HAND ON EXERCISE-WEEK 6

**1.How can you implement a basic full stack user registration and retrieval system using React as the frontend and Java Spring Boot as the backend?**

**Backend - Java (Spring Boot)**

**User.java (Model)**

package com.example.demo.model;

public class User {

private Long id;

private String name;

private String email;

// Constructors

public User() {}

public User(Long id, String name, String email) {

this.id = id;

this.name = name;

this.email = email;

}

// Getters and Setters

// ...

}

**UserController.java**

package com.example.demo.controller;

import com.example.demo.model.User;

import org.springframework.web.bind.annotation.\*;

import java.util.\*;

@RestController

@RequestMapping("/api/users")

@CrossOrigin(origins = "http://localhost:3000") // Allow React to access backend

public class UserController {

private List<User> users = new ArrayList<>();

@PostMapping("/add")

public String addUser(@RequestBody User user) {

user.setId((long) (users.size() + 1));

users.add(user);

return "User added successfully";

}

@GetMapping("/list")

public List<User> getUsers() {

return users;

}

}

**Frontend - React**

**App.js**

import React, { useState, useEffect } from 'react';

import axios from 'axios';

function App() {

const [users, setUsers] = useState([]);

const [formData, setFormData] = useState({ name: '', email: '' });

const fetchUsers = async () => {

const res = await axios.get('http://localhost:8080/api/users/list');

setUsers(res.data);

};

useEffect(() => {

fetchUsers();

}, []);

const handleSubmit = async (e) => {

e.preventDefault();

await axios.post('http://localhost:8080/api/users/add', formData);

fetchUsers(); // Refresh list

};

return (

<div>

<h2>User Form</h2>

<form onSubmit={handleSubmit}>

<input type="text" placeholder="Name"

value={formData.name}

onChange={(e) => setFormData({ ...formData, name: e.target.value })} />

<input type="email" placeholder="Email"

value={formData.email}

onChange={(e) => setFormData({ ...formData, email: e.target.value })} />

<button type="submit">Add User</button>

</form>

<h3>User List:</h3>

<ul>

{users.map((u) => (

<li key={u.id}>{u.name} - {u.email}</li>

))}

</ul>

</div>

);

}

export default App;

**2.How can we build a complete CRUD application using React (frontend), Spring Boot (backend), and MySQL (database)?**

**Backend (Spring Boot + MySQL)**

**Product.java (Entity)**

package com.example.demo.model;

import jakarta.persistence.\*;

@Entity

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private double price;

// Constructors, Getters, Setters

}

**ProductRepository.java**

java

CopyEdit

package com.example.demo.repository;

import com.example.demo.model.Product;

import org.springframework.data.jpa.repository.JpaRepository;

public interface ProductRepository extends JpaRepository<Product, Long> {}

**ProductController.java**

package com.example.demo.controller;

import com.example.demo.model.Product;

import com.example.demo.repository.ProductRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/api/products")

@CrossOrigin(origins = "http://localhost:3000")

public class ProductController {

@Autowired

private ProductRepository repo;

@GetMapping

public List<Product> getAllProducts() {

return repo.findAll();

}

@PostMapping

public Product addProduct(@RequestBody Product product) {

return repo.save(product);

}

@PutMapping("/{id}")

public Product updateProduct(@PathVariable Long id, @RequestBody Product product) {

Product existing = repo.findById(id).orElseThrow();

existing.setName(product.getName());

existing.setPrice(product.getPrice());

return repo.save(existing);

}

@DeleteMapping("/{id}")

public void deleteProduct(@PathVariable Long id) {

repo.deleteById(id);

}

}

**Frontend (React)**

**ProductApp.js**

import React, { useEffect, useState } from 'react';

import axios from 'axios';

function ProductApp() {

const [products, setProducts] = useState([]);

const [form, setForm] = useState({ name: '', price: '' });

const [editId, setEditId] = useState(null);

const fetchData = async () => {

const res = await axios.get('http://localhost:8080/api/products');

setProducts(res.data);

};

useEffect(() => { fetchData(); }, []);

const handleSubmit = async (e) => {

e.preventDefault();

if (editId) {

await axios.put(`http://localhost:8080/api/products/${editId}`, form);

} else {

await axios.post('http://localhost:8080/api/products', form);

}

setForm({ name: '', price: '' });

setEditId(null);

fetchData();

};

const handleEdit = (product) => {

setForm({ name: product.name, price: product.price });

setEditId(product.id);

};

const handleDelete = async (id) => {

await axios.delete(`http://localhost:8080/api/products/${id}`);

fetchData();

};

return (

<div>

<h2>Product Manager</h2>

<form onSubmit={handleSubmit}>

<input type="text" placeholder="Name" value={form.name}

onChange={(e) => setForm({ ...form, name: e.target.value })} />

<input type="number" placeholder="Price" value={form.price}

onChange={(e) => setForm({ ...form, price: e.target.value })} />

<button type="submit">{editId ? "Update" : "Add"}</button>

</form>

<ul>

{products.map(p => (

<li key={p.id}>

{p.name} - ₹{p.price}

<button onClick={() => handleEdit(p)}>Edit</button>

<button onClick={() => handleDelete(p.id)}>Delete</button>

</li>

))}

</ul>

</div>

);

}

export default ProductApp;