Bilkent University
Department of Computer Engineering

Senior Design Project

T2333

Stock Vision

Final Report

21802713 Remzi Tepe remzi.tepe@ug.bilkent.edu.tr
21802228 Ekrem Polat ekrem.polat@ug.bilkent.edu.tr
21703049 Abdulkadir Erol abdulkadir.erol@ug.bilkent.edu.tr
21802520 Mert Atakan Onrat atakan.onrat@ug.bilkent.edu.tr
21702301 Nihat Bartu Serttaş bartu.serttas@ug.bilkent.edu.tr

Supervisor: Shervin Rahimzadeh Arashloo
Course Instructors: Erhan Dolak and Tağmaç Topal

19.05.2023

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Final Report **3**
1 Introduction 3
1.1 Purpose of the System 3
1.2 Definitions, Acronyms, and Abbreviations 4
1.3 Overview 4
2 Requirements Details 5
2.1 Functional Requirements 5
2.2 Non-functional Requirements 5
2.2.1 Accessibility 5
2.2.2 Availability 6
2.2.3 Performance 6
2.2.4 Reliability 6
2.2.5 Scalability 6
2.2.6 Security 6
2.2.7 Usability 6
2.2.8 Portability 7
2.2.9 Marketability 7
2.2.10 Extendibility 7
2.2.11 Maintainability 7
2.2.12 Flexibility 7
2.2.13 Modularity 7
2.2.14 Aesthetics 7
3 Final Architecture and Design Details 8
3.1 Overview 8
3.2 Subsystem Composition 9
3.3 Hardware/Software Mapping 9
3.4 Persistent Data Management 10
3.5 Access Control and Security 10
3.6 Global Software Control 11
3.7 Boundary Conditions 11
3.7.1 Initialization 11
3.7.2 Termination 11
3.7.3 Failure 12
3.8 Customer Subsystem 12
3.8.1 Customer UI Subsystem 12
3.8.2 User Subsystem 13

3.9 Admin Subsystem 14
3.9.1 Admin UI Subsystem 14
3.9.2 Admin Subsystem 15
3.10 Server Subsystem 16
3.10.1 Remote Server Layer 16
3.10.2 Data Storage Layer 16
4 Development/Implementation Details 16
5 Test Cases and Results 17
6 Maintenance Plans and Details 44
6.1 Flutter 44
6.2 Django 44
6.3 SDKs 44
6.4 API 44
6.5 Machine Learning Models 44
6.6 Chart Pattern Algorithms 45
6.7 Coins 45
6.8 Language 45
6.9 Bug Fixes 45
7 Other Project Elements 45
7.1 Consideration of Various Factors in Engineering Design 45
7.2. Ethics and Professional Responsibilities 46
7.3 Teamwork Details 47
7.4.1 Contributing and functioning effectively on team 47
7.4.2 Helping create a collaborative and inclusive environment 47
7.4.3 Taking the lead role and sharing leadership on the team 47
7.4.4 Meeting Objectives 47
8 Conclusion and Future Work 48
9 Glossary 49
References 50

Final Report

Project Short-Name: Stock Vision

1 Introduction

The cryptocurrency market has evolved significantly over the last decade. As of 2022, the market cap has reached over $1.00 trillion, and over 300+ million people use/own cryptocurrencies [1]. This huge demand led to much research regarding the stock market for users to learn about. Even so, reaching for trustworthy information about the market and analyzing the stock market have always been struggles for users. Many so-called economists have emerged and tried to direct people with some controversial or even incorrect concepts about the market. It takes a lot of time for beginners to choose which cryptocurrencies to use and how to invest, while limitless ideas exist. Moreover, many cryptocurrencies arise and fail due to a lack of security, weak teams, scamming, etc. This thriving industry has become very risky, especially for beginners who need to learn what and how to invest their money.

Therefore, in the project, we aim to design a hands-on experience simulation app where users can invest the application's fake currency (given in any real currency such as Bitcoin or even dollars) and get AI-based results and tips in real-time. This way, users can learn the fundamentals of investing in cryptocurrencies without losing money or relying on anyone.

This final report describes requirements, final architecture, design details, development/implementation details, test cases and results, maintenance plan and details, other project elements, and conclusion and future work.

1.1 Purpose of the System

Stock Vision is a mobile application that provides an educational environment for people interested in the stock market. The application will allow individuals to buy/sell coins with StockV’s currency, which enables them to invest comfortably without risking any money loss. The most important purpose of the application is to give investment tips to the user with the

help of machine learning algorithms by reading the graphic values of the coin selected by the user. The user can invest by considering these tips that the application gives him, and more importantly, by learning these tips, he can have a better idea of how to invest better. Moreover, he can make profitable investments by putting these tips learned in practice into real life. In this regard, to increase the probability of giving correct tips to users, Stock Vision will collect feedback from the users, which will be used to check the corresponding algorithm's correctness.

1.2 Definitions, Acronyms, and Abbreviations

V-Tip: The expected behavior of the coin given by the StockVision as a hint when the coin's values matched at least %75 of the training dataset.

V-Prob: The expected probability of realization of a given V-Tip as a percentage.

V-Plans: The available premium plans for the Stock Vision.

1.3 Overview

Stock Vision is a mobile application that works in iOS and Android. It provides a real-time stock/ETF simulation using past stock/ETF data. It allows users to invest using our application's fake currency and get real-time results/tips about their investments. At first, the user will sign up, if not signed yet, and then log in to the system. After logging in, users can see and search real-time stock/ETF values. Stock market graphs of each cryptocurrency can be seen daily, weekly, monthly, or even yearly. Users can go through these cryptocurrencies and see the data provided and read the tips given by our AI algorithms. More importantly, each user will start with a specific amount of fake currency and be able to use this fake currency to invest in stocks/ETFs. When users make investments, they can see tips from our AI on investing their money, the probability of values increasing/decreasing, and receive real-time results when gaining or losing money. These tips allow users to practice their graph reading abilities and see the results in real-time. If the users run out of the application's fake currency, they can start over with the initial fake currency.

Furthermore, Stock Vision has graph-reading algorithms using reliable stock graph patterns and providing a real-time tipping mechanism for the graphs. Analyzing stock values will be done with image processing, and a wide range of datasets of stock graph patterns will be used. Moreover, users' investments will be stored in our database, and the user data will be analyzed. Some tips and results will be provided after these analyses.

2 Requirements Details

2.1 Functional Requirements

- Users must log in or register to the system before using the application.
- The system should define some fake coins to users for investing in the market.
- The system must advise users by evaluating users' investments/transactions with AI.
- The system must give tips to users by evaluating the previous data about the currencies.
- The system must have two types of accounts: Free and Premium. Premium users can get more detailed tips and fake coins to invest in the market.
- The system must show users extended graphs related to predictions on currencies.
- The system must save all information about the currencies' progress, predictions, and graphs in the database so other users can benefit from it.
- Users can save their favorite currency pairs to see their progress.
- Users can change/delete their profile and related data in the database.

2.2 Non-functional Requirements

2.2.1 Accessibility

- Android Jelly Bean, v16, 4.1, and iOS 8 or newer versions are required for users to use the system.

2.2.2 Availability

- The application will use past and present cryptocurrency data from reliable stock market sites. Unless these websites are down (it is very rare), the application can be used properly.

2.2.3 Performance

- Displaying the home screen to users should be under 5 seconds.
- Updating users' data on investments should take under 1 second.

2.2.4 Reliability

- Authentication is required to join the system.
- A server crash or power outage should not result in data loss.

2.2.5 Scalability

- The servers can be extended easily.
- Many other features can be added easily without losing performance.

2.2.6 Security

- A unique sign-up/log-in process will be done.
- The data will be put in reliable servers like Firebase, so data loss will not be encountered.

2.2.7 Usability

- The GUI of the application will be very user-friendly since the users are not expected to be an expert in using mobile applications.
- Any problems encountered by users can be reported so that sustainability can be managed and the bugs can be fixed.

