

Build a CRUD Rest API Project using the MySQL Database and JPA concept on Customer Entity created in the previous assignment.

1. Create a CustomerController to handle CRUD operations for the Customer entity.
2. Create a CustomerService to manage business logic and interact with the CustomerRepository.
3. In the CustomerRepository interface, which should extend JpaRepository, implement the following query methods:
  - Find Customers by First Name
  - Find Customers by Last Name
  - Find Customers by Email
  - Find Customers by Salary Range
  - Find Customers with Salary Greater Than a Specific Value
  - Find Customers by Name Starting With
  - Find Customers by Email Domain
4. Test the Query Methods using Postman

## Code-

### Customer.java (Entity)

```
1 package com.ProductRestApi.model;
2
3 import jakarta.persistence.Entity;
4
5 @Entity
6 public class Customer {
7
8     @Id
9     @GeneratedValue(strategy = GenerationType.IDENTITY)
10    private Long id;
11
12    @NotNull(message = "First name must not be null")
13    @Size(min = 3, message = "First name must contain at least 3 characters")
14    private String firstName;
15
16    @NotNull(message = "Last name must not be null")
17    @Size(min = 3, message = "Last name must contain at least 3 characters")
18    private String lastName;
19
20    @NotNull(message = "Email must not be null")
21    @Email(message = "Email should be valid")
22    private String email;
23
24    @NotNull(message = "Password must not be null")
25    @Pattern(regexp = "^(?=.*[a-zA-Z])(?=.*\\d)[A-Za-z\\d]{6,20}$",
26            message = "Password must be alphanumeric and 6-20 characters long")
27    private String password;
28
29    @NotNull(message = "Confirm password must not be null")
30    @Pattern(regexp = "^(?=.*[a-zA-Z])(?=.*\\d)[A-Za-z\\d]{6,20}$",
31            message = "Password must be alphanumeric and 6-20 characters long")
32    // @Transient // Exclude from persistence since it won't be stored in the database
33    private String confirmPassword;
```

```

45• @NotNull(message = "Mobile number must not be null")
46 @NotEmpty(message = "Mobile number must not be empty")
47 @Pattern(regexp = "\\d{10}", message = "Mobile number must be exactly 10 digits")
48 private String mobileNo;
49
50• @Min(value = 10000, message = "Salary must be at least 10,000")
51 @Max(value = 50000, message = "Salary must not exceed 50,000")
52 private double salary;
53
54 // Getters and setters
55
56• public Long getId() {
57     return id;
58 }
59
60• public void setId(Long id) {
61     this.id = id;
62 }
63
64• public String getFirstName() {
65     return firstName;
66 }
67
68• public void setFirstName(String firstName) {
69     this.firstName = firstName;
70 }
71
72• public String getLastName() {
73     return lastName;
74 }
75
76• public void setLastName(String lastName) {
77     this.lastName = lastName;

```

```

78 }
79
80• public String getEmail() {
81     return email;
82 }
83
84• public void setEmail(String email) {
85     this.email = email;
86 }
87
88• public String getPassword() {
89     return password;
90 }
91
92• public void setPassword(String password) {
93     this.password = password;
94 }
95
96• public String getConfirmPassword() {
97     return confirmPassword;
98 }
99
00• public void setConfirmPassword(String confirmPassword) {
01     this.confirmPassword = confirmPassword;
02 }
03
04• public String getMobileNo() {
05     return mobileNo;
06 }
07
08• public void setMobileNo(String mobileNo) {
09     this.mobileNo = mobileNo;
10 }

```

```

112     public double getSalary() {
113         return salary;
114     }
115
116     public void setSalary(double salary) {
117         this.salary = salary;
118     }
119
120
121 }
122

