**Objective:-** Develop a web application which involves database operations using NodeJS

**Code :-**

**Frontend :-**

**App.jsx**

import { useState, useEffect } from 'react';

import axios from 'axios';

import AddMarks from './components/AddMarks';

import MarksTable from './components/MarksTable';

import './App.css';

const API\_URL = 'http://localhost:3000/api/students';

const App = () => {

  const [marks, setMarks] = useState([]);

  const [editingMark, setEditingMark] = useState(**null**);

  useEffect(() => {

    fetchMarks();

  }, []);

  const fetchMarks = async () => {

    try {

      const response = await axios.get(API\_URL);

      setMarks(response.data);

    } catch (error) {

      console.error('Error fetching marks:', error);

    }

  };

  const addMark = async (*newMark*) => {

    const total = Object.values(newMark.subjects).reduce(

      (*acc*, *val*) => acc + parseInt(val || 0),

      0

    );

    const percentage = ((total / 600) \* 100).toFixed(2);

    if (editingMark) {

      const updatedMark = { ...newMark, total, percentage, id: editingMark.id };

      setMarks((*prevMarks*) =>

        prevMarks.map((*mark*) =>

          mark.id === editingMark.id ? updatedMark : mark

        )

      );

      try {

        await axios.put(`${API\_URL}/${editingMark.id}`, updatedMark);

        setEditingMark(**null**);

        fetchMarks();

      } catch (error) {

        console.error('Error updating mark:', error);

      }

    } else {

*// Add new mark*

      const tempId = Date.now();

      const newStudent = {

        ...newMark,

        total,

        percentage,

        id: tempId

      };

*// Update UI*

      setMarks((*prevMarks*) => [...prevMarks, newStudent]);

      try {

        await axios.post(API\_URL, {

          ...newMark,

          total,

          percentage

        });

        fetchMarks();

      } catch (error) {

        console.error('Error adding mark:', error);

      }

    }

  };

  const handleEdit = (*mark*) => {

    setEditingMark(mark);

  };

  const deleteMark = async (*id*) => {

    try {

      await axios.delete(`${API\_URL}/${id}`);

      setMarks((*prevMarks*) => prevMarks.filter((*mark*) => mark.id !== id));

      fetchMarks();

    } catch (error) {

      console.error('Error deleting mark:', error);

    }

  };

  return (

    <div className="container">

      <h1>Student Result Management</h1>

      <AddMarks onAdd={addMark} editingMark={editingMark} />

      <MarksTable marks={marks} onEdit={handleEdit} onDelete={deleteMark} />

    </div>

  );

};

export default App;

