ï»¿Ankit Raj - Software Engineer Intern Assignment
Date: 2026-02-10

GitHub Repository: [INSERT GITHUB LINK HERE]

==
Assignment 1: Durable Execution Engine (Java)
Location: C:\Users\91637\Desktop\coding\oops

Overview
This project implements a minimal Durable Execution Engine that makes normal Java code durable by wrapping

Structure
- engine/: core durable engine library
- examples/onboarding/: Employee Onboarding example workflow
- main/: CLI app to start/resume the workflow and simulate crashes

Build
```
mvn -q -DskipTests package
```

Run
```
java -jar main/target/app-1.0.0.jar --workflow-id employee-1 --db durable.db
java -jar main/target/app-1.0.0.jar --workflow-id employee-1 --db durable.db --crash-at provision-parallel
```

Design Notes
- Sequence Tracking: Each step(id, ...) uses a monotonically increasing sequence number to support loops ar
- Thread Safety: SQLite writes are guarded by a write lock; WAL mode + busy timeout; each operation uses a
- Zombie Step Handling: If a crash happens between execution and commit, policy is RETRY (default) or FAIL

Prompts Used (Assignment 1)
1) "read the pdf"
2) "install pypdf2"
3) "generate all code file in java and readme pdf"

==
Assignment 2: High-Throughput Fan-Out Engine (Java)
Location: C:\Users\91637\Desktop\coding\fanout-engine

Overview
Streaming fan-out engine that ingests large files and distributes records to multiple sink types (REST, gRP

Architecture Summary
InputReader -> Source Queue -> Dispatcher -> Per-sink Queues -> Sink Workers
Each sink applies a Strategy Transformer (JSON / Protobuf / XML / Avro+CQL map).

Backpressure & Concurrency
- Bounded queues prevent OOM and propagate backpressure.
- Java 21 virtual threads for sink workers.

Resilience
- Per-sink retries (max 3).
- Failures written to dlq/<sink>.jsonl.

Observability
- Every 5 seconds prints ingested, dispatched, throughput, and per-sink success/failure/retry counts.

Architecture Diagram
```
+-------------------+        +-------------------+        +-------------------+
|   InputReader     | -----> |   Source Queue    | -----> |    Dispatcher     |
| (CSV/JSONL/FIXED) |        | (bounded, backpr.)|        | (fan-out)         |
+-------------------+        +-------------------+        +---------+---------+
                                                                      |
                                                                      v
      +---------------------+   +---------------------+   +---------------------+
      |  Sink Queue (REST)  |   |  Sink Queue (gRPC)  |   |   Sink Queue (MQ)   |
      |  workers + limiter  |   |  workers + limiter  |   |  workers + limiter  |
      +----------+----------+   +----------+----------+   +----------+----------+
                 |                         |                         |
                 v                         v                         v
           REST Mock Sink            gRPC Mock Sink            MQ Mock Sink

                                    +---------------------+
                                    | Sink Queue (WIDE_DB)|
                                    | workers + limiter   |
                                    +----------+----------+
                                               |
                                               v
                                        Wide DB Mock Sink
```

Prompts Used (Assignment 2)
- "Generate a Java 21 fan-out engine design using Strategy + Factory."
- "Implement streaming ingestion for CSV/JSONL with backpressure and bounded queues."
- "Provide a minimal Maven project with tests and README."

Suggested Prompts (Not Necessarily Used)
Assignment 1
1. "Summarize the durable execution engine requirements and key deliverables."
2. "Draft a minimal Java API for step(id, Callable<T>) with SQLite persistence."
3. "Explain how to handle sequence IDs for loops and conditionals."
4. "How to handle zombie steps (crash between execute and commit)? Provide policy options."
5. "Review my README for clarity and completeness."

Assignment 2
1. "Design a Java 21 fan-out engine with bounded queues, backpressure, and retries."
2. "Implement streaming CSV/JSONL readers without loading whole file."
3. "How to add per-sink rate limiting with retries?"
4. "List observability metrics to log every 5 seconds."
5. "Provide a simple architecture diagram text."