Transaction Insight System

Project Objective

Project helps you express and showcase your ability to design and implement a scalable, maintainable, and well-structured solution to a real-world engineering problem. It assists in assessing your software engineering skills, problem solving skills, coding standards, system design, and ability to articulate your thought process.

Problem Statement

Design and implement a Transaction Insight System as a Web Application that enables financial organizations to efficiently query, filter, and analyze large volumes of financial transactions in real time with minimal latency. Financial transaction is a transaction that captures source and destination of monies along with supporting details.

Transactions are analysed, Transactions are probed and view more details conveniently, enabling them to make quick decisions

System should allow users to

- Query and list hundreds to thousands of financial transactions efficiently. The results can also include basic insights on qualified financial transactions like # of transactions, average # of transactions each month, # of transactions failed etc. (*You are free to include more relevant insights!)*
- Define and save “Transaction Selection Sets” for reusable transaction queries.
- Use pre-saved “Transaction Selection Sets” to quickly retrieve transactions.

Technical Requirements

- **Backend:** Use Java, C#, Node.js, Python or PHP (Laravel preferred)
- **Mock UI:** You may provide even pencil sketches, do not overspend time on this
- **Database:** Use a relational database (**PostgreSQL, MySQL, or DB2**) and design an optimized schema for fast query performance.
- **API Design:** Implement GraphQL or RESTful APIs, following best practices for API semantics (API-first approach preferred).
- **Authentication** (Optional): If implemented, use Auth0 JWT based authentication

Submission Guidelines

- **Timeline:** Complete the project within 2 to 3 days.
- **Code Submission:** Use a Git repository from the beginning and share it as part of submission
- **README.md:** Include setup instructions, design decisions, API documentation, and any assumptions or trade-offs made.

Evaluation Criteria

Attribute	Description
Code Quality & Best Practices	Clean, readable, modular, maintainable code, with best practices followed. Editor Settings can be included in the repository. VS Code preferred
System Design	Well-structured codebase and system design for future scalability.
API Design & Documentation	Well-defined GraphQL or RESTful APIs with clear endpoints and thorough documentation.
Database Design & Efficiency	Optimized schema design with indexed queries to ensure efficient transaction retrieval.
Security Considerations	Proper authentication, authorization, and input validation to safeguard data integrity.
Overall Feature Implementation & Usability	Core functionalities work as expected
Bonus (Extra Credit)	CI/CD, performance optimizations, monitoring or additional enhancements.