WT LAB Assignment 4: Result App

```
Name : Aditya Sakhare

Roll No: 26 TYCS-D Batch 2
```

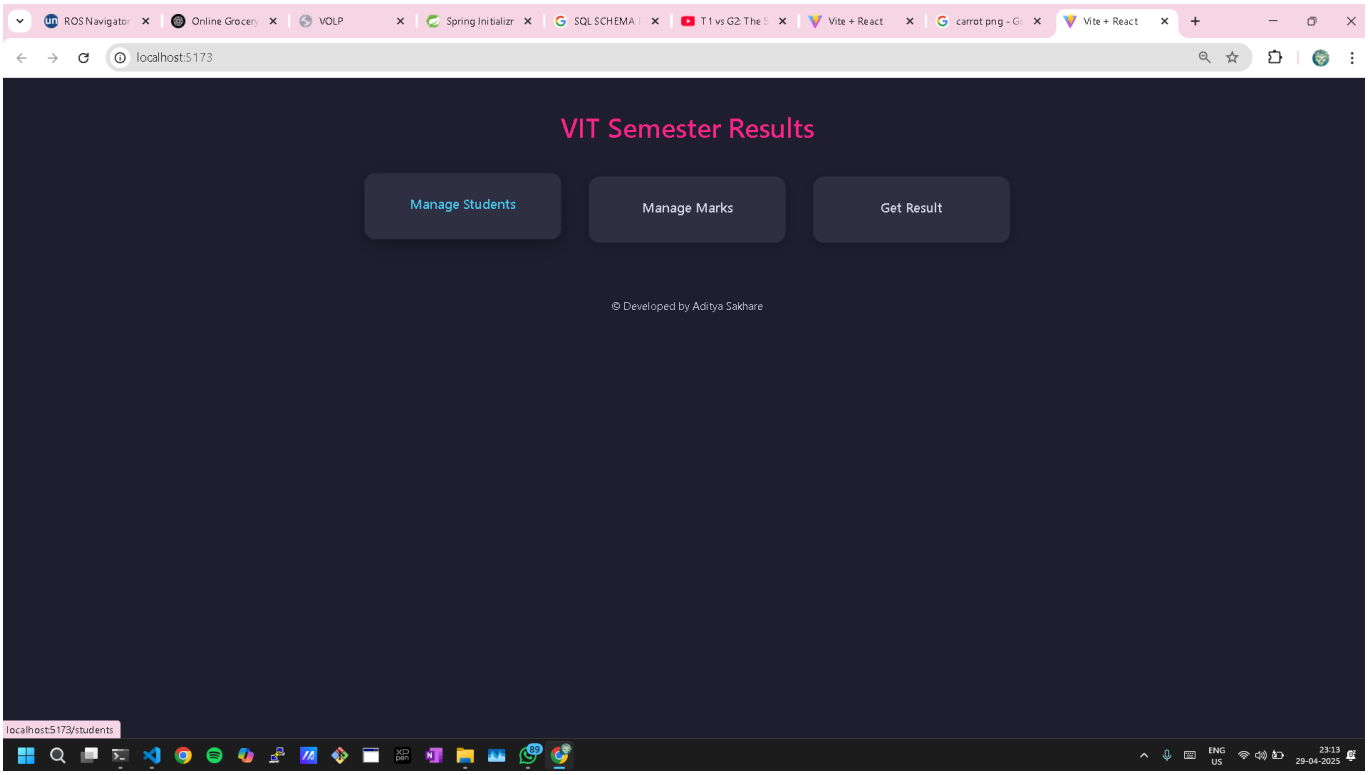
Design:

