



Bilkent University
Department of Computer Engineering

Senior Design Project

T2333
Stock Vision

Detailed Design Report

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

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

13.03.2022

Detailed Design Report	3
1 Introduction	3
1.1 Purpose of the System	3
1.2 Design Goals	4
1.2.1 Accessibility	4
1.2.2 Availability	4
1.2.3 Performance	4
1.2.4 Reliability	4
1.2.5 Scalability	4
1.2.6 Security	5
1.2.7 Usability	5
1.2.8 Portability	5
1.2.9 Marketability	5
1.2.10 Extendibility	5
1.2.11 Maintainability	5
1.2.12 Flexibility	6
1.2.13 Modularity	6
1.2.14 Aesthetics	6
1.3 Definitions, acronyms, and abbreviations	6
1.4 Overview	6
2 Current Software Architecture	7
2.1 Overview	7
2.2 Alternative and Current Solutions	8
2.3 Comparison	9
3 Proposed Software Architecture	10
3.1 Overview	10
3.2 Subsystem Composition	10
3.3 Hardware/Software Mapping	10
3.4 Persistent Data Management	11
3.5 Access Control and Security	11
3.6 Global Software Control	12
3.7 Boundary Conditions	12
3.7.1 Initialization	12
3.7.2 Termination	12
3.7.3 Failure	13
4 Subsystem services	14

4.1 Customer Subsystem	14
4.1.1 Customer UI Subsystem	14
4.1.2 User Subsystem	15
4.2 Admin Subsystem	16
4.2.1 Admin UI Subsystem	16
4.2.2 Admin Subsystem	17
4.3 Server Subsystem	18
4.3.1 Remote Server Layer	18
4.3.2 Data Storage Layer	18
5 Test Cases	18
5.1 User Interface Test Cases	19
5.1.1 Coin Detail Page Test Cases	19
5.1.2 Homepage Test Cases	20
5.1.3 Login Page Test Cases	21
5.2 Database Test Cases	21
5.3 Dataset and AI Test Cases	22
5.3.1 Data Pre-processing Test Cases	22
5.3.2 AI Model Training Test Cases	23
5.3.3 AI Model Evaluation Test Cases	23
6 Consideration of Various Factors in Engineering Design	24
7 Teamwork Details	25
7.1 Contributing and functioning effectively on team	25
7.2 Helping create a collaborative and inclusive environment	25
7.3 Taking the lead role and sharing leadership on the team	25
8 Glossary	25
References	27

Detailed Design 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 it. 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.

In this detailed design report, we describe current software architecture, proposed software architecture, subsystem services, test cases, and consideration of various factors in engineering design.

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 own 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 much 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 Design Goals

1.2.1 Accessibility

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

1.2.2 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.

1.2.3 Performance

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

1.2.4 Reliability

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

1.2.5 Scalability

- The servers can be extended easily.

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

1.2.6 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.

1.2.7 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.

1.2.8 Portability

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

1.2.9 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.

1.2.10 Extendibility

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

1.2.11 Maintainability

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

1.2.12 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.

1.2.13 Modularity

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

1.2.14 Aesthetics

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

1.3 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.

1.4 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 increase/decrease, and receive real-time results when gaining or losing money. With these tips, users can 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 tipping mechanism for the graphs in real-time. 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 Current Software Architecture

2.1 Overview

The proposed software 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 business logic of the application, including the AI-based algorithms that provide real-time investment advice and tips to users. 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 also the dataset for AI-based algorithms to use. 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.

2.2 Alternative and Current Solutions

Alternative solutions and current solutions refer to two different concepts. Alternative solutions refer to any existing solutions that address the same problem or provide similar functionality as our project. Current solutions, on the other hand, refer to any existing solutions that are currently being used by the target audience of our project.

Alternative solutions to our project include various cryptocurrency investment simulation apps, such as "CryptoParrot," "CryptoSim," and "CryptoMaster"(1)(2)(3). These apps provide a hands-on experience of investing in cryptocurrencies using fake currency. However, they do not offer AI-based suggestions to users. Other alternatives include websites and platforms that provide educational resources, such as Investopedia and Udemy.

Current solutions may include popular cryptocurrency exchanges like Coinbase, Binance, and Kraken(4)(5)(6). These exchanges allow users to buy, sell and trade various cryptocurrencies using real money. They also provide educational resources and analysis tools to help users make informed decisions. However, these platforms involve real money, which poses a risk to beginners who are still learning about the market.

2.3 Comparison

Our project's main advantage over alternative cryptocurrency investment simulation apps such as "CryptoParrot," "CryptoSim," and "CryptoMaster" is the use of AI-based suggestions derived from past data of the coins, allowing users to make more informed investment decisions. In that way, we offer an effective educational platform to the users. Our application also provides users with real-time values of coins through the Binance API.

Compared to alternative solutions, our project may not offer the same level of simplicity and ease of use, as some investment simulation apps provide more straightforward interfaces. Since our applications offer more features because of the AI-based suggestions, its interface would be slightly more complicated than the alternative solution applications.

Also, compared to current solutions like Coinbase, Binance, and Kraken, our project offers users a more educational and interactive experience. By providing a simulated investment environment with AI-based suggestions, users can gain more knowledge and confidence in making investment decisions. Another advantage is that users can use fake currency in our application. In that way, users can invest in any coin and gain more experience without losing real money. However, the applications like Binance, Coinbase, and Kraken do not have a feature like this. In these applications, users can only invest with real money.

One of the potential disadvantages of our project is that it offers a more limited range of features compared to some of the current solutions in the market. While our application provides educational resources for users to learn about trading and investment strategies, it may not offer features such as staking, sending coins to another wallet, or advanced trading options. This could be a drawback for users who are looking for a more comprehensive platform that offers a wider range of functionalities. However, education for these kinds of features can be easily added to our application.

3 Proposed Software Architecture

3.1 Overview

This 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

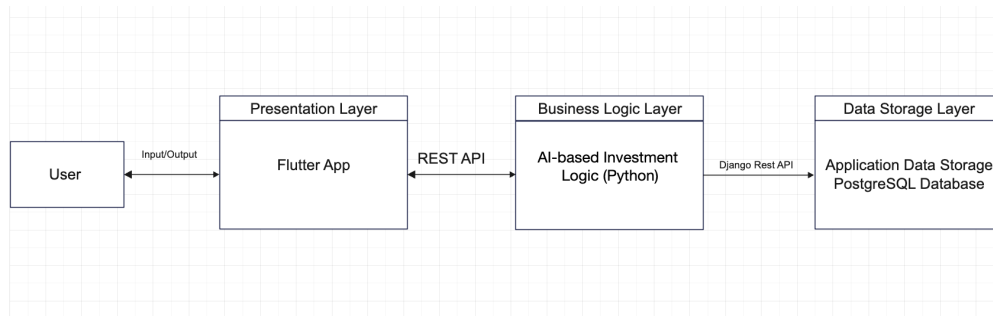


Figure 1: Subsystem Composition

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.

