt(atrData[atrData.length - 1]);
                let payingIndicators = strategyData.payingIndicators
                for (let signal of payingIndicators) {
                    const isAccepted = await isAcceptedTransaction(binanceClient,
                        strategyData.ammount, pair)
                    if (rsiValue == signal && isAccepted) {

                        if (numberOfTrades > 0 && numberOfTrades <= numberOfTrades) {
                            // pair
                            let pairPrice = await getCurrentPrice(binanceClient, pair)
                            let winningPrice = computeWinningMarginPercent(Object.values(pairPrice)[0],
strategyData.winningMarginPercent)
                            let losingPrice = computelosingMarginPercent(Object.values(pairPrice)[0],
strategyData.losingMarginPercent)
                            let monetor = strategyData.monetor
                            // buy
                            monetor.push({
                                pair: Object.keys(pairPrice)[0],
                                pairPrice: Object.values(pairPrice)[0],
                                winningPrice,
                                losingPrice
                            })
                            let newStrategyData = Object.assign(strategyData, {
                                monetor
                            });
                            const strategy = await updateStrategy(strategyId, JSON.stringify(newStrategyData))
                            strategy)
                        numberOfTrades -= 1
```

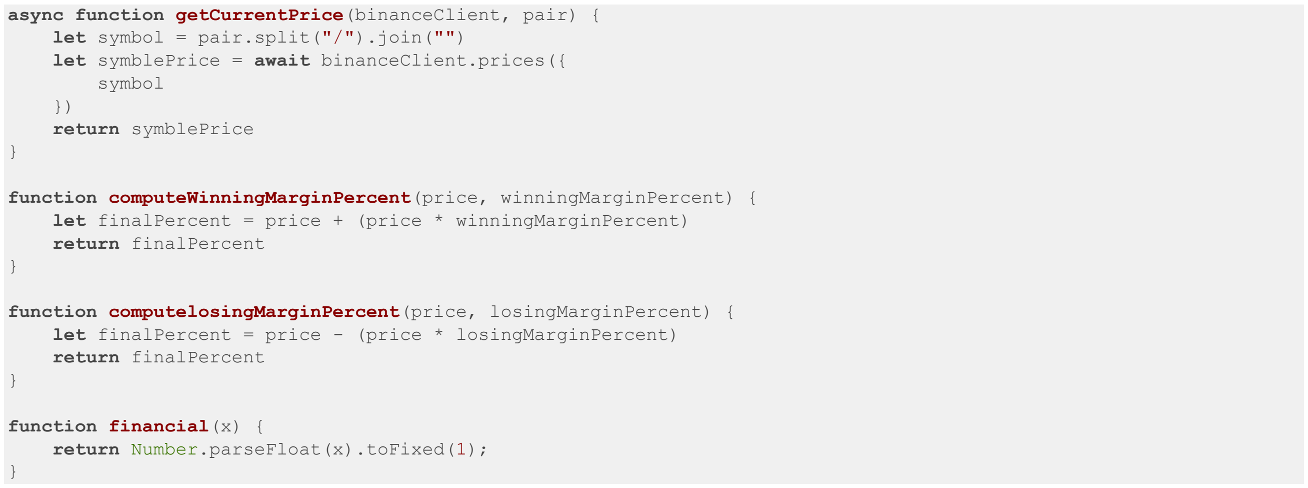
ISSN (E): 2181-4570 ResearchBib Impact Factor: 6,4 / 2023

```
                        let p = await binanceClient.prices({
                            symbol: Object.keys(pairPrice)[0]
                        })
                        let currentPrice = Object.values(p)[0]
                        let order = await binanceClient.order({
                            symbol: Object.keys(pairPrice)[0],
                            side: 'BUY',
                            type: 'market',
                            quantity: Number.parseFloat(strategyData.ammount /
Object.values(pairPrice)[0]).toFixed(1),
                        })