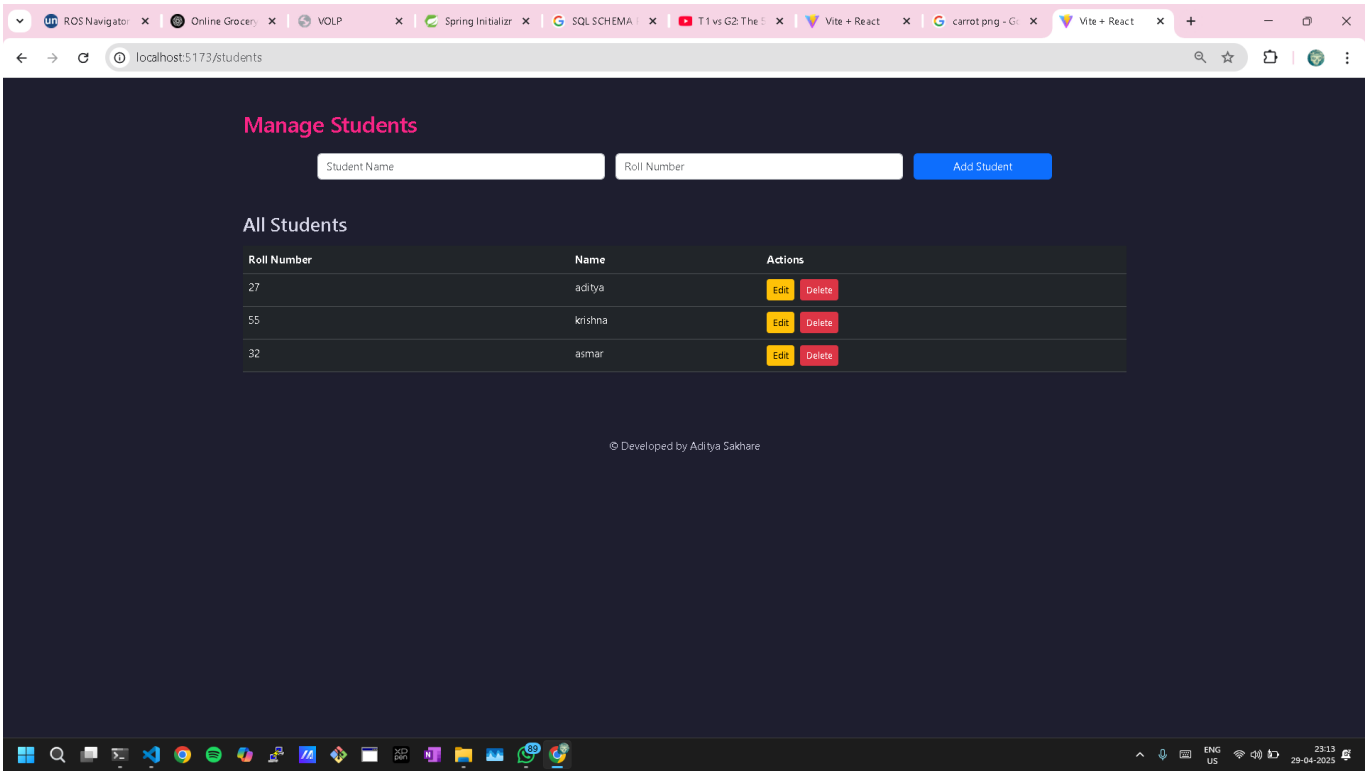
Tables:

