Autonomous Trading Bot
SPROJ Report

LUMS
A Not-for-Profit University

Syed Talal Hasan 23100176
Suleman Mahmood 23100011

Ahmed Tahir Shekhani 23100197
Ali Asghar 23100198

Advisor:
School of Science and Engineering

Lahore University of Management Sciences

Submission Date

Acknowledgement and Dedication

Certificate

I certify that the senior project titled “**Add project title here**” was completed under my supervision by the following students:

____

____

____

and the project deliverables meet the requirements of the program.

------------------------------------ Date:

Advisor (Signature)

------------------------------------ Date:

Co-advisor (if any)

Table of Contents

List of Figures

1. Introduction

a. Introduction

A web application with an autonomous trading bot instance that will trade to generate profitable returns on stocks. The bot will be trained on the PSX data. The bot's decision will be based on the concepts of game theory, mathematical models, financial techniques, and especially artificial intelligence. The primary web app will allow the user to provide the bot's configuration, which includes, target return, risk appetite, and duration of the instance.

With recent advancements in deep learning frameworks and access to faster gpus, training complex models that can predict on time series data has opened new avenues to explore stock market trading. We plan on using models that have a memory component in them, such as LSTM (Long Short Term Memory) to make predictions and trades on the stock market.

The overall objective for the application would be to achieve the return target provided by the analyst while configuring the bot and minimize loss according to the risk factor provided. The potential users of this application would be trade analysts or managers who will use the bot to run its instances according to their requirements.

Technical details:

The project's tech stack would be Next.js for frontend web application, Flask for backend server, and PostgreSQL for our persistent storage. The application would follow three-tier architecture with a repository pattern for the persistent layer, models, and command layer for modifying the state.

b. Objective and Scope

The main objective of this web application is to create an autonomous trading bot that will generate profitable returns on stocks. The bot will be trained on PSX data using game theory, mathematical models, financial techniques, and artificial intelligence. The application's primary objective is to provide users with a configurable platform where they can set their return targets, risk appetite, and duration of the instance to achieve their desired financial goals.

The scope of this project is to develop a web application that will use advanced machine learning techniques to create an autonomous trading bot that can generate profitable returns on stocks. The application will be designed to allow users to configure the bot to their specific requirements, including their desired return targets, risk appetite, and duration of the instance.

The application will be targeted towards trade analysts or managers who will use the bot to run its instances according to their requirements. The bot's decisions will be based on advanced machine learning algorithms, including LSTM, that have a memory component in them to make accurate predictions on time-series data.

By using this web application, analysts can save time and effort in manual trading and optimize their profits. The application can help enhance business operations by providing an autonomous trading bot that can generate consistent returns with minimal manual effort. Additionally, it can provide greater insights into the stock market trends and patterns, leading to better-informed financial decisions.

c. Development Methodology

The tech stack of this application includes Next.js for the frontend, Flask for the backend server, and PostgreSQL for persistent storage. The application will follow the three-tier architecture with a repository pattern for the persistent layer, models, and command layer for modifying the state.

The development methodology used to build this project is the Agile methodology. The Agile methodology is a flexible and iterative approach that emphasizes collaboration, frequent feedback, and continuous improvement.

Using Agile methodology, the development team will work in short iterations, typically lasting one or two weeks, to deliver working software incrementally. The requirements and priorities of the project will be defined and refined over time through continuous feedback from the stakeholders.

The Agile methodology will enable the development team to respond to changes and updates quickly, as the project progresses. It will help to ensure that the project stays on track, is delivered on time, and meets the expectations of the stakeholders.

The project team will work in sprints, with each sprint lasting one or two weeks, and will consist of planning, development, testing, and review phases. The project's progress will be tracked using a project management tool, which will help to ensure that all team members are aware of the project's status and are working towards the same goals.

The Agile methodology will allow the development team to prioritize the features and functionalities that are most important to the stakeholders, which will help to ensure that the project delivers maximum value. Additionally, it will enable the development team to identify and address issues and risks early on in the development process, minimizing the impact of any potential problems.

Overall, using the Agile methodology will help to ensure that the project is delivered on time, meets the expectations of the stakeholders, and delivers maximum value to the end-users.

d. Contributions

The contributions of our work are primarily focused on developing an autonomous trading bot that uses advanced machine learning techniques to generate profitable returns on stocks. The innovative aspects of our work include:

1. Advanced Machine Learning Techniques: We have used advanced machine learning techniques, including LSTM, to create a trading bot that can accurately predict stock prices and make informed trading decisions. Our solution's predictive capabilities are better than traditional trading bots that use basic algorithms and statistical models.
2. User Configuration: Our solution allows users to configure the trading bot according to their specific requirements, including return targets, risk appetite, and duration of the instance. This customization feature sets our solution apart from other similar solutions that do not allow for such detailed user configuration.
3. Three-tier architecture with Repository Pattern: Our solution follows a three-tier architecture with a repository pattern for the persistent layer, models, and command layer for modifying the state. This architecture ensures that the code is modular, scalable, and easy to maintain.
4. Next.js and Flask Tech Stack: Our solution uses Next.js for the frontend and Flask for the backend server, which provides an efficient and fast application development environment. The use of these modern tech stacks sets our solution apart from other similar solutions that use outdated technologies.

Our solution is better than other similar solutions because it uses advanced machine learning techniques and allows for detailed user configuration. Additionally, our three-tier architecture and use of modern tech stacks make our solution more scalable, modular, and efficient than other solutions.

Moreover, the trading bot we developed is trained on PSX data, which gives our solution a unique selling point. PSX data is specific to the Pakistan Stock Exchange, and few trading bots are trained on this data. By using PSX data, our solution can provide better insights and predictions for traders in the Pakistan stock market.

Overall, our solution provides a more advanced, customizable, and efficient platform for autonomous stock trading, making it better than other similar solutions.

2. System Requirements — TALAL

Brief introduction of this chapter in a paragraph highlighting the content

a. System Actors

Actor Name	Description
Analyst	Analysts will configure and start the execution of the bot, analyze it’s statistics and assign investors to their bots.
Investor	Investors will use the bot for investment purposes and can view a summary report of the performance of the bot for the a specific range of time period (monthly, quarterly, half-yearly, yearly, or other combinations of days or months)

b. Functional Requirements

Analyst Requirements:

Sr#	Requirement
1	As an analyst, I want to be able to login with my credentials.
2	As an analyst, I want to be able to view recent highlights on my dashboard.
3	As an analyst, I want to be able to configure the parameters (such as balance, maximum drawdown, list of stocks) on which the bot will execute before it’s executed.
4	As an analyst, I want to be able to set stopping parameters (target percentage return, target balance return, duration) of the bot.
5	As an analyst, I want to be able to initiate the execution of the bot.

6	As an analyst, I want to be able to forcefully stop the execution of the bot.
7	As an analyst, I want to be able to instantiate new instances of the bot based on different configurations.
8	As an analyst, I want to be able to register an investor who can then view their bot's performance.
9	As an analyst, I want to get login credentials for an investor after registering them.
10	As an analyst, I want to be able to assign one or more bots to an investor.
11	As an analyst, I want to view the performance of all my bots that are currently executing.
12	As an analyst, I want to view the entire history of all my bots that I ran.
13	As an analyst, I want to pick any bot in history and view its entire performance.
14	As an analyst, I want to view multiple graphs of a bot such as balance over time, number of trades over time.
15	As an analyst, I want to choose which ML models to use for a bot.
16	As an analyst, I want to choose a list of indicators to use for a bot.
17	As an analyst, I want to configure the parameters of my chosen indicators for the bot.

Investor Requirements:

Sr#	Requirement
1	As an investor, I want to be able to login to my dashboard with the credentials provided by my analyst.
2	As an investor, I want to be able to see various graphs for my portfolio invested.
3	As an investor, I want to be able to see statistics of my bot's performance.

c. Non-functional Requirements

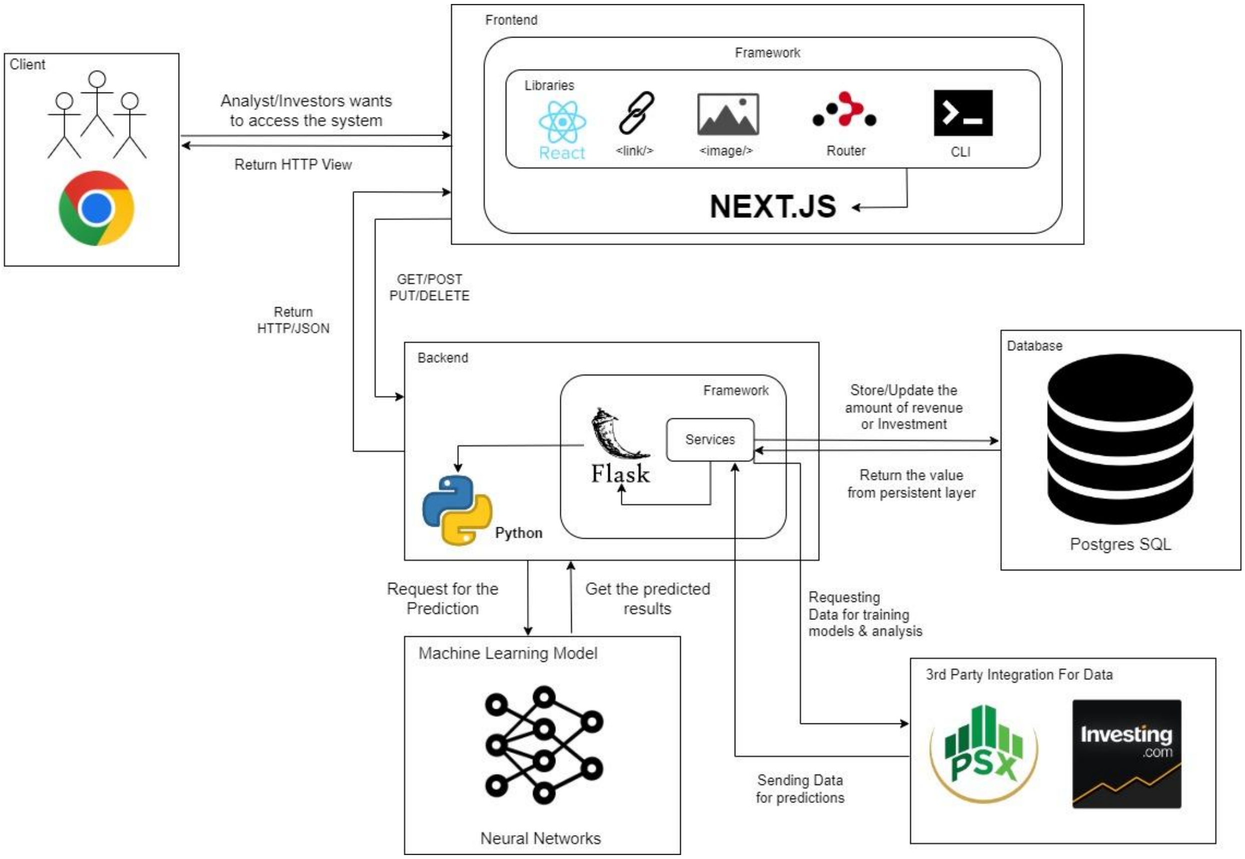
Sr#	Requirements
1	The system should complete execution of any request within 1 minute.
2	The system should entirely and accurately respond to requests at least 90% of the time it is invoked.
3	The time to get current stock prices from pakistan stock exchange should not be more than 5 seconds
4	The system should be available from Monday to Friday 9 am to 5 pm.

5	The bot should ping the trade instance every 1 minute to get the decision
6	The personal data of the investors and analysts will be stored in encrypted form
7	The trade profits, invested amounts, buying price, and selling will be in encrypted form as they are sensitive
8	Passwords should be salted and hashed before storage.
9	Multiple running bots will still ensure the individual performance of decision-making and request handling is catered within 60 seconds.
10	There must be at least 5 unique stock's pairs where the bots can trade

3. System Architecture

Brief introduction of this chapter in a paragraph highlighting the content

a. Architecture Diagram


Frontend
Framework
Libraries
React
<link/>
<image/>
Router
CLI
NEXT.JS
Client
Analyst/Investors wants to access the system
Return HTTP View
GET/POST PUT/DELETE
Return HTTP/JSON
Backend
Framework
Flask
Services
Python
Database
Store/Update the amount of revenue or Investment
Return the value from persistent layer
Postgres SQL
Request for the Prediction
Get the predicted results
Requesting Data for training models & analysis
Machine Learning Model
Neural Networks
3rd Party Integration For Data
PSX
Investing.com
Sending Data for predictions

b. Architecture Description

The project is based on three-tier architecture, front-end, back-end and database. The front-end is using MVVM design pattern, and we are using Next.js framework, which does server side rendering for better user experience. Front-end communicates with back-end through api calls, and we are using a python based flask framework in the back-end which integrates postgre-sql database, machine learning models and 3rd party services such as investing.com and PSX. The client sends a request to the frontend that is based on next.js framework, where the Dom listener is the View Model and the Client request is the view. The services are connected to the backend and it is the controller in this model.

The Backend uses the domain driven design (DDD). This enables the backend to load micro services and connect them. Whenever a command is received by the flask API it checks if it is a view or a service request. If it is a micro service request, the domain of the micro service is loaded and the changes are made and then sent to the user. The repository is connected to the database.

c. Justification of the Architecture

Backend

DDD (Domain Driven Design) allows us to architect our backend server in terms of microservices. Our implementation involves various iterations of strategies with which the bot makes its decisions so DDD helps us to create pure models in python which will be able to process and make decisions. The architecture does not depend on the existence of any specific library for a specific implementation. This allows us to use such frameworks as tools, rather than having to cram our system into the limited constraints. It also allows us to write testable code and follow TDD (Test driven development) and write unit test cases for the domain model, repository model and write some E2E tests to test complete functionality. Writing unit tests for the domain model and commands is very fast since they don't require any implementation details for the databases. FakeRepositories and Unit of work allows abstractions and dependency inversion principle allows this architecture to be independent of its inner layers. This brings more confidence into our code and prevents code that is testable and verifiable. Since we are also working in the domain of finance involving real money, quality code that does what it is intended to do is of the highest importance. This architecture is also Independent of UI which can change very easily, without changing the rest of the system. Our business rules are not bound to any choice of the database. Independent of any external factors since layers aren't dependent on outside layers and they can easily be swapped.