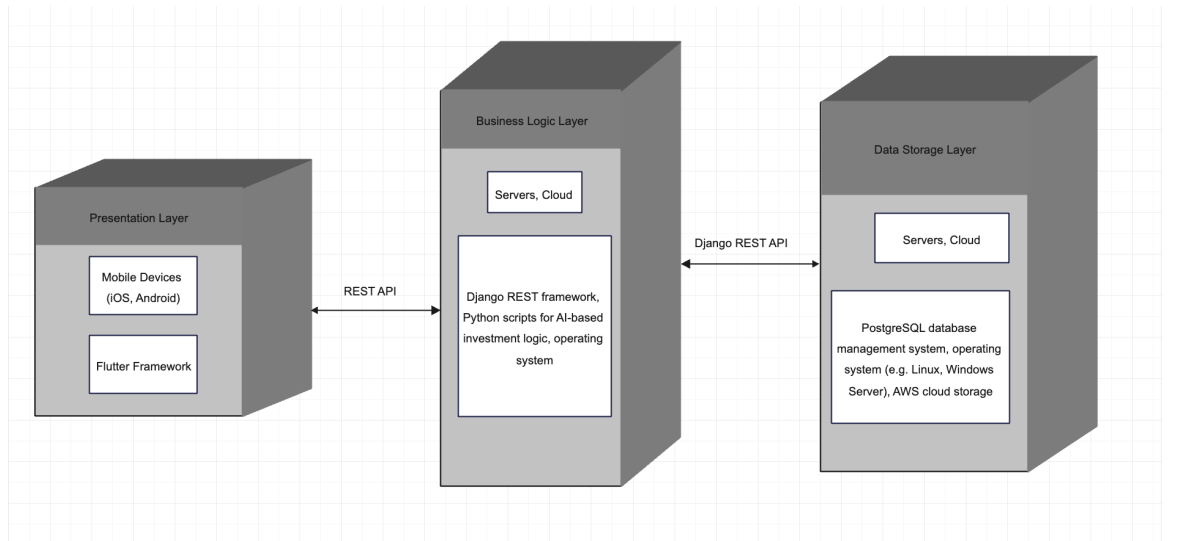


Figure 2: 3-Tier Layer Structure

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 the performance of our AI model 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, which is 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 the

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 security leak. To control and monitor user data, the database will be used actively.

3.6 Global Software Control

In StockVision, event-driven programming will allow our app to be 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 any data loss. 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, if the app crashes, the user should be able to email the support.

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.

4 Subsystem services

4.1 Customer Subsystem

4.1.1 Customer UI Subsystem

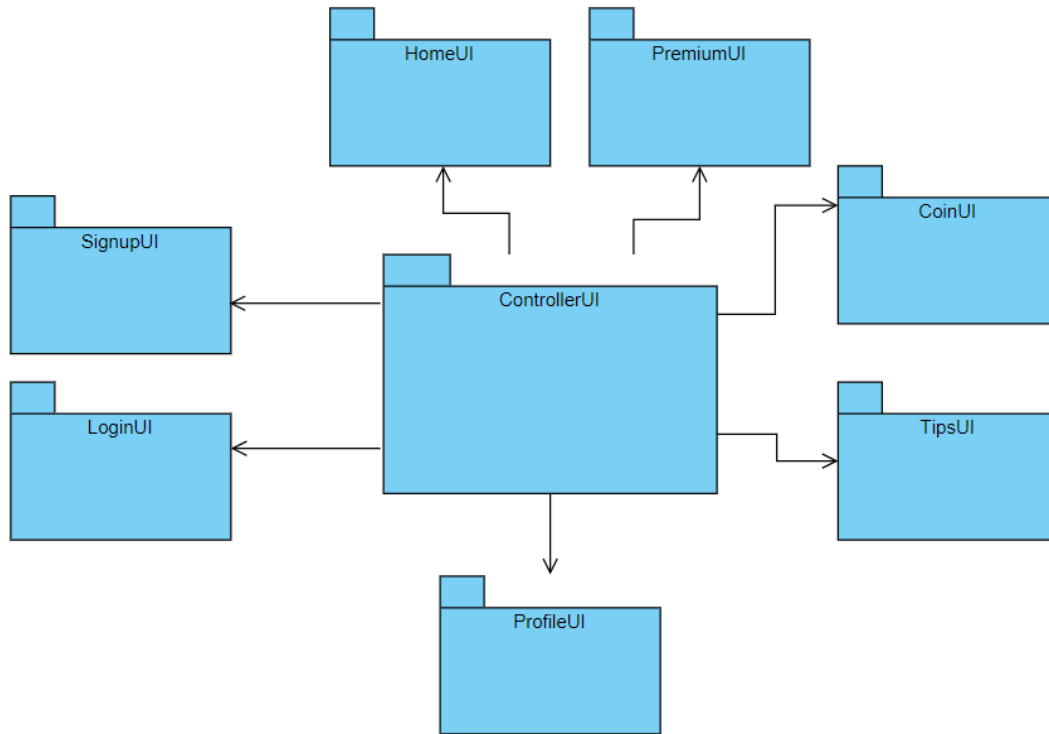


Figure 3: UI Subsystem of a Customer

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.

4.1.2 User Subsystem

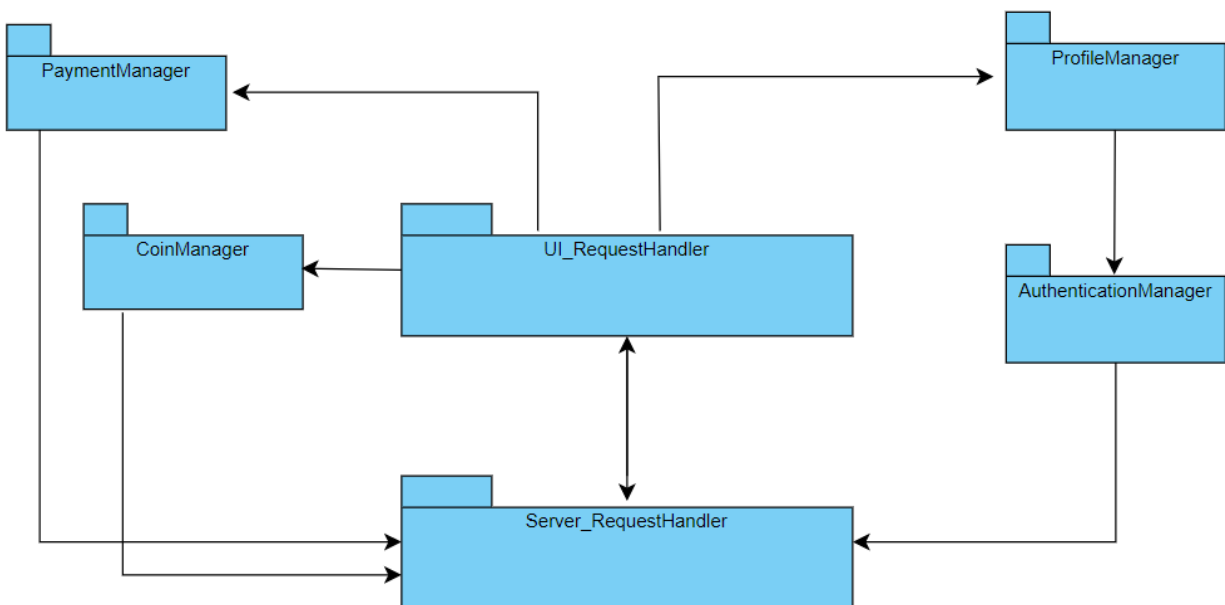


Figure 4: User Subsystem of a Customer

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.

