Group symbol: **C**

Team: **3**

Project title: **<Trading Bot>**

Team members (*filled by PM, Team Leader*):

No	Name	Surname	Student ID	Role
1	Bohdan	Kyryliuk	267855	*PM, Team Leader*
2	Serhii	Ohurtsov	251530	*Team member*
3	Sergiy	Vergun	251203	*Team member*
4	Ilgin	Sogut	282416	*Team member*

2. Requirements specification **(F2)**

2.1. Functional Requirements Specification

In this section, provide the table for functional requirements, including symbol, type (e.g. business logic, user interface, data exchange, etc.) description, significance (MoSCoW) and source (Stakeholder).

Symbol	**Type**	**Description**	**Significance**	**Source**
FR1	User Interface	User registration and profile management	Must have	End User
FR2	User Interface	User login and authentication mechanism	Must have	End User
FR3	Business Logic	Algorithm to analyze market data for trading signals	Must have	ML Engineer
FR4	Data Exchange	Fetching real-time cryptocurrency prices	Must have	Backend Developer
FR5	User Interface	Dashboard showing bot performance & current trades	Should have	End User
FR6	Business Logic	Allow user to set trading parameters (e.g., stop loss)	Should have	End User
FR7	Business Logic	Automatically execute trades when criteria are met	Must have	ML Engineer
FR8	Data Exchange	Integration with cryptocurrency exchange APIs	Must have	Backend Developer
FR9	User Interface	Notifications & alerts mechanism for trade & bot events	Could have	End User
FR10	Business Logic	Mechanism to start, stop, and pause the trading bot	Must have	End User

2.2. Non-Functional Requirements

In this section provide the table for non-functional requirements that includes symbol, type (e.g., efficiency, standards, constraints, etc.), description, significance (MoSCoW) for the project, source (Stakeholder). Each requirement should also have specified a verification method – a description of a confirmation method whether a requirement has been fulfilled or not in the most measurable and objective way.

Sym bol	Type	Description	Significance	Source	Verification Method
NFR 1	Efficiency	The bot must process trading signals in less than 1 second	Must have	ML Engineer	Time the bot from receiving a signal to executing a trade
NFR 2	Reliability	The bot should have an uptime of 99.9%	Must have	Backend Developer	Monitor the bot's uptime over a month
NFR 3	Security	All user data must be encrypted and securely stored	Must have	End User	Security audit and penetration testing
NFR 4	Standards	The bot must comply with cryptocurrency trading regulations	Must have	Legal Team	Legal review and compliance checks
NFR 5	Efficiency	UI should load within 2 seconds	Should have	Frontend Developer	Measure UI load times across various devices
NFR 6	Scalability	The bot should support a minimum of 10,000 simultaneous users	Could have	Business Analyst	Load testing with simulated users
NFR 7	Interoperability	The bot should integrate seamlessly with at least three major cryptocurrency exchanges	Should have	Backend Developer	Integration tests with multiple exchange platforms
NFR 8	Usability	User should be able to navigate all features within three clicks	Could have	End User	Usability testing and feedback
NFR 9	Constraints	The bot should operate within the API rate limits set by exchanges	Must have	Backend Developer	Monitoring of API calls to ensure they remain within allowable limits
NFR 10	Security	Two-factor authentication (2FA) must be implemented for user accounts	Should have	End User	Test the 2FA mechanism during user login

2.3. Use Case Diagram

You should prepare the use case diagram in UML 2.5 depicting the roles of stakeholders who are users of the project. It should also present the high-level concept of system usage divided into modules. Use case diagram should not *cover common business logic (e.g. registration and logging in, CRUD operations).*

Actors (Roles of Stakeholders who are users):

Development Team:

- End user
- Admin
- Trading platform

Use Cases:

Configure Trading Strategies:
Actor: End User
Module: Strategy Configuration
Description: Allows end users to set up and modify their trading strategies based on various indicators and preferences.

Toggle Bot Trading:
Actor: End User
Module: Trading Automation Control
Description: Provides the ability for the end user to enable or disable the automated trading bot function.

