3. Backend Developer

The Scenario: To manage table availability effectively, a system is required to temporarily "lock" a table while a user completes their booking. This mechanism is crucial for preventing simultaneous double-bookings of the same table.

Your Task: Develop the core API logic for this functionality by creating a "Table Reservation Lock" API.

Specific Deliverable: Construct a set of REST API endpoints designed to facilitate the temporary reservation of a table.

Requirements & Tech Stack:

- **Tech Stack:** Node.js with Express.js.
- **Database:** Utilize an in-memory solution (e.g., a simple JavaScript object or array) for storing lock records. No external database setup is necessary.
- **API Endpoints:**
  1. **POST /api/tables/lock**
     - **Request Body:** `{ "tableId": "table-123", "userId": "user-abc", "duration": 600 }` (duration to be specified in seconds).
     - **Logic:**
       - Verify if the `tableId` is already subject to a lock.
       - If no lock exists, create a new lock record containing the `tableId`, `userId`, and an `expiry timestamp` (calculated as `currentTime + duration`).
       - Respond with a `200 OK` status and the body `{ "success": true, "message": "Table locked successfully." }`.
       - If the table is already locked, return a `409 Conflict` status with the body `{ "success": false, "message": "Table is currently locked by another user." }`.
  2. **POST /api/tables/unlock**
     - **Request Body:** `{ "tableId": "table-123", "userId": "user-abc" }`
     - **Logic:**
       - Remove the lock associated with the provided `tableId` only if the `userId` in the request matches the `userId` that established the original lock.
       - Respond with a `200 OK` status.
  3. **GET /api/tables/:tableId/status**
     - **Logic:**

        - Determine if a lock exists for the specified `tableId` and ascertain if the lock has expired.
        - Respond with a `200 OK` status and the body `{ "isLocked": true/false }`.

Deliverables:

- A publicly accessible GitHub repository link containing your Node.js/Express project.
- A Postman collection (or equivalent documentation) detailing and demonstrating the testing procedures for each endpoint.

Assessment Criteria:

- **API Design:** Clarity, logical structure, and adherence to RESTful principles in endpoint design.
- **Business Logic:** Accuracy in implementing the locking and unlocking mechanisms, including comprehensive handling of edge cases.
- **Code Quality:** Organization, readability, and maintainability of the server-side code.