4.2 Admin Subsystem

4.2.1 Admin UI Subsystem

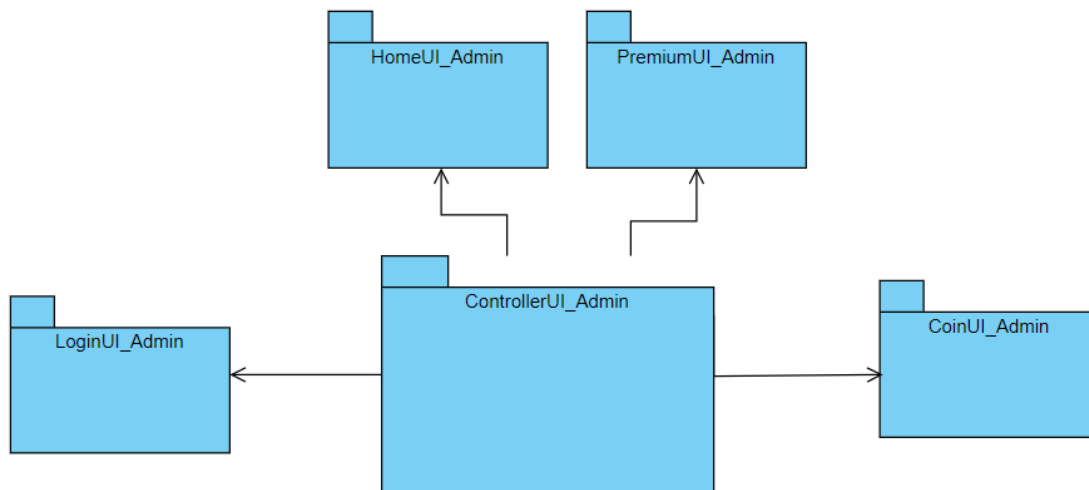


Figure 5: UI Subsystem of an Admin

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.

4.2.2 Admin Subsystem

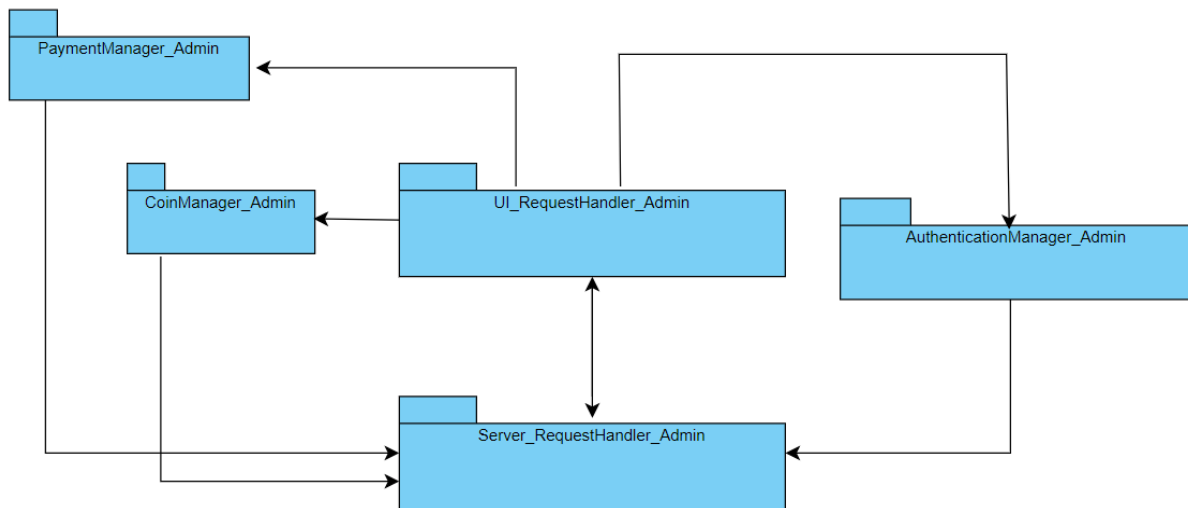


Figure 6: User Subsystem of an Admin

Server_RequestHandler_Admin: Creates responses when requests arrive in the system and controls 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.

4.3 Server Subsystem

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

4.3.1 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.

4.3.2 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.

5 Test Cases

Stock Vision is an application heavily dependent on the back-end software, which includes the AI, database, and the rest of the necessary back-end code. The most important component of Stock Vision needed testing is the AI for the stock exchange. However, most of the test cases were tested using the Android emulator Pixel 6 Pro API.

This section presents the test cases for the front-end application, the necessary REST format back-end component, database persistence and security, and AI training and testing. The AI of the application will be tested much more compared to the other components of Stock Vision.

5.1 User Interface Test Cases

There are many test cases for our Stock Vision front-end mobile application. The first of which is to test whether the coins and the data from the charts are correct. By testing this, we will ensure that after we train the AI to give tips on this page later, the tips will be much more accurate.

5.1.1 Coin Detail Page Test Cases

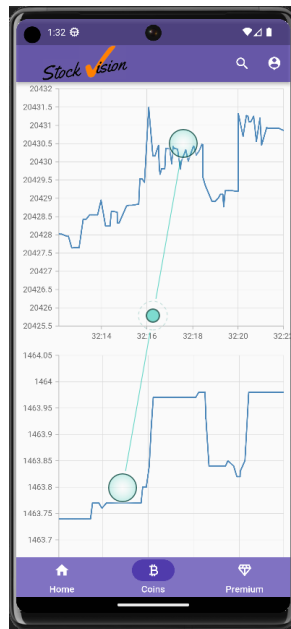


Figure 7: Stock Vision Coin Detail Page.

The first and most important test case is that saved coins will be observed for the price, which will be displayed on the live chart. To make it simpler for users, the charts only show the last time period (last 60 seconds, last hour, last day, etc.). The most important test case here is for performance. When the user displays the live chart for only one ETF, the performance is fine, however, if the user decides to display and compare more than one ETF (See figure X), observing multiple coins and using live charts affects the performance considerably.

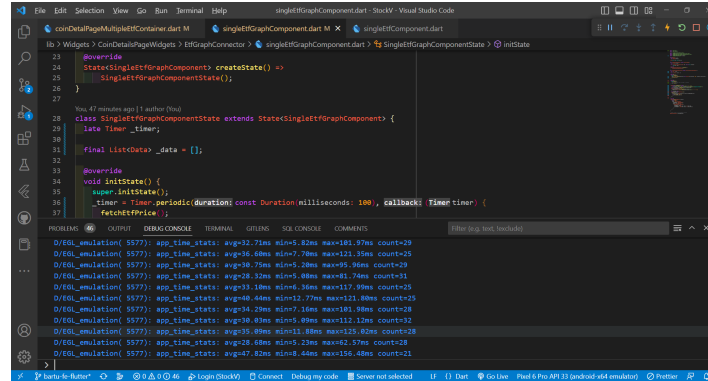


Figure 8: Live performance tests seen on the emulator debug console.

By using Visual Studio Code and an Android emulator, run and debug mode was used to test the performance issues on the coin detail page.

5.1.2 Homepage Test Cases

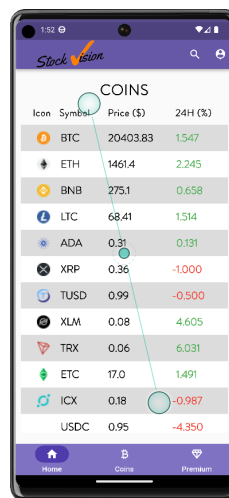


Figure 9: Homepage of Stock Vision.

On this page, users can search and display the price of ETFs, which are updated every second. Also, the users can see the daily changes given for the last 24 hours. There are many ETFs, and keeping track of each ETF displayed on the emulator made a major impact on performance. Same as before, VS Code and Android emulator was used to test the performance issues. Multi-threading is on implementation, however, dart/flutter and Android emulator do

not have the capability for actual multi-threading. Until tests on real mobile phones (both Android and iPhone), it is not possible to test multi-threaded applications.

