Scenario

Our bank's IT team needs a simple **Customer Management Service** to maintain customer records. This is an internal API that other systems (like loan processing and account management) will call. As the backend developer, you are assigned to build a **CRUD API** for customer details using **Spring Boot, Spring Data JPA, and an H2 in-memory database**.

Your deliverable: A set of REST endpoints that support **Create, Read, Update, and Delete operations** for customers.

This lab will walk you through the process step by step, just like you would in a real engineering assignment.

Requirements

1. **Create a `Customer` entity** in the `com.bank.customer` package.
   It should have the following fields:
   - `id` (Long, primary key, auto-generated)
   - `firstName` (String)
   - `lastName` (String)
   - `email` (String, unique)
   - `phoneNumber` (String)

2. **Create a Spring Data JPA repository** named `CustomerRepository`.
   It should extend `JpaRepository<Customer, Long>` *to provide CRUD operations.*

3. **Implement a `CustomerService` class** in the service layer.
   It should provide methods to:
   - Create a customer (`createCustomer`)
   - Retrieve all customers (`getAllCustomers`)
   - Retrieve a customer by ID (`getCustomerById`)
   - Update a customer (`updateCustomer`)
   - Delete a customer (`deleteCustomer`)

4. **Create a REST controller** named `CustomerController` under `/api/customers`.
   Add the following endpoints:

- **POST** `/api/customers`
  Accepts a JSON request to create a new customer and returns the created customer.

- **GET** `/api/customers`
  Returns a list of all customers.

- **GET** `/api/customers/{id}`
  Returns a single customer by ID.

- **PUT** `/api/customers/{id}`
  Updates an existing customer's details and returns the updated customer.

- **DELETE** `/api/customers/{id}`
  Deletes a customer and returns a confirmation message.