**App.css**

*@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600&display=swap');*

*\* {*

*box-sizing: border-box;*

*margin: 0;*

*padding: 0;*

*}*

*body {*

*font-family: 'Poppins', sans-serif;*

*background: linear-gradient(135deg, #74EBD5, #ACB6E5);*

*color: #333;*

*min-height: 100vh;*

*display: flex;*

*justify-content: center;*

*align-items: center;*

*padding: 20px;*

*}*

*.container {*

*background: #fff;*

*box-shadow: 0 10px 20px rgba(0, 0, 0, 0.1);*

*border-radius: 12px;*

*max-width: 1200px;*

*width: 100%;*

*padding: 40px;*

*animation: fadeIn 0.8s ease-in-out;*

*transition: all 0.3s;*

*}*

*h1 {*

*color: #4CAF50;*

*font-size: 36px;*

*margin-bottom: 20px;*

*text-align: center;*

*}*

*form {*

*display: grid;*

*grid-template-columns: repeat(4, 1fr);*

*gap: 20px;*

*padding: 20px 0;*

*}*

*form input {*

*padding: 12px;*

*border: 1px solid #ccc;*

*border-radius: 8px;*

*font-size: 16px;*

*color: #333;*

*transition: all 0.3s;*

*}*

*form input:focus {*

*outline: none;*

*border-color: #4CAF50;*

*box-shadow: 0 0 8px rgba(76, 175, 80, 0.5);*

*}*

*form button {*

*grid-column: span 4;*

*padding: 14px;*

*background: #4CAF50;*

*color: white;*

*border: none;*

*border-radius: 8px;*

*font-size: 16px;*

*cursor: pointer;*

*transition: background 0.3s, transform 0.2s;*

*}*

*form button:hover {*

*background: #45a049;*

*transform: scale(1.05);*

*}*

*form button:active {*

*transform: scale(0.95);*

*}*

*table {*

*width: 100%;*

*border-collapse: collapse;*

*margin-top: 20px;*

*}*

*thead {*

*background: #4CAF50;*

*color: white;*

*}*

*th, td {*

*padding: 10px;*

*text-align: center;*

*border-bottom: 1px solid #ddd;*

*font-size: 16px;*

*}*

*tbody tr:hover {*

*background: #f1f1f1;*

*}*

*tbody tr:nth-child(even) {*

*background: #f9f9f9;*

*}*

*button {*

*cursor: pointer;*

*transition: all 0.3s ease;*

*}*

*.edit-btn {*

*background: #2196F3;*

*color: white;*

*border: none;*

*padding: 10px 20px;*

*border-radius: 8px;*

*font-size: 14px;*

*margin-right: 5px;*

*transition: 0.3s;*

*margin-bottom: 2px;*

*width: 90px;*

*}*

*.edit-btn:hover {*

*background: #1976D2;*

*transform: scale(1.05);*

*}*

*.delete-btn {*

*background: #f44336;*

*color: white;*

*border: none;*

*padding: 10px 20px;*

*border-radius: 8px;*

*font-size: 14px;*

*transition: 0.3s;*

*width: 90px;*

*}*

*.delete-btn:hover {*

*background: #d32f2f;*

*transform: scale(1.05);*

*}*

*@keyframes fadeIn {*

*from {*

*opacity: 0;*

*transform: translateY(-30px);*

*}*

*to {*

*opacity: 1;*

*transform: translateY(0);*

*}*

*}*

*@media (max-width: 1024px) {*

*form {*

*grid-template-columns: repeat(2, 1fr);*

*}*

*form button {*

*grid-column: span 2;*

*}*

*th, td {*

*font-size: 14px;*

*padding: 12px;*

*}*

*}*

*@media (max-width: 768px) {*

*form {*

*grid-template-columns: 1fr;*

*}*

*form button {*

*grid-column: span 1;*

*}*

*th, td {*

*font-size: 12px;*

*padding: 10px;*

*}*

*.container {*

*padding: 20px;*

*}*

*h1 {*

*font-size: 28px;*

*}*

*}*

**Components/AddMarks.jsx**

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

import axios from 'axios';

const AddMarks = ({ *editingMark*, *onSuccess* }) => {

  const [rollNumber, setRollNumber] = useState('');

  const [name, setName] = useState('');

  const [subjects, setSubjects] = useState({

    math: '',

    science: '',

    english: '',

    geography: '',

    computer: '',

    history: ''

  });

*// Pre-fill form with existing data during editing*

  useEffect(() => {

    if (editingMark) {

      setRollNumber(editingMark.roll\_no);

      setName(editingMark.name);

      setSubjects({

        math: editingMark.math,

        science: editingMark.science,

        english: editingMark.english,

        geography: editingMark.geography,

        computer: editingMark.computer,

        history: editingMark.history

      });

    } else {

*// Reset the form when not editing*

      setRollNumber('');

      setName('');

      setSubjects({

        math: '',

        science: '',

        english: '',

        geography: '',

        computer: '',

        history: ''

      });

    }

  }, [editingMark]);

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

*// e.preventDefault();*

    const studentData = {

      roll\_no: rollNumber,

      name: name,

      math: parseInt(subjects.math),

      science: parseInt(subjects.science),

      english: parseInt(subjects.english),

      geography: parseInt(subjects.geography),

      computer: parseInt(subjects.computer),

      history: parseInt(subjects.history)