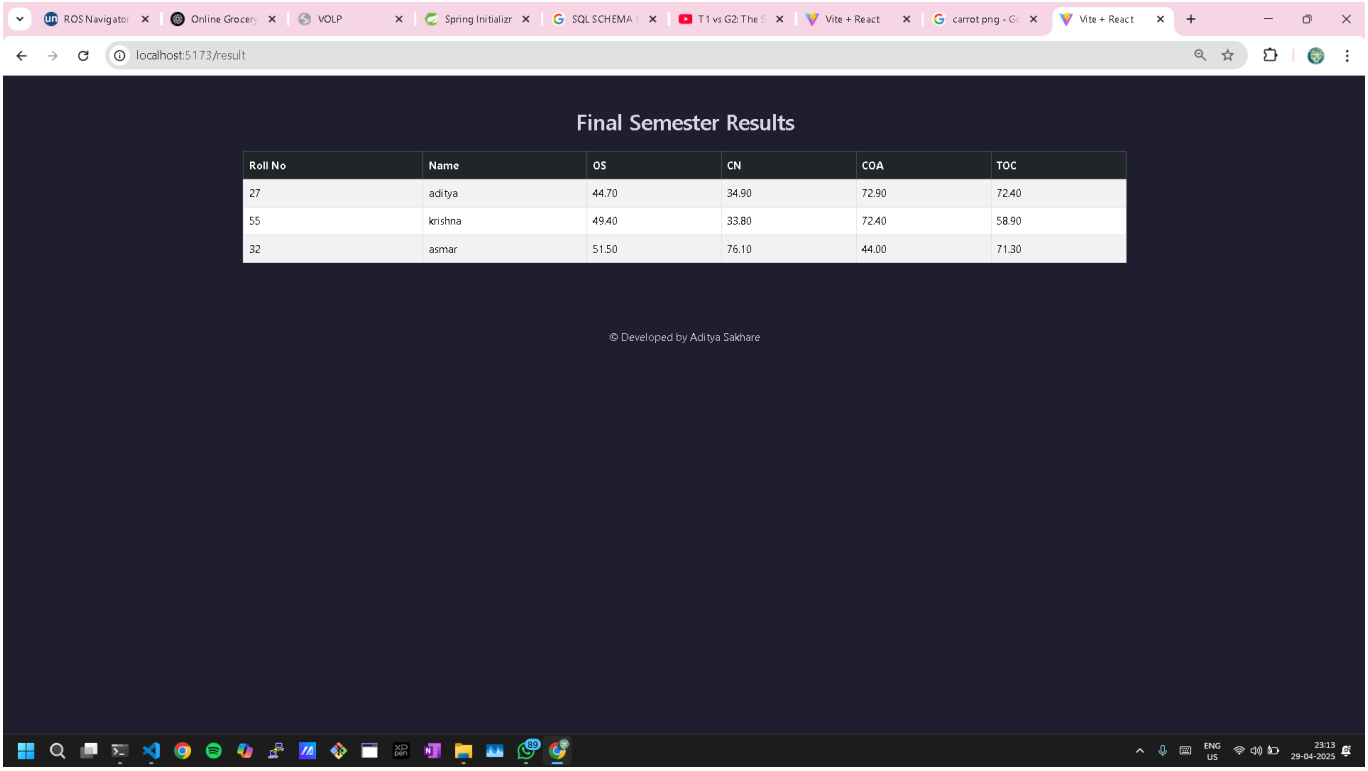
```
mysql> show tables;
+----------------+
| Tables_in_ass4 |
+----------------+
| marks          |
| student        |
+----------------+
2 rows in set (0.01 sec)

mysql>

mysql> select * from student;
+----+---------+---------+
| id | name    | roll_no |
+----+---------+---------+
|  6 | aditya  | 27      |
|  8 | krishna | 55      |
|  9 | asmar   | 32      |
+----+---------+---------+
3 rows in set (0.02 sec)

mysql> select * from marks;
+----+--------+--------+---------+---------+---------+--------+--------+---------
+---------+---------+
| id | cn_ese | cn_mse | coa_ese | coa_mse | name    | os_ese | os_mse | roll_no |
toc_ese | toc_mse |
+----+--------+--------+---------+---------+---------+--------+--------+---------
+---------+---------+
|  1 |     10 |     93 |      90 |      33 | aditya  |     36 |     65 | 27      |
64 |      92 |
|  2 |      8 |     94 |      82 |      50 | krishna |     44 |     62 | 55      |
79 |      12 |
|  3 |     74 |     81 |      29 |      79 | asmar   |     53 |     48 | 32      |
98 |       9 |
+----+--------+--------+---------+---------+---------+--------+--------+---------
+---------+---------+
3 rows in set (0.01 sec)
```

```
mysql>
```

Screenshots:







Codes:

Frontend (React + Vite):

src/App.jsx

```
import { BrowserRouter as Router , Routes, Route } from "react-router-dom"
import './style.css'
import HomePage from "./components/HomePage"
import StudentCRUD from "./components/StudentCRUD"
import MarksCRUD from "./components/MarksCRUD"
import ResultPage from "./components/ResultPage"

function App() {

  return (
    <Router>
    <Routes>
      <Route path="/" element={ <HomePage/>} />
      <Route path="/students" element={<StudentCRUD/>} />
      <Route path="/marks" element={<MarksCRUD/>}/>
      <Route path="/result" element={<ResultPage/>}/>

    </Routes>
    <div className="container text-center mt-5">
    <footer className="footer">
        <p className="text">&copy; Developed by Aditya Sakhare</p>
      </footer>
    </div>
    </Router>
```

```
  )
}

export default App
```

src/componenets/

```
import { Link } from "react-router-dom";
import '../style.css'; // Don't forget to import!

function HomePage() {
  return (
    <div className="container text-center mt-5">
      <h1 className="mb-5 main-heading">VIT Semester Results</h1>

      <div className="d-flex justify-content-center gap-4 flex-wrap">
        <Link to="/students" className="card card-link">
          <div className="card-body">
            <h5 className="card-title">Manage Students</h5>
          </div>
        </Link>

        <Link to="/marks" className="card card-link">
          <div className="card-body">
            <h5 className="card-title">Manage Marks</h5>
          </div>
        </Link>

        <Link to="/result" className="card card-link">
          <div className="card-body">
            <h5 className="card-title">Get Result</h5>
          </div>
        </Link>
      </div>



    </div>
  );
}

export default HomePage;
```

```
import { useEffect, useState } from 'react';
import API from '../api/api';
```

```
function MarksCRUD() {
    const [marksList, setMarksList] = useState([]);
    const [students, setStudents] = useState([]);
    const [selectedStudentRollNo, setSelectedStudentRollNo] = useState('');
    const [marks, setMarks] = useState({
        name: '', // Add name field
        osMse: '',
        osEse: '',
        cnMse: '',
        cnEse: '',
        coaMse: '',
        coaEse: '',
        tocMse: '',
        tocEse: '',
    });

    const [isEditing, setIsEditing] = useState(false); // Flag to track if we are
in edit mode

    function handleDummyFill() {
        setMarks({
            osMse: Math.floor(Math.random() * 101),
            osEse: Math.floor(Math.random() * 101),
            cnMse: Math.floor(Math.random() * 101),
            cnEse: Math.floor(Math.random() * 101),
            coaMse: Math.floor(Math.random() * 101),
            coaEse: Math.floor(Math.random() * 101),
            tocMse: Math.floor(Math.random() * 101),
            tocEse: Math.floor(Math.random() * 101),
        });
    }

    useEffect(() => {
        fetchMarks();
        fetchStudents();
    }, []);


    const fetchMarks = async () => {
        const res = await API.get('/marks');
        setMarksList(res.data);
    };

    const fetchStudents = async () => {
        const res = await API.get('/students');
        setStudents(res.data);
    };

    const handleChange = (e) => {
        setMarks({ ...marks, [e.target.name]: e.target.value });
    };

    const handleSubmit = async (e) => {
```

```
        e.preventDefault();

        // Find the student and get their name
        const student = students.find(s => s.rollNo === selectedStudentRollNo);
        const updatedMarks = { ...marks, name: student?.name };

        if (isEditing) {
            // PUT request to edit marks if we're in edit mode
            await API.put(`/marks/${selectedStudentRollNo}`, updatedMarks);
        } else {
            // POST request to submit new marks with name added
            await API.post('/marks', { rollNo: selectedStudentRollNo,
...updatedMarks });
        }

        resetForm();
        fetchMarks();
    };


    const resetForm = () => {
        setMarks({
            name: '',
            osMse: '',
            osEse: '',
            cnMse: '',
            cnEse: '',
            coaMse: '',
            coaEse: '',
            tocMse: '',
            tocEse: '',
        });
        setSelectedStudentRollNo('');
        setIsEditing(false);
    };

    const handleEdit = (mark) => {
        setSelectedStudentRollNo(mark.rollNo);
        setMarks({
            name: mark.name,
            osMse: mark.osMse,
            osEse: mark.osEse,
            cnMse: mark.cnMse,
            cnEse: mark.cnEse,
            coaMse: mark.coaMse,
            coaEse: mark.coaEse,
            tocMse: mark.tocMse,
            tocEse: mark.tocEse,
        });
        setIsEditing(true);
    };

    return (
        <div className="container mt-5">
```

```
            <h2 className="mb-4 main-heading">Manage Marks</h2>

            <form onSubmit={handleSubmit} className="mb-5">
                <div className="row g-3 justify-content-center">
                    <div className="col-md-4">
                        <select
                            value={selectedStudentRollNo}
                            onChange={(e) =>
setSelectedStudentRollNo(e.target.value)}
                            className="form-select"
                            required
                        >
                            <option value="">Select Student</option>
                            {students.map((student) => (
                                <option key={student.id} value={student.rollNo}>
                                    {student.rollNo} - {student.name}
                                </option>
                            ))}
                        </select>
                    </div>
                </div>

                <div className="row g-3 mt-3 justify-content-center">
                    {/* OS */}
                    <div className="col-md-3">
                        <input
                            type="number"
                            name="osMse"
                            placeholder="OS MSE"
                            value={marks.osMse}
                            onChange={handleChange}
                            className="form-control"
                            required
                        />
                    </div>
                    <div className="col-md-3">
                        <input
                            type="number"
                            name="osEse"
                            placeholder="OS ESE"
                            value={marks.osEse}
                            onChange={handleChange}
                            className="form-control"
                            required
                        />
                    </div>
                </div>
                <div className="row g-3 mt-1 justify-content-center">
                    {/* CN */}
                    <div className="col-md-3">
                        <input
                            type="number"
                            name="cnMse"
                            placeholder="CN MSE"
```

```
                        value={marks.cnMse}
                        onChange={handleChange}
                        className="form-control"
                        required
                    />
                </div>
                <div className="col-md-3">
                    <input
                        type="number"
                        name="cnEse"
                        placeholder="CN ESE"
                        value={marks.cnEse}
                        onChange={handleChange}
                        className="form-control"
                        required
                    />
                </div>
            </div>
            <div className="row g-3 mt-1 justify-content-center">
                {/* COA */}
                <div className="col-md-3">
                    <input
                        type="number"
                        name="coaMse"
                        placeholder="COA MSE"
                        value={marks.coaMse}
                        onChange={handleChange}
                        className="form-control"
                        required
                    />
                </div>
                <div className="col-md-3">
                    <input
                        type="number"
                        name="coaEse"
                        placeholder="COA ESE"
                        value={marks.coaEse}
                        onChange={handleChange}
                        className="form-control"
                        required
                    />
                </div>
            </div>
            <div className="row g-3 mt-1 justify-content-center">
                {/* TOC */}
                <div className="col-md-3">
                    <input
                        type="number"
                        name="tocMse"
                        placeholder="TOC MSE"
                        value={marks.tocMse}
                        onChange={handleChange}
                        className="form-control"
                        required
```

```
                        />
                    </div>
                    <div className="col-md-3">
                        <input
                            type="number"
                            name="tocEse"
                            placeholder="TOC ESE"
                            value={marks.tocEse}
                            onChange={handleChange}
                            className="form-control"
                            required
                        />
                    </div>

                    <div className="col-12 mt-3 d-flex justify-content-center ">
                        <button type="submit" className="btn btn-primary w-25 mx-
2">
                            {isEditing ? 'Update Marks' : 'Submit Marks'}
                        </button>
                        <button type="button" onClick={handleDummyFill}
className="btn btn-secondary w-25 mx-2">
                            Dummy Fill
                        </button>
                    </div>
                </div>
            </form>

            <h3 className="mb-3">All Marks</h3>
            <div className="table-responsive">
                <table className="table table-dark table-hover rounded shadow-sm">
                    <thead>
                        <tr>
                            <th>Student Roll</th>
                            <th>OS (MSE / ESE)</th>
                            <th>CN (MSE / ESE)</th>
                            <th>COA (MSE / ESE)</th>
                            <th>TOC (MSE / ESE)</th>
                            <th>Actions</th>
                        </tr>
                    </thead>
                    <tbody>
                        {marksList.map((mark) => (
                            <tr key={mark.id}>
                                <td>{mark.rollNo}</td>
                                <td>{mark.osMse} / {mark.osEse}</td>
                                <td>{mark.cnMse} / {mark.cnEse}</td>
                                <td>{mark.coaMse} / {mark.coaEse}</td>
                                <td>{mark.tocMse} / {mark.tocEse}</td>
                                <td>
                                    <button onClick={() => handleEdit(mark)}
className="btn btn-warning">
                                        Edit
                                    </button>
                                </td>
```

```
                            </tr>
                        ))}
                    </tbody>
                </table>
            </div>
        </div>
    );
}

export default MarksCRUD;
```

```
import { useEffect, useState } from 'react';
import API from '../api/api';

function ResultPage() {
  const [results, setResults] = useState([]);

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

  const fetchResults = async () => {
    const res = await API.get('/results');
    setResults(res.data);
  };

  const calculateFinalMarks = (mse, ese) => {
    return (mse * 0.3 + ese * 0.7).toFixed(2);
  };

  return (
    <div className="container my-5">
      <h2 className="text-center">Final Semester Results</h2>

      <div className="table-responsive mt-4">
        <table className="table table-striped table-bordered">
          <thead className="table-dark">
            <tr>
              <th>Roll No</th>
              <th>Name</th>
              <th>OS</th>
              <th>CN</th>
              <th>COA</th>
              <th>TOC</th>
            </tr>
          </thead>
          <tbody>
            {results.map((student) => (
              <tr key={student.id}>
                <td>{student.rollNo}</td>
                <td>{student.name}</td>
```

```
                <td>{calculateFinalMarks(student.osMse, student.osEse)}</td>
                <td>{calculateFinalMarks(student.cnMse, student.cnEse)}</td>
                <td>{calculateFinalMarks(student.coaMse, student.coaEse)}</td>
                <td>{calculateFinalMarks(student.tocMse, student.tocEse)}</td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    </div>
  );

}

export default ResultPage;
```

```
import { useState, useEffect } from 'react';
import API from '../api/api';

function StudentCRUD() {
    const [students, setStudents] = useState([]);
    const [name , setName] = useState('');
    const [rollNo , setRollNo] = useState('');
    const [editID , setEditID] = useState(null);


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

    const fetchStudents = async () => {
        const res= await API.get('/students');
        setStudents(res.data);
    }

    const handleSubmit = async (e) => {
        e.preventDefault();

        if(editID){
            await API.put(`/students/${editID}`, {name, rollNo});
        }else{
            await API.post(`/students`, {name, rollNo});
        }

        setName('');
        setRollNo('');
        setEditID(null);
        fetchStudents();


    };
```

```
    const handleEdit = (student) =>{
        setName(student.name);
        setRollNo(student.rollNo);
        setEditID(student.id);

    };

    const handleDelete = async(id) =>{
        await API.delete(`/students/${id}`);
        fetchStudents();
    };

    return (
        <div className="container mt-5">
          <h2 className="mb-4 main-heading">Manage Students</h2>

          <form onSubmit={handleSubmit} className="mb-5">
            <div className="row g-3 justify-content-center">
              <div className="col-md-4">
                <input
                  type="text"
                  className="form-control"
                  placeholder="Student Name"
                  value={name}
                  onChange={(e) => setName(e.target.value)}
                  required
                />
              </div>
              <div className="col-md-4">
                <input
                  type="text"
                  className="form-control"
                  placeholder="Roll Number"
                  value={rollNo}
                  onChange={(e) => setRollNo(e.target.value)}
                  required
                />
              </div>
              <div className="col-md-2">
                <button type="submit" className="btn btn-primary w-100">
                  {editID ? 'Update' : 'Add'} Student
                </button>
              </div>
            </div>
          </form>

          <h3 className="mb-3">All Students</h3>
          <div className="table-responsive">
            <table className="table table-dark table-hover rounded shadow-sm">
              <thead>
                <tr>
                  <th>Roll Number</th>
```

```
                    <th>Name</th>
                    <th>Actions</th>
                  </tr>
                </thead>
                <tbody>
                  {students.map((student) => (
                    <tr key={student.id}>
                      <td>{student.rollNo}</td>
                      <td>{student.name}</td>
                      <td>
                        <button onClick={() => handleEdit(student)} className="btn
btn-sm btn-warning me-2">
                          Edit
                        </button>
                        <button onClick={() => handleDelete(student.id)}
className="btn btn-sm btn-danger">
                          Delete
                        </button>
                      </td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>

          </div>
        );
    }



export default StudentCRUD;
```

