Financial Backend Microservices Architecture Assessment
Date: February 19, 2026
Project Path: C:\Users\super\Desktop\project3

Executive Summary
The project is partially production-ready, but it is not intact yet for a payment-grade microservice architecture.
The main blockers are financial consistency under concurrency, secret management, and release/infrastructure consistency.

Review Scope
- Account service, transaction service, shared deployment assets
- Security model, DB migrations, transactional behavior
- Docker and docker-compose production paths
- CI workflows, Helm, and Terraform
- Build and test execution validation

Validation Run Results
1. account-service compile: PASSED
2. transaction-service compile: PASSED
3. account-service tests (-DskipITs): PASSED
4. transaction-service tests (full suite): PASSED
5. CI-referenced transaction hardening target: FAILED (missing class)

Critical Findings
1. Idempotency is not atomic for payment operations.
   - Evidence:
     - transaction-service/src/main/java/com/suhasan/finance/transaction_service/service/TransactionServiceImpl.java
     - transaction-service/src/main/java/com/suhasan/finance/transaction_service/service/TransactionServiceImpl.java
     - transaction-service/src/main/java/com/suhasan/finance/transaction_service/repository/TransactionRepository.ja
     - transaction-service/src/main/resources/db/migration/V6__add_idempotency_and_processing_state.sql:67
   - Impact:
     - Concurrent duplicates can both pass check-then-insert and double-apply funds.
   - Recommendation:
     - Enforce DB unique constraint for idempotency scope and treat duplicate-key as idempotent replay.

2. Reversal flow can race and allow duplicate reversals.
   - Evidence:
     - transaction-service/src/main/java/com/suhasan/finance/transaction_service/service/TransactionServiceImpl.java
     - transaction-service/src/main/java/com/suhasan/finance/transaction_service/service/TransactionServiceImpl.java
     - transaction-service/src/main/java/com/suhasan/finance/transaction_service/repository/TransactionRepository.ja

- Impact:
  - Multiple reversal transactions may be created for the same original transaction.
- Recommendation:
  - Add DB uniqueness on original_transaction_id for reversal type and lock original row in reversal flow.

3. Secret handling is unsafe in active runtime configs.
   - Evidence:
     - docker-compose.yml:90
     - docker-compose.yml:91
     - docker-compose.yml:130
     - docker-compose.yml:131
     - transaction-service/docker-compose-monitoring.yml:18
   - Impact:
     - Default/fallback secrets and committed secret values violate payment-system security requirements.
   - Recommendation:
     - Remove fallbacks, require injected secrets, and rotate exposed values immediately.

High Findings

4. CI gate references a missing test and fails.
   - Evidence:
     - .github/workflows/ci-cd-pipeline.yml:94
   - Impact:
     - Pipeline quality gate is unreliable.
   - Recommendation:
     - Update workflow target to an existing test class or add missing class.

5. CI policy scan conflicts with committed config.
   - Evidence:
     - .github/workflows/ci-cd-pipeline.yml:41
     - transaction-service/docker-compose-monitoring.yml:18
   - Impact:
     - Security policy consistency is broken.
   - Recommendation:
     - Remove hardcoded secret and align scanning scope with active deployment files.

6. Transaction service Docker healthcheck uses wrong port.
   - Evidence:
     - transaction-service/Dockerfile:47 (checks 9002)
     - transaction-service/src/main/resources/application.properties:3 (service on 8081)
   - Impact:
     - Container may appear unhealthy despite healthy app.
   - Recommendation:
     - Point healthcheck to actual management/app port.

7. Transaction search is not scalable.
   - Evidence:
     -

    transaction-service/src/main/java/com/suhasan/finance/transaction_service/service/TransactionServiceImpl.java
  - Impact:
    - In-memory filtering/sorting of large data can cause latency and memory pressure.
  - Recommendation:
    - Push filtering, sorting, and paging into repository/database queries.

8. Monitoring auth model is inconsistent with scraping model.
  - Evidence:
    -
      account-service/src/main/java/com/suhasan/finance/account_service/security/SecurityConfig.java:47
    -
      transaction-service/src/main/java/com/suhasan/finance/transaction_service/security/SecurityConfig.java:35
    - infrastructure/helm/account-service/templates/servicemonitor.yaml:20
  - Impact:
    - Scrapes may fail, or endpoints may be weakened unsafely.
  - Recommendation:
    - Implement explicit scrape auth strategy (service account, mTLS, or tightly scoped
      network controls).

