Design and implement an efficient in-memory trading system similar to a stock exchange, where registered users can place, execute and cancel trades. The system should demonstrate synchronization and concurrency in a multi-threaded environment.

Your system should be able to perform the following items:

Functional Requirements:

Your system should support the following functionalities:

- A registered user can place, modify, and cancel his orders.
- A user should be able to query the status of his order
- ***[IMPORTANT] - The system should be able to execute trades based on matching buy and sell orders. A trade is executed when the buy and sell price of two different orders match. (Buy price greater than or equal to Sell Price). If multiple eligible orders can be matched with the same price, match the oldest orders first.***
- ***[IMPORTANT] - Concurrent order placement, modification, cancellation, and execution should be handled appropriately.***
- The system should maintain an order book per symbol, which holds all the current unexecuted orders.

Your system should store at least the following mentioned details.

- User details
    - User ID
    - User Name
    - Phone Number
    - Email Id
- Orders
    - Order ID
    - User ID
    - OrderType (Buy/Sell)
    - Stock Symbol (eg: RELIANCE, WIPRO etc.)
    - Quantity
    - Price
    - Order Accepted Timestamp
    - Status (ACCEPTED, REJECTED, CANCELED)
- Trades
    - Trade ID
    - Trade Type (Buy/Sell)
    - Buyer Order Id
    - Seller Order Id

    - Stock Symbol
    - Quantity
    - Price
    - Trade Timestamp

Additional functionality, but not compulsory [implement if time permits]:

- Implement trade expiry. A trade should be automatically canceled if that trade is not executed within a specific time.

Expectation:

- Your code should be executable (at worst partial running would work) & should be clean.
- Your code should be adequately refactored, and exceptions should be gracefully handled.
- Your code should store all the attributes explained under the "Stores" section.
- Your code should cover all the functionality in the "Supports" section.
- If you get extra time, you can code for "Additional functionality, but not compulsory." This will get you extra credit.

Guidelines:

- You don't have to build the user registration part. Consider some dummy users registered and use those in the entire system.
- You should use the in-memory data structure of your preferred language to store the data but have the right abstractions so that other persistent stores can be plugged in.

Evaluation criteria:

- Executable code.
- Code readability and testability
- Refactored code
- Abstraction
- Object-Oriented concepts.
- Language proficiency.
- **[execution time limit] 4 seconds (py3)**
- **[memory limit] 1 GB**