src/api/

```
import axios from 'axios';

const API_URL = axios.create({
    baseURL: 'http://localhost:8080/api'
});


export default API_URL;
```

Backend:

backend\backend\src\main\java\com\result\backend\config

```
package com.result.backend.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class WebConfig implements WebMvcConfigurer {
    @Override
    public void addCorsMappings(CorsRegistry registry) {
        // Allow all origins, methods, and headers
        registry.addMapping("/**")
                .allowedOrigins("*")  // Allow all origins
                .allowedMethods("*")  // Allow all HTTP methods
                .allowedHeaders("*");  // Allow all headers
    }
}
```

backend\backend\src\main\java\com\result\backend\controller

```
package com.result.backend.controller;

import com.result.backend.model.Marks;
import com.result.backend.service.MarksService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/api/marks")
public class MarksController {

    @Autowired
    private MarksService marksService;

    @GetMapping
    public List<Marks> getAllMarks() {
        return marksService.getAllMarks();
    }

    @GetMapping("/{id}")
    public ResponseEntity<Marks> getMarksById(@PathVariable Long id) {
        Optional<Marks> marks = marksService.getMarksById(id);
```

```
        return marks.map(ResponseEntity::ok).orElseGet(() ->
ResponseEntity.notFound().build());
    }

    @PostMapping
    public Marks addMarks(@RequestBody Marks marks) {
        return marksService.addMarks(marks);
    }

    @PutMapping("/{rollNo}")
    public ResponseEntity<Marks> updateMarks(@PathVariable String rollNo,
@RequestBody Marks marks) {
        marks.setRollNo(rollNo);;
        return ResponseEntity.ok(marksService.updateMarks(marks));
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteMarks(@PathVariable Long id) {
        marksService.deleteMarks(id);
        return ResponseEntity.noContent().build();
    }
}
```

```
package com.result.backend.controller;

import com.result.backend.model.Marks;
import com.result.backend.service.MarksService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
@RequestMapping("/api/results")
public class ResultController {

    @Autowired
    private MarksService marksService;

    @GetMapping
    public ResponseEntity<List<Marks>> getAllResults() {
        List<Marks> results = marksService.getAllMarks();

        // Assuming the Marks entity contains fields like 'rollNo', 'name', and
marks for the various subjects
        return ResponseEntity.ok(results);
    }
```

```
}
```

```
package com.result.backend.controller;

import com.result.backend.model.Student;
import com.result.backend.service.StudentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/api/students")
public class StudentController {

    @Autowired
    private StudentService studentService;

    @GetMapping
    public List<Student> getAllStudents() {
        return studentService.getAllStudents();
    }

    @GetMapping("/{id}")
    public ResponseEntity<Student> getStudentById(@PathVariable Long id) {
        Optional<Student> student = studentService.getStudentById(id);
        return student.map(ResponseEntity::ok).orElseGet(() ->
ResponseEntity.notFound().build());
    }

    @PostMapping
    public ResponseEntity<Student> addStudent(@RequestBody Student student) {
        System.out.println("Received Student: " + student.getName() + ", " +
student.getRollNo()); // Log the data
        Student savedStudent = studentService.addStudent(student);
        return ResponseEntity.ok(savedStudent);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Student> updateStudent(@PathVariable Long id,
@RequestBody Student student) {
        // Find the student by id
        Optional<Student> existingStudent = studentService.getStudentById(id);

        if (!existingStudent.isPresent()) {
            return ResponseEntity.notFound().build();  // Student not found
        }
```

```
        // Update the student details
        Student studentToUpdate = existingStudent.get();
        studentToUpdate.setName(student.getName());
        studentToUpdate.setRollNo(student.getRollNo());

        // Save the updated student
        studentService.updateStudent(studentToUpdate);

        return ResponseEntity.ok(studentToUpdate);  // Return the updated student
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteStudent(@PathVariable Long id) {
        studentService.deleteStudent(id);
        return ResponseEntity.noContent().build();
    }
}
```

backend\backend\src\main\java\com\result\backend\model

```
package com.result.backend.model;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Column;

@Entity
public class Marks {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "rollNo", nullable = false, unique = true)
    private String rollNo;

    @Column(name = "name", nullable = false)
    private String name;

    // Marks for different subjects
    @Column(name = "osMse")
    private Integer osMse;

    @Column(name = "osEse")
    private Integer osEse;

    @Column(name = "cnMse")
    private Integer cnMse;
```

```
    @Column(name = "cnEse")
    private Integer cnEse;

    @Column(name = "coaMse")
    private Integer coaMse;

    @Column(name = "coaEse")
    private Integer coaEse;

    @Column(name = "tocMse")
    private Integer tocMse;

    @Column(name = "tocEse")
    private Integer tocEse;

    // Getters and Setters
    public Long getId() {
        return id;
    }

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

    public String getRollNo() {
        return rollNo;
    }

    public void setRollNo(String rollNo) {
        this.rollNo = rollNo;
    }

    public String getName() {
        return name;
    }

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

    public Integer getOsMse() {
        return osMse;
    }

    public void setOsMse(Integer osMse) {
        this.osMse = osMse;
    }

    public Integer getOsEse() {
        return osEse;
    }

    public void setOsEse(Integer osEse) {
```

```
        this.osEse = osEse;
    }

    public Integer getCnMse() {
        return cnMse;
    }

    public void setCnMse(Integer cnMse) {
        this.cnMse = cnMse;
    }

    public Integer getCnEse() {
        return cnEse;
    }

    public void setCnEse(Integer cnEse) {
        this.cnEse = cnEse;
    }

    public Integer getCoaMse() {
        return coaMse;
    }

    public void setCoaMse(Integer coaMse) {
        this.coaMse = coaMse;
    }

    public Integer getCoaEse() {
        return coaEse;
    }

    public void setCoaEse(Integer coaEse) {
        this.coaEse = coaEse;
    }

    public Integer getTocMse() {
        return tocMse;
    }

    public void setTocMse(Integer tocMse) {
        this.tocMse = tocMse;
    }

    public Integer getTocEse() {
        return tocEse;
    }

    public void setTocEse(Integer tocEse) {
        this.tocEse = tocEse;
    }
}
```

```
package com.result.backend.model;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Column;

@Entity
public class Student {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "rollNo", nullable = false, unique = true)
    private String rollNo;

    @Column(name = "name", nullable = false)
    private String name;

    // Getters and Setters
    public Long getId() {
        return id;
    }

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

    public String getRollNo() {
        return rollNo;  // Use rollNo here instead of rollNumber
    }

    public void setRollNo(String rollNo) {  // Use rollNo here instead of
rollNumber
        this.rollNo = rollNo;
    }

    public String getName() {
        return name;
    }

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

backend\backend\src\main\java\com\result\backend\repository

