رقم المادة ( )

جمهورية مصر العربية
وزارة التربية والتعليم والتعليم الفني
امتحان دبلوم مدارس الدولية التكنولوجيا التطبيقية الصناعية (نظام السنوات الثلاث)
تجريبي – يناير 2025

التخصص: تطوير برمجيات المادة: الدراسة الفنية التخصصية المهارية
الزمن: 4 ساعات

كود المهمة: EUHM -SD-08

أجب عن الأسئلة الآتية

Mission description :

You are a software specialist at Company X, tasked with developing a **School Management System (SMS)** using **.NET Web API**. The system aims to streamline various school management functions, allowing administrators, teachers, students, and parents to interact with and manage academic and administrative data. This includes crucial features such as handling student records, class schedules, attendance tracking, and grade reporting.

This project requires you to create a robust, secure, and scalable system. The School Management System will be accessible to various users with different roles, each having different access privileges. Administrators will have full access, teachers will manage student attendance and grades, and students and parents will be able to view academic performance and attendance records. Additionally, the system needs to be built following modern software development practices, ensuring maintainability and extensibility.

This mission tasks you with creating a fully functional **School Management System (SMS)**. The system should allow users (administrators, teachers, students, and parents) to perform basic CRUD operations for student records, class schedules, attendance, and grades. Additionally, the SMS should provide role-based access and support communication features between teachers, students, and parents. The project will follow best practices for Web API development, including the use of Data Transfer Objects (DTOs), repository patterns, and dependency injection (DI).

you'll be responsible for creating and integrating models, implementing a repository pattern for efficient data access, applying dependency injection (DI) for better decoupling, and building controllers that handle HTTP requests. The goal is

to ensure that the application behaves as expected, and thorough testing will be conducted to verify this.

1- Develop a School Management System (SMS) using Web API in .NET, incorporating modern development practices like dependency injection, repository pattern, and controllers. The system will allow school administrators, teachers, students, and parents to manage and view academic and administrative data, such as student information, classes, attendance, and grades.

2- Verify that the output of the application matches the specifications and ensure alignment with user requirements through testing of modules (like attendance tracking and grade reporting).

3- Use version control to manage code changes and foster collaborative development.

Tasks:

1- **Model Creation**
   - Define core models such as Class, Teacher, Student, Attendance, and Grad.
   - Implement appropriate relationships.
     - Class ↔ Teacher (1-to-1)
     - Class ↔ Student (1-to-Many)
     - Student ↔ Grad (One-to-Many)
     - Student ↔ Subject(Many-to-Many)

Models

Class

- Id: Integer (Auto-incremented, Primary Key)
- Name: String (Required, Max Length: 50)
- TeacherId: Integer (Foreign Key to Teacher)
- Teacher: Teacher (Navigation Property, 1-to-1)
- Students: List of Student (Navigation Property, 1-to-Many)

Teacher

- Id: Integer (Auto-incremented, Primary Key)

- Name: String (Required, Max Length: 100)
- Email: String (Required, Valid Email Address)
- Phone: String (Optional)
- Class: Class (Navigation Property, 1-to-1)

Student

- Id: Integer (Auto-incremented, Primary Key)
- Name: String (Required, Max Length: 100)
- Email: String (Required, Valid Email Address)
- ClassId: Integer (Foreign Key to Class)
- Class: Class (Navigation Property, Many-to-1)
- Grad: Grad (Navigation Property, 1-to-Many)
- Subject: List of Subject(NavigationProperty,Many-to-Many)

Grad

- Id: Integer (Auto-incremented, Primary Key)
- Year: Integer (Required)
- Grade: String (Required, Max Length: 10)
- Student: Student (Navigation Property, Many-to-1)

Subject

- d: Integer (Auto-incremented, Primary Key)
- StudentId: Integer (Foreign Key to Student)
- Student: Student (Navigation Property, Many-to-Many)
- Duration: String (Required, Max Length: 10)

3.Create Controllers for Class/Teacher/Student/Grad:

- Each controller should support CRUD operations for its entity.
- Use DTOs to manage binding of data with models.
- Use Dependency Injection to interact with a Repository that manages data.

Implement the following actions in `Class Controller` using the Repository Pattern:

`POST /api/classes`

- Adds a new class with the teacher and students information.
- Required Fields:
- Name: String
- TeacherId: Integer
- Returns: 200 OK on success or 400 Bad Request for validation errors.

Implement the following actions in `Teacher Controller` using the Repository Pattern:

`GET /api/teachers`

- Retrieves a list of all teachers, including assigned class. And students and subject
- Returns: 200 OK or 404 Not Found if no teachers exist.