5.1.3 Login Page Test Cases

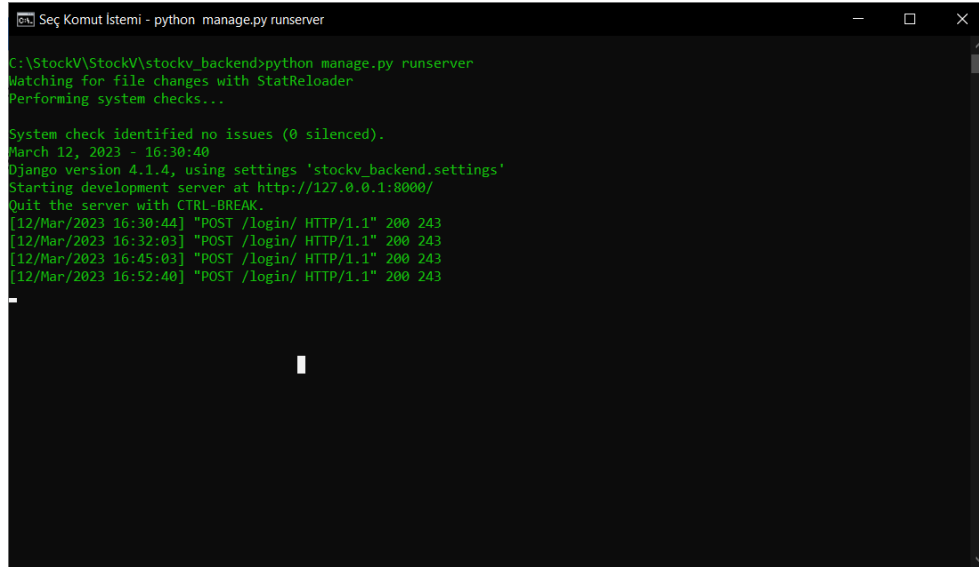


Figure 10: Login Screen of Stock Vision.

Login page and the rest of the helper pages (forgot password, sign-up, etc.), the only test cases on these pages were the REST connections with our back-end. Flutter HTTP package was used to make REST requests to our back-end application frequently. So far, no issue has been found regarding back and front-end application connections. When real users start using Stock Vision, more test cases will be tested.

5.2 Database Test Cases

Database of Stock Vision is implemented with PostgreSQL and Django server running with python. Since the front-end application is connected to the back-end, using REST requests, we have run persistence tests on our database.



```
Seç Komut İstemi - python manage.py runserver

C:\StockV\StockV\stockv_backend>python manage.py runserver
Matching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
March 12, 2023 - 16:30:40
Django version 4.1.4, using settings 'stockv_backend.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[12/Mar/2023 16:30:44] "POST /login/ HTTP/1.1" 200 243
[12/Mar/2023 16:32:03] "POST /login/ HTTP/1.1" 200 243
[12/Mar/2023 16:45:03] "POST /login/ HTTP/1.1" 200 243
[12/Mar/2023 16:52:40] "POST /login/ HTTP/1.1" 200 243
```

Figure 11: Django server running on python.

Also, by running these tests, we ensure the database is ready for the AI and our dataset connections.

5.3 Dataset and AI Test Cases

In this section, the most important test cases for the Stock Vision application are discussed in detail.

5.3.1 Data Pre-processing Test Cases

The quality and accuracy of the data used to train the AI model can significantly impact its performance. Therefore, it's crucial to preprocess the data to remove any outliers, noise, or irrelevant features. To ensure that the data preprocessing is effective, the following test cases are being checked:

Test case 1: Check for missing values and how they are handled during the preprocessing phase.

Test case 2: Test the effectiveness of the data normalization or scaling method used to ensure that the data is within the same range before being fed to the AI model.

Test case 3: Test the feature selection method used to determine which features should be included in the dataset to avoid overfitting or underfitting issues.

5.3.2 AI Model Training Test Cases

The accuracy and effectiveness of the AI model will determine the success of Stock Vision. Hence, it is crucial to test the model's performance during the training phase. Here are some test cases to consider:

Test case 4: Test the accuracy of the AI model during training by comparing the predicted outputs with the actual outputs. This will help identify any overfitting or underfitting issues that must be addressed.

Test case 5: Test the model's generalization ability by using a separate validation dataset to evaluate its performance. This will help ensure that the model can perform well on unseen data.

Test case 6: Test the model's robustness by introducing noise or outliers to the dataset during training to evaluate its performance in real-world scenarios.

5.3.3 AI Model Evaluation Test Cases

Once the AI model is trained, it's essential to evaluate its performance to determine its effectiveness in predicting stock prices/ETFs accurately. Here are some test cases to consider:

Test case 7: Test the model's accuracy by comparing the predicted stock prices with the actual prices. This will help identify any discrepancies and improve the model's accuracy.

Test case 8: Test the model's performance under different market conditions, such as bull or bear markets, to ensure it performs well in various scenarios.

Test case 9: Test the model's sensitivity to changes in the dataset to determine its robustness and identify any potential issues.

6 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 machine 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 in 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 Teamwork Details

7.1 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

7.2 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.

7.3 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.

8 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/>