KOCAELI UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

Simultaneous Order and Inventory Management

Objective: This project aims to design a system that performs concurrent order management and stock updates. The aim is to solve the problem of concurrent access to the same resource using multithreading and synchronization mechanisms and to understand the use of concepts such as process, thread, mutex, semaphore and process priority.

Programming Languages: Java, JavaScript, Python, C, C#.

Databases: MySQL, PostgreSQL, MongoDB, Firebase Realtime Database.

Note: You can make web or desktop applications. Mobile applications will not be evaluated.

PROJECT REQUIREMENTS

Customers and Types:

- Customer information: *CustomerID, CustomerName, Budget,CustomerType (Premium/ Standard), TotalSpent*
- Customers should initially be assigned a random number, budgets should be assigned randomly in a similar way.

    - The number of clients should vary between 5-10.
    - The budget amount should vary between 500-3000 TL.
    - There must be at least 2 premium customers at the start.
- There are two types of customers:
    - Premium Customers: These customers have high priority and are processed
    - before regular customers.
      Regular (Standard) Customers: Their transactions are queued after Premium customers' transactions are completed.
- If a customer makes a purchase of over 2000 TL, he/she automatically becomes a Premium Customer.
- When more than one customer wants to buy the same product at the same time, a proper sequence of transactions between Premium and Regular customers should be ensured.
- Each customer can apply to purchase one or more product variants.
- A customer can purchase a maximum of 5 pieces of each product.

Admin:

- Admin manages the system:
    - Performs product addition, deletion and stock update operations.
- Admin transactions are carried out in parallel with customer transactions.
- Admin puts other operations on hold while accessing product and stock information.

Inventory Management:

- Each product is initially defined by the system with a fixed stock quantity.
- Initially there are 5 different products in the store and the information about the products is given in the table below:

ProductID	*ProductName*	*Stock*	*Price (TL)*
1	Product1	500	10
2	Product2	10	0
3	Product3	200	50
4	Product4	75	45
5	Product5	0	75

- Admin can add new products or delete existing ones.
- Admin can increase or decrease stock quantities.
- Product stocks may need to be updated by multiple customers simultaneously due to purchasing.
- If the stock of a product is not sufficient, the transaction is rejected.
- Stocks should be updated immediately upon purchase.

Budget Management

- Customers have a balance account and payment transactions are made from this balance.
- The transaction is rejected when the customer's balance is insufficient.
- Budgets are assigned randomly (between 500-3000 TL).

Dynamic Priority System:

- During customer transactions, the priority should change dynamically according to the transaction time.
- The priority ranking should be continuously updated based on criteria such as waiting time and customer type.
- Premium customers have high priority by default. For regular customers, the higher the waiting time, the higher the priority ranking.
- If a Standard customer becomes a Premium customer, the customer's current trade queue should not be touched. The client type change should take effect on the next trade.

Dynamic Priority Calculation:

- Each customer is assigned a priority score:

  $$PriorityScore = BasicPriorityScore + (WaitTime \times WaitTimeWeight)$$

  - Basic Priority Score: 20 for Premium customers and 10 for Regular customers.
  - Waiting Time: The time in seconds that the customer waits for the transaction.
  - Dwell Time Weighting: Determines the effect of the waiting time. Each second of waiting time has a weight of 0.5 points.
  - The ranking must be dynamically recalculated during each operation.

Statute of Limitations and Transaction Cancellation :

- The time-out check should start from the moment the client is placed in the queue.
- The timeout period must be reset to zero from the moment the order is processed.
- If a customer waits 15 seconds for a transaction and the transaction is not completed, the transaction is canceled and written to the log file.

Database

- The minimum tables and attributes expected to be in the database are given below:
  - Customers: *CustomerID, CustomerName, Budget*, *CustomerType, TotalSpent*
  - Products: *ProductID, ProductName, Stock, Price*
  - Orders: *OrderID, CustomerID, ProductID, Quantity, TotalPrice, OrderDate, OrderStatus*
  - Logs: *LogID, CustomerID, OrderID, LogDate, LogType, LogDetails.*

Logging and Monitoring :

- Each time a process starts, a log is recorded about the process.
- should be kept for both customer and admin transactions.
- Logs should be created in order of processing.
- Each log record must contain the following information:
    - Log ID
    - Customer ID
    - Log type: "Error", "Warning", "Notification"
    - Customer type: Premium or Standard
    - Product and quantity purchased
    - Transaction (purchase) time information
    - Transaction (purchase) result: If the transaction was completed successfully, "Purchase successful" is recorded as the transaction result. If the transaction failed, the appropriate message from the following error messages is recorded.

  Some Types of Error Messages:

    - "Insufficient product stock": Triggered if the product stock is not sufficient when a customer applies for a purchase.
    - "Time overrun":Processdeterminedmaximum while is triggered if it cannot be completed.
    - "Insufficient customer balance": Triggered if the customer's budget does not cover the total price of the selected product(s).
    - "Database Error": Connection issues or transaction during are triggered in situations such as deadlocks.

- Log Record Example:
    - Log ID: 4
    - Customer ID: 1
    - Log Type: Error
    - Customer Type: Premium
    - Product Product5
    - Quantity Purchased: 5
    - Transaction Time: 2024-11-28 14:32
    - Transaction Result: Product out of stock

UI Integration:

- Customer Panel
    - Customer List:
        - In the table, show information such as CustomerID, Name, Type (Premium/Normal), Budget, Waiting Time, Priority Score.
    - Order Form:

        - Allow the customer to make transactions with Product Selection, Quantity Entry and Place Order button.
    - Waiting Status
        - Her customer order during status (pending, in process, completed) can be visualized with coloring.
- Product Stock Status Panel
    - Product Table:
        - Information such as Product Name, Stock Quantity, Price is shown in a table.
        - Stock quantities are updated after each transaction.
    - Graphic Representation:
        - Stock status can be visualized with a bar or circular chart.
        - A visual warning can be added with a graph that changes color when the stock reaches a critical level.
- Log Panel
    - It lists the result of each transaction with real-time logging:
        - For example: "Customer 1 bought 2 units of Product3. Transaction Successful."
        - For errors: "Customer 2 wanted to buy 3 units of Product5. Insufficient Stock."
    - Logs should be displayed in the UI as a scrolling list in sequence.
- Dynamic Priority and Standby Panel
    - The Wait Time and Priority Score are shown in a table and updated with each trade.
    - You can show the customer queue with animation:
        - For example, the lists move up/down when the order sequence changes.
- Order Processing Animation
    - An animation indicator for orders in process:
        - For example: "Customer 1's order is being processed" with a loading bar or an animated icon.