Frontend

For frontend, Next Js is used which is based on MV-VM Architecture. Our above given architecture for frontend used MV-VM as base design. Next JS helps in server-side rendering, along with client side with optimized routing system. It makes the code further re-usable. It makes fully interactive, performant and highly dynamic websites. Particularly MVVM, also flattens the dependencies. It means that the focus would be on a single function where the function is separated from the interruption of other dependencies.

d. Tools and Technologies

Frontend

Language
- Typescript 4.8.4
- ESLint 8.25.0

JavaScript library
- Next.js 17.0.2
- React.sj 18.2.0

CSS Framework
- Tailwind CSS 3.1.8
- DaisyUI 2.31.0

API library
- Axios 1.1.2

Deployment
- Vercel

Backend

Language
- Python 3.10.7
- PostgreSQL 15

Persistent storage
- PostgreSQL 15

Server
- Flask 2.2.2

Deployment
- Google Cloud’s App Engine flexible

Deep Learning Models:

Language

- Python 3.10.7

Frame work

- Pytorch and Tensorflow 1.12.1

Tool

- Google Colab

4. Requirements Specifications

a. Use Cases

4a.1: Login with credentials

Identifier	UC-001
Purpose	The analyst is verified to be a authorized user
Pre-conditions	User enters the username and password
Post-conditions	User is redirected to the home page of the trading platform
Step #	**Typical Course of Action**
1.	User opens the trading website
2.	Website displays a login prompt to the user
3.	User enters username and password
4.	Username is matched in the database of authorized users
5.	If the username is present, check if the corresponding password is entered correctly
6.	If the password is correct, send a message to redirect to the home page
Step #	**Alternate Courses of Action**
	-
Step #	**Exception Paths**
1.	In step 4: If username doesn't match, display a username error message In step 5: If password doesn't match display a failed login message

4a.2: View Recent Highlights

Identifier	UC-002
Purpose	Analyst want to see recent predictions made by the model
Pre-conditions	User is logged in with analyst credentials
Post-conditions	A highlight page displays recent activity highlights from the database
Step #	**Typical Course of Action**
1	Analyst clicks on the view highlights tab
2	A request is sent to the database to fetch recent activity of the bot
3.	Each entry of the activity is sorted based on time
4.	The activity is displayed on the display page
5.	Use case ends
Step #	**Alternate Courses of Action**
	-
Step #	**Exception Paths**
1.	In step 2: If there are no activities in the database a display message telling the user of no recent activity is shown

4a.3: User initiates a bot for execution

Identifier	UC-005
Purpose	Enable the analyst to enter trades
Pre-conditions	User is logged in with analyst credentials. Time for execution is during trading times of the stock market
Post-conditions	A bot state is run and linked to a chronejob
Step #	**Typical Course of Action**
1.	User defines the trade parameters before running the bot.
2.	Based on the parameters, models from the model directories are run and send their predictions to the bot
3.	The bot chooses the best prediction instance and initiates a trade on the best predicted stock
4.	A cron job is initialized to exit trades based on user defined exit conditions
5.	A running model state is linked to the the cron job
6.	Instance of the model is stored on the analysts running bot directory
7.	The use case ends.
Step #	**Alternate Courses of Action**
	-
Step #	**Exception Paths**
1.	In Step 5: If the analyst tries to enter a trade in non trading times, the model initialization will fail and display an error message

4a.4: Forcefully terminate the execution of the bot

Identifier	UC-006
Purpose	To exit a trade forcefully, before parametric termination condition is met
Pre-conditions	A bot has entered a trade that needs to be exited
Post-conditions	The bots execution is terminated and the associated loss or profits are added to the analysts account
Step #	**Typical Course of Action**
1.	User sends a command to exit an instance of the bot execution
2.	System checks if the bot is executing
3.	If the bot is running, it is exited from the trade
4.	Store profits or losses in the analysts account
5.	Save the model history in user highlights
6.	The use case ends
Step #	**Alternate Courses of Action**
1	-
Step #	**Exception Paths**
1.	In Step 1: If user sends a command to exit in non trading time, it should send an error message

4a.5: Initiate multiple bot instances

Identifier	UC-007
Purpose	Enable analyst to make multiple trades based on different parameters
Pre-conditions	User is logged in with analyst credentials. Time for execution is during trading times of the stock market
Post-conditions	A bot state is run and linked to a chronejob
Step #	**Typical Course of Action**
1.	User defines the trade parameters before running the bot.
2.	Based on the parameters, models from the model directories are run and send their predictions to the bot
3.	The bot chooses the best prediction instance and initiates a trade on the best predicted stock
4.	A cron job is initialized to exit trades based on user defined exit conditions
5.	A running model state is linked to the the cron job
6.	The new model instance is added to the list of running bots with a special bot id
8.	The use case ends.
Step #	**Alternate Courses of Action**
	-
Step #	**Exception Paths**
1.	In Step 5: If the analyst tries to enter a trade in non trading times, the model initialization will fail and display an error message

4a.6: Analyst registers a new investor

Identifier	UC-008
Purpose	The analyst registers an investor who can view their bot’s performance.
Pre-conditions	User is logged in using analyst credentials
Post-conditions	The investor is registered, and his details added to the database.
Step #	**Typical Course of Action**
1.	The analyst is logged in to the website.
2.	The analyst goes to the ‘Register Investor’ page.
3.	The website asks the analyst for new investor details.
4.	The analyst enters the name, phone number, email address, residential address, and NTN number of the investor.
5.	The email address, phone number and NTN number are checked if they are valid.
6.	If details are valid, the database is checked if the email address is already there.
7.	If the email address is not in the database, the analyst is shown a prompt to confirm the details about the investor.
8.	If the analyst confirms, the user is registered, and details are added to the database.
9.	Use case ends.
Step #	**Alternate Courses of Action**
10.	
Step #	**Exception Paths**
1.	In step 5, if any of the details is invalid, the analyst is prompted about the error and be told to enter again. 2. In step 6, if the email address is already in the database, the analyst is prompted that email already exists.

	3. In step 7, if the analyst doesn’t confirm, he/she is redirected to the enter details page.

4a.7: View performance of all bots

Identifier	UC-011
Purpose	Analyst can view the performance of all his/her bots that are currently executing.
Pre-conditions	The Analyst is logged in and authenticated
Post-conditions	Analyst is displayed performance of bots currently running.
Step #	**Typical Course of Action**
1.	Analyst clicks on the view performance tab.
2.	A request is sent to the database to fetch performance(profit%) of all bots currently executing initiated by the analyst.
3.	The result is displayed to the analyst on the website.
4.	Use case ends.
Step #	**Alternate Courses of Action**
1.	
Step #	**Exception Paths**
1.	In step 2, if there are no bots currently running initiated by the analyst, a display message is shown saying “No current bots”.

4a.8: View entire performance of a bot

Identifier	UC-013
Purpose	The analyst can view the entire performance of a bot.
Pre-conditions	The Analyst is logged in and authenticated
Post-conditions	Entire performance of the bot is displayed to the analyst.

Step #	Typical Course of Action
1.	Analyst clicks on the ‘View entire performance of a bot’ tab.
2.	Analyst selects the bot that is required.
3.	A request is sent to the database to fetch the complete trade history of the bot selected by the analyst.
4.	The result is displayed to the analyst on the website.
5.	Use case ends.

Step #	Alternate Courses of Action
1.	
Step #	**Exception Paths**
1.	In step 3, if the bot selected by the analyst is still currently running, the user is prompted with an error and execution proceeds to step 2.

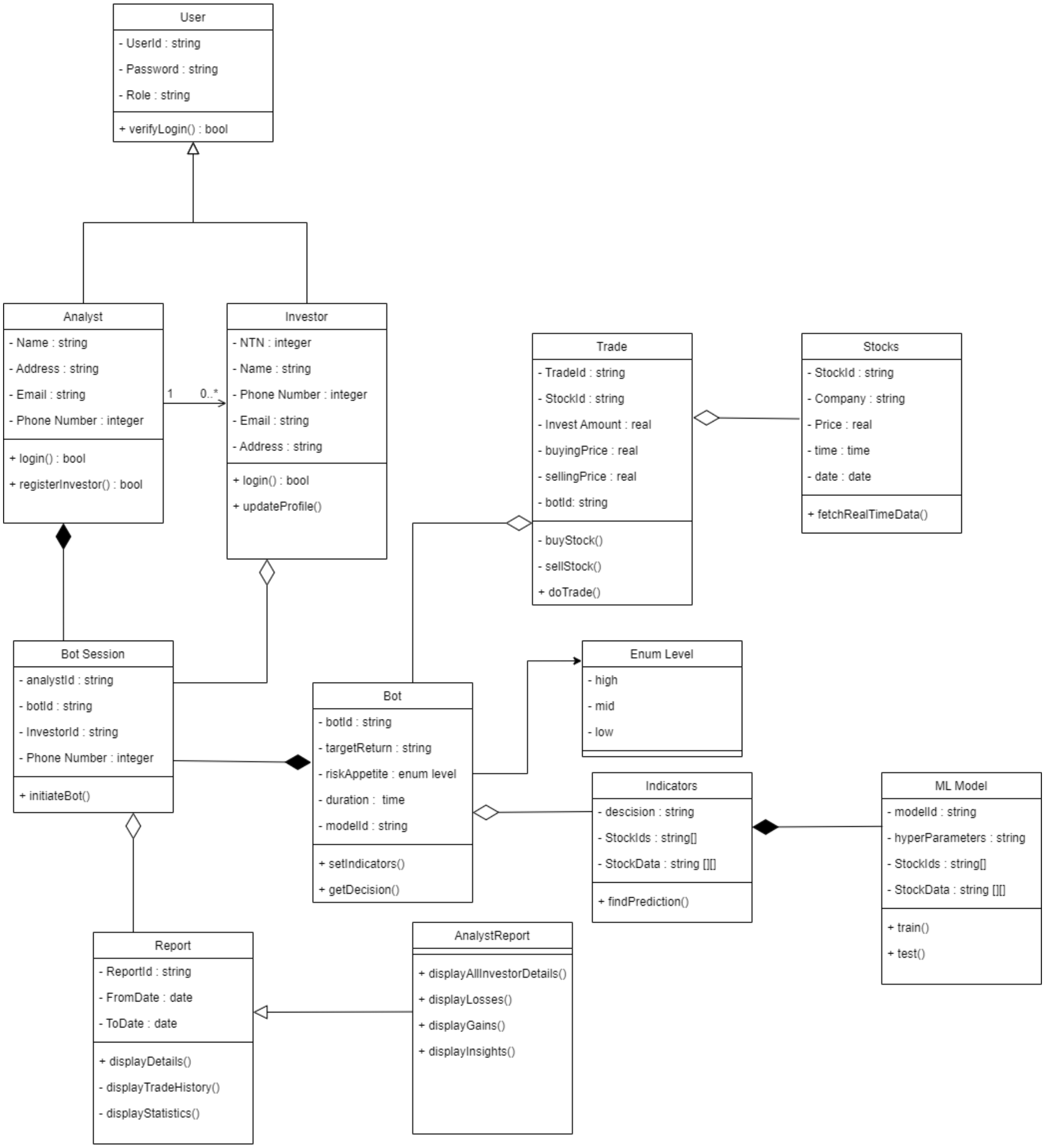
4a.9: View multiple graphs of a bot

Identifier	UC-014
Purpose	The analyst can view the multiple graphs of a bot.
Pre-conditions	The Analyst is logged in and authenticated
Post-conditions	Selected graphs of the bot are displayed to the analyst.
Step #	**Typical Course of Action**
2.	Analyst clicks on the ‘View graphs of a bot’ tab.
3.	Analyst selects the bot that is required.
4.	The analyst is asked to select graphs required from multiple choices
5.	Analyst selects the graphs that are required.
6.	A request is sent to the database to fetch the complete trade history of the bot selected by the analyst.
7.	The graphs are displayed to the analyst on the website.
8.	Use case ends.
Step #	**Alternate Courses of Action**
9.	
Step #	**Exception Paths**
10.	In step 2, if there is no history of bots initiated by the analyst, a display message is shown saying “No history for selected bot”.

4a.10: As an investor, I want to be able to login to my dashboard with the credentials provided by my analyst.

Identifier	UC-018
Purpose	The investor successfully logins in and is authenticated
Pre-conditions	
Post-conditions	Investor is redirected to his dashboard
Step #	**Typical Course of Action**
1.	User opens the trading website
2.	Website displays a login prompt to the user
3.	User enters username and password
4.	Username is matched in the database of authorized users
5.	If the username is present, check if the corresponding password is entered correctly
6.	If the password is correct, redirect the user to the dashboard
Step #	**Alternate Courses of Action**
Step #	**Exception Paths**
1.	In step 4: If username doesn't match, display a username error message In step 5: If password doesn't match display a failed login message

4b.1: Class Diagram



4b.2: Description

User: This is a parent class to keep the stakeholder's id and password. The child classes include analysts and investors with parent features and their separate functions of login. Investors can update their profile which includes the variables of the objects. Analysts are associated with investors where one analyst can register multiple investors.

Bot: It will help to create separate instances of bots with various combinations of configuration. It will use the indicator instances with the configuration to make decisions.

BotSession: It will keep track of bot sessions currently running, which are a composition of bot and analyst objects. They will be destroyed after the session is completed.

Trade: It will keep track of trades and store the stock ids, buying and selling prices etc. It will be responsible for managing trade according to the decisions taken by the bot.

Stocks: Every instance of stocks will represent a single stock with their relevant information for trading. These data will be used by the Trade class to make sales and purchases.

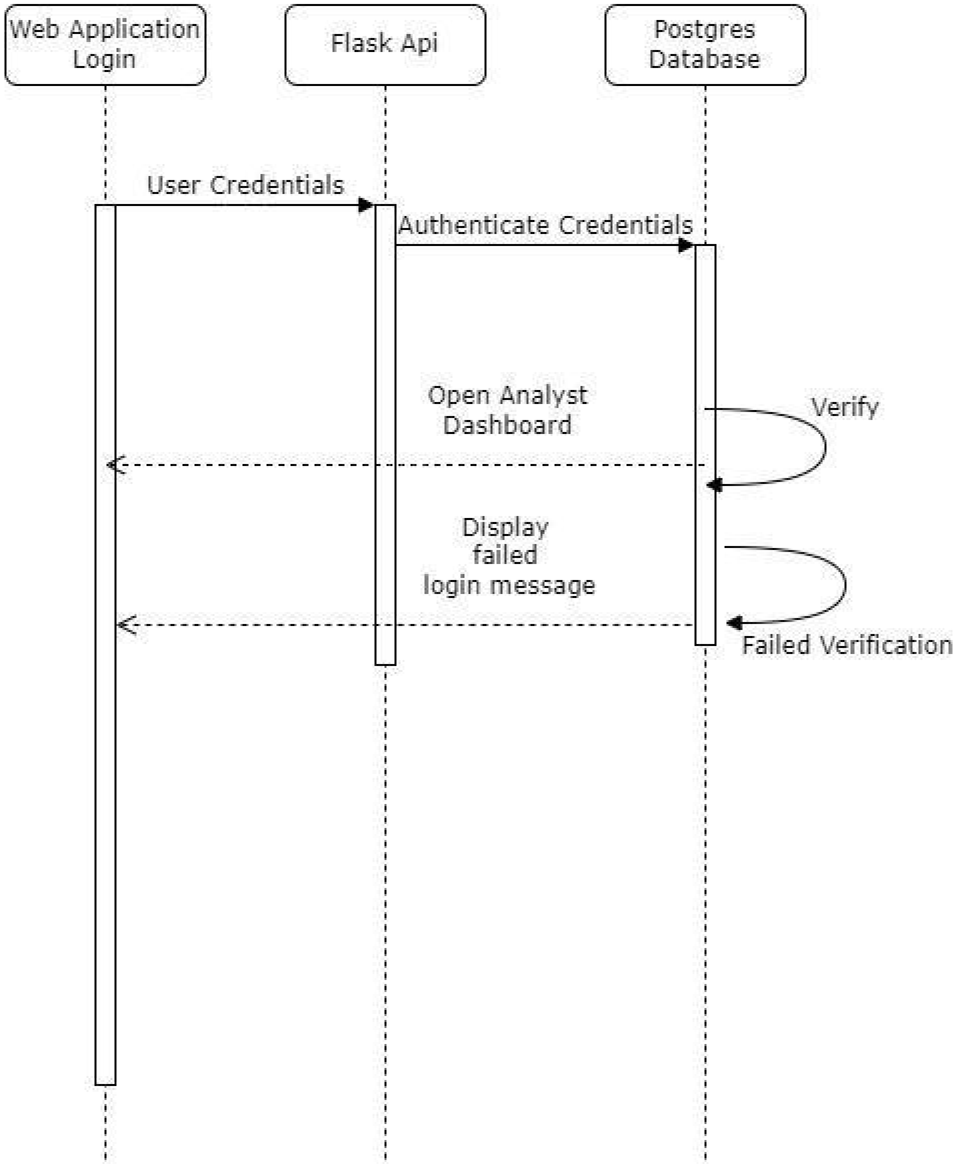
Report: It will be responsible for displaying the statistics, trade history, and comparisons of previous trades with the latest ones. It will help keep track of generated reports.