    };

    try {

      if (editingMark) {

*// Update existing student*

        await axios.put(`http://localhost:3000/api/students/${editingMark.id}`, studentData);

        alert('Student updated successfully!');

      } else {

*// Add new student*

        await axios.post('http://localhost:3000/api/students', studentData);

        alert('Student added successfully!');

      }

      onSuccess(); *// Refresh the list after submission*

*// Clear form after submission*

      setRollNumber('');

      setName('');

      setSubjects({

        math: '',

        science: '',

        english: '',

        geography: '',

        computer: '' ,

        history: ''

      });

    } catch (error) {

      console.error('Error adding/updating student:', error);

      alert('Failed to add/update student');

    }

  };

  const handleChange = (*e*) => {

    setSubjects({ ...subjects, [e.target.name]: e.target.value });

  };

  return (

    <form onSubmit={handleSubmit}>

      <input

        type="text"

        placeholder="Student Name"

        value={name}

        onChange={(*e*) => setName(e.target.value)}

        required

      />

      <input

        type="text"

        placeholder="Roll Number"

        value={rollNumber}

        onChange={(*e*) => setRollNumber(e.target.value)}

        required

      />

      {Object.keys(subjects).map((*subject*) => (

        <input

          key={subject}

          name={subject}

          type="number"

          placeholder={`${subject.charAt(0).toUpperCase() + subject.slice(1)} Marks`}

          value={subjects[subject]}

          onChange={handleChange}

          required

        />

      ))}

      <button type="submit">

        {editingMark ? 'Update Marks' : 'Add Marks'}

      </button>

    </form>

  );

};

export default AddMarks;

**Components/MarksTable.jsx**

import React from 'react';

const MarksTable = ({ *marks*, *onEdit*, *onDelete* }) => {

  if (!marks || !Array.isArray(marks) || marks.length === 0) {

    return <p>No student marks available.</p>;

  }

  return (

    <div>

      <h2>Student Marks</h2>

      <table>

        <thead>

          <tr>

            <th>Roll No</th>

            <th>Name</th>

            <th>Math</th>

            <th>Science</th>

            <th>English</th>

            <th>Geography</th>

            <th>Computer</th>

            <th>History</th>

            <th>Total</th>

            <th>Percentage</th>

            <th>Actions</th>

          </tr>

        </thead>

        <tbody>

          {marks.map((*mark*) => (

            <tr key={mark.id}>

              <td>{mark.roll\_no}</td>

              <td>{mark.name}</td>

              <td>{mark.math}</td>

              <td>{mark.science}</td>

              <td>{mark.english}</td>

              <td>{mark.geography}</td>

              <td>{mark.computer}</td>

              <td>{mark.history}</td>

              <td>{(mark.math + mark.science + mark.english + mark.history + mark.geography + mark.computer)}</td>

              <td>{(mark.math + mark.science + mark.english + mark.history + mark.geography + mark.computer) \* 100 / 600}%</td>

              <td>

                <button

                  className="edit-btn"

                  onClick={() => onEdit(mark)}

                >

                  Edit

                </button>

                <button

                  className="delete-btn"

                  onClick={() => onDelete(mark.id)}

                >

                  Delete

                </button>

              </td>

            </tr>

          ))}

        </tbody>

      </table>

    </div>

  );

};

export default MarksTable;

**Server API :-**

**Index.js :-**

const express = require('express');

const bodyParser = require('body-parser');

const cors = require('cors');

const markRoutes = require('./src/marks/marks.routes');

const db = require('./src/config/db.config');

const app = express();

const PORT = 3000;

app.use(cors());

app.use(bodyParser.json());

app.use('/api/students', markRoutes);

app.get('/', (*req*, *res*) => {

  res.send('Student Result Management System API');

});

app.listen(PORT, () => {

  console.log(`Server running at  http://localhost:${PORT}`);

});

**Config/db.config.js :-**

const mysql = require('mysql2');

const db = mysql.createConnection({

  host: 'localhost',

  user: 'root',

  password: '',

  database: 'student\_results'

