Farheen Shaikh - 989500432 (Lab 3)

Architecture & Data Flow – IndexedDB + Neo4j → MongoDB Lake

[IndexedDB] → [Node.js server API]

↓

[Neo4j Aura (Graph DB)]

↓

[Integration Script → ingestNeo4jToMongo.js]

↓

[MongoDB Atlas – Data Lake]

Flow explanation (right/bullets):

- IndexedDB stores browser-side readings (JSON objects).
- Neo4j represents farm network graph: Farm → Device → Reading.
- readAgriGraph.js queries Neo4j using Cypher.
- ingestNeo4jToMongo.js merges both data sources.
- Final output stored in MongoDB Lake with metadata (sourceDB, ingestedAt, tags).

Extending Lab 2 Pipeline – From IndexedDB Only → Hybrid Graph Integration

- **What was added**
  - Neo4j connection layer (neo4j-driver)
  - Cypher query to fetch (Farm-Device-Reading) triples
  - Merge logic to insert Neo4j data into MongoDB Lake
  - Metadata fields:
    - sourceDB: IndexedDB | Neo4j
    - ingestedAt: UTC timestamp
    - tags: custom keywords
- **Data validation**
  - Verified count of documents per source DB in MongoDB Compass.
  - Used db.lake.find({sourceDB:"Neo4j"}) to confirm integration.

Visual suggestion: small code snippet block showing .table().

Lessons Learned from Neo4j + MongoDB Integration

What Worked Well

- Successful connection to Neo4j Aura via secure URI.
- Easy integration with MongoDB Driver (JSON compatibility).
- Clear graph queries for agriculture relationships.
- Console logging and console.table() helped debug outputs.

Challenges / What Did Not Work Well

- Neo4j connection errors when .env was missing.
- Formatting Integer types from Neo4j → JavaScript.
- Sync issues when multiple sessions ran simultaneously.
- Needed manual testing before pushing data to MongoDB.

Key Takeaway

"Integrating a graph database with a document lake requires clean schema mapping and consistent metadata design."