Carbon Crunch

Objective:

Build a lightweight tool that analyzes a given React (JavaScript) or FastAPI (Python) code file and scores it on clean code practices, while also offering recommendations for improvement.

What You Need to Build:

1. Input Handling:

- Accept a single .js, .jsx, or .py file for analysis.
- Accept file via:
  - A simple frontend interface in React (upload + submit)
  - A backend endpoint in FastAPI (/analyze-code)

2. Code Analysis Logic:

- Your tool should return:
  - A score out of 100
  - Naming conventions (10)
  - Function length and modularity (20)
  - Comments and documentation (20)
  - Formatting/indentation (15)
  - Reusability and DRY (15)
  - Best practices in web dev (20)
  - A summary with:
    - Category-wise breakdown
    - 3–5 clear recommendations (e.g., “Use consistent camelCase for function names.”)

3. Output Format:

- Return JSON like:

```
{
  "overall_score": 82,
  "breakdown": {
    "naming": 8,
    "modularity": 17,
```

```
    "comments": 20,
    "formatting": 12,
    "reusability": 10,
    "best_practices": 15
  },
  "recommendations": [
    "Avoid deeply nested components in your React render logic.",
    "Function 'calculateTotal' in app.py is too long—consider refactoring.",
    "Use camelCase for variable 'Total_Amount'."
  ]
}
```

Bonus (Optional):

- Add a GitHub Action YAML file that runs this tool on every commit to check code quality.
- Output the result as a comment on the pull request or commit log.

Tech Stack:

- Frontend: React (basic file upload + display result)
- Backend: FastAPI
- Optional Libraries:
- Python: pylint, radon, flake8, or your own logic
- JS: eslint or static regex-based checks

Deliverables:

- GitHub repo or zipped folder
- README.md with setup instructions
- Sample test files used
- (Bonus) .github/workflows/code-quality.yml

Examples Input:

```
# app.py

def CalculateTotal(orders):
```

```
    sum = 0
    for i in orders:
        sum += i["value"]
    return sum
```

Expected Output:

```
{
  "overall_score": 64,
  "recommendations": [
    "Use snake_case for function names in Python (should be 'calculate_total').",
    "Avoid using 'sum' as a variable name—it's a built-in.",
    "Add a docstring to explain the purpose of the function."
  ]
}
```