■ Mobile Technical Challenge

Automated Transport Cycle Simulator in Mining

■ Problem Context

In large-scale mining, equipment such as trucks, excavators, drills, and loaders work together to extract, transport, and unload ore. This challenge focuses on automating the complete cycle of a mining haul truck without manual operator intervention.

The app you will build aims to automatically interpret embedded sensor data, identify the current stage of the truck's cycle, and display this information to the operator. Each cycle should be recorded, stored locally, and later synchronized with a simulated "server" by exporting to a log file.

■ Objective

Create a mobile app that:

- Loads simulated sensor readings line by line from a `.jsonl` file
- Interprets the readings to determine the current transport cycle stage
- Updates the UI with processed data
- Stores complete cycles locally
- Simulates data transmission by writing to a `.jsonl` output file
- Avoids duplication of already synchronized data
- Works fully offline

■ Business Rules – Cycle Stages

1. EM FILA CARREGAMENTO (IN LOADING QUEUE)
- Speed = 0 for more than 5 seconds
- AND detects truck in queue or loading at the same excavator

2. EM CARREGAMENTO (LOADING)
- Speed = 0 for more than 5 seconds
- Detects excavator less than 2m away
- AND no other truck is loading at the same excavator

3. TRÂNSITO CHEIO (FULL TRANSIT)
- Speed > 0
- Excavator more than 2m away
- Current state is EM CARREGAMENTO

4. EM FILA BASCULAMENTO (IN DUMPING QUEUE)
- Speed = 0 for more than 5 seconds
- GPS equal or close to dumping point
- Tipper sensor OFF
- AND detects truck in queue or current state is TRÂNSITO CHEIO

5. EM BASCULAMENTO (DUMPING)
- Speed = 0
- GPS at dumping point
- Tipper sensor ON

6. TRÂNSITO VAZIO (EMPTY TRANSIT)
- Speed > 0
- GPS 5m or more away from dumping point
- Current state is EM BASCULAMENTO

■ Sensors Used
Sensor data will be provided in the simulation as:
- Beacons: array of nearby equipment (type: excavator, truck, or tipper_sensor)
- GPS: current location and speed (in m/s)

The tipper sensor will appear in the beacons array. Example: installed in truck ID CAM-001

■ Input File Structure (simulacao.jsonl)
Format: `.jsonl` (JSON Lines). Each line is a sensor reading at a specific moment in time.

■ Expected Interface
With each simulated reading (via SIMULATE button), the app should display:
- CURRENT STAGE: status as determined by logic
- LOADING EQUIPMENT: as per reading
- DUMPING POINT: as per reading
- CURRENT SPEED: converted from m/s to km/h
- SYNC STATUS: shows if there are pending or synced data

■ Offline → Online Synchronization
- The app should store complete cycles (up to TRÂNSITO VAZIO) locally
- When the app detects “network available” (simulated), it should export unsynced cycles to:

`sync_servidor.jsonl` (1 line per complete cycle)

■ Expected Deliverables
The candidate must deliver:

1. ■ Link to GitHub repository
2. ■ Full app source code
3. ■ APK for installation (link in README)
4. ■ README.md file with:

- Installation and usage instructions
- Explanation of architecture and technical decisions (optional, but a plus)

- Location of `sync_servidor.jsonl` output file
- Technical challenges or incomplete implementations and reasoning