**Week 4 – Day 6 Report** **Date:** 04-11-2025

**Module:** Full Stack Project – Student Management System

## 1. Objective of the Day

The goal for Day 6 was to **add Edit (Update) functionality** to the Student Management System and enhance the **frontend UI** with **Bootstrap modal popups** and **React Router navigation**.
This stage focused on providing a **complete interactive experience** — the ability to modify existing records from the frontend while keeping the backend connected to SQL Server via ASP.NET Core Web API.

## 2. Tasks Completed

1. Added **EditStudentForm** component using a Bootstrap modal.
2. Updated the **StudentList** component to include **Edit** and **Delete** buttons.
3. Implemented **PUT request** to the backend API for editing student records.
4. Installed and configured **React Router DOM** for smooth page navigation.
5. Tested all operations (GET, POST, PUT, DELETE) from both Swagger and React UI.

## 3. Folder Structure

```
student-management-app/
├── src/
│   ├── components/
│   │   ├── StudentList.js
│   │   ├── AddStudentForm.js
│   │   ├── EditStudentForm.js
│   ├── services/
│   │   └── api.js
│   ├── App.js
│   ├── index.js
```

**4. Code Implemented**

**EditStudentForm.js**

```
import React, { useState, useEffect } from "react";
import API_BASE_URL from "../services/api";

function EditStudentForm({ student, onClose, onUpdate }) {
  const [form, setForm] = useState(student);

  useEffect(() => {
    setForm(student);
  }, [student]);

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

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const response = await fetch(`${API_BASE_URL}/${form.id}`, {
        method: "PUT",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(form),
      });

      if (response.ok) {
        alert("Student updated successfully!");
        onUpdate();
        onClose();
      } else {
        alert("Error updating student.");
      }
    } catch (error) {
      console.error("Error updating student:", error);
    }
  };

  if (!student) return null;

  return (
    <div className="modal show" style={{ display: "block" }}>
      <div className="modal-dialog">
        <div className="modal-content">
          <div className="modal-header">
            <h5 className="modal-title">Edit Student</h5>
            <button type="button" className="btn-close" onClick={onClose}></button>
          </div>
          <div className="modal-body">
            <form onSubmit={handleSubmit}>
              <input
                className="form-control mb-2"
                name="name"
                value={form.name}
                onChange={handleChange}
                placeholder="Name"
                required
              />
              <input
                className="form-control mb-2"
                name="age"
                value={form.age}
                onChange={handleChange}
                placeholder="Age"
                required
              />
              <input
                className="form-control mb-2"
                name="grade"
                value={form.grade}
                onChange={handleChange}
                placeholder="Grade"
                required
              />
              <input
                className="form-control mb-2"
                name="courseId"
                value={form.courseId}
                onChange={handleChange}
                placeholder="Course ID"
                required
              />
              <button className="btn btn-primary" type="submit">
                Save Changes
              </button>
            </form>
          </div>
        </div>
```

```
      </div>
    </div>
  );
}

export default EditStudentForm;
```

**StudentList.js (Updated)**

```
import React, { useEffect, useState } from "react";
import API_BASE_URL from "../services/api";
import EditStudentForm from "./EditStudentForm";

function StudentList({ refresh }) {
  const [students, setStudents] = useState([]);
  const [selectedStudent, setSelectedStudent] = useState(null);

  const fetchStudents = async () => {
    const response = await fetch(API_BASE_URL);
    const data = await response.json();
    setStudents(data);
  };

  const deleteStudent = async (id) => {
    if (window.confirm("Are you sure you want to delete this student?")) {
      await fetch(`${API_BASE_URL}/${id}`, { method: "DELETE" });
      alert("Student deleted successfully.");
      fetchStudents();
    }
  };
```

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

return (
  <div>
    <h4 className="mt-4 mb-3">Student List</h4>
    <table className="table table-bordered table-striped">
      <thead className="table-dark">
        <tr>
          <th>ID</th>
          <th>Name</th>
          <th>Age</th>
          <th>Grade</th>
          <th>Course ID</th>
          <th>Actions</th>
        </tr>
      </thead>
      <tbody>
        {students.length > 0 ? (
          students.map((student) => (
            <tr key={student.id}>
              <td>{student.id}</td>
              <td>{student.name}</td>
              <td>{student.age}</td>
              <td>{student.grade}</td>
              <td>{student.courseId}</td>
              <td>
          <button
            className="btn btn-warning btn-sm me-2"
            onClick={() => setSelectedStudent(student)}
          >
            Edit
          </button>
          <button
            className="btn btn-danger btn-sm"
            onClick={() => deleteStudent(student.id)}
          >
            Delete
          </button>
        </td>
      </tr>
    ))
  ) : (
    <tr>
      <td colSpan="6" className="text-center">
        No records found.
      </td>
    </tr>
  )}
 </tbody>
</table>

{selectedStudent && (
 <EditStudentForm
  student={selectedStudent}
  onClose={() => setSelectedStudent(null)}
```

```
          onUpdate={fetchStudents}
        />
      )}
    </div>
  );
}

export default StudentList;
```

## 5. Output Verification

**Tested functionalities:**

- **Update existing student** using Edit modal.
- **Verify database update** in SQL Server.
- **Confirm UI refresh** after successful update.
- **Navigation:** Tested page reload, CORS setup, and component rendering.

## 6. Challenges Faced

- Modal re-rendering issue after saving changes. Solved by using state cleanup (setSelectedStudent(null)).
- Ensured CORS policies remained active in ASP.NET Core backend.
- Small UI alignment issue fixed using Bootstrap spacing utilities.