```

## CustomerController.java

```

1 package com.ProductRestApi.controller;
2
3 import java.util.List;
4
19
20 @RestController
21 @RequestMapping("/api/customers")
22 public class CustomerController {
23
24     @Autowired
25     private CustomerService customerService;
26
27     @GetMapping
28     public List<Customer> getAllCustomers() {
29         return customerService.getAllCustomers();
30     }
31
32     @GetMapping("/{id}")
33     public ResponseEntity<Customer> getCustomerById(@PathVariable Long id) {
34         return customerService.getCustomerById(id)
35             .map(ResponseEntity::ok)
36             .orElse(ResponseEntity.notFound().build());
37     }
38
39     @PostMapping
40     public ResponseEntity<Customer> addCustomer(@RequestBody Customer customer) {
41         return ResponseEntity.ok(customerService.addCustomer(customer));
42     }
43
44     @PutMapping("/{id}")
45     public ResponseEntity<Customer> updateCustomer(@PathVariable Long id, @RequestBody Customer customer) {
46         return ResponseEntity.ok(customerService.updateCustomer(id, customer));
47     }
48
49     @DeleteMapping("/{id}")
50     public ResponseEntity<String> deleteCustomer(@PathVariable Long id) {
51         customerService.deleteCustomer(id);
52         return ResponseEntity.ok("Customer deleted successfully!");
53     }
54
55     // Custom query endpoints
56     @GetMapping("/search/firstName/{firstName}")
57     public List<Customer> findByFirstName(@PathVariable String firstName) {
58         return customerService.findByFirstName(firstName);
59     }
60
61     @GetMapping("/search/lastName/{lastName}")
62     public List<Customer> findByLastName(@PathVariable String lastName) {
63         return customerService.findByLastName(lastName);
64     }
65
66     @GetMapping("/search/email/{email}")
67     public ResponseEntity<Customer> findByEmail(@PathVariable String email) {
68         return ResponseEntity.ok(customerService.findByEmail(email));
69     }
70
71     @GetMapping("/search/salaryRange")
72     public List<Customer> findBySalaryRange(@RequestParam double minSalary, @RequestParam double maxSalary) {
73         return customerService.findBySalaryRange(minSalary, maxSalary);
74     }
75
76     @GetMapping("/search/salaryGreaterThan/{salary}")
77     public List<Customer> findBySalaryGreaterThan(@PathVariable double salary) {
78         return customerService.findBySalaryGreaterThan(salary);
79     }
80

```

```

    @GetMapping("/search/nameStartingWith/{prefix}")
    public List<Customer> findByFirstNameStartingWith(@PathVariable String prefix) {
        return customerService.findByFirstNameStartingWith(prefix);
    }

    @GetMapping("/search/emailDomain")
    public List<Customer> findByEmailDomain(@RequestParam String domain) {
        return customerService.findByEmailDomain(domain);
    }
}

```

## CustomerService.java

```

1 package com.ProductRestApi.service;
2
3 import java.util.List;
4
11
12 @Service
13 public class CustomerService {
14
15     @Autowired
16     private CustomerRepository customerRepository;
17
18     public List<Customer> getAllCustomers() {
19         return customerRepository.findAll();
20     }
21
22     public Optional<Customer> getCustomerById(Long id) {
23         return customerRepository.findById(id);
24     }
25
26     public Customer addCustomer(Customer customer) {
27         if (!customer.getPassword().equals(customer.getConfirmPassword())) {
28             throw new IllegalArgumentException("Passwords do not match");
29         }
30         return customerRepository.save(customer);
31     }
32
33     public Customer updateCustomer(Long id, Customer customer) {
34         if (!customerRepository.existsById(id)) {
35             throw new IllegalArgumentException("Customer not found");
36         }
37         customer.setId(id);
38         return customerRepository.save(customer);
39     }
40

```

```

41●     public void deleteCustomer(Long id) {
42         if (!customerRepository.existsById(id)) {
43             throw new IllegalArgumentException("Customer not found");
44         }
45         customerRepository.deleteById(id);
46     }
47
48     // Query methods
49●     public List<Customer> findByFirstName(String firstName) {
50         return customerRepository.findByFirstName(firstName);
51     }
52
53●     public List<Customer> findByLastName(String lastName) {
54         return customerRepository.findByLastName(lastName);
55     }
56
57●     public Customer findByEmail(String email) {
58         return customerRepository.findByEmail(email);
59     }
60
61●     public List<Customer> findBySalaryRange(double minSalary, double maxSalary) {
62         return customerRepository.findBySalaryBetween(minSalary, maxSalary);
63     }
64
65●     public List<Customer> findBySalaryGreaterThan(double salary) {
66         return customerRepository.findBySalaryGreaterThan(salary);
67     }
68
69●     public List<Customer> findByFirstNameStartingWith(String prefix) {
70         return customerRepository.findByFirstNameStartingWith(prefix);
71     }
72