`POST /api/teachers`

- Adds a new teacher.
- Required Fields:
- Name: String
- Email: String
- Phone: String (Optional)
- Returns: 200 OK or 400 Bad Requests for validation errors.

`GET /api/teachers/{id}`

- Retrieves a specific teacher by Id, including assigned class.
- Returns: 404 Not Found if the teacher does not exist.

Implement the following actions in `Student Controller` using the Repository Pattern:

`GET /api/students`

- Retrieves a list of all students, including the associated class and grad information ,teacher name.
- Returns: 200 OK or 404 Not Found if no students exist.

`POST /api/students`

- Adds a new student. With teacher & class
- Required Fields:
- Name: String
- Email: String
- Returns: 200 OK or 400 Bad Requests for validation errors.

`GET /api/students/{id}`

- Retrieves a specific student by **Id**, including the associated class and grad details teacher
- Returns: **404 Not Found** if the student does notexist.

`PUT /api/students/{id}`

- Updates a student's details (e.g., class or grad changes).
- Required Fields:Name: String, Email: String
- Returns: 200 OK or 404 Not Found if the student does not exist.

`DELETE /api/students/{id}`

- Deletes a student by Id.
- Returns: 200 OK or 404 Not Found if the student does not exist.

Implement the following actions in `Grad Controller` using the Repository Pattern:

`POST /api/grads`

- Adds a new grad record for a student.
- Required Fields:
- Year: Integer
- Grade: String
- Returns: 200 OK or 400 Bad Request for validation errors.

Implement the following actions in `SubjectController` using the Repository Pattern:

`GET /api/Subject`

- Retrieves a list of all Subjects with Students and Grads.
- Returns: 200 OK or 404 Not Found if no teachers exist.

`POST /api/Subject`

- Adds a new subject with students and teacher and grade
- Required Fields:
- Name: String
- Email: String
- Phone: String (Optional)
- Returns: 200 OK or 400 Bad Requests for validation errors.

`GET /api/Subject/{id}`

- Retrieves a specific Subject by Id, including assigned student and All Other data.
- Returns: 404 Not Found if the teacher does not exist.
-

4. Create DTOs

- Use Data Transfer Objects (DTOs) for efficient data transfer between client and server.
- Create DTOs for Class, Teacher,Student and Grad.

5. Implement Repository Pattern & Dependency Injection

- Implement repository interfaces for, Class, Teacher,Student and Grad
- Register services in the Program.cs file for dependency injection.

6.Testing and Validations.

- Perform thorough testing using tools like **Swagger** to ensure all the endpoints are working correctly. You can test your methods (GET, POST, PUT, and DELETE). on your controllers, as well as your validations on your properties.

7.Set up a source code repository.

- Commit changes regularly.
- Pull updates from the cloud repository.
- Push updated code to the cloud.
- Create branches and manage updates.

(END of assessment)

Note that you will be assessed according to the following criterias , Please Read carfully .

Assessment criterias – Mission Code : EUHM-SD-08

Assessment Element Title	#	Assessment Criteria	Degree
Develop an application using API Technology	1	Install necessary packages and libraries	10
	2	Configure a connection string to connect to the database in app.json file and program.cs	10
	3	Create ApplicationDbContext file.	10
	4	Apply migration and update database	10
	5	Define core models/class	10
	6	Apply appropriate validations on your properties.	10
	7	Implement appropriate relationships between tables.	10
	8	Create appropriate controllers	15
	9	Inject your dependencies into your controllers.	15
	10	Develop your controllers to handle HTTP requests.	15
	11	Create repository interfaces for entities.	15
	12	Create repository concrete classes.	15
	13	Register your dependencies in your program.cs file.	15
	14	Inject the appropriate dependencies into the repository class.	15
	15	Apply required methods in the repository file.	15
	16	Return the appropriate response from the repository file.	15
	17	Return the appropriate status code from the controller class.	15
	18	Apply input validations on added data.	15
Verify that the output of the application matches the specifications	19	Design the test scenario and test cases for each test scenario	15
	20	Create ten test cases on your endpoints on attribute validations	15
	21	Record the expected results	15
	22	Execute the application using the created testing template	15
	23	Record the actual result	15
		the issues that noticed after testing	10

باقي الأسئلة في الورقة التالية

رقم المادة ( )

برنامج : تطوير برمجيات

Ensure using version control	25	Initialize local and remote repositories	10
	26	Commit changes to the local repository	10
	27	Connect local repository to the remote	6
	28	Commit and push changes to the remote	2
	29	Create branches and update files	2
الدرجة الكلية			**350**

Best Wishes