DOCUMENTATION

PYTHON DEVELOPMENT

WEEK-2

PROJECT NAME: Real-Time FinTech Fraud Detection System (SentinelStream)

Week 2 – Core Transaction Pipeline Implementation
Organization -Zaalima Development Pvt. Ltd.

CREATED BY - PITCHE ESHWAR

1. Week Overview:

Week 2 focuses on building the core transaction processing pipeline, which is the backbone of the SentinelStream fraud detection engine. The goal of this phase is to implement a high-performance, scalable transaction endpoint capable of handling concurrent requests while minimizing database load.
This week transitions the project from design to implementation, emphasizing speed, reliability, and scalability.

2. Goal & Focus:

Core Transaction Pipeline Development
The primary objective of Week 2 is to:

- Develop a high-speed transaction ingestion API
- Integrate Redis caching to reduce database latency
- Establish asynchronous database communication
- Validate system performance using load testing

3. Key Features Implemented:

3.1 High-Speed Transaction Endpoint

- Implemented a POST /transaction API endpoint
- Designed to process incoming financial transactions with:
    - Low latency
    - High concurrency
    - Strict input validation

The endpoint acts as the entry point for all transaction data flowing into SentinelStream.

3.2 Redis Caching for User Profiles:

To minimize repeated database access:

- Integrated Redis caching for frequently accessed user profile data
- Cached data includes:
    - User identifiers
    - Transaction limits
    - Risk-related metadata

Benefits:

- Reduced database hits
- Faster response times
- Improved system scalability

3.3 Asynchronous Database Integration:

- Connected to PostgreSQL using SQLAlchemy AsyncIO
- Ensured non-blocking database operations
- Enabled the system to handle multiple concurrent transactions efficiently

This approach supports high throughput without compromising stability.

4. Performance & Load Testing:

4.1 Load Testing Tool:

- Used Locust for performance validation
- Simulated concurrent users sending transaction requests

4.2 Performance Benchmark:

- Successfully demonstrated the ability to sustain:
  - Minimum of 100 requests per second
  - On local infrastructure

This validates the core pipeline's readiness for further scaling and production-level enhancements.

5. Technology Stack Used (Week 2):

- FastAPI – High-performance backend framework
- Redis – In-memory caching layer
- PostgreSQL – Persistent transaction storage
- SQLAlchemy AsyncIO – Asynchronous database ORM
- Locust – Load testing and performance benchmarking

6. Deliverables:

- Fully functional POST /transaction API
- Redis-integrated caching mechanism
- Async PostgreSQL database connectivity
- Load testing report validating throughput goals

7. Challenges & Mitigation:

Challenges:

- Managing concurrent transaction requests
- Avoiding database bottlenecks
- Ensuring stable performance under load

Mitigation Strategies:

- Redis caching to offload database queries
- Asynchronous database operations
- Incremental load testing to identify bottlenecks early

8. Learning Outcomes:

By the end of Week 2:

- Gained hands-on experience with asynchronous backend development
- Understood real-world performance optimization techniques
- Implemented and tested a scalable transaction pipeline
- Learned practical load testing using Locust

9. Conclusion:

Week 2 successfully established the core transaction processing pipeline of SentinelStream. With a high-speed API, optimized caching, and validated performance benchmarks, the system is now prepared for advanced fraud detection layers, including rule-based evaluation and machine learning integration in upcoming weeks.