

Create a new Data Rest Project as per below details given below.

Project Name: Application

Classname: Customer

Attribute Names for Customer class: customerId, firstName, lastName, email, address

Create Repository class for the Customer Entity to map the following APIs with the query methods

HTTP Methods	routes	Action
POST	/customers	Create a new Customer
GET	/customers	Read all customers
GET	/customers/{customerId}	Read a single Customer
DELETE	/customers/{customerId}	Delete a single Customer
PUT	/customers/{customerId}	Update a single Customer

## Code-

### Customer.java (Entity)

```
1 package com.ProductRestApi.model;
2
3 import java.time.Instant;
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;
34}
```

```

53• @NotNull(message = "Mobile number must not be null")
54 @NotEmpty(message = "Mobile number must not be empty")
55 @Pattern(regexp = "\\d{10}", message = "Mobile number must be exactly 10 digits")
56 private String mobileNo;
57
58• @Min(value = 10000, message = "Salary must be at least 10,000")
59 @Max(value = 50000, message = "Salary must not exceed 50,000")
60 private double salary;
61
62 // @CreateDate
63 // private Instant createdAt; // Automatically set the creation timestamp
64 //
65 // @LastModifiedDate
66 // private Instant modifiedAt; // Automatically set the modification timestamp
67 //
68• @Column(name = "created_at", updatable = false)
69 private Instant createdAt;
70
71• @Column(name = "modified_at")
72 private Instant modifiedAt;
73
74• @NotNull(message = "Address must not be null")
75 private String address; // Added address field
76
77• @PrePersist
78 protected void onCreate() {
79     createdAt = Instant.now();
80 }
81
82• @PreUpdate
83 protected void onUpdate() {
84     modifiedAt = Instant.now();
85 }

```

```
87 // Getters and setters
88
89 public Long getId() {
90     return id;
91 }
92
93 public void setId(Long id) {
94     this.id = id;
95 }
96
97 public String getFirstName() {
98     return firstName;
99 }
100
101 public void setFirstName(String firstName) {
102     this.firstName = firstName;
103 }
104
105 public String getLastName() {
106     return lastName;
107 }
108
109 public void setLastName(String lastName) {
110     this.lastName = lastName;
111 }
112
113 public String getEmail() {
114     return email;
115 }
116
117 public void setEmail(String email) {
118     this.email = email;
119 }
```

```
121• public String getPassword() {
122     return password;
123 }
124
125• public void setPassword(String password) {
126     this.password = password;
127 }
128
129• public String getConfirmPassword() {
130     return confirmPassword;
131 }
132
133• public void setConfirmPassword(String confirmPassword) {
134     this.confirmPassword = confirmPassword;
135 }
136
137• public String getAddress() {
138     return address;
139 }
140
141• public void setAddress(String address) {
142     this.address = address;
143 }
144
145• public String getMobileNo() {
146     return mobileNo;
147 }
148
149• public void setMobileNo(String mobileNo) {
150     this.mobileNo = mobileNo;
151 }
152
```

```
153• public double getSalary() {
154     return salary;
155 }
156
157• public void setSalary(double salary) {
158     this.salary = salary;
159 }
160
161• public Instant getCreatedAt() {
162     return createdAt;
163 }
164
165• public void setCreatedAt(Instant createdAt) {
166     this.createdAt = createdAt;
167 }
168
169• public Instant getModifiedAt() {
170     return modifiedAt;
171 }
172
173• public void setModifiedAt(Instant modifiedAt) {
174     this.modifiedAt = modifiedAt;
175 }
176
177 }
178
```

## CustomerController.java

```
1 package com.ProductRestApi.controller;
2
3 import java.util.List;
4
19 @RestController
20 @RequestMapping("/api/customers")
21 public class CustomerController {
22
23     @Autowired
24     private CustomerService customerService;
25
26     @GetMapping
27     public List<Customer> getAllCustomers() {
28         return customerService.getAllCustomers();
29     }
30
31     @GetMapping("/{id}")
32     public ResponseEntity<Customer> getCustomerById(@PathVariable Long id) {
33         return customerService.getCustomerById(id)
34             .map(ResponseEntity::ok)
35             .orElse(ResponseEntity.notFound().build());
36     }
37
38     @PostMapping
39     public ResponseEntity<Customer> addCustomer(@RequestBody Customer customer) {
40         return ResponseEntity.ok(customerService.addCustomer(customer));
41     }
42
43     @PutMapping("/{id}")
44     public ResponseEntity<Customer> updateCustomer(@PathVariable Long id, @RequestBody Customer customer) {
45         return ResponseEntity.ok(customerService.updateCustomer(id, customer));
46     }
47
48     @DeleteMapping("/{id}")
49     public ResponseEntity<String> deleteCustomer(@PathVariable Long id) {
50         customerService.deleteCustomer(id);
51         return ResponseEntity.ok("Customer deleted successfully!");
52     }
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
81 @GetMapping("/search/nameStartingWith/{prefix}")
82 public List<Customer> findByFirstNameStartingWith(@PathVariable String prefix) {
83     return customerService.findByFirstNameStartingWith(prefix);
84 }
85
86 @GetMapping("/search/emailDomain")
87 public List<Customer> findByEmailDomain(@RequestParam String domain) {
88     return customerService.findByEmailDomain(domain);
89 }
90
91 }
```