```
package com.result.backend.repository;

import com.result.backend.model.Marks;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.Optional;

public interface MarksRepository extends JpaRepository<Marks, Long> {
    Optional<Marks> findByRollNo(String rollNo);

}
```

```
package com.result.backend.repository;

import com.result.backend.model.Student;
import org.springframework.data.jpa.repository.JpaRepository;

public interface StudentRepository extends JpaRepository<Student, Long> {
}
```

backend\backend\src\main\java\com\result\backend\service

```
package com.result.backend.service;

import com.result.backend.model.Marks;
import com.result.backend.repository.MarksRepository;

import jakarta.persistence.EntityNotFoundException;

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

import java.util.List;
import java.util.Optional;

@Service
public class MarksService {

    @Autowired
    private MarksRepository marksRepository;

    public List<Marks> getAllMarks() {
        return marksRepository.findAll();
    }

    public Optional<Marks> getMarksById(Long id) {
        return marksRepository.findById(id);
```

```
    }

    public Marks addMarks(Marks marks) {
        return marksRepository.save(marks);
    }

    public Marks updateMarks(Marks marks) {
    // Check if the Marks entry with the provided rollNo exists
        Optional<Marks> existingMarks =
marksRepository.findByRollNo(marks.getRollNo()); // or findById if using ID
        if (existingMarks.isPresent()) {
            Marks updatedMarks = existingMarks.get();

            // Only update fields that are present in the request
            updatedMarks.setName(marks.getName());
            updatedMarks.setCnEse(marks.getCnEse());
            updatedMarks.setCnMse(marks.getCnMse());
            updatedMarks.setCoaEse(marks.getCoaEse());
            updatedMarks.setCoaMse(marks.getCoaMse());
            updatedMarks.setOsEse(marks.getOsEse());
            updatedMarks.setOsMse(marks.getOsMse());
            updatedMarks.setTocEse(marks.getTocEse());
            updatedMarks.setTocMse(marks.getTocMse());

            // Return the updated entity
            return marksRepository.save(updatedMarks); // This should perform an
update now
        } else {
            // If no matching entry found, return null or throw exception
            throw new EntityNotFoundException("Marks not found for roll number: "
+ marks.getRollNo());
        }
    }


    public void deleteMarks(Long id) {
        marksRepository.deleteById(id);
    }
}
```

```
package com.result.backend.service;

import org.springframework.stereotype.Service;

@Service
public class ResultService {

    public double calculateFinalMarks(double mse, double ese) {
        return (mse * 0.30) + (ese * 0.70);
    }
```

```
}
```

```
package com.result.backend.service;

import com.result.backend.model.Student;
import com.result.backend.repository.StudentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class StudentService {

    @Autowired
    private StudentRepository studentRepository;

    public List<Student> getAllStudents() {
        return studentRepository.findAll();
    }

    public Optional<Student> getStudentById(Long id) {
        return studentRepository.findById(id);
    }

    public Student addStudent(Student student) {
        return studentRepository.save(student);
    }

    public Student updateStudent(Student student) {
        return studentRepository.save(student);
    }

    public void deleteStudent(Long id) {
        studentRepository.deleteById(id);
    }
}
```