2.2.8 Portability

- If the newer Android or IOS is installed, the system should not cause any errors.

2.2.9 Marketability

- The application can satisfy the users' needs since there is a high demand on the stock market.
- A marketing strategy on social media will be applied to the target audience.

2.2.10 Extendibility

- The application is scalable and can extend to new features without problems.

2.2.11 Maintainability

- The maintenance of the app can be done easily since the code, and the documentation is very-well designed.

2.2.12 Flexibility

- It's a cross-platform application: Android and IOS users can use it.
- In the application, Dart/Flutter is used, which is very trendy and follows the newest technologies.

2.2.13 Modularity

- The application consists of reusable components that do not overlap on top of each other.

2.2.14 Aesthetics

- Throughout the application, GUI is user-friendly and very easy to use.
- The visual design of the application is also very futuristic.

3 Final Architecture and Design Details

3.1 Overview

The final architecture for the application follows a 3-tier architecture approach consisting of a presentation layer, business logic layer, and data storage layer.

The presentation layer, or the front-end component of the application, is developed using the Flutter framework. It will be responsible for providing a user-friendly interface for the application, allowing users to simulate cryptocurrency investments using the app's virtual fake currency. The front-end component will communicate with the business logic layer through RESTful APIs provided by the back-end component.

The business logic layer or the back-end component of the application is developed using the Django REST framework. It will be responsible for handling the application's business logic, including the AI-based algorithms that provide users with real-time investment advice and tips. The back-end component will also provide the necessary APIs for communication between the front-end and data storage layers. In addition, the back-end component will handle the persistent data storage using the PostgreSQL database.

The data storage layer is responsible for storing all the data generated by the application and the dataset for AI-based algorithms. In this project, AWS cloud storage will be utilized to provide scalable and reliable storage options. PostgreSQL will be used as the relational database management system to store and retrieve application data.

Overall, this 3-tier architecture approach aims to provide a modular, scalable, and maintainable software architecture for the application. Using modern technologies and frameworks such as Flutter, Django REST framework, PostgreSQL, and AI-based algorithms provides a cutting-edge and user-friendly experience for users to learn about cryptocurrency investments.

The next section aims to provide a clear understanding of the StockVision app's architecture and logic through the following topics: subsystem decomposition,

hardware/software mapping, persistent data management, access control, software control, and boundary conditions. Each topic will be explained in detail to provide a comprehensive overview of the application structure.

3.2 Subsystem Composition

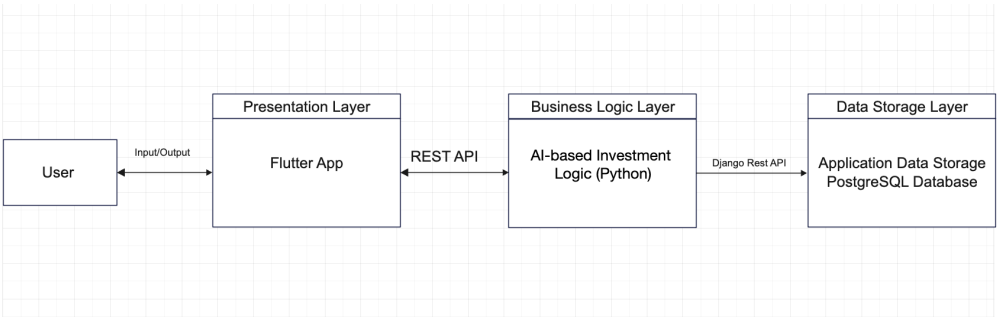

Figure 1: Subsystem Composition

3.3 Hardware/Software Mapping

As can be seen, the StockVision app relies on 3-tier architecture. So that it can easily be scalable, modifiable, and logical, each tier requires different hardware and software components to support its functionality. The presentation layer relies on mobile devices to run the Flutter framework and the operating system (e.g., iOS, Android) that supports it. The business logic layer requires servers or cloud infrastructure to run the Django REST framework, the Python scripts for AI-based investment logic, and the operating system (e.g., Linux, Windows Server) that supports them. Finally, the data storage layer requires servers or cloud infrastructure to run the PostgreSQL database management system, the operating system (e.g., Linux, Windows Server), and AWS cloud storage for scalable and reliable storage options.

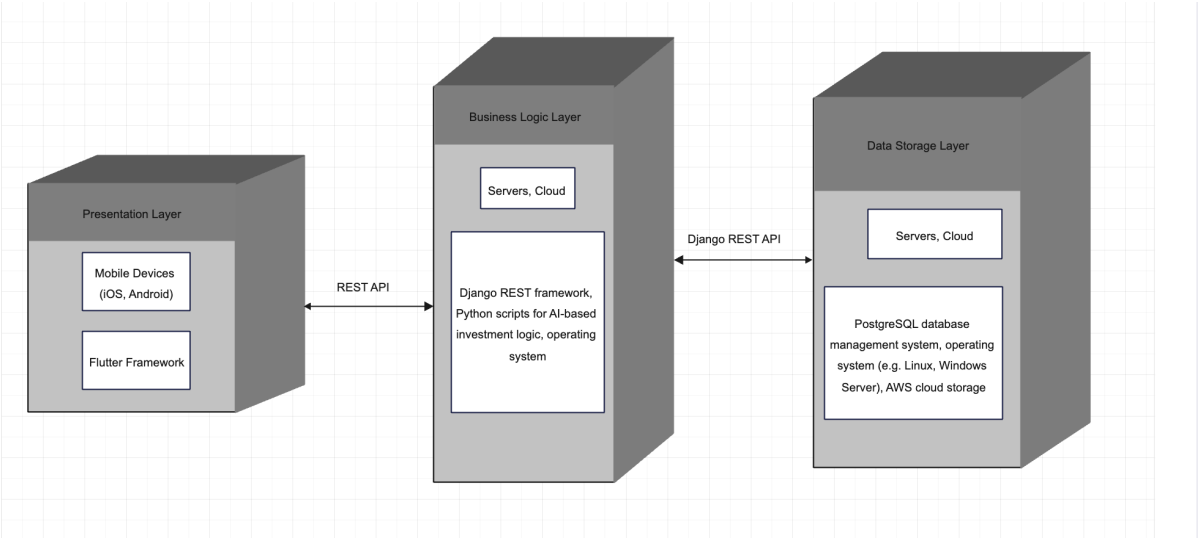

Figure 2: 3-Tier Layer Structure

3.4 Persistent Data Management

For StockVision, users' personal information, their balance, and the data about AI models are persistent data.

For StockVision, a persistent data management system is crucial to keep track of user information, transaction history, and AI training data. To accomplish this, we chose to use PostgreSQL as our database management system and Amazon Web Services (AWS) as our cloud computing platform. We opted for PostgreSQL due to its reliability, scalability, and robust features such as indexing, data integrity checks, and transactional support. In AWS, we utilized the Relational Database Service (RDS) to deploy and manage our PostgreSQL instance, ensuring high availability and scalability. With this setup, we can easily store and retrieve user data, transaction history, and AI training data, allowing us to improve our AI model's performance continuously.

3.5 Access Control and Security

In StockVision, users have a Free Trial and Premium. Both types of users' sign-in methods are the same: only entering a name, mail, and password. All the passwords are hashed and secured in the server in case of leaks. Moreover, if users want to buy premium packages,

they should enter their credit card information and buy the product. Also, their credit card information will not be saved for security reasons.

So, since their password is secured with hashing, they will be safe against any secured control and monitor user data, and the database will be used actively.

3.6 Global Software Control

In StockVision, event-driven programming will make our app more responsive and interactive. By triggering actions in response to user input or system notifications, we can provide a more intuitive and user-friendly experience. For example, when a user wants to learn the approximate value of Bitcoin in two months, they can find it by clicking on the coin's details.

Also, multithreading will be used to improve the performance of our app. We can handle multiple requests by running multiple threads since the cryptocurrency data is constantly coming, and we need to maintain its durability and stability. Otherwise, it is going to be a problem.

3.7 Boundary Conditions

3.7.1 Initialization

The app must be installed and launched successfully on a compatible device. The user must have a stable internet connection to access real-time data and features. The user must have a valid account with the app, either created by signing up or logging in.

3.7.2 Termination

The user can close the app or log out of their account easily without losing data. The app can properly save any changes or data the user makes before closing or logging out. The app should not cause any errors or crashes during the termination process.

3.7.3 Failure

If any bug happens while the user interacts with the application, for example, while trying to see the prediction on Bitcoin in 2024, the user should be able to email the support if the app crashes.

When there is no internet connection, users should be notified to connect to the Internet immediately to get the current data and predictions.

If there is a technical problem with AI models, all users should be notified by email to wait until the problem is solved.

If there is a danger of losing data on the server side, all users should also be notified to avoid any conflict and data loss in the process.

3.8 Customer Subsystem

3.8.1 Customer UI Subsystem

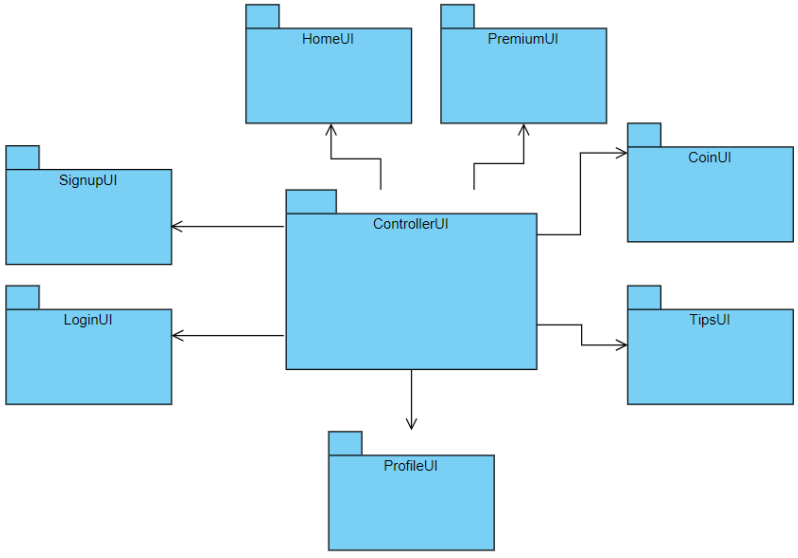

Figure 3: UI Subsystem of a Customer

ControllerUI: Controller of the user interface of the user. Controls the user input and manages the responses of the user.

SignupUI: Signup User Interface

LoginUI: Login User Interface

HomeUI: Homepage User Interface where users can see the coins with their graphs, current values, and percentage increase/decrease. Moreover, users can access some other pages, such as the coin page, profile page, premium page, etc.

PremiumUI: Premium User Interface where users can upgrade their service and make the payment on the page.

CoinUI: Coin User Interface where users can see the details of a coin's current value and percentage increase/decrease from here. Daily, weekly, monthly, and yearly changes can be seen.

TipsUI: Tips User Interface where users can access some tips (more tips with premium options) provided through our algorithms.

ProfileUI: Profile User Interface where users can see their profile information and update some changes.

3.8.2 User Subsystem

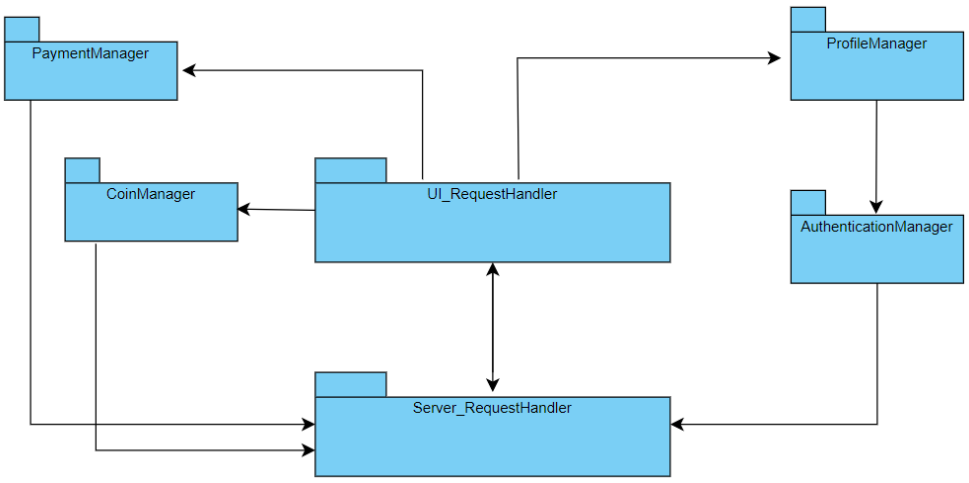

Figure 4: User Subsystem of a Customer

Server_RequestHandler: Creates responses when requests arrive in the system and control the UI system with service requirements.

UI_RequestHandler: Sends information to the server and manages responses that come from UI.

ProfileManager: Manages profile information and updates when required.

AuthenticationManager: Controls the safe connection to the system.

PaymentManager: Controls the payment for becoming premium users.

CoinManager: Controls the coins and their tips in the system.

3.9 Admin Subsystem

3.9.1 Admin UI Subsystem

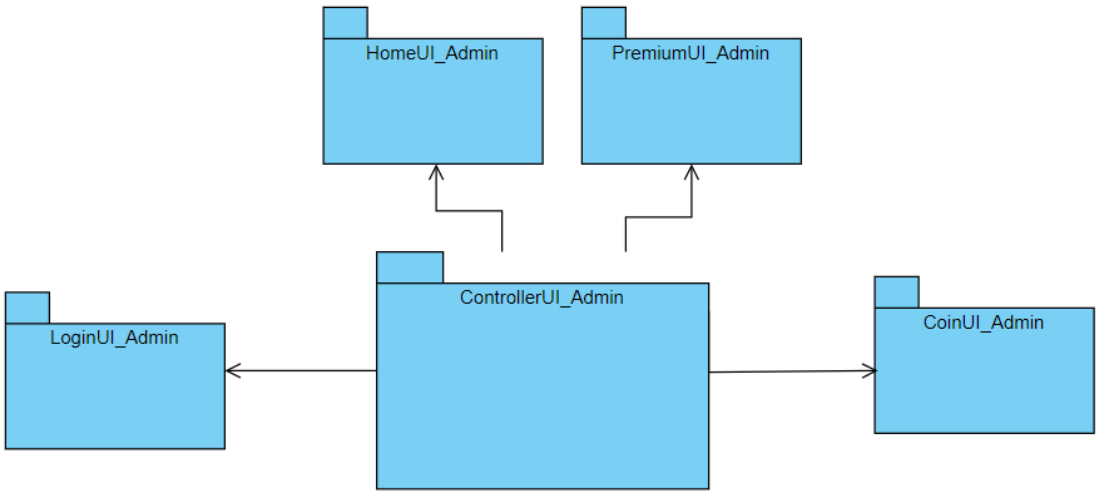

Figure 5: UI Subsystem of an Admin

ControllerUI_Admin: Controller of the user interface of the admin. Controls the user input and manages the responses of the admin.

LoginUI_Admin: Login User Interface

HomeUI_Admin: Homepage User Interface where the admin can add or remove the coins with their graphs, current values, and percentage increase/decrease.

PremiumUI_Admin: Premium User Interface where the admin can update and manage premium users and their service.

CoinUI_Admin: Coin User Interface where the admin can update and manage coins and their tips according to being a premium or a regular user.