});

db.connect((*err*) => {

  if (err) {

    console.error('Database connection failed: ' + err.stack);

    return;

  }

  console.log('Connected to MySQL database.');

});

module.exports = db;

**marks/marks.controller.js**

const studentService = require('./marks.services');

exports.createStudent = (*req*, *res*) => {

  const student = req.body;

  studentService.createStudent(student, (*err*, *result*) => {

    if (err) {

      res.status(500).send({ message: 'Error creating student', error: err });

    } else {

      res.status(201).send({ message: 'Student created', data: result });

    }

  });

};

exports.getAllStudents = (*req*, *res*) => {

  studentService.getAllStudents((*err*, *results*) => {

    if (err) {

      res.status(500).send({ message: 'Error fetching students', error: err });

    } else {

      res.status(200).send(results);

    }

  });

};

exports.getStudentById = (*req*, *res*) => {

  const id = req.params.id;

  studentService.getStudentById(id, (*err*, *result*) => {

    if (err) {

      res.status(500).send({ message: 'Error fetching student', error: err });

    } else {

      res.status(200).send(result[0]);

    }

  });

};

exports.updateStudent = (*req*, *res*) => {

  const id = req.params.id;

  const student = req.body;

  studentService.updateStudent(id, student, (*err*, *result*) => {

    if (err) {

      res.status(500).send({ message: 'Error updating student', error: err });

    } else {

      res.status(200).send({ message: 'Student updated successfully' });

    }

  });

};

exports.deleteStudent = (*req*, *res*) => {

  const id = req.params.id;

  studentService.deleteStudent(id, (*err*, *result*) => {

    if (err) {

      res.status(500).send({ message: 'Error deleting student', error: err });

    } else {

      res.status(200).send({ message: 'Student deleted successfully' });

    }

  });

};

**marks/marks.model.js**

const db = require('../config/db.config');

class Student {

  static create(*student*, *callback*) {

    const query = `

      INSERT INTO students (roll\_no, name, math, science, english, geography, computer , history)

      VALUES (?, ?, ?, ?, ?, ?, ?, ?)`;

    db.query(query, [student.roll\_no, student.name, student.math, student.science, student.english, student.geography, student.computer , student.history], callback);

  }

  static findAll(*callback*) {

    const query = 'SELECT \* FROM students';

    db.query(query, callback);

  }

  static findById(*id*, *callback*) {

    const query = 'SELECT \* FROM students WHERE id = ?';

    db.query(query, [id], callback);

  }

  static update(*id*, *student*, *callback*) {

    const query = `

      UPDATE students

      SET roll\_no = ?, name = ?, math = ?, science = ?, english = ?, geography = ?, computer = ?, history = ?

      WHERE id = ?`;

    db.query(query, [student.roll\_no, student.name, student.math, student.science, student.english, student.geography, student.computer, student.history , id], callback);

  }

  static delete(*id*, *callback*) {

    const query = 'DELETE FROM students WHERE id = ?';

    db.query(query, [id], callback);

  }

}

module.exports = Student;

**marks/marks.routes.js**

const express = require('express');

const router = express.Router();

const markController = require('./marks.controller');

router.post('/', markController.createStudent);

router.get('/', markController.getAllStudents);

router.get('/:id', markController.getStudentById);

router.put('/:id', markController.updateStudent);

router.delete('/:id', markController.deleteStudent);

module.exports = router;

**marks/marks.services.js**

const Student = require('./marks.model');

const createStudent = (*data*, *callback*) => {

  Student.create(data, callback);

};

const getAllStudents = (*callback*) => {

  Student.findAll(callback);

};

const getStudentById = (*id*, *callback*) => {

  Student.findById(id, callback);

};

const updateStudent = (*id*, *data*, *callback*) => {

  Student.update(id, data, callback);

};

const deleteStudent = (*id*, *callback*) => {

  Student.delete(id, callback);