```

```

●     public List<Customer> findByEmailDomain(String domain) {
        return customerRepository.findByEmailDomain(domain);
    }
}

```

## CustomerRepository.java

```
1 package com.ProductRestApi.repository;
2
3 import java.util.List;
4
5
6
7
8
9
10
11 public interface CustomerRepository extends JpaRepository<Customer, Long>{
12
13     // Find Customers by First Name
14     List<Customer> findByFirstName(String firstName);
15
16     // Find Customers by Last Name
17     List<Customer> findByLastName(String lastName);
18
19     // Find Customers by Email
20     Customer findByEmail(String email);
21
22     // Find Customers by Salary Range
23     List<Customer> findBySalaryBetween(double minSalary, double maxSalary);
24
25     // Find Customers with Salary Greater Than a Specific Value
26     List<Customer> findBySalaryGreaterThan(double salary);
27
28     // Find Customers by Name Starting With
29     List<Customer> findByFirstNameStartingWith(String prefix);
30
31     // Find Customers by Email Domain
32     @Query("SELECT c FROM Customer c WHERE c.email LIKE %:domain")
33     List<Customer> findByEmailDomain(@Param("domain") String domain);
34
35 }
36
```

## Adding the Customer through Postman

Postman interface showing a successful POST request to `http://localhost:8080/api/customers`. The request body is a JSON object representing a customer:

```
{  "firstName": "Rohan",  "lastName": "Kadam",  "email": "rohankadam@gmail.com",  "password": "pass123",  "confirmPassword": "pass123",  "mobileNo": "1234567890",  "salary": 20000}
```

The response is a 200 OK status with a JSON object containing the customer's details and an assigned ID:

```
{  "id": 1,  "firstName": "Rohan",  "lastName": "Kadam",  "email": "rohankadam@gmail.com",  "password": "pass123",  "confirmPassword": "pass123",  "mobileNo": "1234567890",  "salary": 20000.0}
```

Postman interface showing a successful POST request to `http://localhost:8080/api/customers`. The request body is a JSON object representing a customer:

```
{  "firstName": "Ketan",  "lastName": "Jadhav",  "email": "ketan@gmail.com",  "password": "pass123",  "confirmPassword": "pass123",  "mobileNo": "1234567890",  "salary": 30000}
```

The response is a 200 OK status with a JSON object containing the customer's details and an assigned ID:

```
{  "id": 2,  "firstName": "Ketan",  "lastName": "Jadhav",  "email": "ketan@gmail.com",  "password": "pass123",  "confirmPassword": "pass123",  "mobileNo": "1234567890",  "salary": 30000.0}
```

## Find by First Name

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/api/customers/search/firstName/Rohan`. The request body is a JSON object representing a customer. The response is also a JSON object, showing a customer with the first name 'Rohan'.

**Request:**

```
GET http://localhost:8080/api/customers/search/firstName/Rohan
```

**Request Body:**

```
{  "firstName": "Ketan",  "lastName": "Jadhav",  "email": "ketan@gmail.com",  "password": "pass123",  "confirmPassword": "pass123",  "mobileNo": "1234567890",  "salary": 30000}
```

**Response:**

```
{  "id": 1,  "firstName": "Rohan",  "lastName": "Kadam",  "email": "rohankadam@gmail.com",  "password": "pass123",  "confirmPassword": "pass123",  "mobileNo": "1234567890",  "salary": 20000.0}
```

## Find by Last Name

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/api/customers/search/lastName/Jadhav`. The request body is a JSON object representing a customer. The response is also a JSON object, showing a customer with the last name 'Jadhav'.

