Project Title:

Dynamic Form Generator

Project Objective:

The objective of this project is to design and implement a dynamic web-based form generator capable of rendering a form based on a JSON configuration. The application should handle the following:

1. **Render Forms:** Dynamically generate and display a form defined by the provided JSON structure.
2. **Data Collection:** Collect user input values entered into the form fields.
3. **Validation:** Apply basic and custom validation rules to the input fields, such as required fields, email formatting, and number range checks.
4. **Generate Response:** Return the collected form data as a JSON object upon submission.

Note: Development time is estimated at 10±2 hours. Focus on features you find most critical, ensuring a balance between functionality, simplicity, and code quality.

Perspective: This challenge represents a simplified version of a larger feature set in our product. Future capabilities may include cross-field validations, WYSIWYG form design, triggers, spreadsheet data integration, workflow automation, and more (not included in this challenge).

Technical Requirements:

Input Specification (Form JSON):

Example JSON:

```
{
  "title": "Sample Form",
  "fields": [
   { "type": "text", "label": "Name", "required": true },
   { "type": "email", "label": "Email", "required": true },
   { "type": "number", "label": "Age", "min": 18, "max": 100 },
   { "type": "dropdown", "label": "Industry", "values": ["Tech", "Production", "Health"], "required": true },
   { "type": "checkbox", "label": "Subscribe to Newsletter", "required": false }
  ]
}
```

- **Field Types:**
  - text, email, number, checkbox, dropdown, etc.
- **Validation Rules:**
  - required, email formatting, number min and max, etc.
- **Metadata:**
  - Form title and field labels.

Output Specification (Response JSON):

Example Response:

```
{
  "Name": "John Doe",
  "Email": "john.doe@example.com",
  "Age": 25,
  "Industry": "Technology",
  "Subscribe to Newsletter": true
}
```

UI Expectations:

- **Framework:** Use **Blazor WebAssembly** for the front-end.
- **Styling/Components:** Optionally utilize **MudBlazor** for enhanced design and user experience.

Development Constraints:

1. **No Database Integration:** Handle data in-memory.
2. **Modern Frameworks:** Utilize **.NET 8** for development.
3. **Focus:** Prioritize simplicity, maintainability, and code quality.

Encouraged Practices:

- Feel free to **use AI and internet searches** for assistance.
- Write clean, modular, and high-quality code.
- Use a simple and efficient approach for solution implementation.

Deliverables:

1. **Functioning Application:**
    - A fully functional dynamic form generator with the following features:
        - Form rendering from JSON configuration.
        - Validation and error handling.
        - JSON output generation.
2. **Git Repository:**
    - Share a public **GitHub repository** link containing the complete solution.
3. **Documentation:**
    - Include a **README** with:
        - Steps to run the application.
        - Design considerations and architecture decisions.
        - Any assumptions made during development.

Duration:

You have **one week** to complete this project.

Evaluation Metrics:

1. **Problem Solving:**
   - How effectively the candidate breaks down and solves the problem.
2. **Code Quality:**
   - Readability, structure, adherence to best practices, and simplicity.
3. **Functionality:**
   - Completeness and correctness of the implemented features.
4. **Knowledge of Technology:**
   - Understanding and application of **.NET 8**, **Blazor**, and optionally **MudBlazor**.
5. **Optional Review Meeting:**
   - Candidates may be invited to a review meeting to present and discuss their solution.