Week 4 – Day 1 Report **Date:** 30-10-2025

Module: Full Stack Project – Student Management System

1. Objective of the Day

The goal of Day 1 was to begin the setup for the **Full Stack Student Management System project**. This includes planning the project structure, creating the database in SQL Server, and initializing the backend project in ASP.NET Core Web API. The aim was to lay the foundation for connecting the backend and database in the following days.

2. Topics Covered / Tasks Completed

a. Project Planning and Folder Setup

- Created the main folder: Week4_Project
- Inside it, added the subfolder StudentManagementSystem to hold backend, frontend, and SQL files
- Created the folder structure for daily reports (Day-1 to Day-7)

b. Database Design (SQL Server)

- Designed a database named StudentCourseDB
- Identified the required tables and relationships:
    - **Students** table for student details
    - **Courses** table for storing available courses
- Decided to use **stored procedures** for all CRUD operations

Database Schema

```
CREATE DATABASE StudentCourseDB;
GO

USE StudentCourseDB;
GO

CREATE TABLE Courses (
   CourseId INT IDENTITY(1,1) PRIMARY KEY,
   CourseName NVARCHAR(100) NOT NULL
);
```

```
CREATE TABLE Students (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    Name NVARCHAR(100) NOT NULL,
    Age INT CHECK (Age BETWEEN 1 AND 100),
    Grade NVARCHAR(10),
    CourseId INT FOREIGN KEY REFERENCES Courses(CourseId)
);
```

Inserted Sample Data

```
INSERT INTO Courses (CourseName) VALUES
('C# Fundamentals'),
('ASP.NET Core Development'),
('ReactJS Frontend'),
('SQL Server Management');


INSERT INTO Students (Name, Age, Grade, CourseId) VALUES
('Udaya Kumar Shetty', 22, 'A', 1),
('Prajwal M', 23, 'B', 2),
('Sneha R', 21, 'A', 3);
```

3. Backend Setup (ASP.NET Core Web API)

Steps:

1. Opened **Visual Studio 2022**
2. Created a new project → “ASP.NET Core Web API”
3. Named it **StudentApi**
4. Set framework to **.NET 9.0**
5. Verified that the project runs successfully with the default Swagger UI

Project Configuration Done

- Configured connection string in appsettings.json
- Added folder structure:
    - Models
    - Controllers
    - Data
- Installed required NuGet packages:
    - Microsoft.EntityFrameworkCore.SqlServer
    - Microsoft.EntityFrameworkCore.Tools

Connection String Example

"ConnectionStrings": {

"DefaultConnection":
"Server=YOUR_SERVER_NAME;Database=StudentManagementDB;Trusted_Connection=True;TrustServerCertificate=True;"

}

4. Stored Procedures Created

```
CREATE PROCEDURE sp_GetStudents
AS
BEGIN
    SELECT s.Id, s.Name, s.Age, s.Grade, c.CourseName, s.CourseId
    FROM Students s
    JOIN Courses c ON s.CourseId = c.CourseId;
END;

CREATE PROCEDURE sp_AddStudent
    @Name NVARCHAR(100),
    @Age INT,
```

```
    @Grade NVARCHAR(10),
    @CourseId INT
AS
BEGIN
    INSERT INTO Students (Name, Age, Grade, CourseId)
    VALUES (@Name, @Age, @Grade, @CourseId);
END;
```

5. Testing and Validation

- Tested database using **SQL Server Management Studio (SSMS)**.
- Ran the stored procedures manually to confirm that data retrieval and insertion worked correctly.
- Confirmed that the database structure aligns with project requirements.

6. Challenges Faced

- Initially had to fix SQL foreign key reference error when inserting student data.
- Verified table relationships manually using SSMS.
- Ensured that the database naming matched the backend connection string.