Task: Online Exam System (Admin Panel & Exam Website)

Task Description

You are required to develop a **mini online exam system** that consists of two sections:

1. **Admin Panel:** A secure area where an administrator can **manage exams and questions**.
2. **User Website:** A public-facing section where registered users can **log in, take exams, and view their scores**.

The system should be built using **ASP.NET Core (MVC, Razor Pages, or Blazor) and Entity Framework Core**, with authentication handled through **ASP.NET Identity**.

Key Requirements:

- **Secure authentication system** (Users are pre-added manually to the database).
- **Well-structured database design** (You must design the database schema yourself).
- **Proper use of Entity Framework Core** with migrations and relationships.
- **A user-friendly interface** with JavaScript and AJAX for smooth interactions.
- **Ensure the code follows best practices** in structure, maintainability, and performance.

Requirements:

1. Admin Panel

Develop an **Admin Panel** using **ASP.NET Core MVC or Blazor**, where an admin can:

- **Log in using ASP.NET Identity.**
- **Create, Edit, Delete Exams.**
- **Add, Edit, Delete Questions for each Exam.**
  - Each question should have:
    - A **title**.
    - A **list of four choices**.
    - A **correct answer**.
- **Manually add users to the database** (No self-registration).

---

2. User Website (Frontend)

Develop a **User Website** where users can:

- **Log in** (Credentials will be manually added to the database).
- **See Available Exams** and select an exam.
- **Solve the Exam** (Answer multiple-choice questions).
- **Submit the Exam** and see the **score**.

---

3. Exam Evaluation Criteria

After submitting the exam:

- Each **question is worth 1 point**.
- The final score is calculated as: Score=(Correct AnswersTotal Questions)×100\text{Score} = \left(\frac{\text{Correct Answers}}{\text{Total Questions}}\right) \times 100
- The **passing percentage is 60%**:
  - If the **score is 60% or higher**, the exam is **passed**.
  - If the **score is below 60%**, the exam is **failed**.
- The evaluation screen should display:
  - **Total Score (in percentage).**
  - **Number of correct vs. incorrect answers.**
  - **Pass/Fail status.**

Technical Requirements

Backend:

- **ASP.NET Core (MVC, Razor Pages, or Blazor).**
- **Entity Framework Core** for database handling.
- **ASP.NET Identity** for authentication and user management.
- **Repository Pattern** (Preferred for better structure).
- **Use Migrations** to create and update the database schema.

Frontend:

- **JavaScript (Vanilla JS or jQuery)** for handling UI interactions.
- **Bootstrap** (or Tailwind CSS) for styling.

- **AJAX** for submitting exam answers without page reloads.

Database:

- **SQL Server** (or In-Memory DB for simplicity).
- **You must design the database structure yourself** based on the requirements.
- **Use EF Core Migrations** to manage schema changes.

Submission Guidelines

Candidates should:

1. Upload their project to **GitHub** (public repo).
2. Ensure the project runs without issues.
3. **No README file is required**.

Evaluation Criteria

ria	**ription**
base Design	-structured relational database with proper relationships
Quality	-organized, follows best practices
y Framework Usage	er use of DbContext, Migrations, Relationships
NET Identity	re authentication and user management
tend Implementation	UI, proper use of JavaScript/jQuery
tionality	equirements implemented correctly
Evaluation Logic	er calculation of scores and pass/fail status
sage	mit history showing progress

Estimated Time to Complete: 3 days from receiving the task