Indicators: These will create the indicators with specific values for each bots according to their configuration parameters. Machine Learning Model Class will be used in it to assist the decision

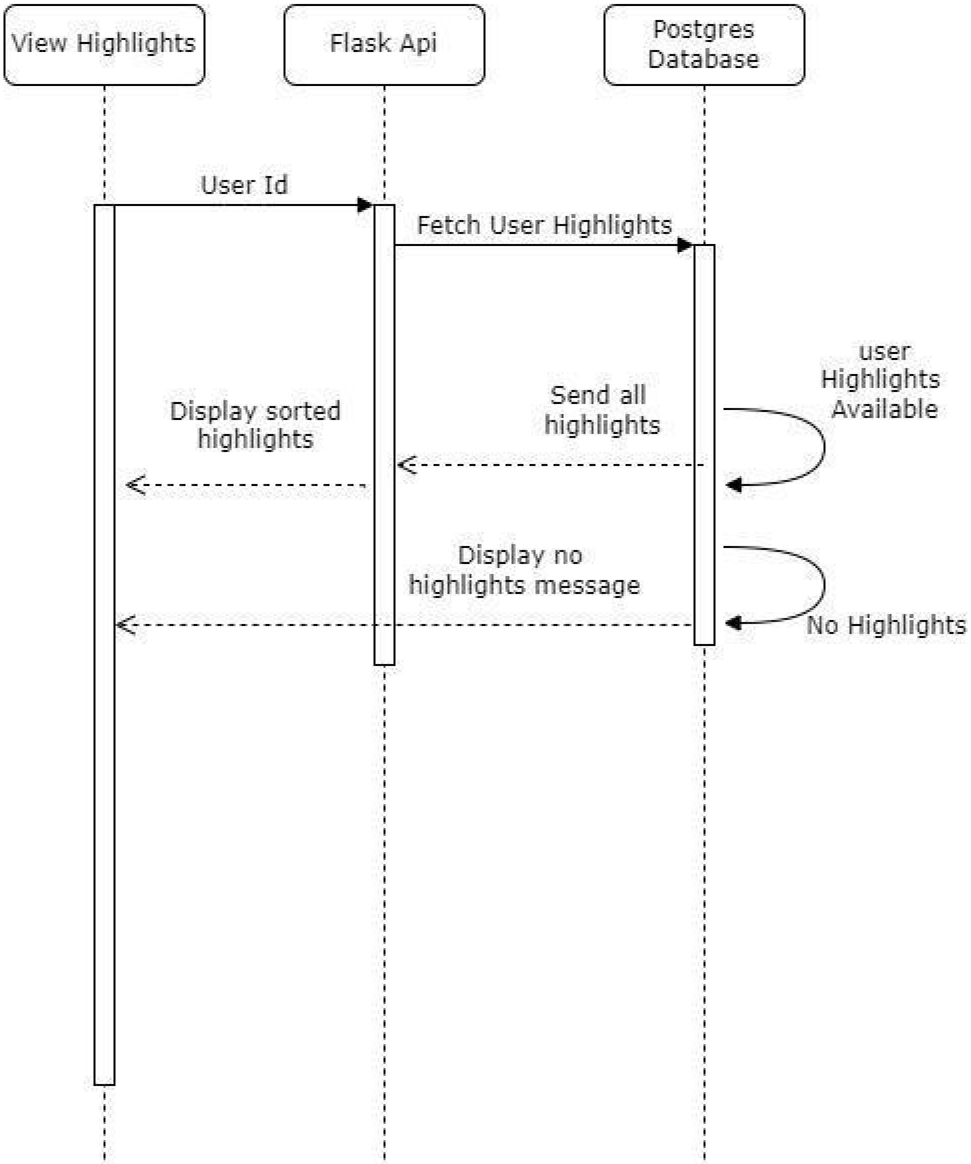
c: Sequence Diagrams

Draw sequence diagrams of 10 core use cases. Draw the diagrams using standard UML notation

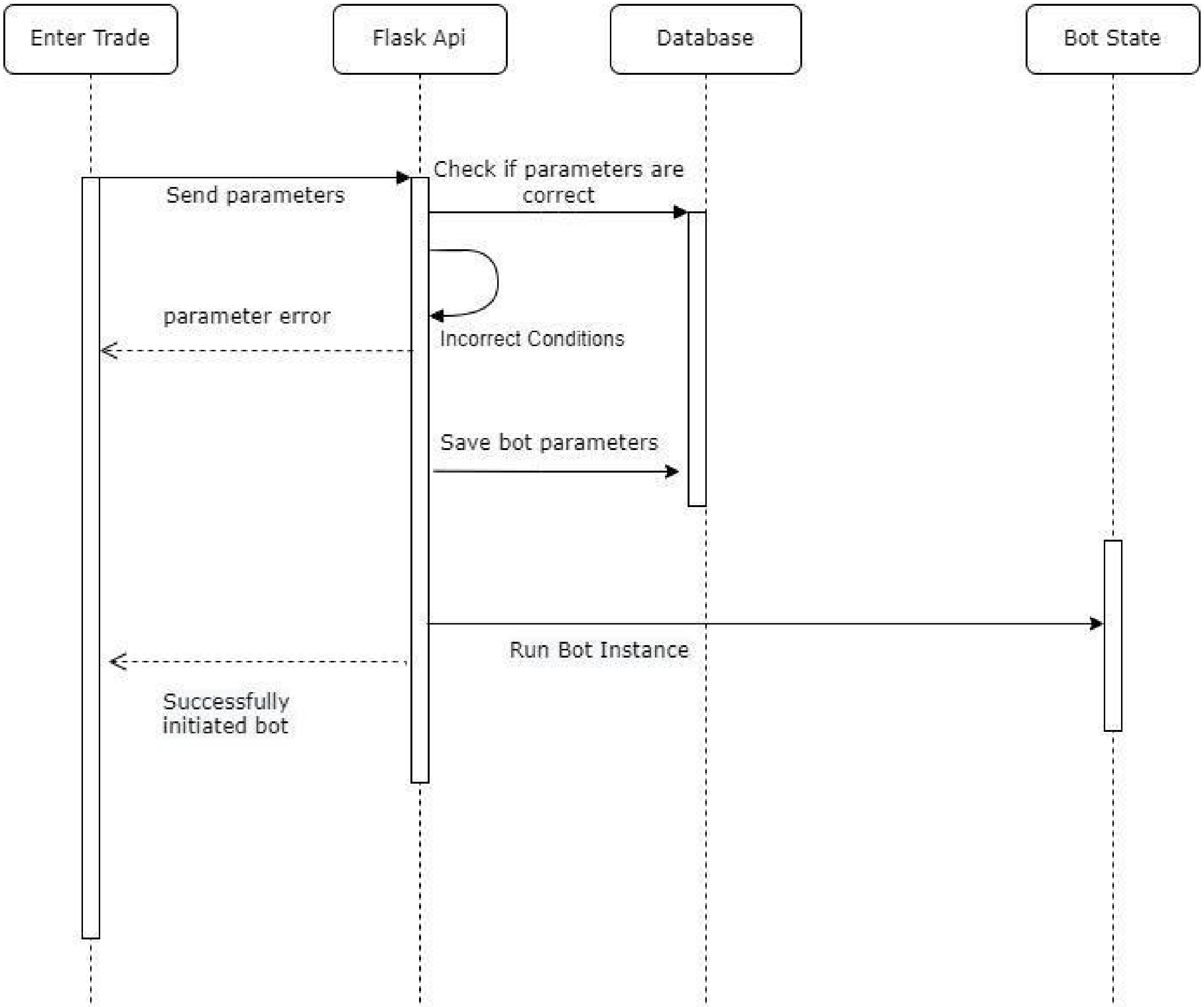
4c.1: Login with credentials



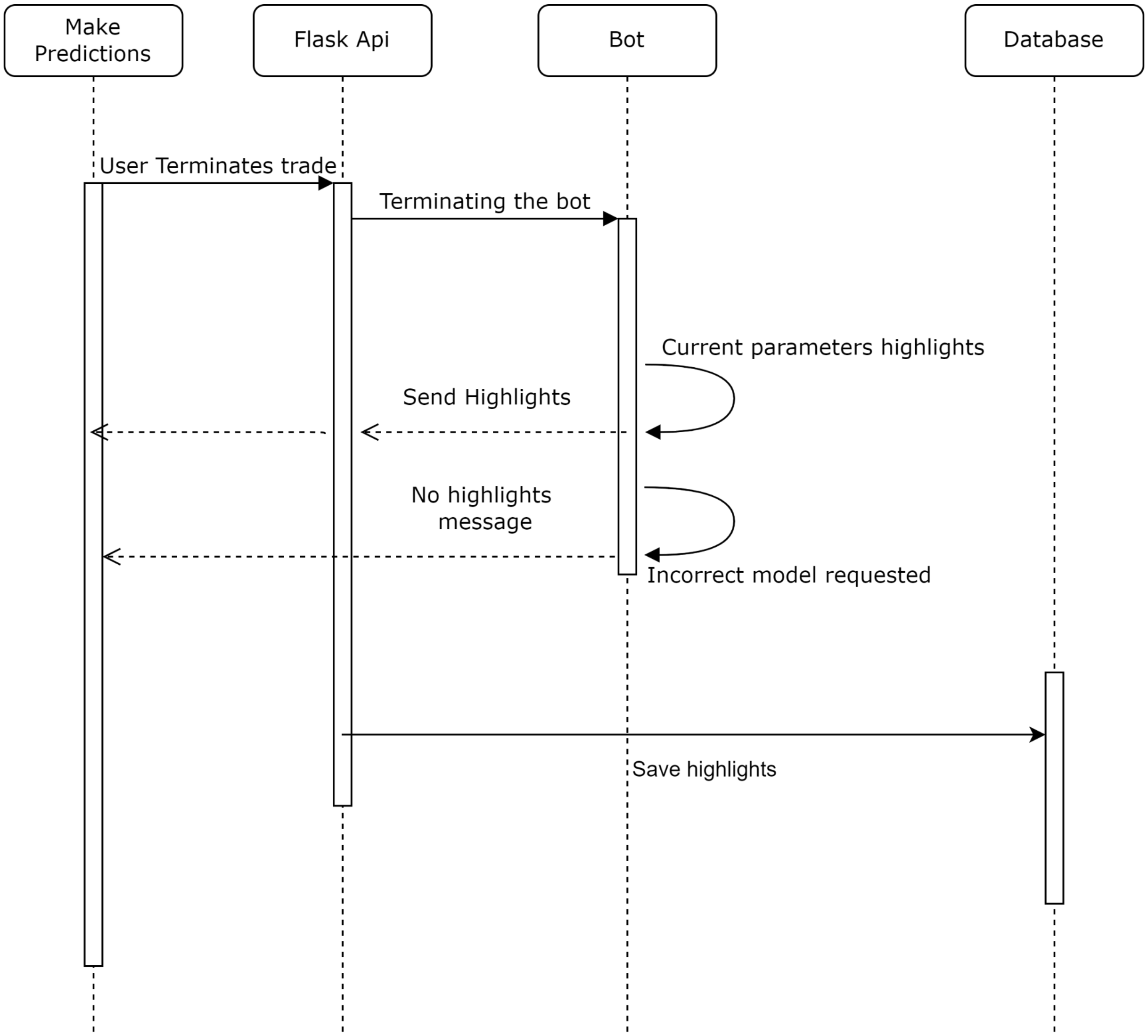
4c.2: View Recent Highlights



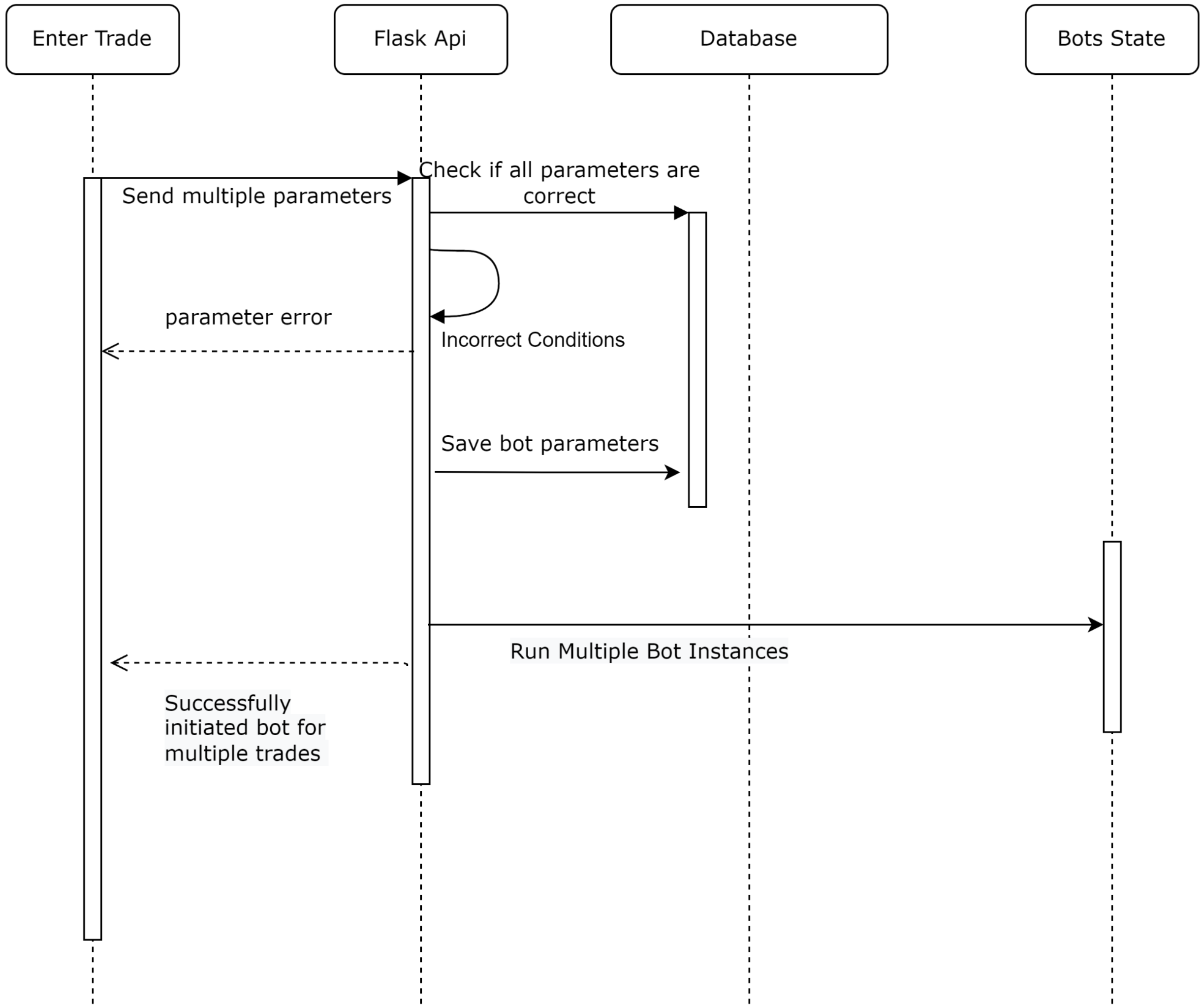
4c.3: User initiates a bot for execution