3.9.2 Admin Subsystem

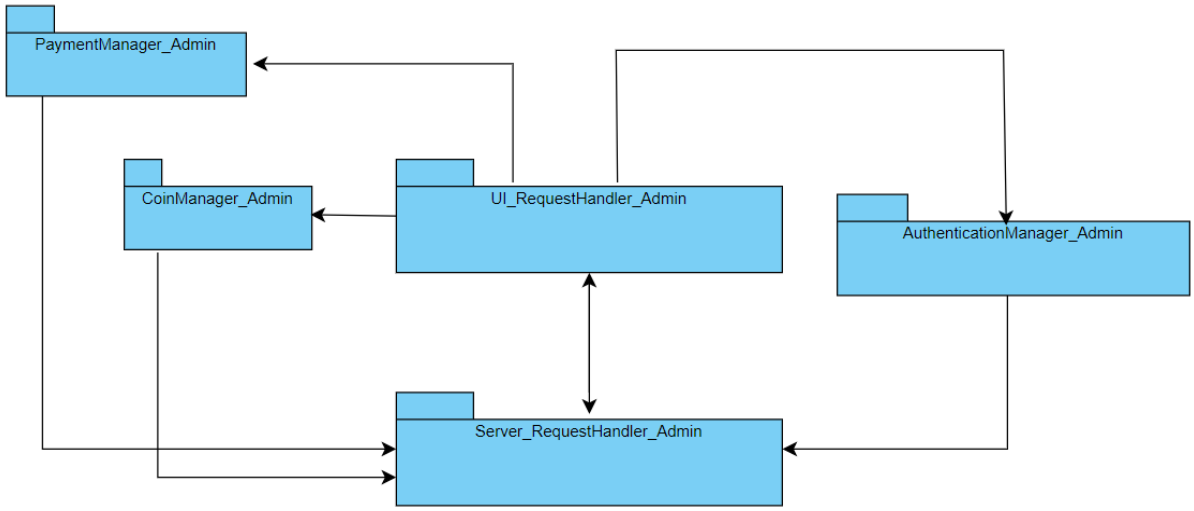

Figure 6: User Subsystem of an Admin

Server_RequestHandler_Admin: Creates responses when requests arrive in the system and control the UI system with service requirements.

UI_RequestHandler_Admin: Sends information to the server and manages responses that come from UI.

AuthenticationManager_Admin: Controls the safe connection to the system.

PaymentManager_Admin: Controls the payment for becoming premium users in case of a problem in the payment and can report to the user.

CoinManager_Admin: Controls the coins (additionally, adding and removing can be made here) and their tips in the system.

3.10 Server Subsystem

The server subsystem has two important layers: The remote server layer and the data storage layer.

3.10.1 Remote Server Layer

Remote Server Layer handles the HTTPS requests using REST API to our server (Django) with providing security. It is used for authentication. For the coins and their information, we use Binance API to get the details of the coins.

3.10.2 Data Storage Layer

Data required to be used later is stored in our data storage. PostgreSQL is used for the relational database management system to store and retrieve application data. Moreover, it is used for user information. Local storage is used for the coin logos to receive data faster and improve performance.

4 Development/Implementation Details

- Dart: We have used the Flutter framework to develop our application that uses Dart language. We have used a variety of Dart packages, such as convert, material, async, HTTP, kChart, etc., while implementing the application.
- Python: We have used the Django framework to develop the backend part of our application that uses Python language. In addition to Django packages, we have used several Python packages such as requests, numpy, pandas, matplot.lib, scipy.signal, io, base64, etc., while developing the backend.
- PostgreSQL: Our application uses PostgreSQL as a relational database management system.

- AWS: We have deployed our database to Amazon Relational Database Systems (RDS) to access the same data as a team.
- API Usage: We have used Binance API to fetch coin information such as sell price, buy price, volume, daily change, etc. Moreover, we have used Rest API to make communication between the front-end, i.e., Flutter application, and the back-end, i.e., Django application.
- Machine Learning: Our application uses each coin's 10 years' historical price values as a dataset. We have implemented the ARIMA model to train our datasets. After training our datasets, we displayed the predicted results in the application.
- Artificial Intelligence: We have used different Artificial Intelligence algorithms implemented in Python to capture chart patterns in the given coin's graph, such as Rectangle, Triangle, Head & Shoulders, and Round Bottom.
- Git / Github: We have used Git as a version control system. Also, we have used Github to share the code with the team. We can open issues, assign teammates to the issues, and open/close pull requests using Github.

5 Test Cases and Results

In the Test Cases section, there are 61 test cases designed for StockVision. When assigning a priority/severity level, we have used the following criteria:

Critical: The tested functionality has a critical role in the application and it's a must.
Major: The tested functionality prevents the users from using the application correctly.
Minor: The tested functionality has no major effects.

Test ID	TC#1
Test Type/Category	Functional, Safety
Title	Check if the users successfully sign up
Procedure of testing steps	1. Check whether the email is not already used 2. Check whether the email is valid 3. Check whether the password is valid

Expected results	Already used emails should not be selected. The user should use a unique, valid email and a valid password.
Priority/Severity	Critical
Date Tested and Test Result	01.04.2023 - Successful

Test ID	TC#2
Test Type/Category	Functional, Safety
Title	Check if the users successfully log in
Procedure of testing steps	1. Check whether the email is already in the database 2. Check whether the password matches the email
Expected results	Not signed-up emails should not be written. The user should write the matched password with the email.
Priority/Severity	Critical
Date Tested and Test Result	01.04.2023 - Successful

Test ID	TC#3
Test Type/Category	Functional, Safety
Title	Check if the forgotten password process is working
Procedure of testing steps	1. Check whether the forgotten password button sends a valid link to the responding email 2. Check whether the link in the email directs users to change their password 3. Check whether the new password chosen by the user is valid 4. Check whether the new password chosen by the user is updated with the old one
Expected results	The forgotten password was updated correctly with the new password.
Priority/Severity	Major

Date Tested and Test Result	01.04.2023 - Successful

Test ID	TC#4
Test Type/Category	Functional, Safety
Title	Check whether the profile of the users are matched and can be updated successfully
Procedure of testing steps	1. Check whether users can see their profile with the correct information and update it successfully.
Expected results	Users should be able to see their profile with their own information and can update their data correctly.
Priority/Severity	Major
Date Tested and Test Result	01.04.2023 - Successful

Test ID	TC#5
Test Type/Category	Functional, Integration
Title	Check whether the coins are retrieved accurately
Procedure of testing steps	1. Check whether the coins are retrieved with the correct logo, information, and data.
Expected results	All the coins retrieved with APIs should be correctly seen on the homepage.
Priority/Severity	Critical
Date Tested and Test Result	01.04.2023 - Successful

Test ID	TC#6
Test Type/Category	Functional, Integration
Title	Check whether the coins are updated in real-time

Procedure of testing steps	1. Check whether the coins are updated with the correct data.
Expected results	All the coins should be updated with their corresponding data in real-time on the homepage.
Priority/Severity	Critical
Date Tested and Test Result	01.04.2023 - Successful

Test ID	TC#7
Test Type/Category	Functional, Usability
Title	Check whether users can add a coin to their favorite coins.
Procedure of testing steps	1. Check whether the add to the favorite button of a coin is working and adding a coin to favorite coins
Expected results	Favourite coins should be added to the favorite coin list correctly with the add to the favorite button.
Priority/Severity	Minor
Date Tested and Test Result	05.04.2023 - Successful

Test ID	TC#8
Test Type/Category	Functional, Usability
Title	Check whether users can remove a coin to their favorite coins.
Procedure of testing steps	1. Check whether the add to the favorite button of a coin is working and adding a coin to favorite coins
Expected results	Selected coins should be removed from the favorite coin list correctly with the remove from the favorite button.
Priority/Severity	Minor
Date Tested and Test Result	05.04.2023 - Successful

Test ID	TC#9
Test Type/Category	Functional, Usability
Title	Check whether favorite coins can be listed on the homepage
Procedure of testing steps	1. Check whether favorite coins can be listed on the homepage correctly with its button
Expected results	Favourite coins should be listed with the list button on the homepage.
Priority/Severity	Minor
Date Tested and Test Result	05.04.2023 - Successful

Test ID	TC#10
Test Type/Category	Functional, Integration
Title	Check whether each coin's details are real-time and accurate
Procedure of testing steps	1. Check whether each coin's details are updated in real-time with the correct data. 2. More importantly, the graphs of each coin should be retrieved from APIs fast and accurately.
Expected results	Each coin details retrieved with APIs should be fast and correctly seen on the coin details page.
Priority/Severity	Critical
Date Tested and Test Result	05.04.2023 - Successful

Test ID	TC#11
Test Type/Category	Functional, Usability
Title	Check whether users can search and browse through each coin with their 15-min graphs.
Procedure of testing steps	1. Check 15-min time range of each coin's button is

	working correctly.
Expected results	15-min time range of each coin retrieved with APIs should be fast and correctly seen on the coin details page.
Priority/Severity	Major
Date Tested and Test Result	05.04.2023 - Successful

Test ID	TC#12
Test Type/Category	Functional, Usability
Title	Check whether users can search and browse through each coin with their 1-hour graphs.
Procedure of testing steps	1. Check 1-hour time range of each coin's button is working correctly.
Expected results	1-hour time range of each coin retrieved with APIs should be fast and correctly seen on the coin details page.
Priority/Severity	Major
Date Tested and Test Result	05.04.2023 - Successful

Test ID	TC#13
Test Type/Category	Functional, Usability
Title	Check whether users can search and browse through each coin with their 4-hours graphs.
Procedure of testing steps	1. Check 4-hour time range of each coin's button is working correctly.
Expected results	4-hour time range of each coin retrieved with APIs should be fast and correctly seen on the coin details page.
Priority/Severity	Major
Date Tested and Test Result	05.04.2023 - Successful

Test ID	TC#14
Test Type/Category	Functional, Usability
Title	Check whether users can search and browse through each coin with their daily graphs.
Procedure of testing steps	1. Check daily time range of each coin's button is working correctly.
Expected results	Daily time range of each coin retrieved with APIs should be fast and correctly seen on the coin details page.
Priority/Severity	Major
Date Tested and Test Result	05.04.2023 - Successful

Test ID	TC#15
Test Type/Category	Functional, Usability
Title	Check whether users can search and browse through each coin with their weekly graphs.
Procedure of testing steps	1. Check weekly time range of each coin's button is working correctly.
Expected results	Weekly time range of each coin retrieved with APIs should be fast and correctly seen on the coin details page.
Priority/Severity	Major
Date Tested and Test Result	05.04.2023 - Successful

Test ID	TC#16
Test Type/Category	Functional, Usability
Title	Check whether users can search and browse through each coin with their monthly graphs.

Procedure of testing steps	1. Check monthly time range of each coin's button is working correctly.
Expected results	Monthly time range of each coin retrieved with APIs should be fast and correctly seen on the coin details page.
Priority/Severity	Major
Date Tested and Test Result	05.04.2023 - Successful

Test ID	TC#17
Test Type/Category	Functional, Usability
Title	Check whether users can search and browse through each coin with their yearly graphs.
Procedure of testing steps	1. Check yearly time range of each coin's button is working correctly.
Expected results	Yearly time range of each coin retrieved with APIs should be fast and correctly seen on the coin details page.
Priority/Severity	Major
Date Tested and Test Result	05.04.2023 - Successful

Test ID	TC#18
Test Type/Category	Functional, Integration
Title	Check whether the coin’s buy and sell prices are retrieved from the APIs accurately
Procedure of testing steps	1. Check whether the sell prices and buy prices are displayed on the coin details page. 2. Check whether these prices match the data of Binance API.
Expected results	Buying and selling prices should be displayed correctly.
Priority/Severity	Major

Date Tested and Test Result	10.04.2023 - Successful

Test ID	TC#19
Test Type/Category	Functional, Integration
Title	Check whether the coin's buy and sell prices are updated from the APIs in real-time accurately
Procedure of testing steps	1. Check whether the sell prices and buy prices are displayed and updated in real-time on the coin details page. 2. Check whether these prices match the data of Binance API.
Expected results	Buying and selling prices should be updated correctly.
Priority/Severity	Major
Date Tested and Test Result	10.04.2023 - Successful

Test ID	TC#20
Test Type/Category	Functional, Security
Title	Check whether users cannot buy the coin after 2 minutes expires.
Procedure of testing steps	1. Check whether the user clicks can buy coin after 2 minutes expiration time.
Expected results	After 2 minutes, the user should get a "Time is up!" warning pop-up and be directed to the coin page.
Priority/Severity	Minor
Date Tested and Test Result	10.04.2023 - Successful

Test ID	TC#21

Test Type/Category	Functional, Usability
Title	Check whether the user cannot enter the amount value that its corresponding price exceeds their balance in their wallet while buying the coin.
Procedure of testing steps	1. Check whether the users can click on the buy button after the amount value that its corresponding price exceeds their balance in their wallet.
Expected results	If the amount value that its corresponding price exceeds their balance, a warning should be displayed, and buying operation should not be completed.
Priority/Severity	Major
Date Tested and Test Result	10.04.2023 - Successful

Test ID	TC#22
Test Type/Category	Functional, Usability
Title	Check whether the users cannot enter the price value that exceeds their balance in their wallet while buying the coin.
Procedure of testing steps	1. Check whether the users can click on the buy button after the price value exceeds their balance in their wallet
Expected results	If the price value exceeds their balance in their wallet, a warning should be displayed, and buying operation should not be completed.
Priority/Severity	Major
Date Tested and Test Result	10.04.2023 - Successful

Test ID	TC#23
Test Type/Category	Functional, Usability
Title	Check whether users cannot sell the coin after 2 minutes

	expires.
Procedure of testing steps	1. Check whether the user clicks can sell coins after 2 minutes expiration time.
Expected results	After 2 minutes, the user should get a “Time is up!” warning pop-up and be directed to the coin page.
Priority/Severity	Minor
Date Tested and Test Result	10.04.2023 - Successful

Test ID	TC#24
Test Type/Category	Functional, Usability
Title	Check whether the user cannot enter the amount value that its corresponding price exceeds their balance in their wallet while selling the coin.
Procedure of testing steps	2. Check whether the users can click on the sell button after the amount value that its corresponding price exceeds their balance in their wallet.
Expected results	If the amount value that its corresponding price exceeds its balance, a warning should be displayed, and the selling operation should not be completed.
Priority/Severity	Major
Date Tested and Test Result	10.04.2023 - Successful

Test ID	TC#25
Test Type/Category	Functional, Usability
Title	Check whether the users cannot enter the price value that exceeds their balance in their wallet while selling the coin.
Procedure of testing steps	2. Check whether the users can click on the sell button after the price value exceeds their balance in their wallet

Expected results	If the price value exceeds their balance in their wallet, a warning should be displayed, and the selling operation should not be completed.
Priority/Severity	Major
Date Tested and Test Result	10.04.2023 - Successful

Test ID	TC#26
Test Type/Category	Functional, Usability
Title	Check whether the users can buy coins with their balance.
Procedure of testing steps	1. Check whether the users can buy a coin with their balance. 2. Check whether the balance of the user is updated after the operation. 3. Check whether the wallet of the user is updated after the operation.
Expected results	If the user has the amount of money while buying, the coin can be bought from the user, and the balance and wallet should be updated accordingly.
Priority/Severity	Critical
Date Tested and Test Result	10.04.2023 - Successful

Test ID	TC#27
Test Type/Category	Functional, Usability
Title	Check whether the users can sell coins with their balance.
Procedure of testing steps	1. Check whether the users can sell a coin with their balance. 2. Check whether the balance of the user is updated after the operation. 3. Check whether the wallet of the user is updated after the operation.