9. Transaction monitoring API is too broadly accessible.
  - Evidence:
    -
      transaction-service/src/main/java/com/suhasan/finance/transaction_service/security/SecurityConfig.java:40
  - Impact:
    - Internal operational metrics can be exposed to regular authenticated users.
  - Recommendation:
    - Restrict monitoring endpoints to admin/internal roles.

10. Transaction Prometheus config has structural issues.
  - Evidence:
    - transaction-service/src/main/resources/prometheus.yml:2
    - transaction-service/src/main/resources/prometheus.yml:121
    - transaction-service/src/main/resources/prometheus.yml:183
  - Impact:
    - Prometheus config may parse incorrectly or behave unexpectedly.
  - Recommendation:
    - Normalize to valid Prometheus schema (single global block, proper rule_files use).

11. Monitoring aspect targets legacy client, not active resilient client.
  - Evidence:
    -
      transaction-service/src/main/java/com/suhasan/finance/transaction_service/aspect/MonitoringAspect.java:33
    -
      transaction-service/src/main/java/com/suhasan/finance/transaction_service/service/TransactionServiceImpl.java
  - Impact:
    - Missing metrics and alerts for actual account-service calls.
  - Recommendation:
    - Update pointcuts to include ResilientAccountServiceClient.

12. Terraform prod backend/state and network policy labeling are weak.
  - Evidence:
    - infrastructure/terraform/environments/prod/main.tf:18 (local backend)
    - infrastructure/terraform/modules/kubernetes/main.tf:118 (namespace selector expects label ’name’)
    - infrastructure/terraform/modules/kubernetes/main.tf:27 (namespace labels do not include ’name’)
  - Impact:
    - Risky state management and potentially ineffective policy behavior.
  - Recommendation:
    - Use remote backend for prod and align namespace labels/selectors.

13. Infrastructure is asymmetric across services.
  - Evidence:
    - infrastructure/helm/account-service/Chart.yaml:1 exists
    - infrastructure/helm/transaction-service not present
  - Impact:
    - Incomplete production IaC path for full microservice architecture.
  - Recommendation:
    - Add transaction-service Helm/Terraform parity for deploy, policy, and monitoring.

Medium Findings

14. Public health route includes mutable deployment endpoints.
  - Evidence:
    - account-service/src/main/java/com/suhasan/finance/account_service/security/SecurityConfig.java:45
    - account-service/src/main/java/com/suhasan/finance/account_service/controller/HealthController.java:94
    - account-service/src/main/java/com/suhasan/finance/account_service/controller/HealthController.java:122
  - Impact:
    - Unnecessary exposed operational surface.
  - Recommendation:
    - Split public liveness from privileged operations.

15. Health checks include placeholders that always report healthy.
  - Evidence:
    - account-service/src/main/java/com/suhasan/finance/account_service/service/DeploymentTrackingService.java:2
    - account-service/src/main/java/com/suhasan/finance/account_service/service/DeploymentTrackingService.java:2
    - account-service/src/main/java/com/suhasan/finance/account_service/service/DeploymentTrackingService.java:2
  - Impact:
    - False-positive health and delayed incident detection.
  - Recommendation:
    - Implement real dependency checks.

16. Top-level production compose still exposes DB ports publicly.

  - Evidence:
    - docker-compose.yml:33
    - docker-compose.yml:56
  - Impact:
    - Larger attack surface.
  - Recommendation:
    - Keep databases internal unless explicit external access is required.

What Is Intact / Strong

1. Clear service boundaries between account and transaction domains.
2. Flyway migrations and schema versioning are implemented.
3. Account balance operations use idempotent operation table and row-level locking.
4. Both services compile and core tests pass locally.
5. Container non-root runtime is configured.

Overall Production Readiness

Current state: NOT READY for payment-grade production.
Primary blockers: atomic transactional guarantees, secret hygiene, and CI/IaC consistency.

Prioritized Remediation Plan

1. Transactional correctness:
   - Atomic idempotency and reversal uniqueness/locking.
2. Security hardening:
   - Remove fallback/committed secrets and rotate all exposed credentials.
3. Pipeline integrity:
   - Fix CI test references and enforce consistent policy checks.
4. Runtime reliability:
   - Correct healthcheck ports and reconcile monitoring auth/scraping.
5. Architecture completeness:
   - Add transaction-service IaC parity (Helm/Terraform).

Conclusion

The platform has a strong foundation, but payment-grade resilience and security guarantees are incomplete.
Addressing the critical items above should be treated as mandatory before production go-live.