4c.4: Forcefully terminate the execution of the bot

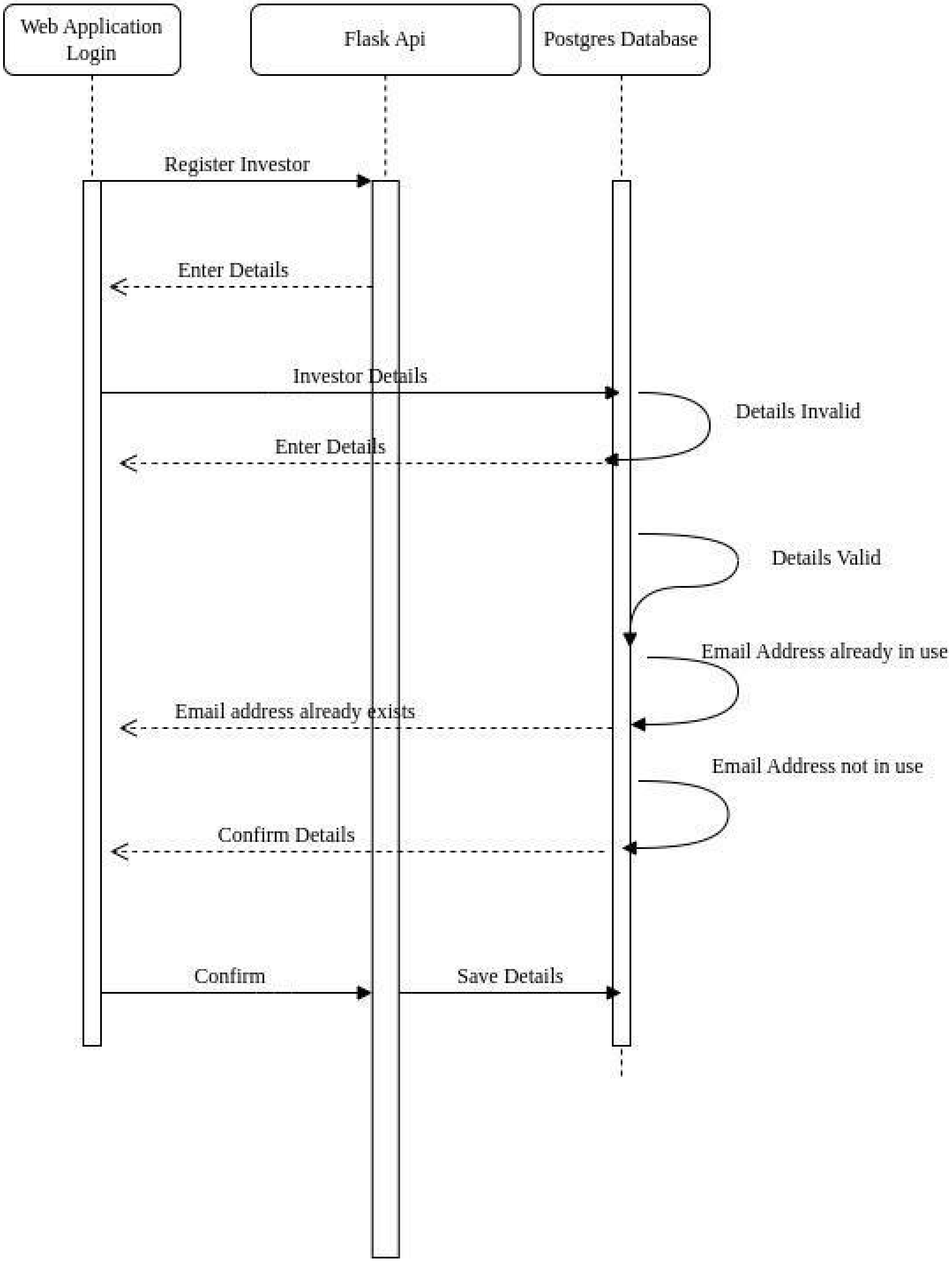

Make Predictions
Flask Api
Bot
Database
User Terminates trade
Terminating the bot
Current parameters highlights
Send Highlights
No highlights message
Incorrect model requested
Save highlights

4c.5: Initiate multiple bot instances



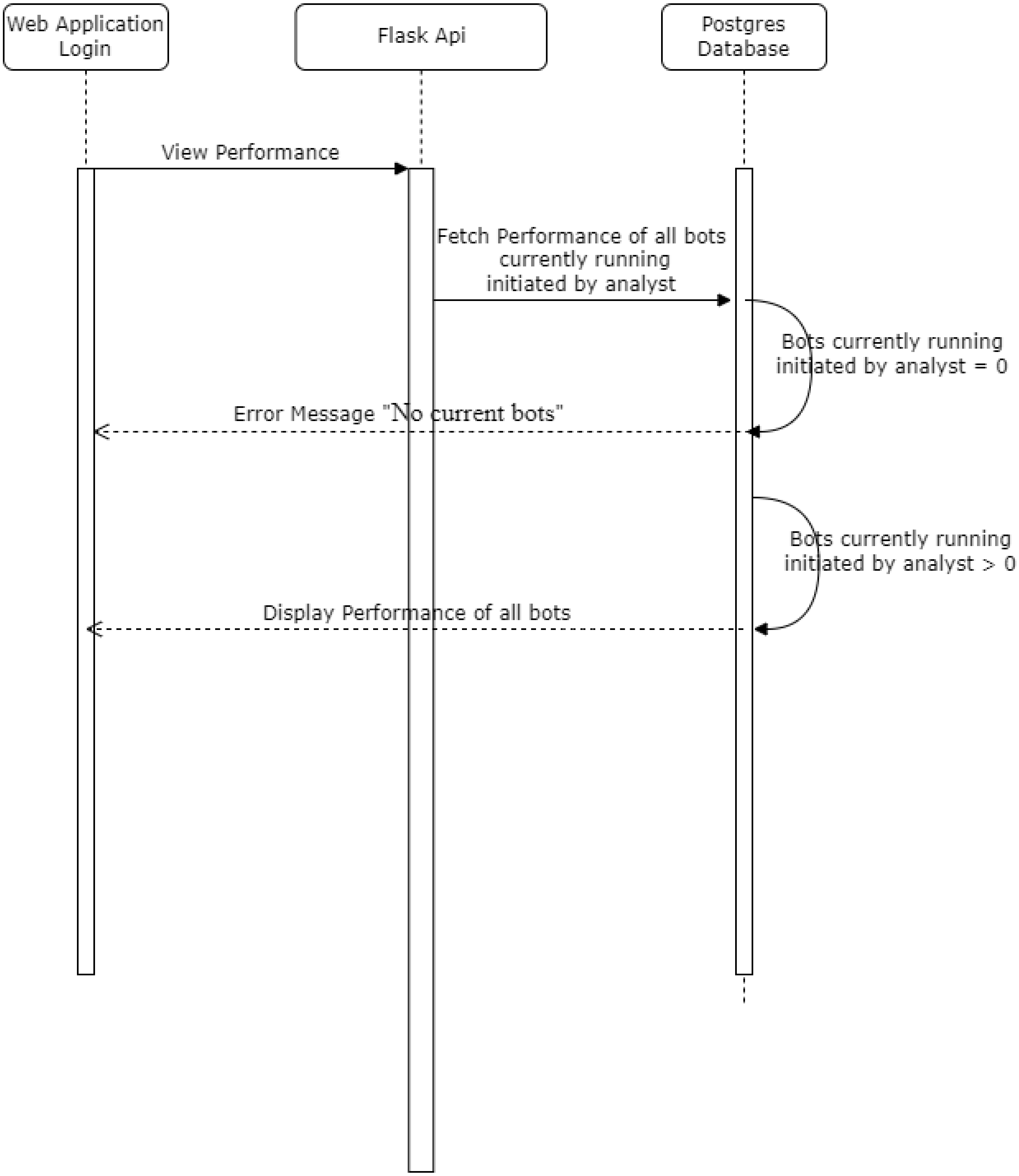
4c.6: Analyst registers a new investor

UC-008



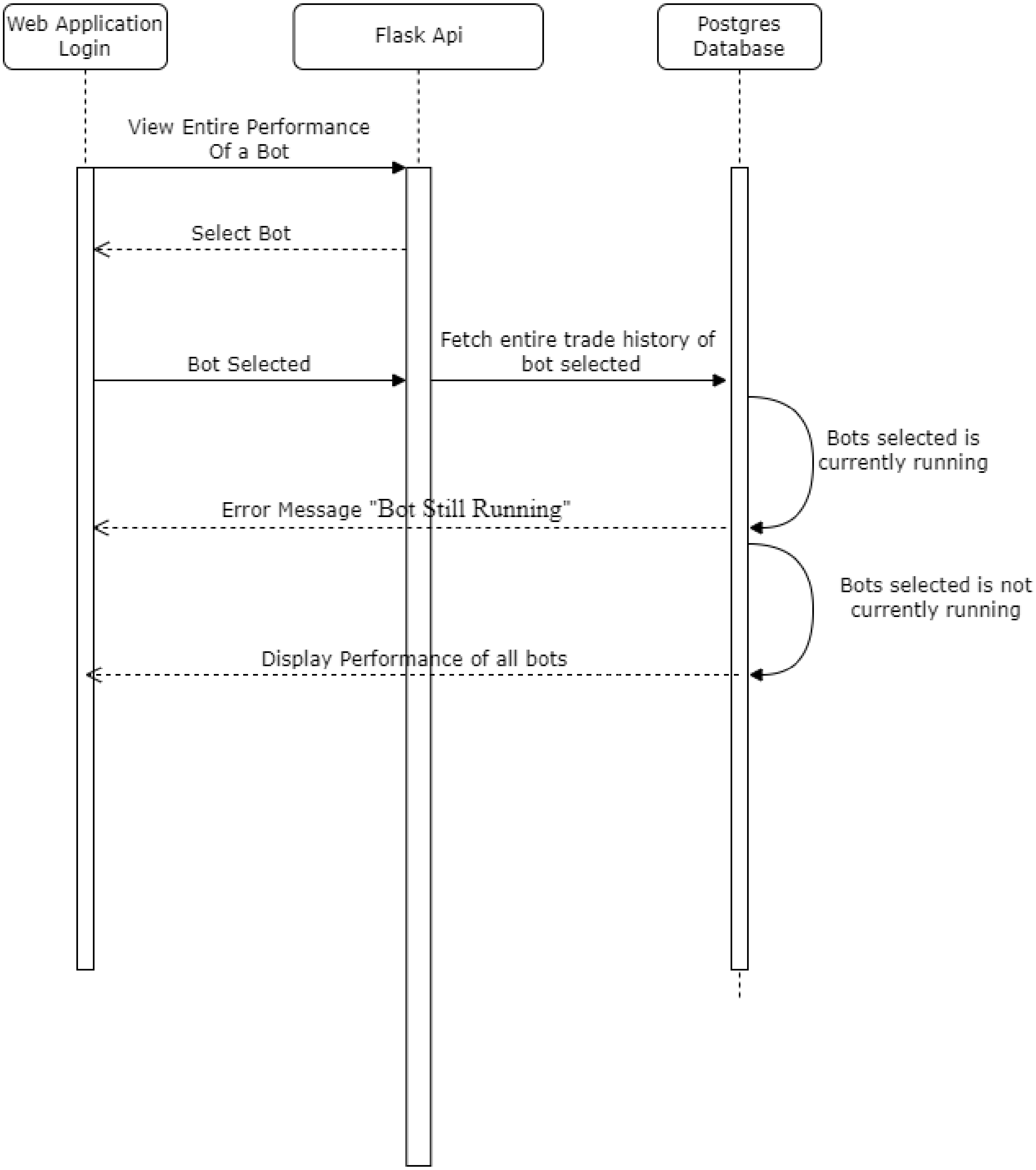
4c.7: View performance of all bots

UC-011


Web Application Login
Flask Api
Postgres Database
View Performance
Fetch Performance of all bots currently running initiated by analyst
Bots currently running initiated by analyst = 0
Error Message "No current bots"
Bots currently running initiated by analyst > 0
Display Performance of all bots

4c.8:The analyst can view the entire performance of a bot.

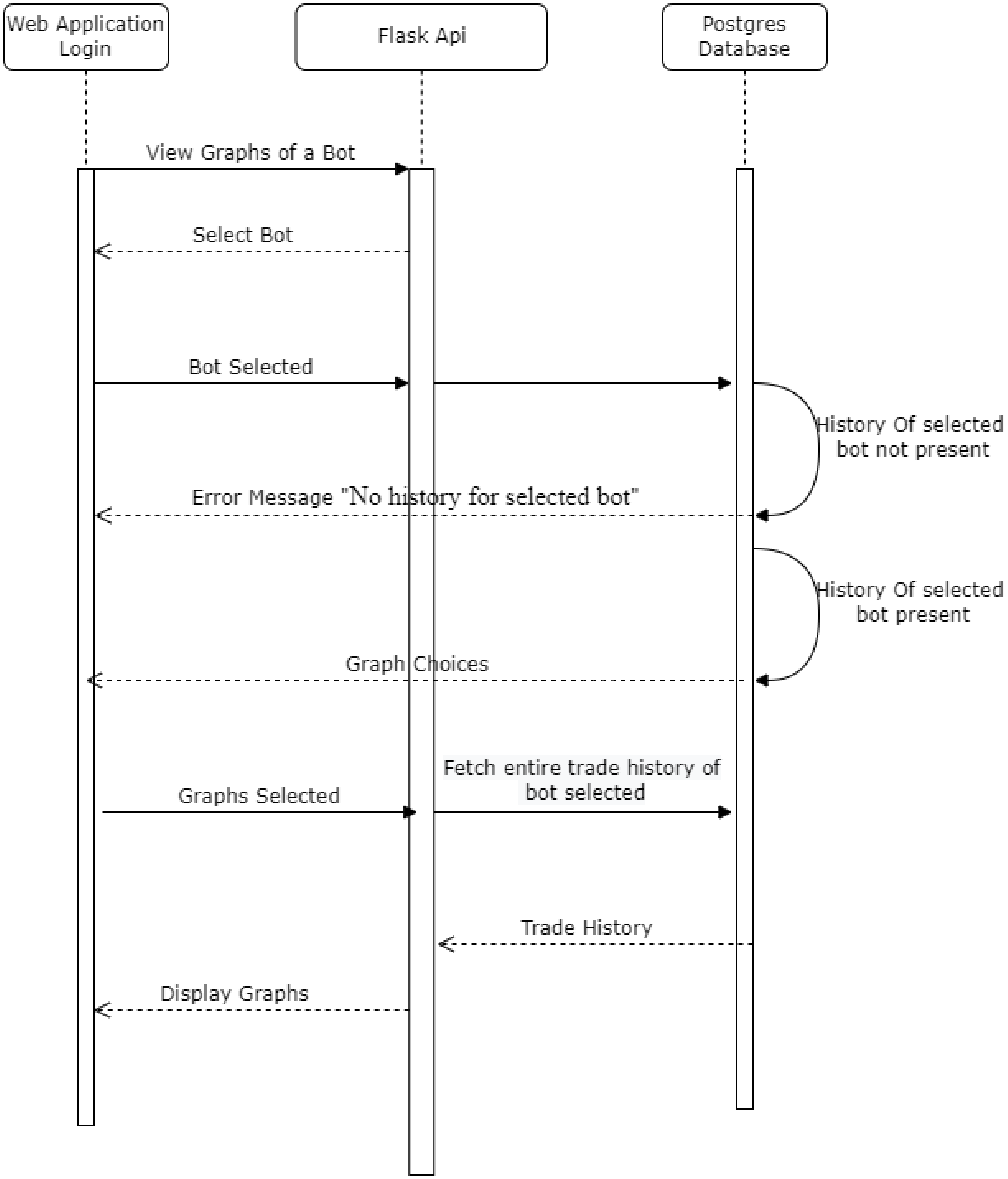
UC-013



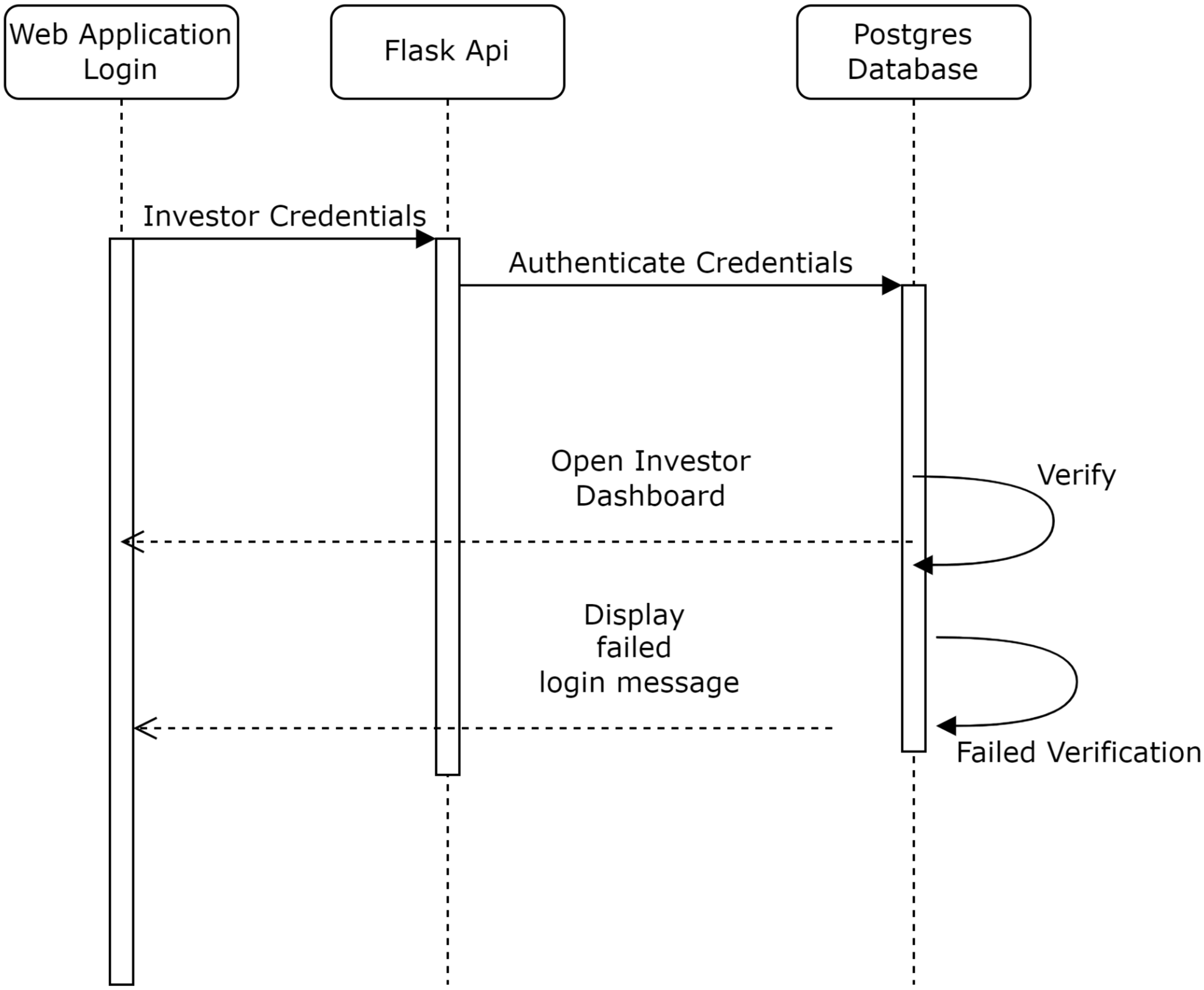
Web Application Login
Flask Api
Postgres Database
View Entire Performance Of a Bot
Select Bot
Bot Selected
Fetch entire trade history of bot selected
Bots selected is currently running
Error Message "Bot Still Running"
Bots selected is not currently running
Display Performance of all bots

4c.9: The analyst can view the multiple graphs of a bot.

UC-014


Web Application Login
Flask Api
Postgres Database
View Graphs of a Bot
Select Bot
Bot Selected
History Of selected bot not present
Error Message "No history for selected bot"
History Of selected bot present
Graph Choices
Fetch entire trade history of bot selected
Graphs Selected
Trade History
Display Graphs

4c.10: As an investor, I want to be able to login to my dashboard with the credentials provided by my analyst.



-

5. Software Development Methodology and Plan

a. Software Process Selection

Waterfall:
The pros of waterfall model include:

1. A finished product that is well-defined and predictable.
2. Clearly defined roles and responsibilities for each member from the start.
3. Detailed project planning and strict deadlines
4. After the requirement phase is finished, client engagement is not required except reviews, approvals, and status updates.
5. Since the project's whole scope was previously defined, progress can be simply tracked.

The cons include:

1. Rigidness and inflexibility following a specification.
2. Fewer chances to change course.
3. There are too many new inventions and commercial techniques by the time the final product is rolled out.
4. It takes too long for bugs/glitches to be found since testing doesn't start until the major part of the project is finished.

Agile:
The pros of agile model include:

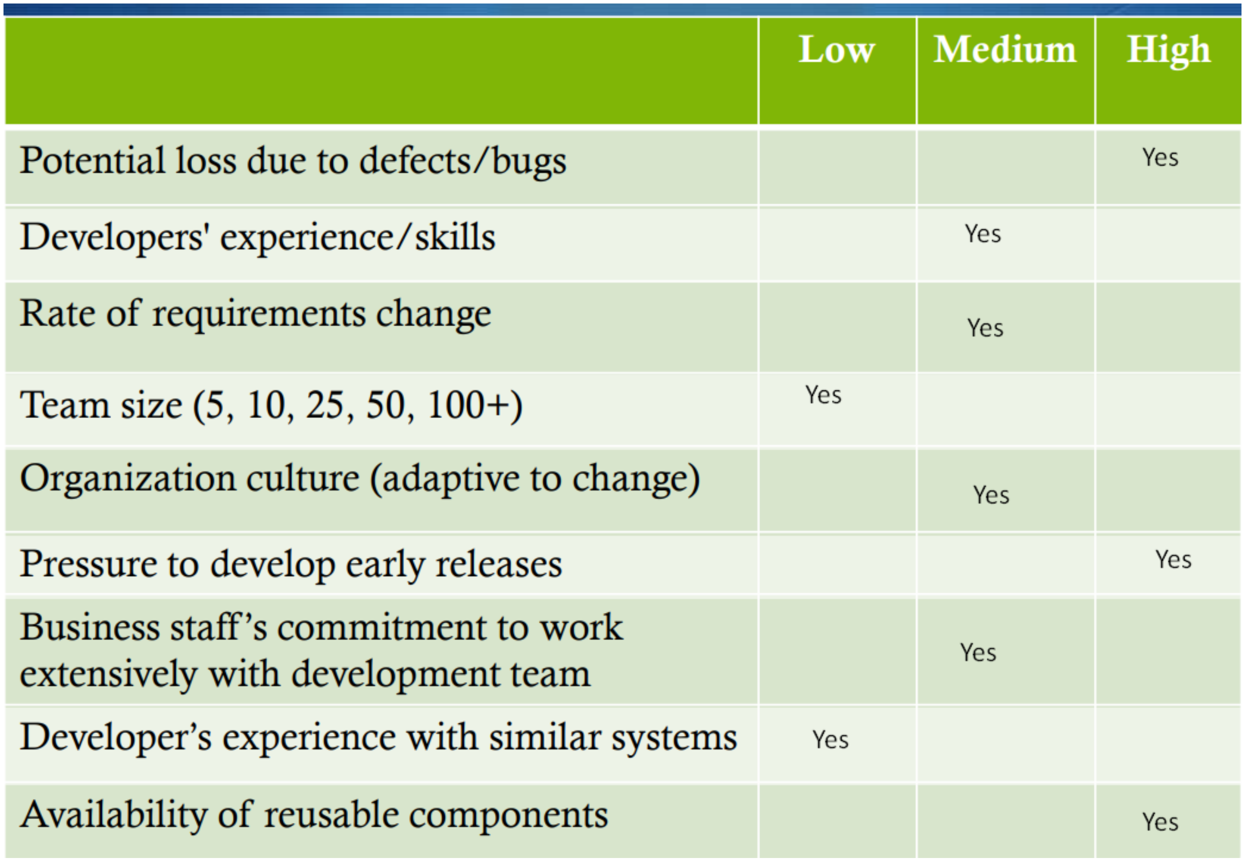
1. Ability to adapt to market conditions and emerging innovations.
2. The team members have room for bringing creative solutions to new challenges.
3. Self-organizing teams and resource allocation
4. Regular updates and enhanced consumer feedback.
5. Deadline flexibility.
6. Because sprints are frequent and early in the project lifecycle, the client sees results quickly.
7. Each sprint includes the identification and resolution of bugs, allowing for modifications.

The cons include:

1. Uncertainty in the final product and delays might result from loose planning.
2. Prone to distractions and a lack of focus from one sprint to another.
3. Testing criteria that are too lenient may result in bugs.
4. No room for modification during a Sprint.
5. Additional sprints could be added if a sprint is not finished within the allotted window, which would increase both the overall time and cost.

Our application is based on Financial models and techniques so the loss could be high if a defect and bug is left there unattended. We have developers with a variety of experience low to high, in general we all are in the learning phase from the project so the skills are medium in general. No large team is required as it would increase communication gap so our 4 developers can easily manage so if anyone leaves we have a replacement from the current 4 developers available to cover the work of the other. We are adaptive to change and expect changes from clients later on so its priority is medium. We have hard deadlines so there is a pressure of early release. Hence, Agile will help to create a working system after every sprint and scrum meetings will help us in gauging the overall performance and catchup is required.
We used the waterfall process at the start to make detailed documentation so we get clear on the requirements but for development we will follow Agile as it will help us develop the project gradually and fixing bugs timely. We are also open to creative solutions from our developers so Agile would help us cater that and since developers need to learn some other skills while developing so again Agile is the best choice for the development module of the project.

Additionally, we are using Artificial Intelligence to keep our bot updated with the market price so Agile will help us to adapt these systems into our main application as we can integrate it and make changes easily later on if required.

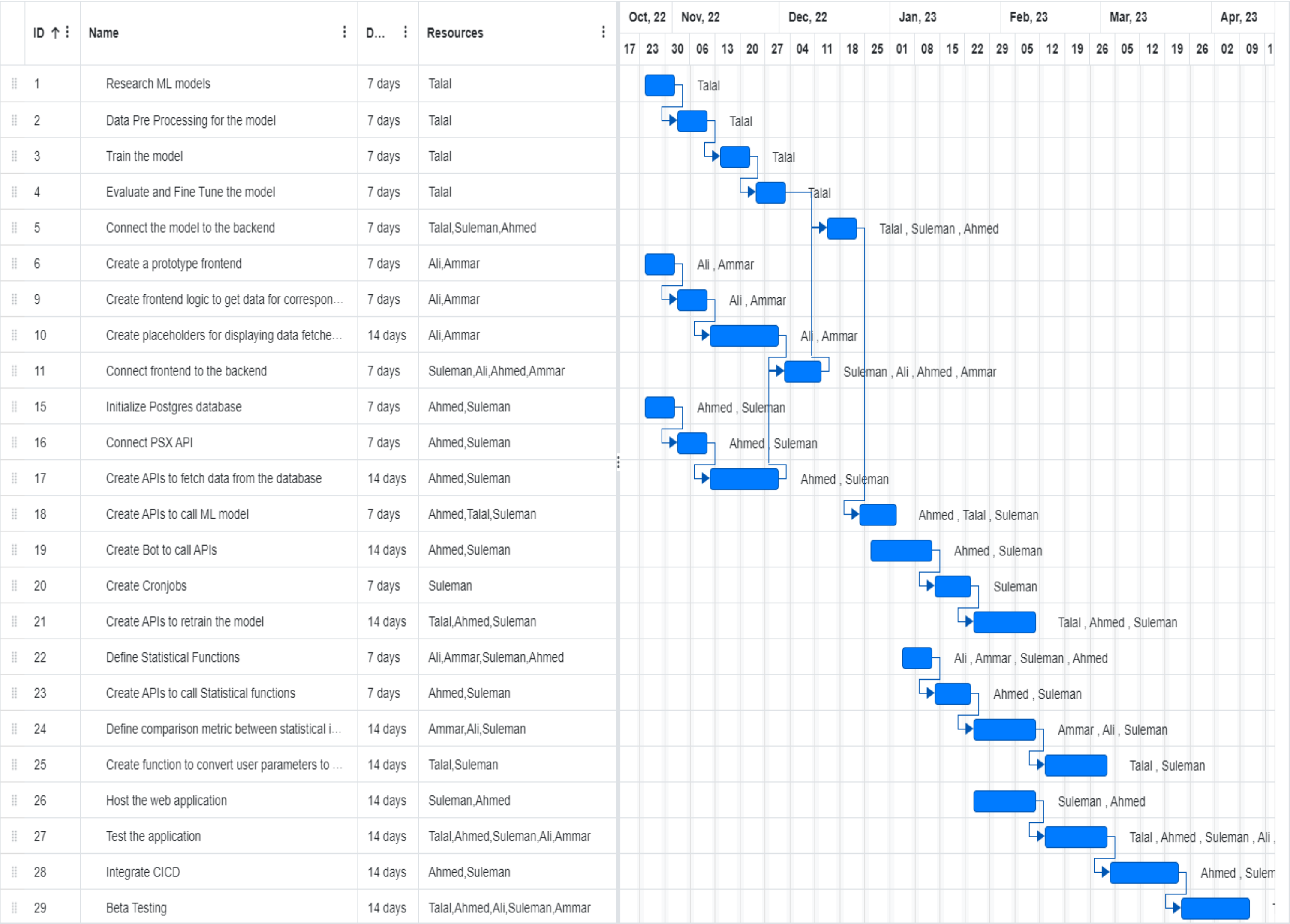


	Low	Medium	High
Potential loss due to defects/bugs			Yes
Developers' experience/skills		Yes	
Rate of requirements change		Yes	
Team size (5, 10, 25, 50, 100+)	Yes		
Organization culture (adaptive to change)		Yes	
Pressure to develop early releases			Yes
Business staff's commitment to work extensively with development team		Yes	
Developer's experience with similar systems	Yes		
Availability of reusable components			Yes

