BSc Computer Science

CM2005 Object Oriented Programming

Midterm coursework assignment

INTRODUCTION

The goal for this assignment is to develop a technical analysis toolkit for the currency exchange platform. A technical analysis toolkit typically provides features and methods to process raw data, visualise, analyse, and perform computations to gain meaningful insights so that suitable actions can be taken by the decision makers. For this coursework, there are five main tasks which involve developing functionalities to read and update CSV files containing market trading data and user trading activities/wallet. The market trading file provided should not be modified. However, ALL user-related data should be stored, updated and read in a separate file (*this includes user login and personal details, wallet and trading transaction information*). A suitable program menu should be displayed to the users to showcase the functionalities involved in these tasks and capture user inputs robustly.

REPORT

A mandatory single report must be created containing suitable discussion and evidence for ALL tasks. The document must contain at least (i) a suitable explanation, (ii) implementation logic discussion (*with fragments of pseudocode where relevant*) and (iii) a screenshot and evidence of the final output. A total of 30 marks (*six marks per task for five tasks*) are dedicated to report writing.

IMPLEMENTATION

TASK 1: Loading Trading Data and Computing Candlestick Data

Task 1 involves loading trading orders from the provided CSV file and creating a summary table for a particular product.

1. The user should be able to enter a product combination, i.e., '*ETH/USDT*' and by default yearly summary should be displayed.
2. The user should also be able to specify and filter the product's record by date range *(i.e., daily, monthly or yearly)*.

Here is an example of the fields needed for candlestick data for '*asks on ETH/UDST*'.

Date	Open	High	Low	Close
2023-06-01	100	120	80	110
2023-06-02	110	130	100	120
2023-06-03	90	100	70	80
2023-06-04	95	110	70	100
2023-06-05	80	120	75	115

The columns represent the following information:
- **Open**: is the first trading price that occurs during the specified time period.
- **Close**: the last trading price that occurred during the specific time period.
- **High**: the highest price value seen in this time frame.
- **Low**: lowest price value seen in this time frame.

Please note that you will need to do this summary table for asks and bids for a given product separately. In addition, you should implement a suitable candlestick class structure to be used as a collection of objects later on throughout the program.

TASK 2: Login or Register a Trading User profile

Task 2 involves allowing users to register or log in with their existing username and password.

1. Register a new user if the record is not found within your own temporary CSV file containing user details.
   1.1. *Request the user's full name, email address and password.*
   1.2. *Generate a unique 10-digit number for the username and show it to the user to log in later on.*
   1.3. *The login password should be stored as an encrypted hash value using $std::hash$ function and later used to compare against user input as a value to log in.*
   1.4. *Store user details and login credentials in a temporary CSV file (typically, a database server is utilised here, but this is out of scope for our purpose) in the structure you prefer.*
   1.5. *To avoid duplicates or users attempting to re-register, you should check if the user with the same email address and full name already exists in the temporary CSV file.*
   1.6. *Link the user with their own wallet to track their trading activities.*

2. User must be logged in with valid credentials and verified by checking the records in a temporary password CSV file.
    2.1. *Compare the hashed password input by the user and the stored record.*
3. Decide on an appropriate step to remind the user of their login details or reset their password.

TASK 3: User Wallet & Trading Transactions History

Task 3 involves maintaining the user wallet & trading transaction history records and displaying suitable summary statistics.

1. Allow the user to simulate depositing more money or taking away money (*i.e., to pay themselves or after trading*) and view the wallet balance from the temporary CSV file.
2. Log all user transactions and wallet balance information in a temporary CSV file. *Ensure to include suitable discussions on the data structure and evidence in the report.*
3. Display recent trading transactions made by the user (*i.e., last five transactions or by product*).
4. Provide a summary of statistics of user activity (*i.e., no. of asks and bids for all products and for a specific product, total money spent within a given timeframe*).

TASK 4: Simulating User Trading Activities

Task 4 involves simulating new user trading activities and updating their wallet accordingly.

1. Create at least FIVE new ask and bid orders for ALL products available with a suitable price using the current system timestamp.
2. Discuss and justify how ask and bid prices were calculated, and how you managed the large date gap between the last record (i.e., year 2022) and the current system timestamp (i.e., year 2025 or 2026).
3. All user transactions should be appended and persisted to the temporary CSV file.

TASK 5: Printing user interaction menu to support user login & trading operations

Task 5 requires you to print the user interaction menu to support the trading platform operations discussed in previous tasks.

1. All user inputs should be gracefully handled when incorrect user inputs are detected and avoid abruptly terminating.
2. Discuss key measures taken to verify and validate user inputs in the report and suitable comments in your code.

WHAT TO SUBMIT

1. A **PDF** file containing all the **code**. You can concatenate all .cpp and .h files into a single text file and then save them as a PDF.
2. A **ZIP** file containing all your code and supplementary files.
3. A **PDF** file containing your **report**, wherein you should describe how you carried out each task with a supporting screenshot and description of the final output.
4. A maximum of a 3-minute system demonstration **video** (in an MP4 format) with a voice narrative showing key features developed and the code logic behind it. The video should contain the following:
   a. *Briefly review your code structure in your IDE, then launch the program.*
   b. *Include voice narration (audibly- not computer-generated voice or text) explaining the feature and code logic as you enter commands and other user interface elements.*
   c. *Demonstration of functionality pertaining to ALL the requirements.*

MARKING CRITERIA

Item	Marks
REPORT: *must include (i) a suitable explanation [2], (ii) implementation logic discussion (with a fragment of pseudocode where relevant) [2] and (iii) a screenshot [2] as evidence of the final output for all tasks.*	**30**
TASK 1: Loading Trading Data and Computing Candlestick Data	6
TASK 2: Login or Register a Trading User profile	6
TASK 3: User Wallet & Trading Transactions History	6
TASK 4: Simulating User Trading Activities	6
TASK 5: Printing user interaction menu to support user login & trading operations	6
CODE IMPLEMENTATION *	**50**
TASK 1: Loading Trading Data and Computing Candlestick Data	10
TASK 2: Login or Register a Trading User profile	10
TASK 3: User Wallet & Trading Transactions History	10
TASK 4: Simulating User Trading Activities	10
TASK 5: Printing user interaction menu to support user login & trading operations	5
Code style (*i.e., appropriate indentation and descriptive comments*) and OOP concepts for modular & reusable code.	5
DEMO & PDFs	**20**
Video demonstration with voice narration of features implemented and code execution.	10
Submit correct items: code as text in PDF, report as PDF, a ZIP file for the code & a video +.	4
Clearly label all sections of the code that you personally wrote without assistance *	6
TOTAL	**100**

* **Please note**: your **CODE** implementation will be marked based on your evidence provided in the **CODE PDF** file, where you clearly indicate where the code is located for ALL TASKS that you have personally written without assistance. Remember to submit a REPORT as a PDF file providing details of how all the tasks are implemented, what logic/methods were adopted and why. Also, screenshots of the results/output with reference to code fragments (*i.e., copy/ paste code in the report or provide page numbers where relevant to the CODE PDF file*).

+ If you do NOT submit CODE PDF AND REPORT PDF files along with the program files in a ZIP file, NO MARKS will be awarded for the respective tasks for the CODE implementation marking criteria.

[END OF COURSEWORK ASSIGNMENT]