**Dear Applicant**,

In your test round, you are expected to complete the following assignment.

- Create two projects in a solution.
  - MVC Project (Models-Views-Controllers)
    - Create a login page and a user to login into the web application.
    - Connect to your local database (SQL or MySQL)
    - Use only **Entity Framework Core** (Code First Approach or DB First Approach) or **ADO .Net** for connecting to the database.
    - Create a table named '**Items**' in the database with the following columns
      - Id (Primary Key, int)
      - Name (Unique Key, string) --item's name
      - Item Count (int) --indicates the item's count.
      - Create Date (Datetime) --indicates the item's create date
      - InStock (bit) --indicates whether the item is in stock or not.
    - Create a page to add items.
    - Create another page to display the items in a table/grid form. The items in this grid must have the status as In Stock or Out of Stock.
    - Update and Delete items functionality must be implemented too in the items grid page in order to update or delete an item.

    Note: The add items page and items grid page should only be accessible if the user is logged in. If the user tries to access those pages, the user should be redirected to login page. Follow any **Design Pattern**. **SOLID principles** should be implemented. Try to use **OOP**. Use Services instead of all the logic in the controllers.

  - Web API Project
    - Create a POST API to add items to the database.
    - Create a GET API to add all the items.
    - Create a GET API to get an item by Id.
    - Create an API to Update an item based on item Id.
    - Create an API to Delete an item based on item Id.
    - Create a swagger page for the above API's.

    Note: Implement all the basic validations in the above API's. Every API should have a failure or pass response based on the request/logic. Try to use Services instead of all the logic in the controllers.

- Create a Stored procedure which will need to return a Json result which easily can be mapped with below Model.

```
public class People : BaseEntity

{
      public String FirstName {get; set;}
      public String LastName {get; set;}
      public String Email {get; set;}
      public bool IsActive {get; set;}
      public String ImageThumbUrl {get; set;}
      Public List<PeoplePosition> PeoplePositions{get;set;}
      Public List<PeopleDepartment> PeopleDepartment{get;set;}
}
public class PeoplePosition
{
      public String PositionName {get; private set;}
      public DateTime? StartDate {get; private set;}
      public DateTime? EndDate {get; private set;}
}
public class PeopleDeartment
{
      public String DepartmentName {get; private set;}
}

Public class BaseEntity
{
      Public int Id {get;set;}
}
```

Focus should be on code structure, architecture, cleanliness, interfaces etc.

Preferred Tech: C#, .Net; Database -> SQL, MySql