b. Gantt Chart

●

>



ID ↑	Name	D...	Resources
1	Research ML models	7 days	Talal
2	Data Pre Processing for the model	7 days	Talal
3	Train the model	7 days	Talal
4	Evaluate and Fine Tune the model	7 days	Talal
5	Connect the model to the backend	7 days	Talal,Suleman,Ahmed
6	Create a prototype frontend	7 days	Ali,Ammar
9	Create frontend logic to get data for correspon…	7 days	Ali,Ammar
10	Create placeholders for displaying data fetche…	14 days	Ali,Ammar
11	Connect frontend to the backend	7 days	Suleman,Ali,Ahmed,Ammar
15	Initialize Postgres database	7 days	Ahmed,Suleman
16	Connect PSX API	7 days	Ahmed,Suleman
17	Create APIs to fetch data from the database	14 days	Ahmed,Suleman
18	Create APIs to call ML model	7 days	Ahmed,Talal,Suleman
19	Create Bot to call APIs	14 days	Ahmed,Suleman
20	Create Cronjobs	7 days	Suleman
21	Create APIs to retrain the model	14 days	Talal,Ahmed,Suleman
22	Define Statistical Functions	7 days	Ali,Ammar,Suleman,Ahmed
23	Create APIs to call Statistical functions	7 days	Ahmed,Suleman
24	Define comparison metric between statistical i…	14 days	Ammar,Ali,Suleman
25	Create function to convert user parameters to …	14 days	Talal,Suleman
26	Host the web application	14 days	Suleman,Ahmed
27	Test the application	14 days	Talal,Ahmed,Suleman,Ali,Ammar
28	Integrate CICD	14 days	Ahmed,Suleman
29	Beta Testing	14 days	Talal,Ahmed,Ali,Suleman,Ammar

6. Database Design and Web Services

This chapter focuses on the database design entities used in the data model of the autonomous trading bot web application. The entities in the data model include Analysts, Investors, and the Bot. Analysts will register investors, configure and start the execution of the bot, analyze its statistics and assign investors to their bots. Investors will use the bot for investment purposes and can view a summary report of the performance of the bot for a specific range of time period. The Bot entity will be trained on PSX data and make decisions based on game theory, mathematical models, financial techniques, and artificial intelligence. Users of the web application will be able to configure the bot's target return, risk appetite, and duration of the instance through the primary web app. The database design entities will help ensure the efficient storage and retrieval of data related to analysts, investors, and the trading bot, enabling the application to function effectively.

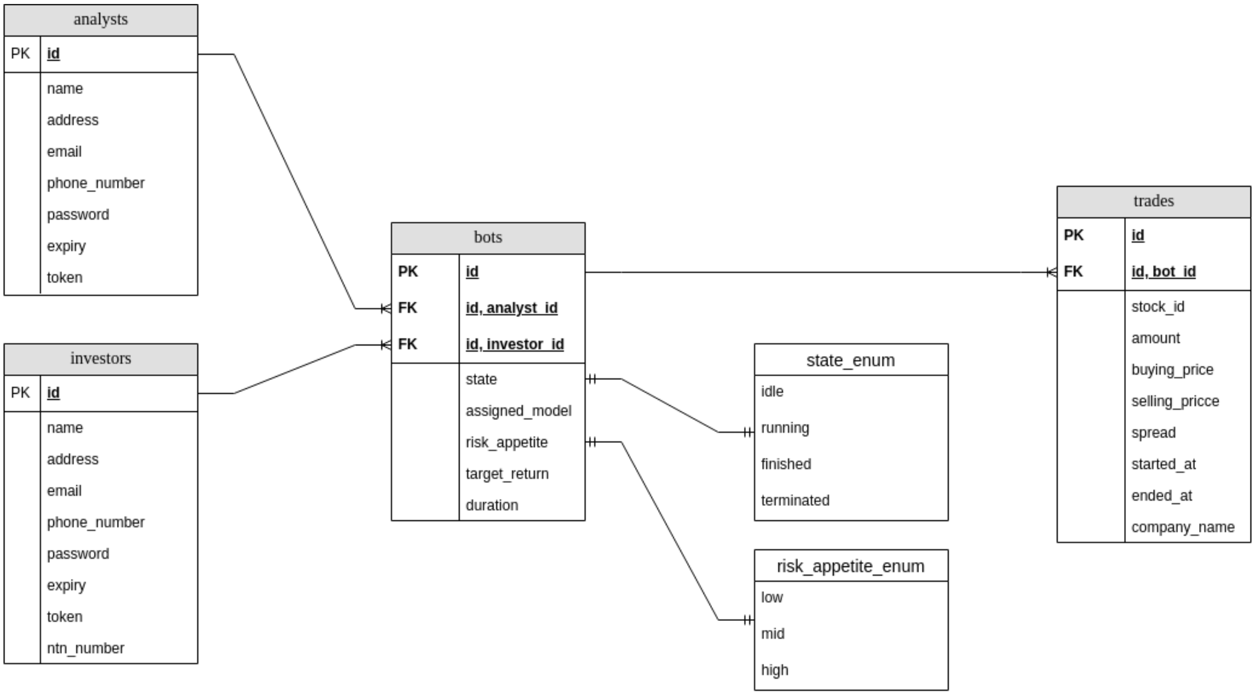
a. Database Design

Entities:

Analyst: Analysts will register investors, configure and start the execution of the bot, analyze its statistics and assign investors to their bots.

Analyst: Investors will use the bot for investment purposes and can view a summary report of the performance of the bot for a specific range of time period (monthly, quarterly, half-yearly, yearly, or other combinations of days or months).

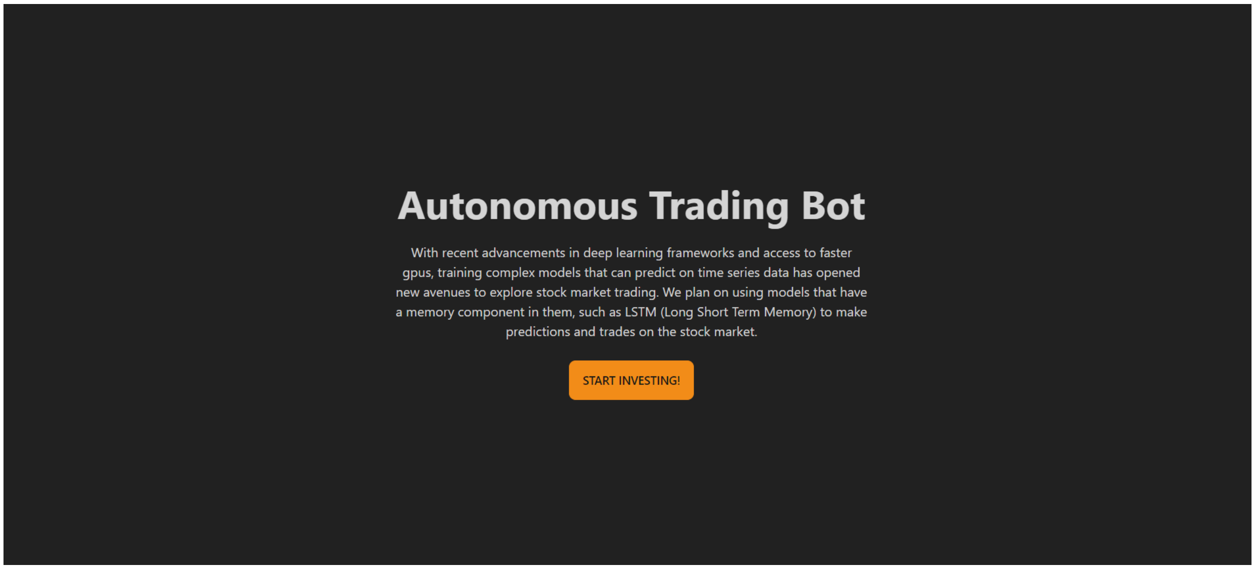
Bot: The bot will be trained on the PSX data. The bot's decision will be based on the concepts of game theory, mathematical models, financial techniques, and especially artificial intelligence. The primary web app will allow the user to provide the bot's configuration, which includes, target return, risk appetite, and duration of the instance.



b. API Specification

We have used the python’s package, psx-data-reader, which allows us to scrape the data of Pakistan stock exchange by just specifying the company's stock data we want and how much we want, and the rest is done for you using the library. This is our primary api to cover all of our data requirements for training our ML models and for simulating predictions on the new data.

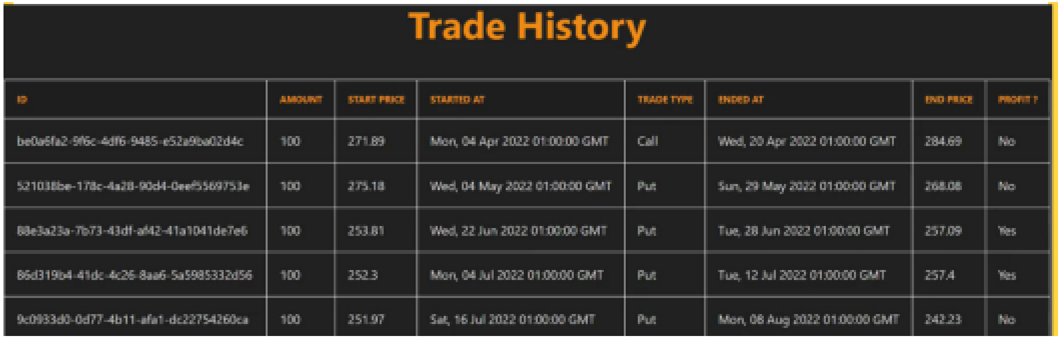
7. System User Interface



Autonomous Trading Bot

With recent advancements in deep learning frameworks and access to faster gpus, training complex models that can predict on time series data has opened new avenues to explore stock market trading. We plan on using models that have a memory component in them, such as LSTM (Long Short Term Memory) to make predictions and trades on the stock market.

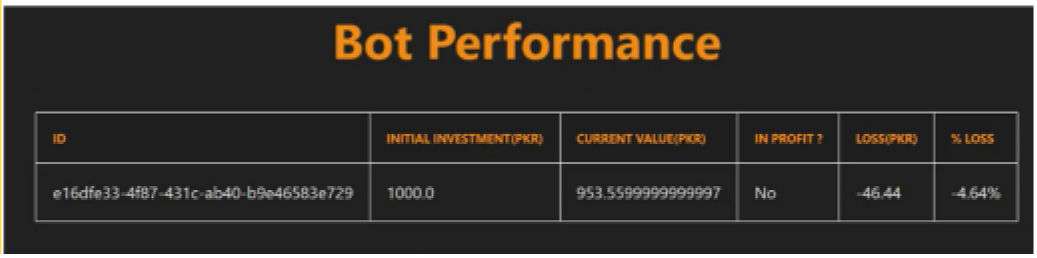
This is the landing page.



Trade History

ID	AMOUNT	START PRICE	STARTED AT	TRADE TYPE	ENDED AT	END PRICE	PROFIT ?
be0a6fa2-9f6c-4df6-9485-e52a9ba02d4c	100	271.89	Mon, 04 Apr 2022 01:00:00 GMT	Call	Wed, 20 Apr 2022 01:00:00 GMT	284.69	No
521038be-178c-4a28-90d4-0eef5569753e	100	275.18	Wed, 04 May 2022 01:00:00 GMT	Put	Sun, 29 May 2022 01:00:00 GMT	268.08	No
88e3a23a-7b73-43df-af42-41a1041de7e6	100	253.81	Wed, 22 Jun 2022 01:00:00 GMT	Put	Tue, 28 Jun 2022 01:00:00 GMT	257.09	Yes
86d319b4-41dc-4c26-8aa6-5a5985332d56	100	252.3	Mon, 04 Jul 2022 01:00:00 GMT	Put	Tue, 12 Jul 2022 01:00:00 GMT	257.4	Yes
9c0933d0-0d77-4b11-afa1-dc22754260ca	100	251.97	Sat, 16 Jul 2022 01:00:00 GMT	Put	Mon, 08 Aug 2022 01:00:00 GMT	242.23	No

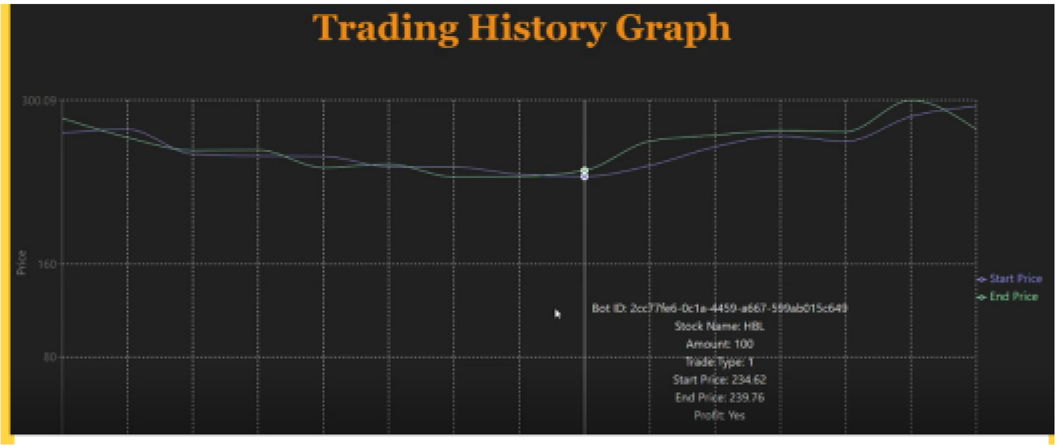
This shows the trading history of the selected bot to the investor/analyst.



Bot Performance

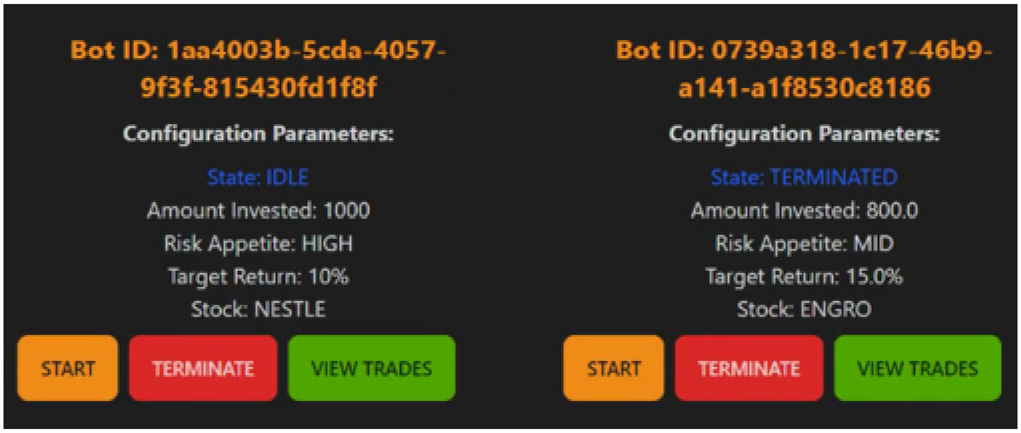
ID	INITIAL INVESTMENT(PKR)	CURRENT VALUE(PKR)	IN PROFIT ?	LOSS(PKR)	% LOSS
e16dfe33-4f87-431c-ab40-b9e46583e729	1000.0	953.5599999999997	No	-46.44	-4.64%

This shows the overall performance report of the selected bot.



Bot ID: 2cc77fe6-0c1a-4459-a667-599ab015c649
Stock Name: HBL
Amount: 100
Trade Type: 1
Start Price: 234.62
End Price: 239.76
Profit: Yes

This screen shows the trading history graph of the selected bot with start and end price of the trades as points on the graph. All the details of the trade are shown when a point is hovered upon.



Bot ID: 1aa4003b-5cda-4057-9f3f-815430fd1f8f

Configuration Parameters:

State: IDLE
Amount Invested: 1000
Risk Appetite: HIGH
Target Return: 10%
Stock: NESTLE

Bot ID: 0739a318-1c17-46b9-a141-a1f8530c8186

Configuration Parameters:

State: TERMINATED
Amount Invested: 800.0
Risk Appetite: MID
Target Return: 15.0%
Stock: ENGRO

These are the current bot instances shown to the analyst when he/she selects an investor. The analyst can either start, terminate, or view trades for the selected investor.

8. Project Security

Project security is a crucial aspect of any web application, especially those that involve financial transactions. Security measures must be implemented at every stage of the application's development to prevent unauthorized access, data breaches, and other security threats. The security of the application encompasses several aspects, including user authentication and authorization, data encryption, secure communication protocols, and secure storage of sensitive data. The application must be designed with security in mind, and thorough testing and evaluation must be conducted to identify and address any potential vulnerabilities or weaknesses. By prioritizing project security, the application can provide users with a secure and trustworthy platform for their financial transactions, protecting them from potential harm.

a. Project Threats

There are several potential security threats to the autonomous trading bot web application. Some of the most significant threats include:

1. Unauthorized Access: This is one of the most common threats to web applications. Unauthorized access to sensitive data can result in financial losses and damage to the application's reputation. The most vulnerable information/functionality from a security perspective in our project is user data, including their financial information, personal data, and trading history.
2. Malware Attacks: Malware attacks can cause significant harm to web applications, including data breaches and disruption of service. The most vulnerable information/functionality from a security perspective in our project is the trading bot's code and algorithm. Malware attacks can exploit vulnerabilities in the code to manipulate the bot's trading decisions and cause financial losses.
3. Denial of Service Attacks: Denial of service attacks can disrupt the application's availability and prevent users from accessing their accounts. The most vulnerable information/functionality from a security perspective in our project is the server and the database. Denial of service attacks can overload the server, causing the application to crash and preventing users from accessing their accounts.

To mitigate these security threats, the development team must implement robust security controls, including strong user authentication and authorization protocols, data encryption, and regular security testing and evaluation. Additionally, access controls and monitoring should be in place to identify and prevent unauthorized access attempts. By prioritizing security in the application's development, the team can reduce the risk of security breaches and ensure that the application is secure and trustworthy for its users.

b. Potential Losses

Potential losses from security breaches can be significant, ranging from financial losses due to stolen data or unauthorized access to funds, loss of customer trust and reputation, and legal ramifications. The loss of customer trust and reputation can be especially damaging as it can take a long time to rebuild and can result in a loss of business.

1. Unauthorized Access: If unauthorized access to sensitive data occurs, the potential losses can include financial loss due to stolen data or unauthorized access to funds, loss of customer trust and reputation, and legal ramifications. Additionally, there could be a total business loss if the security breach is significant enough to force the application to shut down.
2. Malware Attacks: If malware attacks exploit vulnerabilities in the trading bot's code, they can manipulate the bot's trading decisions and cause financial losses for investors. Additionally, malware attacks can lead to a loss of customer trust and reputation, and legal ramifications.
3. Denial of Service Attacks: If a denial of service attack disrupts the application's availability, it can prevent users from accessing their accounts, leading to a loss of business and revenue. Additionally, a significant denial of service attack can damage the application's reputation and lead to legal ramifications.

In summary, the potential losses from these security risks are significant, ranging from financial loss and legal ramifications to a loss of customer trust and reputation. It is essential to implement robust security controls to mitigate these risks and ensure that the application is secure and trustworthy for its users.

c. Security Controls

Controls to address the identified security threats in the autonomous trading bot web application include:

Unauthorized Access:

- Multi-factor authentication for user login: This control falls under the category of protective controls, as it helps prevent unauthorized access by requiring additional verification beyond the username and password.
- User role management: This control falls under the category of detective and protective controls, as it helps detect unauthorized access attempts and limits access to sensitive data and functionality to authorized users.
- Audit logs: This control falls under the category of detective controls, as it helps detect unauthorized access attempts by logging all user activity and login attempts.

Malware Attacks:

- Input validation: This control falls under the category of protective controls, as it helps prevent malware attacks by validating user inputs and preventing malicious inputs from being executed.
- Code reviews: This control falls under the category of detective controls, as it helps detect vulnerabilities in the trading bot's code that can be exploited by malware attacks.

Denial of Service Attacks:

- Load balancing and server redundancy: This control falls under the category of responsive and recovery controls, as it helps respond to and recover from a denial of service attack by distributing the load across multiple servers and maintaining application availability.
- Rate limiting: This control falls under the category of protective controls, as it helps prevent denial of service attacks by limiting the number of requests that can be made to the server in a given time period.

In summary, the controls to address the identified security threats in the autonomous trading bot web application fall into the categories of detective, protective, responsive, and recovery. By implementing these controls, the development team can mitigate the risks of security breaches and ensure that the application is secure and trustworthy for its users.

d. Static and Dynamic Security Scanning Tools

For the autonomous trading bot web application, the following static and dynamic security scanning tools can be used:

Static Scanning Tools:

- SonarQube: SonarQube is a widely used open-source static scanning tool that supports multiple languages, including Java, Python, and JavaScript. It provides a comprehensive analysis of the application's source code and detects potential vulnerabilities and weaknesses.
- ESLint: ESLint is a popular static scanning tool for JavaScript. It provides automated code analysis and identifies potential security issues, coding errors, and other issues that could affect the application's security.

Dynamic Scanning Tools:

- OWASP ZAP: OWASP ZAP is a widely used open-source dynamic scanning tool that can detect potential vulnerabilities in web applications. It provides an automated testing framework that simulates real-world attacks to identify potential vulnerabilities.
- Burp Suite: Burp Suite is a popular dynamic scanning tool that provides a comprehensive testing framework for web applications. It includes a proxy server, a scanner, and other tools that can be used to identify potential vulnerabilities in the application.

Considering the technologies used in the project, Next.js for the frontend, Flask for the backend server, and PostgreSQL for persistent storage, SonarQube and ESLint are suitable static scanning tools. For dynamic scanning, OWASP ZAP and Burp Suite are appropriate tools to identify potential vulnerabilities in the web application. By using these scanning tools, the development team can identify and address potential security threats early on in the development process, reducing the risk of security breaches in the future.

9. Risk Management

Potential Risks and Mitigation Strategies

Sr.	Risk Description	Mitigation Strategy
1.	Cyber Security Breaches	As mentioned in the above section, if someone can steal the token will get access to the system so we have assumed that user will keep it secure. We can use Paseto instead of JWT. Additionally we can use Firewall and Two factor authentication
2.	Financial Losses	Train models more efficiently for results. Regularly monitor the transactions to avoid potential losses.
3.	Team Management	Strong team management and scheduling is required as the system is based on financial values. Use of management applications such as Jira for smooth management and slack or team for smooth official communication.
4.	Market Volatility	Conduct market analysis regularly to avoid potential risk and opportunities. Develop strong and diversified investment strategies.
5.	Realtime-data error	Since, system is dependent on PSX stock APIs so if there system gets down somehow, our system would get freezed. To avoid this we will make a smooth communication with PSX and readily resolve the issue at earliest
6.	Data Loss	Create backups and recovery procedures. Regularly test backup systems and conduct security audits to identify vulnerabilities.
7.	News change fluctuation and dependency effect	In next version of the system integrate sentiment analysis and dependency techniques to mitigate the issue.
8.	Compliance violations	Stay upto date with laws and regulations and accordingly. Develop compliance policies and implement them along with regular training of the employees.
9.	Financial Fraud	Keep rigorous accounting procedures and financial controls. Regularly audit the transactions and have independent auditor to manage it.

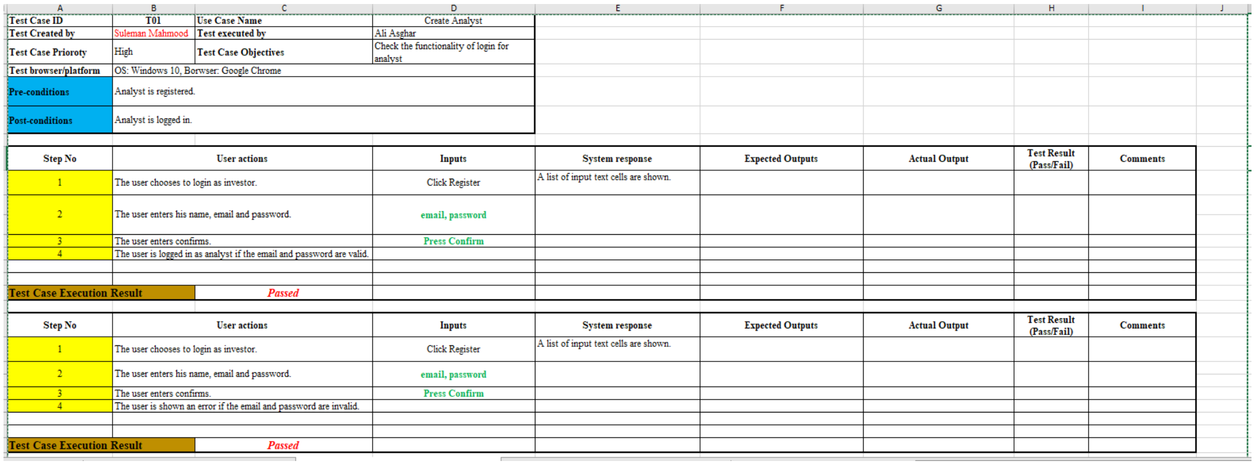
10.	Legal disputes	Disputes can be raised between an investor and analyst. Make legal contracts and agreements within the parties. Maintain insurance coverage and retain legal council to manage disputes in order to reduce the risk

10. Testing and Evaluation

For testing we made a procedure. Since, we were using unit testing so we use pytest for our backend testing. We divided our project in sub domains and updated the code on regular basis to avoid confusions. In terms of development strategy, we used test driven design, writing unit test code before implementing the logic. We also performed pair programming for the most part, allowing for quicker debugging of each other's code.

Our sub domains included:

- Authentication
- Creation of the investor and analyst
- Creation of Bots and Management
- Data object testing

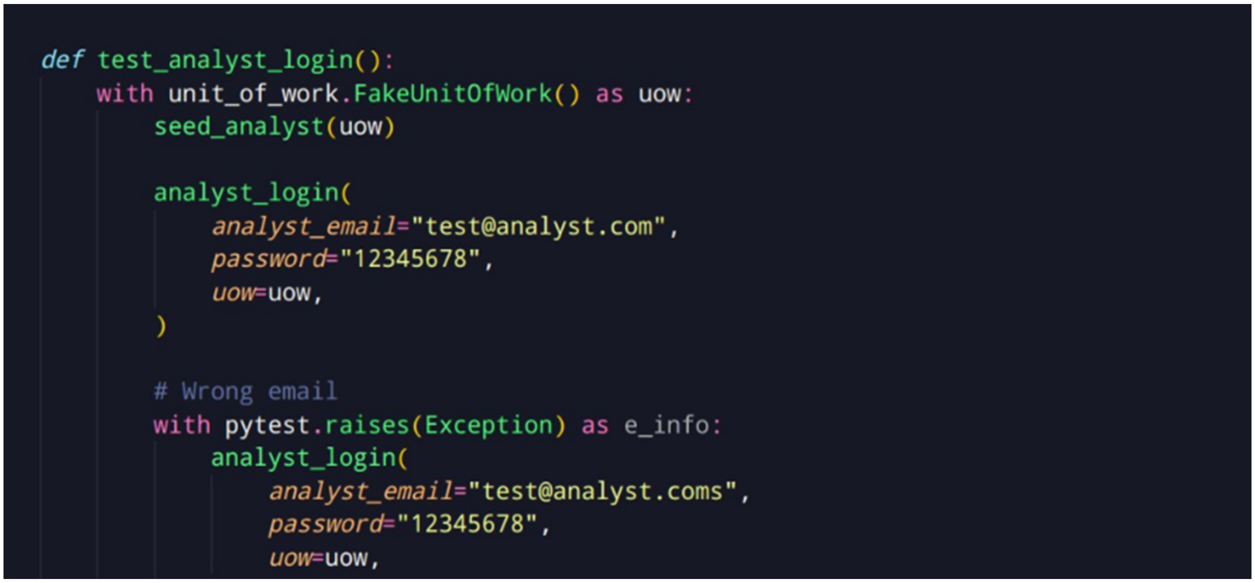


Test Case ID	T01	Use Case Name	Create Analyst
Test Created by	Suleman Mahmood	Test executed by	Ali Asghar
Test Case Prioroty	High	Test Case Objectives	Check the functionality of login for analyst
Test browser/platform	OS: Windows 10, Borwser: Google Chrome		
Pre-conditions	Analyst is registered.		
Post-conditions	Analyst is logged in.		

Step No	User actions	Inputs	System response	Expected Outputs	Actual Output	Test Result (Pass/Fail)	Comments
1	The user chooses to login as investor.	Click Register	A list of input text cells are shown.				
2	The user enters his name, email and password.	email, password					
3	The user enters confirms.	Press Confirm					
4	The user is logged in as analyst if the email and password are valid.						
Test Case Execution Result	Passed						

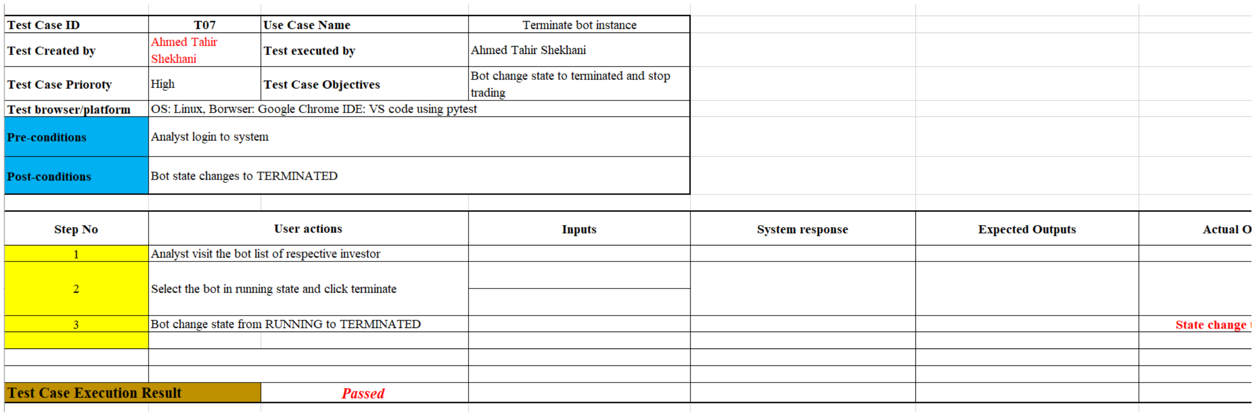
Step No	User actions	Inputs	System response	Expected Outputs	Actual Output	Test Result (Pass/Fail)	Comments
1	The user chooses to login as investor.	Click Register	A list of input text cells are shown.				
2	The user enters his name, email and password.	email, password					
3	The user enters confirms.	Press Confirm					
4	The user is shown an error if the email and password are invalid.						
Test Case Execution Result	Passed						

Analyst successfully logged in:



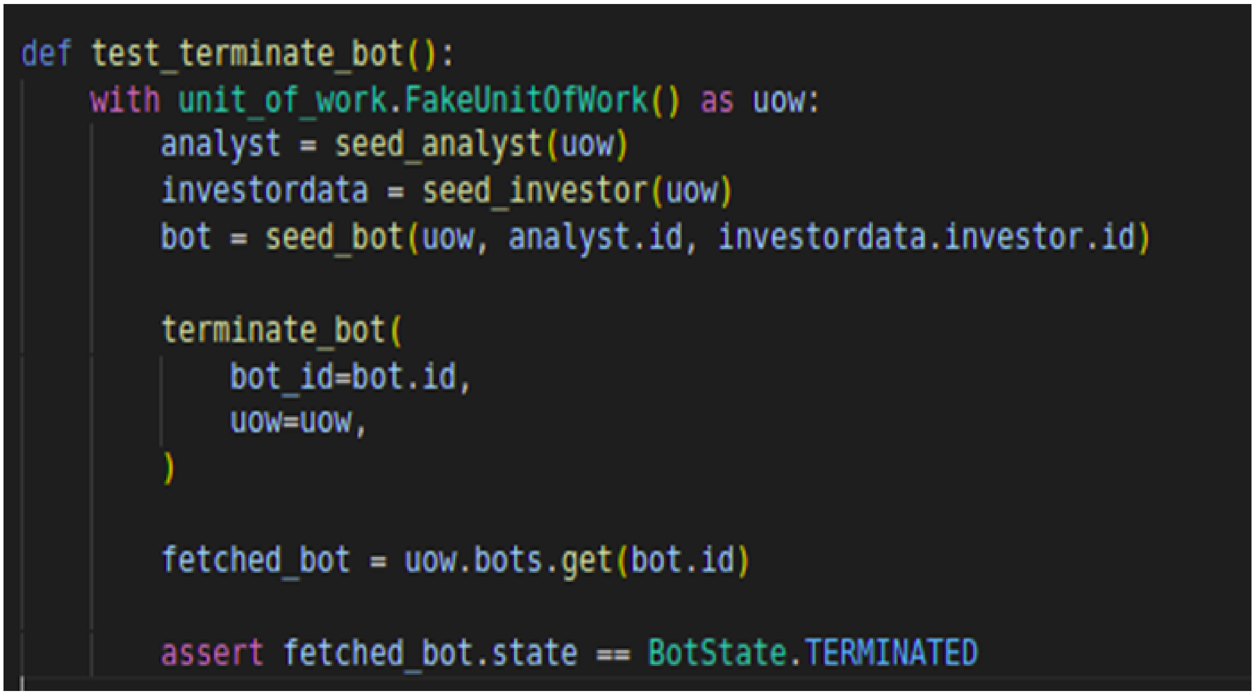
```python
def test_analyst_login():
    with unit_of_work.FakeUnitOfWork() as uow:
        seed_analyst(uow)

        analyst_login(
            analyst_email="test@analyst.com",
            password="12345678",
            uow=uow,
        )

        # Wrong email
        with pytest.raises(Exception) as e_info:
            analyst_login(
                analyst_email="test@analyst.coms",
                password="12345678",
                uow=uow,
```

Bot state changed to terminate:



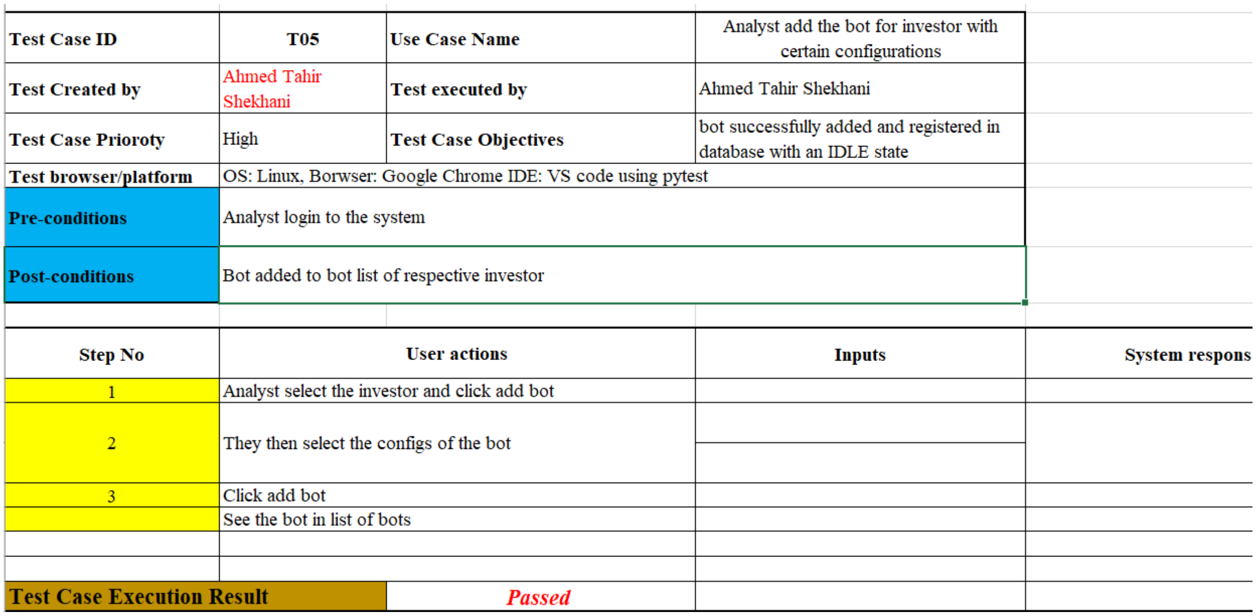
Test Case ID	**T07**	**Use Case Name**	Terminate bot instance
Test Created by	Ahmed Tahir Shekhani	**Test executed by**	Ahmed Tahir Shekhani
Test Case Prioroty	High	**Test Case Objectives**	Bot change state to terminated and stop trading
Test browser/platform	OS: Linux, Borwser: Google Chrome IDE: VS code using pytest		
Pre-conditions	Analyst login to system		
Post-conditions	Bot state changes to TERMINATED		

Step No	**User actions**	**Inputs**	**System response**	**Expected Outputs**	**Actual O**
1	Analyst visit the bot list of respective investor				
2	Select the bot in running state and click terminate				
3	Bot change state from RUNNING to TERMINATED				**State change**
Test Case Execution Result	***Passed***				



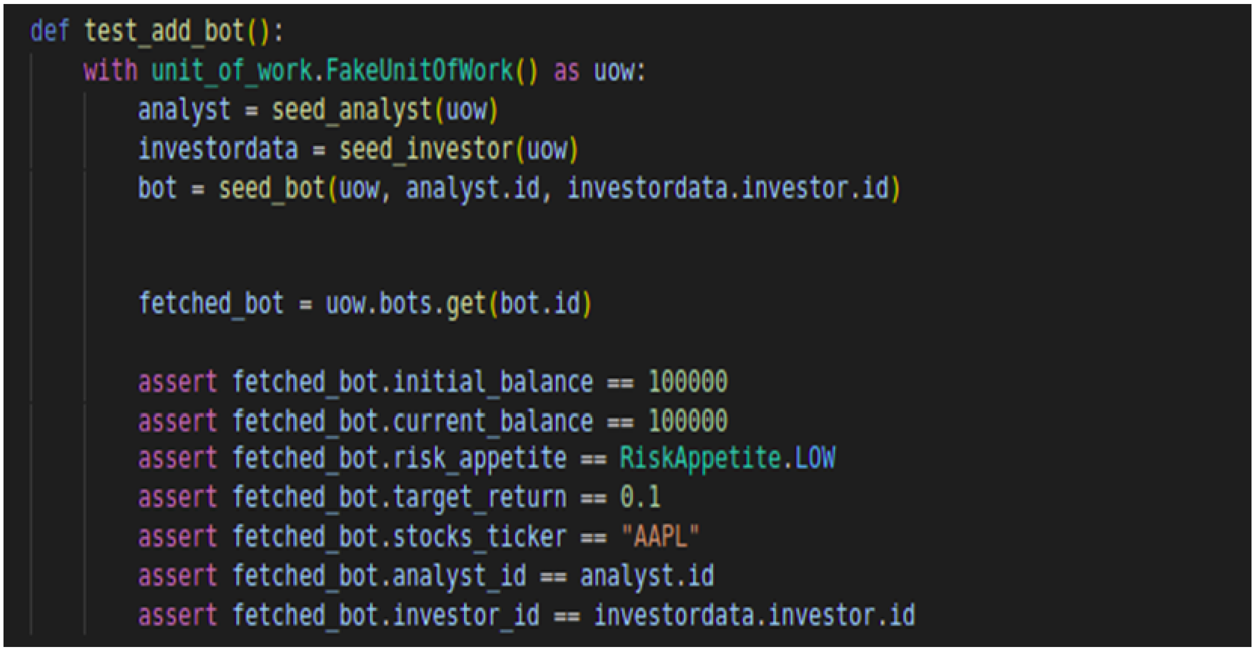
```python
def test_terminate_bot():
    with unit_of_work.FakeUnitOfWork() as uow:
        analyst = seed_analyst(uow)
        investordata = seed_investor(uow)
        bot = seed_bot(uow, analyst.id, investordata.investor.id)

        terminate_bot(
            bot_id=bot.id,
            uow=uow,
        )

        fetched_bot = uow.bots.get(bot.id)

        assert fetched_bot.state == BotState.TERMINATED
```

Adding a Bot:



Test Case ID	**T05**	**Use Case Name**	Analyst add the bot for investor with certain configurations
Test Created by	Ahmed Tahir Shekhani	**Test executed by**	Ahmed Tahir Shekhani
Test Case Prioroty	High	**Test Case Objectives**	bot successfully added and registered in database with an IDLE state
Test browser/platform	OS: Linux, Borwser: Google Chrome IDE: VS code using pytest		
Pre-conditions	Analyst login to the system		
Post-conditions	Bot added to bot list of respective investor		

Step No	User actions	Inputs	System respons
1	Analyst select the investor and click add bot		
2	They then select the configs of the bot		
3	Click add bot		
	See the bot in list of bots		
Test Case Execution Result	***Passed***		



```
def test_add_bot():
    with unit_of_work.FakeUnitOfWork() as uow:
        analyst = seed_analyst(uow)
        investordata = seed_investor(uow)
        bot = seed_bot(uow, analyst.id, investordata.investor.id)


        fetched_bot = uow.bots.get(bot.id)

        assert fetched_bot.initial_balance == 100000
        assert fetched_bot.current_balance == 100000
        assert fetched_bot.risk_appetite == RiskAppetite.LOW
        assert fetched_bot.target_return == 0.1
        assert fetched_bot.stocks_ticker == "AAPL"
        assert fetched_bot.analyst_id == analyst.id
        assert fetched_bot.investor_id == investordata.investor.id
```

Tools used for testing:

- Pytest library
- Selenium

11. Deployment Guidelines

For deployment of our backend, we used GCP AppEngine B2 instance and for frontend we used Vercel. Our postgresSQL instance is deployed on elephantSQL.

Deployment guidelines our follows:

- Developed the application at the github repo link mentioned (https://github.com/ahmedtahirshekhani/P03-AutonomousTradingBot/tree/main/P03-AutonomousTradingBot/Development/Sprint-4/backend)
- Build the application
- Created a GCP instance
- Configure AppEngine settings with allocation of required resources (for validation we used B2)
- Configured the application in Linux environment along with environment variables with secret codes.
- Deployed using GCP single-line terminal statements

Frontend:

- Developed the application at the github repo link mentioned (https://github.com/ahmedtahirshekhani/P03-AutonomousTradingBot/tree/main/P03-AutonomousTradingBot/Development/Sprint-4/frontend)
- Connected the Github repo with Vercel
- Configured the root folder with package.json
- Selected the application type (React NextJS)
- Added the environment variables
- Started the deployment settings.

Frontend: https://autonomous-trading-bot.vercel.app/

Backend on GCP and will run on-demand (due to pricing constraints)

Database: https://www.elephantsql.com/

Testing Credentials:

Analyst credentials:

Email: suleman@analyst.com

Pass: 12345678

Investor credentials:

Email: talal@investor.com

Pass: 3d80f883

12. Conclusion

a. Summary

The project allowed the members to work around each other's area of expertise and experience. Ahmed and Suleman have industry experience with software development and shared best practices with Talal and Ali. Ali is passionate for frontend development and steered the UI and frontend development. Talal worked on ML models, their apis and the architecture of the end to end predictions. This project forced us to read up on different ML based approaches for stock predictions and also made us aware of the challenges faced in the domain.

In terms of development strategy, we used test driven design, writing unit test code before implementing the logic. We also performed pair programming for the most part, allowing for quicker debugging of each other's code.

The project conclusion has ended with a complete frontend design, and the ML model completely isolated from the rest of the system. This allows for any new improved predictive algorithm to be replaced in the model class and use it for prediction. The software that we have developed is the first step that will allow any future system to be built on top of our software.

b. Challenges

The first biggest challenge was finding free psx api that provided real time data for our bot to operate on. We could only manage an api that still isn't 100% reliable and gives data at the end of each day. Secondly stock market prediction further requires sentiment analysis of the markets and global as well as local politics, however we could not find data corresponding to it, and if we would have scraped data from news sites it would include bias in our data set.

c. Future

1) Adding sentiment analysis as an additional parameter to make prediction on the stock market
2) Create a unified model that decides which stock to trade on
3) Integrate crypto trading on the platform
4) Allow model to have even greater granularity white training based on trader preference
5) Find stock correlation to make better informed predictions
6) Make a psx data scraper that provides real time granular data, for each minute, hour and day separately to better be able to make trades

13. Review checklist

Before submission of this report, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

Chapter/Section Name	Reviewer Name(s)
Deployment Guidelines, Database Design and Web Services, Requirements Specifications	Ali Asghar
System User interface, Project security, Software Development methodology and plan, System Architecture	Ahmed Tahir Shekhani

14. References