Backend Assignment – Database Synchronization System (Express.js)

This assignment evaluates your ability to design and implement a real-world backend system using Node.js, Express.js, MySQL, and cron jobs. The focus is on database consistency, synchronization logic, and reliability when working with offline and online systems.

Experience Level

6–12 months experience in backend development with Node.js / Express.js

Problem Statement

Build a Data Synchronization Service that syncs data between two similar MySQL databases. One database represents a local/offline system (DB_A), while the other represents a central/online system (DB_B). Whenever the system is online, pending data changes from DB_A must be synchronized to DB_B using a cron job.

Technology Requirements

1. Node.js
2. Express.js
3. MySQL (no ORM; raw SQL queries only)
4. node-cron (or equivalent)
5. dotenv for environment configuration

Database Structure

DB_A (Local / Offline Database)

1. users (id, name, email, updated_at)
2. orders (id, user_id, amount, status, updated_at)
3. sync_queue (id, table_name, record_id, operation, last_updated_at, sync_status, retry_count)

DB_B (Central / Online Database)

1. users (id, name, email, updated_at)
2. orders (id, user_id, amount, status, updated_at)

Sync Queue Rules

1. Every INSERT, UPDATE, or DELETE in DB_A must create a record in sync_queue.

2. sync_status must be set to PENDING initially.
3. retry_count must increase on sync failure.

Synchronization Logic

1. Only sync records where sync_status = PENDING.
2. Sync should occur only when the system is online.
3. Support INSERT, UPDATE, and DELETE operations.
4. Use MySQL transactions to ensure consistency.
5. After successful sync, mark records as SYNCED.

Network Availability Simulation

Network availability should be simulated using an environment variable:

IS_ONLINE=true | false

Cron Job Requirements

1. Cron job should run every 2 minutes.
2. It must check network availability before syncing.
3. The job must safely handle retries and partial failures.

API Requirements (DB_A)

1. POST /users – Create a user and log sync operation.
2. PUT /users/:id – Update a user and log sync operation.
3. POST /orders – Create an order and log sync operation.
4. PUT /orders/:id – Update an order and log sync operation.
5. GET /sync/status – View pending, synced, and failed records.

Project Structure

1. src/db (localDb.js, centralDb.js)
2. src/routes
3. src/controllers
4. src/services
5. src/cron/syncCron.js
6. src/utils

Submission Guidelines

1. Provide a Git repository or ZIP archive.
2. Include SQL schema and sample data.
3. Include a README with setup and run instructions.