};

module.exports = {

  createStudent,

  getAllStudents,

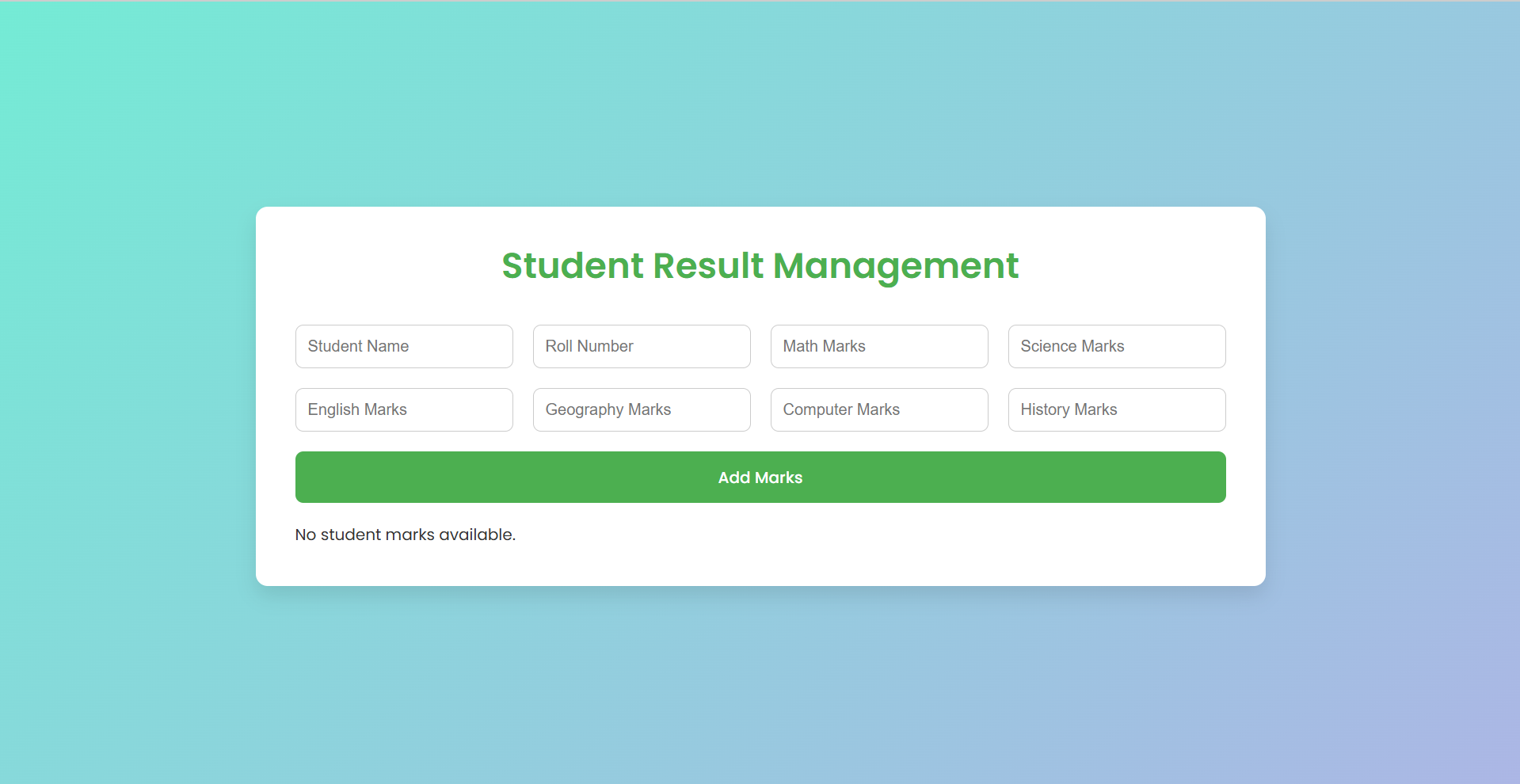
  getStudentById,

  updateStudent,

  deleteStudent

};

**Output :-**

****