## CustomerService.java

```
1 package com.ProductRestApi.service;
2
3 import java.util.List;
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
73     public List<Customer> findByEmailDomain(String domain) {
74         return customerRepository.findByEmailDomain(domain);
75     }
76
77 }
78

```

## CustomerRepository

```

1  package com.ProductRestApi.repository;
2
3  import java.util.List;
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

```

## Output

### Add the Customer

Postman interface showing a successful POST request to `http://localhost:8080/api/customers`. The request body is a JSON object with the following fields:

```
{
  "firstName": "John",
  "lastName": "Doe",
  "email": "johndoe@example.com",
  "address": "123 Main St",
  "password": "password123",
  "confirmPassword": "password123",
  "mobileNo": "1234567890",
  "salary": 30000
}
```

The response is a 200 OK status with a 62 ms response time and 423 B of data. The response body is a JSON object with the following fields:

```
{
  "id": 1,
  "firstName": "John",
  "lastName": "Doe",
  "email": "johndoe@example.com",
  "password": "password123",
  "confirmPassword": "password123",
  "mobileNo": "1234567890",
  "salary": 30000.0,
  "createdAt": "2025-01-02T14:59:20.583516400Z",
  "modifiedAt": null,
  "address": "123 Main St"
}
```

### Get the All Customer

Postman interface showing a successful GET request to `http://localhost:8080/api/customers`. The response is a 200 OK status with a 32 ms response time and 1.27 KB of data. The response body is a JSON array of two customer objects:

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



## Get the single customer by id

GET http://localhost:8080/api/customers/5 Send

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

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary JSON Beautify

Body Cookies Headers (5) Test Results 200 OK 24 ms 420 B Save Response

Pretty Raw Preview Visualize JSON ≡

```
1 {
2   "id": 5,
3   "firstName": "John",
4   "lastName": "Doe",
5   "email": "johndoe@example.com",
6   "password": "password123",
7   "confirmPassword": "password123",
8   "mobileNo": "1234567890",
9   "salary": 30000.0,
10  "createdAt": "2025-01-02T14:59:20.583516Z",
11  "modifiedAt": null,
12  "address": "123 Main St"
13 }
```

## Update the single customer data by id

PUT http://localhost:8080/api/customers/5 Send

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

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary JSON Beautify

Body Cookies Headers (5) Test Results 200 OK 319 ms 425 B Save Response

Pretty Raw Preview Visualize JSON ≡

```
1 {
2   "firstName": "John",
3   "lastName": "David",
4   "email": "johndoe@example.com",
5   "address": "123 Main St",
6   "password": "password123",
7   "confirmPassword": "password123",
8   "mobileNo": "1234567890",
9   "salary": 30000
10 }
```

```
1 {
2   "id": 5,
3   "firstName": "John",
4   "lastName": "David",
5   "email": "johndoe@example.com",
6   "password": "password123",
7   "confirmPassword": "password123",
8   "mobileNo": "1234567890",
9   "salary": 30000.0,
10 }
```

## Delete the Data of Customer by id

The screenshot displays a REST client interface with a DELETE request configured. The URL is `http://localhost:8080/api/customers/4`. The request body is a JSON object representing a customer. The response is a 200 OK status with a message indicating successful deletion.

**Request:**

- Method: DELETE
- URL: `http://localhost:8080/api/customers/4`
- Body (JSON):

```
{  "firstName": "John",  "lastName": "David",  "email": "johndoe@example.com",  "address": "123 Main St",  "password": "password123",  "confirmPassword": "password123",  "mobileNo": "1234567890",  "salary": 30000}
```

**Response:**

- Status: 200 OK
- Time: 68 ms
- Size: 194 B
- Message: `1 Customer deleted successfully!`