Technical Exercise - Bank Account with UI

Problem Statement

We need to implement a solution that is able to track the balance of a bank account supporting credits, debits and balance enquiries. This system has an audit requirement where the last 1000 transactions need to be sent to a downstream audit system. This audit system requires transactions to be sent in batches however the total value of transactions in each batch must not exceed £1,000,000. We are charged for each batch sent therefore to save costs must minimize the number of batches sent. The expectation is that the solution will have the following key components:

- Producer which is responsible for generating transactions
- Balance Tracker which is responsible for:
    - processing the transactions
    - tracking the balance resulting from the transactions
    - publishing batches of balances to an audit system

Requirements

General

1	Your application should make use of Java, ReactJS and Maven
2	Your implementation is free to use libraries however an assessment should be made on the credibility of the library
3	Your solution should have appropriate and sufficient automated tests which validate requirements have been met.
4	AI tooling (e.g, ChatGPT, CoPilot, Codeium) can be used in the completion of this exercise however anything produced by AI should be well understood as it could be discussed as part of the interview process
5	Your implementation and any supporting documentation (if required) should make no reference to any real company names or individuals - everything should be fictitious. You should also make no reference to the role or interview process.
6	Your completed solution should be uploaded to a publically available repository on GitHub. Once the interview process is complete, the repository should be deleted.
7	You should reach out to your recruiter if you have any issues with completing the exercise

Producer

1	The producer application must be a separate JVM to the Balance Tracker
2	The logic to produce transactions should randomly generate a transaction that has an ID and an Amount. Transactions should be a mixture of credits and debits with values between £200 and £500,000.
3	An appropriate domain object should be created and called Transaction. Debits will have a negative amount and credits a positive amount.
4	The transaction ID can be any randomly generated value.
5	You can assume that all transactions being produced are for the same account therefore an account ID is not required.
6	The production of the transactions should be performed on two separate dedicated threads - one for credits and one for debits
7	You should produce credits and debits at a rate of 50 per second. i.e. 25 debits per second and 25 credits per second

Balance Tracker

1	Your solution should have sufficient functional test coverage to ensure any calculations are accurate

2	Your solution should define and implement the following interface (this can be extended if deemed necessary):
	
	`/**`
	` * Service to aggregate transactions tracking the overall balance for an account.`
	` */`
	`public interface BankAccountService {`
	
	`    /**`
	`     * Process a given transaction - this is to be called by the credit and debit generation threads.`
	`     *`
	`     * @param transaction transaction to process`
	`     */`
	`    void processTransaction(Transaction transaction);`
	
	`    /**`
	`     * Retrieve the balance in the account`
	`     */`
	`    double retrieveBalance();`
	`}`

3	You can assume that all transactions being produced are for the same account therefore only a single instance of the above class is required
4	The functionality to retrieve the balance should be exposed over an appropriate REST API.

Balance Tracker UI

1	The balance tracker user interface should display Account ID and the balance. • Clearly label the fields for Bank Account Number and Balance. • Ensure that the balance displayed on the UI is refreshed periodically.
2	You can use any CSS framework for rapidly making the user interface.

Audit System

1	Submissions to the audit system must contain **exactly** 1000 transactions
2	Each submission to the audit system should be made up of 1 or more batches. However, there is a charge for each batch in a submission therefore the number of batches in a submissions should be minimised
3	The **absolute** value of the transactions being submitted in each batch must not exceed £1,000,000. Credits do not offset debits. For example, if you have a debit and a credit transaction for £10 the total value in the batch is £20, not £0

4	Each time a submission occurs you should print to the console the details of the batches. For example:
	
	`{`
	`        submission: {`
	`                batches: [`
	`                        {`
	`                                totalValueOfAllTransactions: 75000`
	`                                countOfTransactions: 18`
	`                        },`
	`                         {`
	`                                totalValueOfAllTransactions: 98000`
	`                                countOfTransactions: 12`
	`                        },`
	`                        ...`
	`                ]`
	`        }`
	`}`

5	You should consider the performance of your implementation and demonstrate how the implementation would scale should the number of transactions in a submission be increased (i.e. submissions triggered every 100,000 transactions)