Expected results	If the user has the amount of money while selling, the coin can be bought from the user, and the balance and wallet should be updated accordingly.
Priority/Severity	Critical
Date Tested and Test Result	10.04.2023 - Successful

Test ID	TC#28
Test Type/Category	Functional, Usability
Title	Check whether the users view their assets in their wallet
Procedure of testing steps	1. Check whether the user can view their assets in their wallet 2. Check whether the assets are updated in real-time
Expected results	Wallet should be updated correctly and seen by the users in the wallet page.
Priority/Severity	Major
Date Tested and Test Result	10.04.2023 - Successful

Test ID	TC#29
Test Type/Category	Functional, Usability
Title	Check whether the users view their gains and losses rate in their wallet page
Procedure of testing steps	1. Check whether the user can view the gains and losses of their assets in their wallet 2. Check whether the gains and the losses of assets are updated in real-time
Expected results	Wallet should be updated correctly and seen from the users in the wallet page.
Priority/Severity	Major

Date Tested and Test Result	10.04.2023 - Successful

Test ID	TC#30
Test Type/Category	Functional, Usability
Title	Check whether transactions are working properly.
Procedure of testing steps	1. Check whether the transactions are updated after each buys and sells operation. 2. Check whether the transactions give responses appropriately.
Expected results	Transactions should be working correctly after each buy and sell operation. If the buying or selling operations are not applicable, transactions should not be changed.
Priority/Severity	Major
Date Tested and Test Result	10.04.2023 - Successful

Test ID	TC#31
Test Type/Category	Functional, Usability
Title	Check whether transactions can be seen from the transaction list.
Procedure of testing steps	1. Check whether the transactions are added to the transaction list according to their corresponding order. 2. Check whether the transaction list can be seen from the transaction list page.
Expected results	Transactions should be seen correctly so that each user can see previous transactions in the transaction list.
Priority/Severity	Major
Date Tested and Test Result	10.04.2023 - Successful

Test ID	TC#32
Test Type/Category	Functional, Usability
Title	Check whether the user can insert line to the coins graph in the coin details page.
Procedure of testing steps	1. Check whether the insert line button adds a line to the graph on the coin details page if the line is not already on the graph.
Expected results	Line should be visible on the graph.
Priority/Severity	Minor
Date Tested and Test Result	20.04.2023 - Successful

Test ID	TC#33
Test Type/Category	Functional, Usability
Title	Check whether the user can remove the line from the coins graph in the coin details page.
Procedure of testing steps	1. Check whether the remove line button removes a line from the graph on the coin details page if the line is on the graph.
Expected results	Line should be removed from the graph.
Priority/Severity	Minor
Date Tested and Test Result	20.04.2023 - Successful

Test ID	TC#34
Test Type/Category	Functional, Usability
Title	Check whether the user can view the moving average prices on the graph on the coin details page.
Procedure of testing steps	1. Check whether the MA button views the moving

	average prices on the graph in the coin details page if moving average prices is not already on the graph.
Expected results	Moving average prices should be visible on the graph.
Priority/Severity	Minor
Date Tested and Test Result	25.04.2023 - Successful

Test ID	TC#35
Test Type/Category	Functional, Usability
Title	Check whether the user can view the Bollinger Bands indicator on the graph on the coin details page.
Procedure of testing steps	1. Check whether the Boll button views the Bollinger Bands indicator on the graph in the coin details page if the Bollinger Bands indicator is not already on the graph.
Expected results	Bollinger Bands indicator should be visible on the graph.
Priority/Severity	Minor
Date Tested and Test Result	25.04.2023 - Successful

Test ID	TC#36
Test Type/Category	Functional, Usability
Title	Check whether the user can view the Moving Average Convergence Divergence indicator on the graph on the coin details page.
Procedure of testing steps	1. Check whether the MACD button views the Moving Average Convergence Divergence indicator on the graph in the coin details page if the Moving Average Convergence Divergence indicator is not already on the graph.
Expected results	Moving Average Convergence Divergence indicator should be

	visible on the graph.
Priority/Severity	Minor
Date Tested and Test Result	25.04.2023 - Successful

Test ID	TC#37
Test Type/Category	Functional, Usability
Title	Check whether the user can view the KDJ indicator on the graph on the coin details page.
Procedure of testing steps	1. Check whether the KDJ button views the KDJ indicator on the graph in the coin details page if the KDJ indicator is not already on the graph.
Expected results	KDJ indicator should be visible on the graph.
Priority/Severity	Minor
Date Tested and Test Result	25.04.2023 - Successful

Test ID	TC#38
Test Type/Category	Functional, Usability
Title	Check whether the user can view the Relative Strength Index indicator on the graph on the coin details page.
Procedure of testing steps	1. Check whether the RSI button views the RSI indicator on the graph in the coin details page if the RSI indicator is not already on the graph.
Expected results	RSI indicator should be visible on the graph.
Priority/Severity	Minor
Date Tested and Test Result	25.04.2023 - Successful

Test ID	TC#39
Test Type/Category	Functional, Usability
Title	Check whether the user can view the William Overbought/Oversold Index indicator on the graph on the coin details page.
Procedure of testing steps	1. Check whether the WR button views the WR indicator on the graph in the coin details page if the WR indicator is not already on the graph.
Expected results	WR indicator should be visible on the graph.
Priority/Severity	Minor
Date Tested and Test Result	25.04.2023 - Successful

Test ID	TC#40
Test Type/Category	Functional, Usability
Title	Check whether the user can remove the secondary state on the graph on the coin details page.
Procedure of testing steps	1. Check whether the Remove Secondary State button removes the RSS on the graph in the coin details page if the RSS indicator is already on the graph.
Expected results	RSS indicator should be visible on the graph.
Priority/Severity	Minor
Date Tested and Test Result	25.04.2023 - Successful

Test ID	TC#41
Test Type/Category	Functional, Usability
Title	Check whether the user can hide volume from the graph on the coin details page.

Procedure of testing steps	1. Check whether the Hide Volume button hides the volume on the graph in the coin details page if the volume is already on the graph.
Expected results	Volume should be hidden from the graph.
Priority/Severity	Minor
Date Tested and Test Result	25.04.2023 - Successful

Test ID	TC#42
Test Type/Category	Functional, Usability
Title	Check whether the user can show volume on the graph on the coin details page.
Procedure of testing steps	2. Check whether the Show Volume button shows the volume on the graph in the coin details page if the volume is not already on the graph.
Expected results	Volume should be visible on the graph.
Priority/Severity	Minor
Date Tested and Test Result	25.04.2023 - Successful

Test ID	TC#43
Test Type/Category	Functional, Usability
Title	Check whether the user clicks on the chart patterns, list of the patterns is displayed on the screen
Procedure of testing steps	1. Check whether the chart pattern page is displayed when see chart patterns button is clicked. 2. Check whether captured chart patterns' buttons are enabled 3. Check whether non-captured chart patterns' buttons are disabled.
Expected results	When see chart patterns button is clicked, enabled and

	disabled chart patterns are seen according to our AI algorithms.
Priority/Severity	Major
Date Tested and Test Result	01.05.2023 - Successful

Test ID	TC#44
Test Type/Category	Functional, Integration
Title	Check whether the user can view captured Rectangle Patterns detected by our AI algorithm if the corresponding button is enabled.
Procedure of testing steps	1. Check captured patterns in the See Chart Patterns. 2. Check whether the pattern button is enabled if our AI algorithm detects it. 3. Check whether the captured patterns can be viewed.
Expected results	Captured patterns should be displayed if our AI algorithm catches the pattern.
Priority/Severity	Major
Date Tested and Test Result	01.05.2023 - Successful

Test ID	TC#45
Test Type/Category	Functional, Integration
Title	Check whether the user can view captured Head & Shoulders Patterns detected by our AI algorithm if the corresponding button is enabled.
Procedure of testing steps	1. Check captured patterns in the See Chart Patterns. 2. Check whether the pattern button is enabled if our AI algorithm detects it. 3. Check whether the captured patterns can be viewed.
Expected results	Captured patterns should be displayed if our AI algorithm catches the pattern.

