EIU
TRƯỜNG ĐẠI HỌC
QUỐC TẾ
MIỀN ĐÔNG
EASTERN
INTERNATIONAL
UNIVERSITY

Practice Assignment – Quarter 4, 2024-2025
Course Name: Backend Development
Course Code: CSW 306
Student's Full Name:
Student ID:

Practice Assignment 6

STUDENT MANAGEMENT SYSTEM

*Build a Web API using ASP.NET Core that supports CRUD operations for **students**, where each student belongs to a **class**.*

Scenario

- A **Class** contains many **Students**.
- Each **Student** belongs to **one Class**.
-

```
public class Class
{
   public int Id { get; set; }
   public string Name { get; set; } // e.g., "SE1301", "CS2025"
   public List<Student> Students { get; set; } = new();
}

public class Student
{
   public int Id { get; set; }
   [Required, MaxLength(100)]
   public string FullName { get; set; }
   [Required, EmailAddress]
   public string Email { get; set; }
   [Required]
   public DateTime EnrollmentDate { get; set; }
   public int ClassId { get; set; }
   public Class Class { get; set; }
}
```

Exercise Instructions – 5 Tasks

Task 1 – Design the Models (10 points)

- Create two classes: Student and Class.
- Set up a one-to-many relationship (one class → many students).
- Add validation:
  - FullName: required, max 100 characters
  - Email: required, must be a valid email
  - EnrollmentDate: required

Task 2 – Setup DbContext and Seed Classes (20 points)

- Create AppDbContext, declare DbSet<Student> and DbSet<Class>.
- Configure the one-to-many relationship if needed.
- Seed a few classes (e.g., "SE1301", "CS2025") using OnModelCreating.

Task 3 – Create Student via Nested JSON (25 points)

- Create endpoint: POST /api/students
- Accept input in this format:

```
{
  "fullName": "Alice Nguyen",
  "email": "alice@example.com",
  "enrollmentDate": "2023-09-01",
  "class": {
    "name": "SE1301"
  }
}
```

Requirements:

- If the class name exists → assign it to the student.
- If it doesn't exist → create a new class and assign it.

Task 4 – Get List of Students with Class Info (20 points)

- Create GET /api/students endpoint.
- Return student list with class info nested in JSON:

Task 5 – Update and Delete Student (25 points)

- PUT /api/students/{id}: Update student data, support class info via nested JSON (same as in task 3).
- DELETE /api/students/{id}: Delete student by ID.

Bonus (optional)

- Add endpoints to manage **classes** (GET, POST, PUT, DELETE)
- Validate EnrollmentDate to ensure it is not in the future