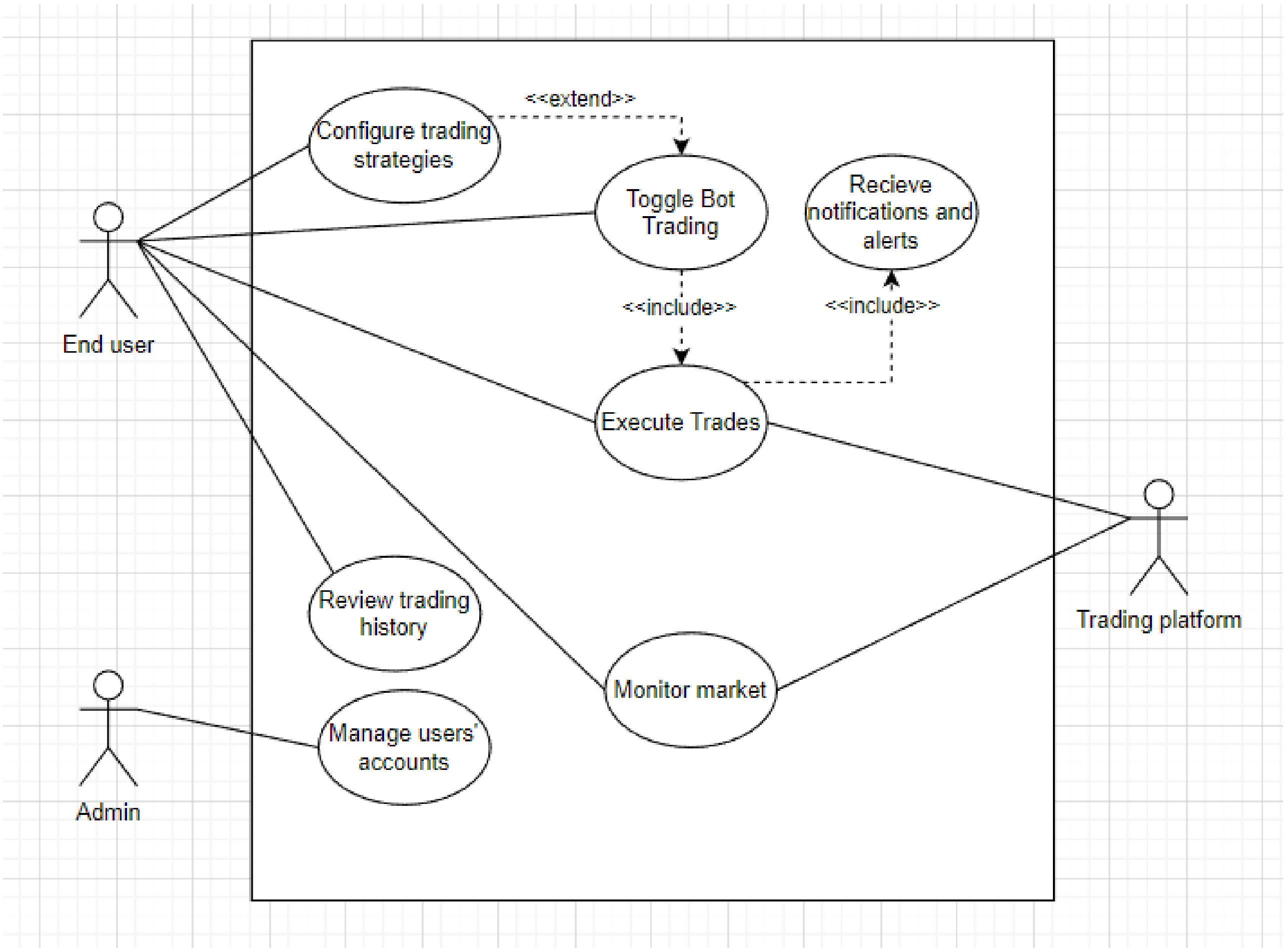
Execute Trades:
Actor: End User, Trading Bot (system)
Module: Trade Execution
Description: Executes trades on behalf of the user automatically or manually, based on the predefined strategies and settings.

Review Trading History:
Actor: End User Module: User Portfolio Analysis
Description: Enables end users to view past trading activities and performance metrics.

Manage Users' Accounts:
Actor: Admin
Module: User Account Management
Description: Allows administrators to manage account settings, user roles, and access permissions.

Monitor Market:
Actor: End User, ML Engineer Module: Market Data Monitoring
Description: Continuous monitoring of market conditions and data to inform trading strategies and decisions.

Receive Notifications and Alerts:
Actor: End User
Module: Notification Management
Description: Sends timely alerts and notifications to end users about significant events or changes in the market, or the status of trades.


Configure trading strategies
<<extend>>
Toggle Bot Trading
Recieve notifications and alerts
End user
<<include>>
<<include>>
Execute Trades
Trading platform
Review trading history
Monitor market
Admin
Manage users' accounts