Priority/Severity	Major
Date Tested and Test Result	01.05.2023 - Successful

Test ID	TC#46
Test Type/Category	Functional, Integration
Title	Check whether the user can view captured Triples Patterns detected by our AI algorithm if the corresponding button is enabled.
Procedure of testing steps	1. Check captured patterns in the See Chart Patterns. 2. Check whether the pattern button is enabled if our AI algorithm detects it. 3. Check whether the captured patterns can be viewed.
Expected results	Captured patterns should be displayed if our AI algorithm catches the pattern.
Priority/Severity	Major
Date Tested and Test Result	01.05.2023 - Successful

Test ID	TC#47
Test Type/Category	Functional, Integration
Title	Check whether the user can view captured Wedge Patterns detected by our AI algorithm if the corresponding button is enabled.
Procedure of testing steps	1. Check captured patterns in the See Chart Patterns. 2. Check whether the pattern button is enabled if our AI algorithm detects it. 3. Check whether the captured patterns can be viewed.
Expected results	Captured patterns should be displayed if our AI algorithm catches the pattern.
Priority/Severity	Major

Date Tested and Test Result	01.05.2023 - Successful

Test ID	TC#48
Test Type/Category	Functional, Integration
Title	Check whether the user can view captured Triangle Patterns detected by our AI algorithm if the corresponding button is enabled.
Procedure of testing steps	1. Check captured patterns in the See Chart Patterns. 2. Check whether the pattern button is enabled if our AI algorithm detects it. 3. Check whether the captured patterns can be viewed.
Expected results	Captured patterns should be displayed if our AI algorithm catches the pattern.
Priority/Severity	Major
Date Tested and Test Result	01.05.2023 - Successful

Test ID	TC#49
Test Type/Category	Functional, Integration
Title	Check whether the user can view captured Support & Resistance Patterns detected by our AI algorithm if the corresponding button is enabled.
Procedure of testing steps	1. Check captured patterns in the See Chart Patterns. 2. Check whether the pattern button is enabled if our AI algorithm detects it. 3. Check whether the captured patterns can be viewed.
Expected results	Captured patterns should be displayed if our AI algorithm catches the pattern.
Priority/Severity	Major
Date Tested and Test Result	01.05.2023 - Successful

Test ID	TC#50
Test Type/Category	Functional, Integration
Title	Check whether the user can view captured Rounding Bottom Patterns detected by our AI algorithm if the corresponding button is enabled.
Procedure of testing steps	1. Check captured patterns in the See Chart Patterns. 2. Check whether the pattern button is enabled if our AI algorithm detects it. 3. Check whether the captured patterns can be viewed.
Expected results	Captured patterns should be displayed if our AI algorithm catches the pattern.
Priority/Severity	Major
Date Tested and Test Result	01.05.2023 - Successful

Test ID	TC#51
Test Type/Category	Functional, Integration
Title	Check whether the user can view captured Flag Patterns detected by our AI algorithm if the corresponding button is enabled.
Procedure of testing steps	1. Check captured patterns in the See Chart Patterns. 2. Check whether the pattern button is enabled if our AI algorithm detects it. 3. Check whether the captured patterns can be viewed.
Expected results	Captured patterns should be displayed if our AI algorithm catches the pattern.
Priority/Severity	Major
Date Tested and Test Result	01.05.2023 - Successful

Test ID	TC#52

Test Type/Category	Functional, Integration
Title	Check whether the user can view captured Double Patterns detected by our AI algorithm if the corresponding button is enabled.
Procedure of testing steps	1. Check captured patterns in the See Chart Patterns. 2. Check whether the pattern button is enabled if our AI algorithm detects it. 3. Check whether the captured patterns can be viewed.
Expected results	Captured patterns should be displayed if our AI algorithm catches the pattern.
Priority/Severity	Major
Date Tested and Test Result	01.05.2023 - Successful

Test ID	TC#53
Test Type/Category	Functional, Integration
Title	Check whether the prediction is made properly.
Procedure of testing steps	1. Check the results of our ML algorithms 2. Check whether the result is appropriate
Expected results	Prediction should be an accurate prediction so that the tips we are giving to the user are working.
Priority/Severity	Critical
Date Tested and Test Result	10.05.2023 - Successful

Test ID	TC#54
Test Type/Category	Functional, Usability
Title	Check whether the predictions are displayed accurately.
Procedure of testing steps	1. Check the results of our ML algorithms 2. Compare it with the results of predictions and see

	whether they are displayed properly.
Expected results	Predictions should be accurate with their corresponding graphs.
Priority/Severity	Major
Date Tested and Test Result	10.05.2023 - Successful

Test ID	TC#55
Test Type/Category	Functional, Security
Title	Check if the users successfully log out
Procedure of testing steps	1. Check whether the logout button is clicked 2. Check whether the user is logged out
Expected results	The user should log out correctly after clicking the logout button.
Priority/Severity	Major
Date Tested and Test Result	10.04.2023 - Successful

Test ID	TC#56
Test Type/Category	Non-Functional, Integration
Title	Check whether the server is connected to Binance API correctly.
Procedure of testing steps	1. Check whether the settings of connecting Binance API are correct 2. Check whether the connection is stable. 3. Check whether the retrieved data is accurate.
Expected results	The connection should be made between the server and the API, and it should be working accurately.
Priority/Severity	Major

Date Tested and Test Result	05.04.2023 - Successful

Test ID	TC#57
Test Type/Category	Non-Functional, Performance
Title	Check whether the coins are updated every 2 seconds.
Procedure of testing steps	1. Check whether the coins are updated in 2 seconds.
Expected results	The coin data is retrieved from the APIs every 2 seconds.
Priority/Severity	Major
Date Tested and Test Result	01.05.2023 - Successful

Test ID	TC#58
Test Type/Category	Non-Functional, Performance
Title	Check whether the buy prices are updated every 0.5 seconds.
Procedure of testing steps	1. Check whether the buy prices are updated every 0.5 seconds.
Expected results	The buy price data is retrieved from the APIs every 0.5 seconds.
Priority/Severity	Major
Date Tested and Test Result	01.05.2023 - Successful

Test ID	TC#59
Test Type/Category	Non-Functional, Performance
Title	Check whether the sell prices are updated every 0.5 seconds.
Procedure of testing steps	1. Check whether the sell prices are updated every 0.5 seconds.

Expected results	The sell price data is retrieved from the APIs every 0.5 seconds.
Priority/Severity	Major
Date Tested and Test Result	01.05.2023 - Successful

Test ID	TC#60
Test Type/Category	Non-Functional, Performance
Title	Check whether the user wallet is updated every 0.5 seconds.
Procedure of testing steps	1. Check whether the user's wallet is updated every 0.5 seconds.
Expected results	The user wallet data is updated from the back-end every 0.5 seconds.
Priority/Severity	Major
Date Tested and Test Result	01.05.2023 - Successful

Test ID	TC#61
Test Type/Category	Non-Functional, Performance
Title	Check whether the save coins data are updated every 2 seconds.
Procedure of testing steps	1. Check whether saved coins are updated every 2 seconds.
Expected results	The saved coins data is updated from the back-end every 2 seconds.
Priority/Severity	Major
Date Tested and Test Result	01.05.2023 - Successful

6 Maintenance Plans and Details

Maintenance is a significant part of our project since it relies on the server.

6.1 Flutter

New upgrades on Flutter arrive, and the application should also be caught up with the new upgrades. Flutter packages can also get updated depending on the developers, and using the last version of it is very important due to UI or other related parts not being deprecated.

6.2 Django

Similar to Flutter, new upgrades on Django arrive, and the application should also be caught up with the new upgrades. Django packages can get updated depending on the developers, and using the last version of it is very important to implement backend features in the most efficient way.

6.3 SDKs

There are also some SDK updates that happen over time, and catching up with the SDKs is also important.

6.4 API

As an API, public Binance API is used, and it is a well-known API for developers who are in the stock market. Any upgrades should be followed through time so that not any errors occur throughout the application.

6.5 Machine Learning Models

New machine learning models can be invented in Computer Science. To predict the coin prices in the most proper way, new models can be integrated into the application.