**Request:**

```
GET http://localhost:8080/api/customers/search/lastName/Jadhav
```

**Request Body:**

```
{  "firstName": "Ketan",  "lastName": "Jadhav",  "email": "ketan@gmail.com",  "password": "pass123",  "confirmPassword": "pass123",  "mobileNo": "1234567890",  "salary": 30000}
```

**Response:**

```
{  "id": 2,  "firstName": "Ketan",  "lastName": "Jadhav",  "email": "ketan@gmail.com",  "password": "pass123",  "confirmPassword": "pass123",  "mobileNo": "1234567890",  "salary": 30000.0}
```



## Find by Email

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** `http://localhost:8080/api/customers/search/email/rohankadam@gmail.com`
- Body:** A JSON object representing a customer:

```
{  "firstName": "Rohan",  "lastName": "Kadam",  "email": "rohankadam@gmail.com",  "password": "pass123",  "confirmPassword": "pass123",  "mobileNo": "1234567890",  "salary": 20000}
```
- Response:** 200 OK, 16 ms, 376 B. The response body is a JSON object:

```
{  "id": 1,  "firstName": "Rohan",  "lastName": "Kadam",  "email": "rohankadam@gmail.com",  "password": "pass123",  "confirmPassword": "pass123",  "mobileNo": "1234567890",  "salary": 20000.0}
```

## Find by Salary Range

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** `http://localhost:8080/api/customers/search/salaryRange?minSalary=15000&maxSalary=25000`
- Body:** A JSON object representing a customer:

```
{  "firstName": "Rohan",  "lastName": "Kadam"}
```
- Response:** 200 OK, 16 ms, 334 B. The response body is a JSON object:

```
{  "id": 1,  "firstName": "Rohan",  "lastName": "Kadam",  "email": "rohankadam@gmail.com",  "password": "pass123",  "confirmPassword": "pass123",  "mobileNo": "1234567890",  "salary": 20000.0}
```

## Find by Salary Greater Than

GET ▼ http://localhost:8080/api/customers/search/salaryGreaterThan/15000 Send ▼

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings Cookies

● none ● form-data ● x-www-form-urlencoded ● raw ● binary JSON ▼ Beautify

```
1 {
2   "firstName": "Rohan",
3   "lastName": "Kadam"
}
```

Body Cookies Headers (5) Test Results 200 OK 12 ms 499 B Save Response ▼

Pretty Raw Preview Visualize JSON ▼ ⇌

```
2 {
3   "id": 1,
4   "firstName": "Rohan",
5   "lastName": "Kadam",
6   "email": "rohankadam@gmail.com",
7   "password": "pass123",
8   "confirmPassword": "pass123",
9   "mobileNo": "1234567890",
10  "salary": 20000.0
11 },
12 {
13   "id": 2,
14   "firstName": "Ketan",
15   "lastName": "Jadhav",
16   "email": "ketan@gmail.com",
17   "password": "pass123",
18   "confirmPassword": "pass123",
19   "mobileNo": "1234567890",
20   "salary": 30000.0
21 }
```

