Backend Engineer Assignment

Objective:

This assignment assesses your skills in designing and implementing key functionalities of a payment service platform. We expect high-quality, maintainable, and scalable code from a senior backend engineer.

Task Overview:

Develop a simple PSP system that handles payment requests and simulates interactions with two payment acquirers based on the BIN (Bank Identification Number) card number routing rules.

Requirements:

1. API for Payment Processing:

- Endpoint Development: Create an API endpoint to accept payment details: card number, expiry date, CVV, amount, currency, and merchant ID.
- Validation: Validate the card number using Luhn's algorithm. Ensure all fields are correctly formatted and complete.
- Transaction Status: Initialize transaction status as 'Pending' in your data storage.

2. Acquirer Routing:

- BIN Routing Logic: Implement a routing mechanism based on the BIN card number. Calculate the sum of the digits in the BIN:
    - If the sum is even, route the transaction to Acquirer A.
    - If the sum is odd, route the transaction to Acquirer B.

3. Mock Acquirer Communication:

- **Simulate Transaction Processing:** Develop a mock function to simulate sending the transaction to the appropriate acquirer:
    - Acquirer A: Decides to approve (even last digit of card number) or deny (odd last digit) the transaction.
    - Acquirer B: Uses the same logic as Acquirer A.
- Update Status: Based on the mock acquirer's decision, update the transaction status to 'Approved' or 'Denied'.
- Response to Merchant: Return the final transaction status to the merchant, along with the transaction ID and any relevant messages.

4. Mock Data Storage:

- In-memory Storage: Use a simple in-memory data structure (e.g., a map or dictionary) to store transaction records, including transaction ID, card details, status, and merchant ID.

5. Documentation and Code Quality:

- Documentation: Include a README for setting up and running your solution. Describe your design choices.
- Code Quality: Your code should be clean, well-documented, and use appropriate design patterns. Include basic error handling to manage common input and processing errors.

Bonus Task (Optional):

- Containerization: Use Docker to containerize your application. Include the Dockerfile in your submission.

Delivery:

- Repository: Establish a new repository on GitHub or GitLab. Commit your code, any scripts, and documentation.
- Public Access: Set the repository to public access and share the link upon completion.
- Please send the link to oravraham@exirom.com.

Time Frame:

- Duration: You have 2 days from receipt of this assignment to submit your work.

FAQ:

- Languages: You may use Java, Scala, or Kotlin. Choose whichever you are most comfortable with.
- Data Storage: Use in-memory data structures to simulate database interactions; no need to set up a real database.
- Encryption Details: While not required for this task, briefly discuss what encryption methods you would recommend for protecting sensitive data.