6.6 Chart Pattern Algorithms

New chart patterns may arrive in the stock market. To provide the best experience to the users, new patterns algorithms should be implemented in the application.

6.7 Coins

New coins may arrive in the stock market. To provide the best experience to the users, new coins should be included in the application.

6.8 Language

In the application, everything is written in English, and it is not meant to be translated into another language. However, if it's requested, everything written in English could be translated into another language as well.

6.9 Bug Fixes

Due to the nature of programming, bugs, and other issues can occur while the application is used. Therefore, it is important to fix the issues so that it does not occur in the future.

7 Other Project Elements

7.1 Consideration of Various Factors in Engineering Design

Stock Vision is a mobile application that provides an educational environment for people interested in the stock market. The most important purpose of the application is to give investment tips to the user with the help of ine learning algorithms by reading the graphic values of the coin selected by the user. The aim of using machine learning algorithms is to provide more concise and accurate results for the users. Users should have a deeper understanding of analyzing the graphs.

	Importance Level	Effect
Cost	3	Application requires only developer cost. Therefore, cost does not have an immense effect on our application.
Reliability	5	Reliability is the most important factor in our app because the main purpose of the application is to provide reliable information for the user.
Safety	3	Since we do not store very important data of users, there is no data to be leaked. However, tips require come a certain level of safety.
Efficiency	4	Efficiency is important because
Environmental Considerations	0	There is nothing to consider due to environmental considerations in our app.
Availability	4	Availability is important in our app because the stock market is huge, and users should access our app easily.
Quality	4	Quality is important because our market is competitive, and our application should be high quality to attract users.

7.2. Ethics and Professional Responsibilities

In the application, although we do not collect private data of our users, like addresses, national IDs, etc., we collect emails and passwords, so we should be able to keep those data private.

Moreover, it is crucial to be careful about the APIs, libraries, and other things because of copyright issues. We used public Binance API and other free libraries so that we do not expect to encounter any copyright issues.

Throughout the application, the well-being and rights of individuals, communities, and the environment are prioritized. Moreover, there are a set of moral principles followed, such as honesty, integrity, and respect for others. While implementing the project, ethics is embraced, and professional responsibilities are fulfilled to create a trustworthy work environment. It was

crucial to build strong relationships with colleagues and clients and make a positive impact on society as a whole.

7.3 Teamwork Details

7.4.1 Contributing and functioning effectively on team

WP#	Work package title	Leader	Members involved
WP1	Project Specification Document	Atakan	Everyone
WP2	Analysis and Requirement Report	Bartu	Everyone
WP3	Back-end implementation	Kadir	Everyone
WP4	First Prototype	Ekrem	Everyone
WP5	Database Implementation	Remzi	Everyone
WP6	UI/UX Implementation	Atakan	Everyone
WP7	Second Prototype	Ekrem	Everyone
WP8	Final Implementation	Kadir	Everyone
WP9	Final Report	Remzi	Everyone

7.4.2 Helping create a collaborative and inclusive environment

We regularly hold meetings. Mostly it is weekly, but from time to time, we hold meetings two times a week. This way, we can discuss all the details we need to do before our implementations, reports, etc. This allows us to work more efficiently. We replace when there is an urgency in one of our team members and help the ones who are in more help.

7.4.3 Taking the lead role and sharing leadership on the team

On every step of the application, we try to hold a different leader and work with him efficiently. Each member is a responsible leader, and everyone dedicates himself to the projects more efficiently this way. Additionally, everyone improves his leadership skills with the help of this course, and this will be an important skill that we will use later on.

7.4.4 Meeting Objectives

In order to meet objectives as a team, deadlines are set at each step of development. Everyone on the team tried to catch up with the deadlines, and in case of emergencies, other team members

tried to compensate and help other teammates for the team spirit. Google Meet is used for team meetings, and these meetings are sent to Google Calendars for everyone. Google Docs is used for the reports so that everyone can work in the same workspace and track each other. WhatsApp is used for team communication instead of other communication channels like Slack, Gmail, etc., to have faster communication rather than using a professional one. Due to team members' schedules, some deadlines and meetings are postponed without affecting the course deadlines.

8 Conclusion and Future Work

Stock Vision is a mobile application designed to provide users with a real-time stock/ETF simulation using past stock/ETF data. It allows users to invest using our application's fake currency and get real-time results/tips about their investments. The application leverages advanced machine learning algorithms and data analysis techniques to forecast the future trends of various stocks and teaches users the fundamentals of stock/ETF predictions with hands-on experience simulation.

The primary goal of the application is to offer users a reliable tool for predicting stock market movements and providing tips and results by showing the analysis, especially for users who are new to the stock market. The application's intuitive and user-friendly interface allows users to invest in our fake currency, get real-time tips, and become familiar with the stock market.

Additionally, Stock Vision has algorithms that analyze stock graph patterns and provide real-time tips for the graphs. Analyzing stock values will be done with Machine Learning, and a wide range of datasets of stock graph patterns will be used. Moreover, users' investments will be stored in our database, and the user data will be analyzed. Some tips and results will be provided after these analyses.

The application employs cutting-edge algorithms to capture complex patterns and correlations within stock data. Users capture different patterns for different time range graphs, such as daily, weekly, monthly, or annually. There are 9 fundamental patterns captured by our algorithms, such as doubles, flag, head and shoulders, rectangle, rounding bottom, support and resistance, triangle, triples, and wedge. After analyzing graphs and capturing complex patterns, a unique machine-learning

technique, ARIMA, is used to predict results and provide accurate and up-to-date tips. After implementing LSTM, RNN, and Logistic Regression, ARIMA gave the most accurate results and was used in the application. The accuracy of the predictions is further enhanced through integration with a wide range of datasets, from a coin's everyday results for the last ten years, enabling the application to capture the predicted stock prices.

While implementing the project, there were some constraints that we encountered that could be used and researched more in future work. First of all, in the stock market, there are lots of factors that affect the prices of the coins. Our project would be more efficient if we could have made some analysis, like browsing tweets about the coins or other related factors that could affect coin prices. We primarily focused on only statistics and the current values of coins and ignored the social factors that could affect stock prices. Secondly, there could be more automated users implemented into the application with some characteristics (ignorant, risk taker, amateur, professional, etc.) so that the user can observe how different characters might act in the stock market, and this could be a great example for users as a guide.

9 Glossary

ETF: An exchange-traded fund is a mutual fund traded on stock exchanges. An ETF holds assets such as stocks, commodities, or bonds and often works with an arbitrage mechanism designed to keep diversions close to the net asset value.

Stock: A company's stock is all of the shares into which the company's ownership is divided. In American English, shares are collectively known as "stocks". A single share of stock represents partial ownership of the company in proportion to the total number of shares. This usually deserves the shareholder.

Amateurs: The ones that are still learning the stock market

Flunks: The ones that are not good at the stock market

Currency System: The currency system that we will provide

References

1- *Cryptocurrency trading simulator: Crypto Parrot*. Cryptocurrency Trading Simulator | Crypto Parrot. (n.d.). Retrieved March 13, 2023, from https://cryptoparrot.com/

2- Ltd., B. P. (2019, July 2). *CryptoSim: Market simulator*. App Store. Retrieved March 13, 2023, from https://apps.apple.com/us/app/cryptosim-market-simulator/id1468838417

3- Google. (n.d.). *Crypto master: Market analysis*. Google Play'de Uygulamalar. Retrieved March 13, 2023, from https://play.google.com/store/apps/details?id=com.astontek.crypto&hl=tr&gl=US

4- *Cryptocurrency exchange for Bitcoin, Ethereum & Altcoins*. Binance. (n.d.). Retrieved March 13, 2023, from https://www.binance.com/en

5- *Cryptocurrency prices, Market Cap & Live Crypto charts | kraken*. (n.d.). Retrieved March 12, 2023, from https://www.kraken.com/prices

6- *Coinbase - buy and sell Bitcoin, Ethereum, and more with trust*. (n.d.). Retrieved March 12, 2023, from https://www.coinbase.com/