                    }
                    if (numberOfTrades <= 0 && monetor.length == 0) {
                        parentPort.postMessage({
                            message: "operation done"
                        })
                        clearInterval(interval);
                    }
                }
            }

            for (let obj of strategyData.monetor) {
                let p = await binanceClient.prices({
                    symbol: obj.pair
                })
                let currentPrice = Object.values(p)[0]
                if (currentPrice >= obj.winningPrice) {
                    let order = await binanceClient.order({
                        symbol: obj.pair,
                        side: 'SELL',
                        type: 'MARKET',
                        price: obj.price,
                        quantity: obj.quantity
                    })
                    // sell
                }
                if (currentPrice <= obj.losingPrice) {
                    let order = await binanceClient.order({
                        symbol: obj.pair,
                        side: 'SELL',
                        type: 'MARKET',
                        price: obj.price,
                        quantity: obj.quantity
                    })
                }
            }

        }
    } catch (error) {
        parentPort.postMessage(error)
    }
}
let interval = setInterval(startRsi, 5000, strategyData, strategyId, updateStrategy)
parentPort.on('message', async (data) => {
    strategy = await findStrategyById(strategyId)
    strategyData = JSON.parse(strategy.strategyData)
    clearInterval(interval);
    interval = setInterval(startRsi, 5000, strategyData, strategyId)
    parentPort.postMessage('Hello from the worker thread!');
});
}
```

Figure 6. RSI strategy worker thread

ISSN (E): 2181-4570 ResearchBib Impact Factor: 6,4 / 2023

Functions Assisting the RSI Bot



```
async function getCurrentPrice(binanceClient, pair) {
    let symbol = pair.split("/").join("")
    let symblePrice = await binanceClient.prices({
        symbol
    })
    return symblePrice
}

function computeWinningMarginPercent(price, winningMarginPercent) {
    let finalPercent = price + (price * winningMarginPercent)
    return finalPercent
}

function computelosingMarginPercent(price, losingMarginPercent) {
    let finalPercent = price - (price * losingMarginPercent)
    return finalPercent
}

function financial(x) {
    return Number.parseFloat(x).toFixed(1);
}
```

Figure 7. Assisting functions

Implementation

Interfaces Design

In the following section, we will showcase the design of the web-based interfaces in our system.

Home Page

The Home page interface serves as an introductory overview of our site, showcasing the range of services we offer and providing convenient means of communication with our technical support team. It offers a comprehensive introduction to our platform and facilitates easy access to support.

The Academy page

The Academy page on our website is dedicated to providing clear and concise information about the services we offer. Its primary objective is to guide users, enhance their understanding, and empower them to make the most of our services. By simplifying complex concepts, the Academy page ensures that users can easily navigate and leverage our offerings to their fullest potential.

The Login page

serves as a secure gateway for users to enter their authentication credentials and verify their identity in order to access their authorized accounts on our site. Its primary purpose is to establish a safe and reliable access point, granting users the ability to explore exclusive content and utilize special functions tailored to their needs. By

ISSN (E): 2181-4570 ResearchBib Impact Factor: 6,4 / 2023

implementing stringent security measures, the Login page guarantees the protection of user accounts and facilitates seamless navigation throughout the site's features.

The Register Page

acts as a user-friendly gateway for individuals to create new accounts on our website, offering a streamlined process that simplifies the entry of required information. It serves as the starting point for users embarking on their journey as registered members, ensuring a smooth and hassle-free registration experience. By providing intuitive interfaces and clear instructions, the Registered Page facilitates the seamless creation of accounts, empowering users to fully engage with our platform's offerings.

The API interface

The API (Application Programming Interface) interface simplifies the process of connecting the trading robot to the site services, enhancing its functions and expanding its capabilities.

My Bots

My Bot: This feature is designed to assist users in automating their tasks by utilizing pre-built bots. When pre-configured bots are available, My Bot provides easy access and implementation for users. In cases where pre-built bots are not available, My Bot guides the user through the process of creating their own custom bot. This empowers users to tailor their automation experience to their specific needs and preferences.

Portfolio

My Bot: The Portfolio feature displays the comprehensive value of your trading portfolio, providing an overview of your current balance. It showcases the combined worth of your trading portfolio, including the various currencies held within it.

Detailed Portfolio: The Detailed Portfolio feature presents a holistic view by showcasing the total value of your balance and the specific currencies held within your trading portfolio. This comprehensive representation enables you to accurately assess the overall performance and composition of your portfolio.

ISSN (E): 2181-4570 ResearchBib Impact Factor: 6,4 / 2023

Trading Bot

Main Settings: In the Main Settings section, users can configure the bot and select the platform on which the bot will operate.

Currencies: Users have the ability to specify the currencies in which the bot will engage in trading.

Trading Strategy: Users can define the maximum number of open trades and set the purchase amount for each order, thereby establishing their preferred trading strategy.

Selected Strategy: Users can choose the specific strategy they wish to implement for their trading activities.

Targeted Profits: Users have control over determining the desired percentage increase in currency value, at which point the bot will execute profit-taking actions. This feature also offers two options for selecting the profit-taking method.

Price Average Calculation: The system calculates the percentage increase in currency value based on the average pricing of the product.

Stop Loss: Users can set the currency rate at which the stop loss mechanism will be triggered to safeguard against potential losses in a trade.

Settings Page

Settings Page: The Settings page serves as a versatile platform for users to allocate and control various aspects of their accounts. It offers a wide range of customization options, empowering users to personalize their experience according to their unique preferences. On this page, users can easily manage notifications, update their profile information, adjust privacy settings, and access other configurable options. The Settings page ensures that users have full control over their accounts, enabling them to tailor their browsing experience and optimize their interactions with the platform.

Testing

Test-Driven Development

With TDD (Test-Driven Development), before writing implementation code, the developer writes automated unit test cases for the new functionality they are about to

ISSN (E): 2181-4570 ResearchBib Impact Factor: 6,4 / 2023

implement. After writing test cases that generally will not even compile, the developers write implementation code to pass these test cases. The developer writes a few test cases, implements the code, writes a few test cases, implements the code, and so on [4].

The work is kept within the developer's intellectual control because he or she is continuously making small design and implementation decisions and increasing functionality at a relatively consistent rate. New functionality is not considered properly implemented unless these new unit test cases and every other unit test case written for the code base run properly.

For example, we want to create a bot that uses Relative Strength Index (RSI) strategy so we should ensure that our bot.

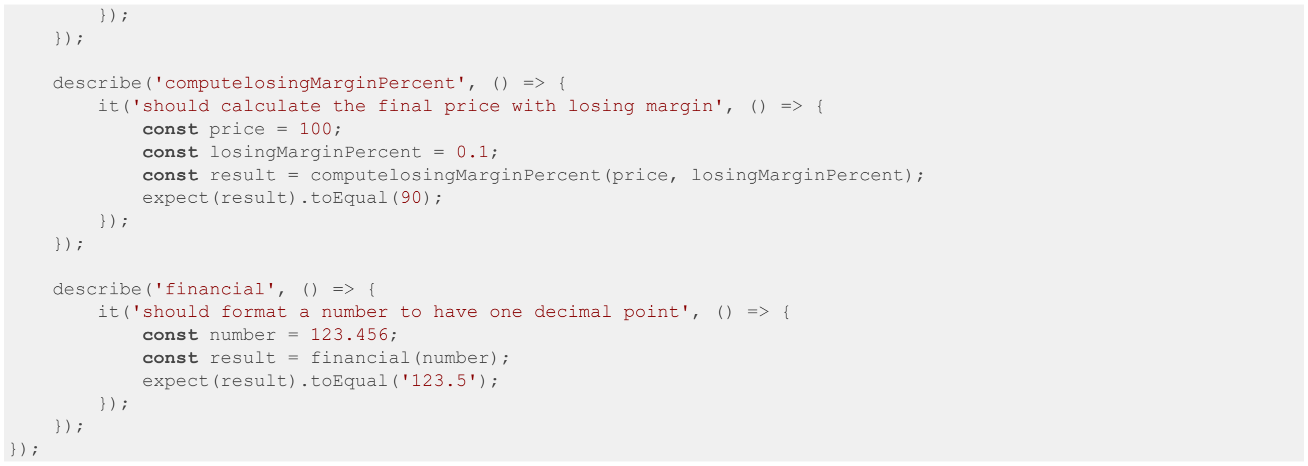
accurately represents a set of orders using the trading platform chosen by the user.

Unit Testing for RSI Bot Functions and Worker Threads

Jest was utilized as the unit testing framework to create the unit tests. Each test case is written using the Jest syntax and follows the Arrange-Act-Assert pattern to set up the test scenario, perform the necessary actions, and verify the expected outcomes. Fig. 8 show sample of test code.

```
const {
    runRsiIndecator,
    getCurrentPrice,
    computeWinningMarginPercent,
    computelosingMarginPercent,
    financial
} = require('./path-to-your-file'); // Replace with the actual path

describe('RSI Bot Functions', () => {
    describe('getCurrentPrice', () => {
        it('should fetch the current price from Binance', async () => {
            const binanceClient = {
                prices: jest.fn().mockResolvedValue({
                    SYMBOL: '123.45'
                }),
            };
            const pair = 'BTC/USDT';
            const result = await getCurrentPrice(binanceClient, pair);
            expect(result).toEqual({
                SYMBOL: '123.45'
            });
            expect(binanceClient.prices).toHaveBeenCalledWith({
                symbol: 'BTCUSDT'
            });
        });
    });

    describe('computeWinningMarginPercent', () => {
        it('should calculate the final price with winning margin', () => {
            const price = 100;
            const winningMarginPercent = 0.05;
            const result = computeWinningMarginPercent(price, winningMarginPercent);
            expect(result).toEqual(105);
```

ISSN (E): 2181-4570 ResearchBib Impact Factor: 6,4 / 2023



```
        });
    });

    describe('computelosingMarginPercent', () => {
        it('should calculate the final price with losing margin', () => {
            const price = 100;
            const losingMarginPercent = 0.1;
            const result = computelosingMarginPercent(price, losingMarginPercent);
            expect(result).toEqual(90);
        });
    });

    describe('financial', () => {
        it('should format a number to have one decimal point', () => {
            const number = 123.456;
            const result = financial(number);
            expect(result).toEqual('123.5');
        });
    });
});
```

Figure 8. Sample of test code

Conclusion

We have developed a bot-powered cryptocurrency trading platform. Its purpose is to enable simultaneous and diversified cryptocurrency trading. Following the Agile methodology.

We constructed the application through the following steps:

- Collecting requirements
- Designing
- Building
- Testing
- Deploying
- Reviewing

Our system uses case diagrams, use case tables, and sequence diagrams of system functions.

For system design, we created the ER diagram and user interface design. In the implementation and testing phase, we provided a detailed explanation of the implementation process and thoroughly tested the site's key functionalities. The resulting site features a subscriber interface that allows users to register, initiate trades, and select preferred currencies for trading.

ISSN (E): 2181-4570 ResearchBib Impact Factor: 6,4 / 2023

References

[1] "Medium" [Online]. Available: https://medium.com/@syantien/introduction-to-model-view-controller-mvc-pattern-8cbc693f043.

[2] J. Welles Wilder Jr. "New Concepts in Technical Trading Systems." Trend Research, 1978.

[3] "Investopedia" [Online]. Available: https://www.investopedia.com/terms/r/rsi.asp#citation-2.

[4] A. Shadi and P. Dennis, "Using Test Oracles and Formal Specifications with Test-Driven Development", International Journal of Software Engineering and Knowledge Engineering, July 2013, doi: 10.1142/S0218194013500113