09: CONTACT MANAGEMENT CRUD APP

Name: Shourya Ojha

Registration Number: 202401100300239

Library ID: 2428CSEAI1182

2nd Year CSE AI Sec D

Shift: 11 am to 1 pm

Subject: OOPS LAB Practicals

Github Repository Link:

shouryaojha/Contact-Management-CRUD-Application: This is a simple website created using springboot. It handles CRUD Operations.

Contact-Management-CRUD-Application

```
|
├── pom.xml
├── README.md
|
├── src
|   └── main
|       ├── java
|       |   └── com
|       |       └── example
```

```
│   │       └── contact
│   │           │
│   │           ├── ContactManagementApplication.java
│   │           │
│   │           ├── controller
│   │           │   └── ContactController.java
│   │           │
│   │           ├── service
│   │           │   └── ContactService.java
│   │           │
│   │           ├── repository
│   │           │   └── ContactRepository.java
│   │           │
│   │           └── model
│   │               └── Contact.java
│   │
│   └── resources
│       ├── static
│       │   ├── index.html
│       │   ├── style.css
│       │   └── script.js
│       │
│       └── application.properties
│
└── .gitignore   (optional, but fine if present)
```

Index.html code:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <title>Contact Management</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

<div class="container">
  <h2>📞 Contact Management System</h2>

  <div class="form">
    <input type="hidden" id="contactId">

    <input type="text" id="name" placeholder="Enter Name">
    <input type="text" id="phone" placeholder="Enter Phone Number">

    <button onclick="saveContact()" id="saveBtn">Add Contact</button>
    <button onclick="resetForm()" class="secondary">Clear</button>
  </div>

  <table>
    <thead>
    <tr>
      <th>ID</th>
      <th>Name</th>
      <th>Phone</th>
      <th>Actions</th>
    </tr>
    </thead>
    <tbody id="contactTable"></tbody>
  </table>
</div>

<script src="script.js"></script>
</body>
</html>
```

Style.css code:

```
body {
  background: #f4f6f8;
  font-family: Arial, sans-serif;
}
```

```
.container {
  width: 70%;
  margin: 40px auto;
  background: white;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 0 10px rgba(0,0,0,0.1);
}

h2 {
  text-align: center;
  margin-bottom: 20px;
}

.form {
  display: flex;
  gap: 10px;
  justify-content: center;
  margin-bottom: 20px;
}

input {
  padding: 8px;
  width: 200px;
}

button {
  padding: 8px 14px;
  border: none;
  cursor: pointer;
  background: #007bff;
  color: white;
  border-radius: 4px;
}

button.secondary {
  background: #6c757d;
}

button.edit {
  background: #28a745;
}

button.delete {
```

```
  background: #dc3545;
}

table {
  width: 100%;
  border-collapse: collapse;
}

th, td {
  padding: 10px;
  border-bottom: 1px solid #ddd;
  text-align: center;
}
```

script.js code:

```
const API_URL = "http://localhost:8082/contacts";

function loadContacts() {
  fetch(API_URL)
    .then(res => res.json())
    .then(data => {
      let rows = "";
      data.forEach(c => {
        rows += `
          <tr>
            <td>${c.id}</td>
            <td>${c.name}</td>
            <td>${c.phone}</td>
            <td>
              <button class="edit" onclick="editContact(${c.id}, '${c.name}',
'${c.phone}')">Edit</button>
              <button class="delete" onclick="deleteContact(${c.id})">Delete</button>
            </td>
          </tr>
        `;
      });
      document.getElementById("contactTable").innerHTML = rows;
    });
}

function saveContact() {
  const id = document.getElementById("contactId").value;
  const name = document.getElementById("name").value.trim();
```

```
  const phone = document.getElementById("phone").value.trim();

  if (!name || !phone) {
    alert("Please fill all fields");
    return;
  }

  const method = id ? "PUT" : "POST";
  const url = id ? `${API_URL}/${id}` : API_URL;

  fetch(url, {
    method: method,
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ name, phone })
  }).then(() => {
    resetForm();
    loadContacts();
  });
}

function editContact(id, name, phone) {
  document.getElementById("contactId").value = id;
  document.getElementById("name").value = name;
  document.getElementById("phone").value = phone;
  document.getElementById("saveBtn").innerText = "Update Contact";
}

function deleteContact(id) {
  fetch(`${API_URL}/${id}`, { method: "DELETE" })
    .then(() => loadContacts());
}

function resetForm() {
  document.getElementById("contactId").value = "";
  document.getElementById("name").value = "";
  document.getElementById("phone").value = "";
  document.getElementById("saveBtn").innerText = "Add Contact";
}

// Load on page start
loadContacts();
```

ContactController.java

```
package com.example.contact.controller;

import com.example.contact.model.Contact;
import com.example.contact.service.ContactService;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/contacts")
@CrossOrigin(origins = "*")
public class ContactController {

  private final ContactService service;

  public ContactController(ContactService service) {
    this.service = service;
  }

  // POST /contacts
  @PostMapping
  public Contact createContact(@RequestBody Contact contact) {
    return service.addContact(contact);
  }

