**Programming Assessment**

**### Exercise 1: Customer Relationship Management (CRM) System**

**Database Schema Design**

CREATE TABLE customers (

customer\_id INT PRIMARY KEY AUTO\_INCREMENT,

customer\_name VARCHAR(100) NOT NULL,

customer\_email VARCHAR(100) UNIQUE,

customer\_phone VARCHAR(15),

customer\_address TEXT

);

CREATE TABLE contacts (

contact\_id INT PRIMARY KEY AUTO\_INCREMENT,

customer\_id INT,

contact\_name VARCHAR(100) NOT NULL,

contact\_email VARCHAR(100),

contact\_phone VARCHAR(15),

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

CREATE TABLE opportunities ( opportunity\_id INT PRIMARY KEY AUTO\_INCREMENT,

customer\_id INT,

opportunity\_name VARCHAR(100) NOT NULL,

stage VARCHAR(50), amount DECIMAL(10, 2),

close\_date DATE,

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

CREATE TABLE interactions (

interaction\_id INT PRIMARY KEY AUTO\_INCREMENT,

customer\_id INT,

interaction\_type VARCHAR(50),

interaction\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

notes TEXT,

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

**Backend: RESTful API in Java (Spring Boot)**

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

</dependencies>

**Application Properties (application.properties)**

spring.datasource.url=jdbc:mysql://localhost:3306/crm

spring.datasource.username=root

spring.datasource.password=password

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

**Customer Entity**

public class Customer {

private Long customerId;

private String customerName;

private String customerEmail;

private String customerPhone;

private String customerAddress;

}

**Customer Repository**

public interface CustomerRepository extends JpaRepository<Customer, Long> { }

**Customer Controller**

public class CustomerController {

private CustomerRepository customerRepository;

public List<Customer> getAllCustomers() {

return customerRepository.findAll();

}

public Customer createCustomer(@RequestBody Customer customer) {

return customerRepository.save(customer);

}

public ResponseEntity<Customer> getCustomerById(@PathVariable Long id) {

Customer customer = customerRepository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Customer not found"));

return ResponseEntity.ok(customer);

}

public ResponseEntity<Customer> updateCustomer(@PathVariable Long id, @RequestBody Customer customerDetails) {

Customer customer = customerRepository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Customer not found"));

customer.setCustomerName(customerDetails.getCustomerName());

customer.setCustomerEmail(customerDetails.getCustomerEmail());

customer.setCustomerPhone(customerDetails.getCustomerPhone());

customer.setCustomerAddress(customerDetails.getCustomerAddress());

Customer updatedCustomer = customerRepository.save(customer);

return ResponseEntity.ok(updatedCustomer);

}

public ResponseEntity<Map<String, Boolean>> deleteCustomer(@PathVariable Long id) {

Customer customer = customerRepository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Customer not found"));

customerRepository.delete(customer);

Map<String, Boolean> response = new HashMap<>();

response.put("deleted", Boolean.TRUE);

return ResponseEntity.ok(response);

}

}

**Interaction Entity**

public class Interaction {

private Long interactionId;

private Long customerId;

private String interactionType;

private LocalDateTime interactionDate;

private String notes;

}

**Interaction Repository**

public interface InteractionRepository extends JpaRepository<Interaction, Long> { }

**Interaction Controller**

public class InteractionController {

private InteractionRepository interactionRepository;

public List<Interaction> getAllInteractions() {

return interactionRepository.findAll();

}

public Interaction createInteraction(@RequestBody Interaction interaction) {

return interactionRepository.save(interaction);

}

public ResponseEntity<Interaction> getInteractionById(@PathVariable Long id) {

Interaction interaction = interactionRepository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Interaction not found"));

return ResponseEntity.ok(interaction);

}

public ResponseEntity<Interaction> updateInteraction(@PathVariable Long id, @RequestBody Interaction interactionDetails) {

Interaction interaction = interactionRepository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Interaction not found")); interaction.setInteractionType(interactionDetails.getInteractionType()); interaction.setInteractionDate(interactionDetails.getInteractionDate()); interaction.setNotes(interactionDetails.getNotes());

Interaction updatedInteraction = interactionRepository.save(interaction);

return ResponseEntity.ok(updatedInteraction);

}

public ResponseEntity<Map<String, Boolean>> deleteInteraction(@PathVariable Long id) {

Interaction interaction = interactionRepository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Interaction not found"));

interactionRepository.delete(interaction); Map<String, Boolean> response = new HashMap<>(); response.put("deleted", Boolean.TRUE);

return ResponseEntity.ok(response);

}

}

**Frontend: Web Interface**

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>Customer List</title>

</head>

<body>

<h1>Customer List</h1>

<table>

<thead>

<tr>

<th>Customer Name</th>

<th>Customer Email</th>

<th>Customer Phone</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

<tr th:each="customer : ${customers}">

<td th:text="${customer.customerName}">Customer Name</td>

<td th:text="${customer.customerEmail}">Customer Email</td>

<td th:text="${customer.customerPhone}">Customer Phone</td>

<td> <a th:href="@{/api/customers/{id}(id=${customer.customerId})}">View</a>

<a th:href="@{/api/customers/edit/{id}(id=${customer.customerId})}">Edit</a>

<a th:href="@{/api/customers/delete/{id}(id=${customer.customerId})}">Delete</a>

</td>

</tr>

</tbody>

</table>

<a href="/api/customers/new">Add New Customer</a>

</body>

</html>