RapidPay - Code Challenge

Welcome Candidate!

Meet RapidPay, a fast-growing payment provider. We need YOU to develop our new Authorization System, and we're rewarding success!

The project consists of three main modules:

✅ Card Management Module (50K points)
✅ Card Authorization Module (20K points)
✅ Payment Fees Module (30K points)

For each module you complete, you earn more points. Total possible points: 100K.

💰 Bonus available! Earn up to 30K extra points for additional improvements.

⏳ Time for solution: 8 Hours

🚀 Good luck! We hope you earn a lot of points!

Task Overview

Develop a RESTful API in C# using .NET Core with:
- Database storage (mandatory) – Use an SQL database to store cards and transactions.
- Secure authentication & authorization – Improve security beyond basic authentication.
- Scalability & performance optimizations.
- Thread safety considerations for concurrent access.
- Unit tests and documentation to validate your solution.

1. API Modules & Requirements

1. Card Management Module (50K points)

Develop the following five API endpoints:

📌 Requirements:

1. Create Card
- A new card must have a 15-digit format.
- A card should start with a random balance and an optional credit limit.
- Store card details securely in the database.

2. Authorize Card
- Validate whether a card is active and eligible for transactions.
- Maintain an authorization status in the database.
- Implement basic fraud checks, such as duplicate transactions within seconds.

3. Pay with Card
- Process a payment using an authorized card.
- Apply payment fees (explained in the next module).
- Update the card balance accordingly.

4. Get Card Balance
- Retrieve the current balance of a card.
- Include available credit limit if applicable.

5. Update Card Details
- Allow updating the card's balance, credit limit, or status.
- Ensure that changes are logged for tracking purposes.

2. Card Authorization Module (20K points)

This module ensures that only valid and authorized cards can process payments.

📌 Requirements:
- Implement an authorization mechanism that checks:
- Card validity & status (e.g., active/inactive, fraud detection).
- Sufficient balance before approving payments.
- Store authorization logs in the database.

3. Payment Fees Module (30K points)

Each payment incurs a transaction fee, which fluctuates dynamically based on a random multiplier.

📌 Requirements:
- Every hour, the Universal Fees Exchange (UFE) selects a random decimal between 0 and 2.
- The new fee price is the last fee amount multiplied by the recent random decimal.
- The fee should be applied to every payment.

💡 Implementation Hint:
- Develop a Singleton service to simulate the UFE service updating the fee hourly.
- Store the latest fee history in the database for accurate calculations.

Bonus Challenges (30K extra points)

✓ Performance & Scalability (10K points)
- Optimize API performance using multithreading to increase throughput.

✓ Security (10K points)
- Basic authentication is not ideal. Implement a more secure authentication method (e.g., JWT, OAuth).

✓ Thread Safety & Data Management (10K points)

- Ensure thread safety when handling concurrent transactions.
- Optimize database schema and ORM usage for high efficiency.

Expectations & Evaluation Criteria

We are looking for quality over speed! Your solution will be evaluated based on:

✅ Correctness – Does the API function as expected?
✅ Database Design – Is the data structure efficient and scalable?
✅ Code Quality – Is the code clean, structured, and readable?
✅ Unit Tests – Are there sufficient tests covering different cases?
✅ Scalability – Can the system handle growth and concurrent users?
✅ Security – Have best practices been applied?
✅ Documentation – Is the solution well-documented for maintainability?

Submission Guidelines

1. Upload your solution to a GitHub repo, Google Drive, or any downloadable link.
2. Include a README file with:
   - Setup instructions
   - API documentation (endpoints, request/response examples)
   - Explanation of architectural choices
   - How you implemented thread safety and security improvements (if applicable).
3. Submit the link.

🚀 We look forward to your solution! Happy coding! 🚀