  // GET /contacts
  @GetMapping
  public List<Contact> getAllContacts() {
    return service.getAllContacts();
  }

  // PUT /contacts/{id}
  @PutMapping("/{id}")
  public Contact updateContact(@PathVariable Long id,
                               @RequestBody Contact contact) {
    return service.updateContact(id, contact);
  }

  // DELETE /contacts/{id}
  @DeleteMapping("/{id}")
  public String deleteContact(@PathVariable Long id) {
    service.deleteContact(id);
```

```
        return "Contact deleted successfully";
    }
}
```

contact.java

```
package com.example.contact.model;

public class Contact {

    private Long id;
    private String name;
    private String phone;

    public Contact() {
    }

    public Contact(Long id, String name, String phone) {
        this.id = id;
        this.name = name;
        this.phone = phone;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
```

```
    this.phone = phone;
  }
}
```

ContactRepository.java

```
package com.example.contact.repository;

import com.example.contact.model.Contact;
import org.springframework.stereotype.Repository;

import java.util.*;

@Repository
public class ContactRepository {

  private final Map<Long, Contact> contactStore = new HashMap<>();
  private Long idCounter = 1L;

  public Contact save(Contact contact) {
    contact.setId(idCounter++);
    contactStore.put(contact.getId(), contact);
    return contact;
  }

  public List<Contact> findAll() {
    return new ArrayList<>(contactStore.values());
  }

  public Contact update(Long id, Contact contact) {
    contact.setId(id);
    contactStore.put(id, contact);
    return contact;
  }

  public void delete(Long id) {
    contactStore.remove(id);
  }
}
```

ContactService.java

```
package com.example.contact.service;

import com.example.contact.model.Contact;
```

```
import com.example.contact.repository.ContactRepository;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class ContactService {

  private final ContactRepository repository;

  public ContactService(ContactRepository repository) {
    this.repository = repository;
  }

  public Contact addContact(Contact contact) {
    return repository.save(contact);
  }

  public List<Contact> getAllContacts() {
    return repository.findAll();
  }

  public Contact updateContact(Long id, Contact contact) {
    return repository.update(id, contact);
  }

  public void deleteContact(Long id) {
    repository.delete(id);
  }
}
```

ContactManagementApplication.java

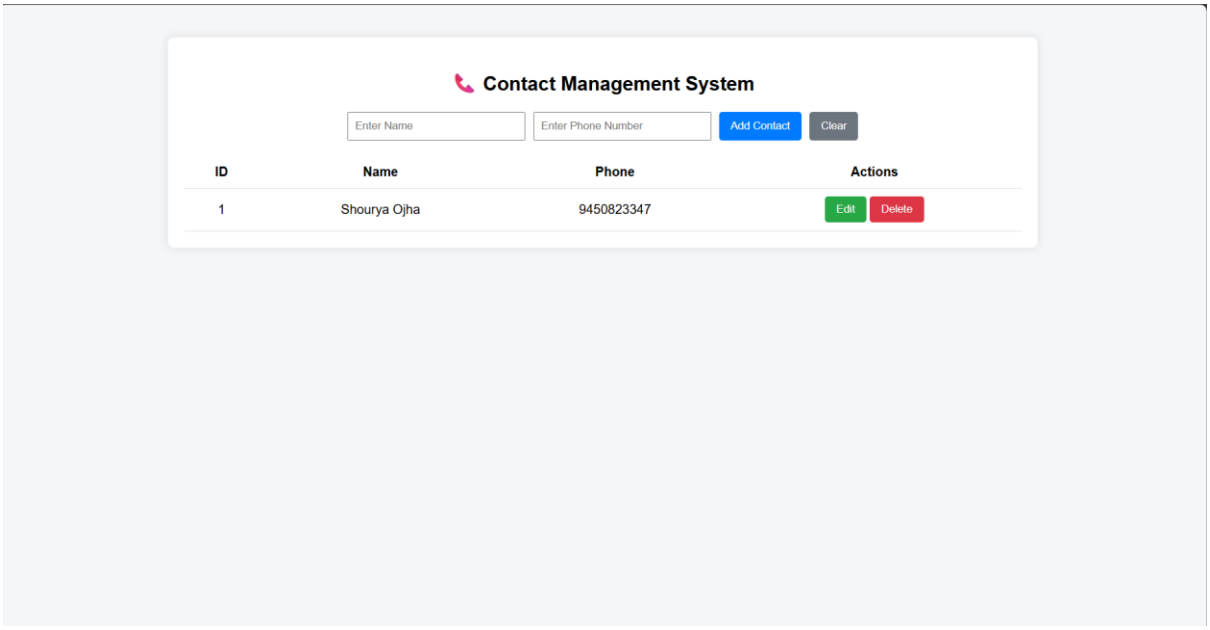
```
package com.example.contact;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ContactManagementApplication {

  public static void main(String[] args) {
    SpringApplication.run(ContactManagementApplication.class, args);
  }
}
```

Frontend: All the crud operations can be easily performed using the respective buttons liked add delete edit



Backend: