Week 4 – Day 7 Report **Date:** 05-11-2025
Module: Full Stack Project – Student Management System

1. Objective of the Day

The primary objective of Day 7 was to **finalize, test, and document** the complete Full Stack Student Management System project.
This included adding **error handling, loading indicators, final UI polishing**, and preparing all **project documentation and deliverables** for submission.

By the end of the day, the project achieved a fully functional **CRUD (Create, Read, Update, Delete)** application using ASP.NET Core Web API, SQL Server, and ReactJS frontend.

2. Tasks Completed

1. Integrated **loading states** and **error messages** in the React frontend.
2. Tested all CRUD API endpoints using **Swagger UI** and **Postman**.
3. Performed **end-to-end testing** to ensure smooth backend–frontend communication.
4. Improved the **UI layout** with consistent Bootstrap styling.
5. Verified **SQL stored procedures** and confirmed data integrity.
6. Prepared **final Word documentation** and **project snapshot folder**.
7. Generated **final project PDF** with complete screenshots and explanation.

3. Final Enhancements in Code

App.js (Final)

```
import React, { useState } from "react";
import "bootstrap/dist/css/bootstrap.min.css";
import StudentList from "./components/StudentList";
import AddStudentForm from "./components/AddStudentForm";

function App() {
  const [refresh, setRefresh] = useState(false);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState("");
```

```
  const refreshData = () => setRefresh(!refresh);

  return (
    <div className="container mt-4">
      <h2 className="text-center text-primary mb-4">Student Management System</h2>
      {error && <div className="alert alert-danger">{error}</div>}
      {loading && <div className="alert alert-info">Loading data...</div>}

      <AddStudentForm onSuccess={refreshData} />
      <StudentList refresh={refresh} />
    </div>
  );
}

export default App;
```

Backend Improvement (Error Handling Example)

```
[HttpPost]
public IActionResult AddStudent([FromBody] Student student)
{
    if (student == null)
        return BadRequest("Invalid student data.");

    try
    {
        _context.Database.ExecuteSqlRaw(
            "EXEC sp_AddStudent @Name={0}, @Age={1}, @Grade={2}, @CourseId={3}",
            student.Name, student.Age, student.Grade, student.CourseId);
```

```
        return Ok(new { message = "Student added successfully!" });
    }
    catch (Exception ex)
    {
        return StatusCode(500, $"Server Error: {ex.Message}");
    }
}
```

4. Output Testing

Frontend:

- Verified that the React UI dynamically updates after adding, editing, or deleting a student.
- Displayed success and error messages using Bootstrap alerts.

Backend:

- Verified all stored procedures execute correctly from SQL Server Management Studio (SSMS).
- Tested API routes in Swagger UI (GET, POST, PUT, DELETE).

End-to-End Test Result:
All CRUD operations are functional and synchronized across React frontend and SQL Server backend.

5. Challenges Faced

- **CORS policy issue** between backend and frontend — resolved by enabling AllowAnyOrigin() in Program.cs.
- **Form validation** errors when empty fields were submitted — fixed by adding required validation.
- **API response delay** due to missing await in fetch calls — resolved by using async/await properly.
- **UI refresh problem** after deletion — solved using useEffect() state re-render trigger.

6. Learning Outcomes

1. Understood **integration of ASP.NET Core Web API with ReactJS frontend**.
2. Learned how to **consume REST APIs** using fetch() and display dynamic data.
3. Enhanced knowledge of **SQL stored procedures** and how to connect them with Entity Framework Core.
4. Practiced **modular project structuring** and **code reusability** for scalable applications.
5. Gained confidence in developing **end-to-end full stack projects** independently.

7. Final Deliverables

1. Complete working full-stack Student Management System project.
2. Documentation with all 7 daily progress reports.
3. Snapshot folder containing step-by-step project visuals.
4. SQL script and database backup file.
5. Final project PDF for submission to HR.

8. Conclusion

The **Student Management System** full-stack project successfully demonstrates end-to-end functionality using modern development tools — **ASP.NET Core (C#)**, **SQL Server**, and **ReactJS**. The project fulfills all objectives of Week 4 by showcasing CRUD operations, API integration, and front-end interactivity. This marks the completion of the 4-week training and project implementation phase under Dhruv Compusoft Consultancy's .NET training module.