## Find by Name Starting With

GET ▼ http://localhost:8080/api/customers/search/nameStartingWith/ke Send ▼ 🔔

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings Cookies

● none ● form-data ● x-www-form-urlencoded ● raw ● binary JSON ▼ Beautify

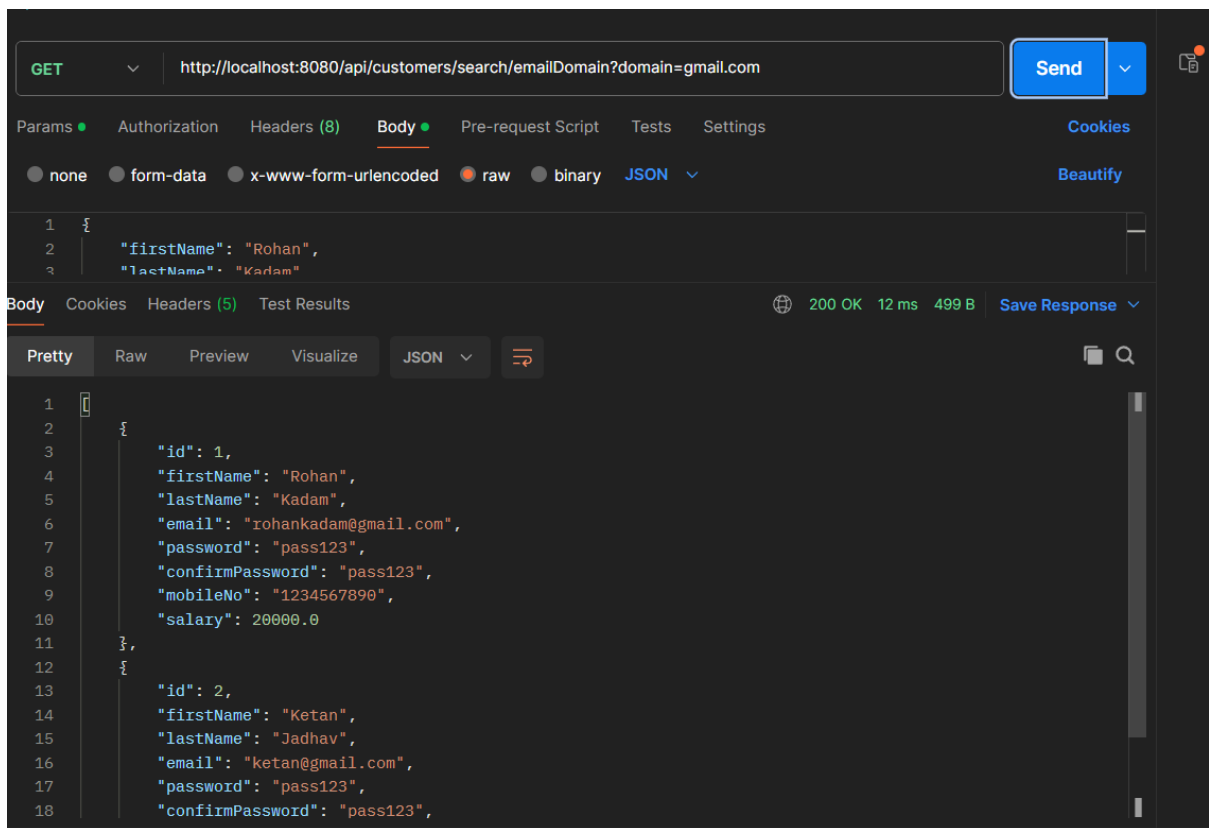
```
1 {
2   "firstName": "Rohan",
3   "lastName": "Kadam"
}
```

Body Cookies Headers (5) Test Results 200 OK 15 ms 330 B Save Response ▼

Pretty Raw Preview Visualize JSON ▼ ⇌

```
1 [
2   {
3     "id": 2,
4     "firstName": "Ketan",
5     "lastName": "Jadhav",
6     "email": "ketan@gmail.com",
7     "password": "pass123",
8     "confirmPassword": "pass123",
9     "mobileNo": "1234567890",
10    "salary": 30000.0
11   }
12 ]
```

## Find by Email Domain



The screenshot displays a REST client interface with a GET request to `http://localhost:8080/api/customers/search/emailDomain?domain=gmail.com`. The request body is a JSON object with `firstName: "Rohan"` and `lastName: "Kadam"`. The response is a 200 OK status with a 12 ms response time and 499 B of data. The response body is a JSON array of two customer objects, displayed in a pretty-printed format.

**Request:**

```
GET http://localhost:8080/api/customers/search/emailDomain?domain=gmail.com
```

**Request Body:**

```
{  "firstName": "Rohan",  "lastName": "Kadam"}
```

**Response:** 200 OK, 12 ms, 499 B

**Response Body:**

```
[  {    "id": 1,    "firstName": "Rohan",    "lastName": "Kadam",    "email": "rohankadam@gmail.com",    "password": "pass123",    "confirmPassword": "pass123",    "mobileNo": "1234567890",    "salary": 20000.0  },  {    "id": 2,    "firstName": "Ketan",    "lastName": "Jadhav",    "email": "ketan@gmail.com",    "password": "pass123",    "confirmPassword": "pass123"  }]
```