■ Project Code Export

■ Frontend File List:

- .gitignore
- Frontend_Code_Export.pdf
- db.json
- postcss.config.js
- public\favicon.ico
- public\index.html
- public\left.png
- public\logo192.png
- public\logo512.png
- public\manifest.json
- public\right.png
- public\robots.txt
- script.py
- src\App.css
- src\App.js
- src\App.test.js
- src\AppContent.js
- src\assets\placeholder.svg
- src\components\admin\AdminPanel.js
- src\components\common\LoadingSpinner.js
- src\components\common\ResetAll.js
- src\components\common\TabNavigation.js
- src\components\common\Toasts.js
- src\components\form\FieldInputRow.js
- src\components\form\InputForm.js
- src\components\report\PDFPreview.js
- src\components\report\ReportOutput.js
- src\components\settings\CategoryManager.js
- src\components\settings\SettingsPanel.js
- src\contexts\FormConfigContext.js
- src\contexts\ThemeContext.js
- src\hooks\useConfigImportExport.js
- src\hooks\useExcelExport.js
- src\hooks\usePDFGeneration.js
- src\hooks\useReportGeneration.js
- src\index.css
- src\index.js
- src\logo.svg
- src\reportWebVitals.js
- src\setupTests.js
- src\utils\constants.js
- src\utils\helpers.js
- tailwind.config.js

■ File: .gitignore

```
================================================================================
[Binary file -  format]

--------------------------------------------------------------------------------
```

■ File: Frontend_Code_Export.pdf

```
================================================================================
[Binary file - .pdf format]

--------------------------------------------------------------------------------
```

■ File: db.json

```
================================================================================
```


```
  1: {
  2:   "settings": {
  3:     "title": "GENOMICS & DIET",
  4:     "quote": "\"YOU ARE WHAT YOU EAT\" - Victor Lindlahr",
  5:     "description": "A right diet is the one that makes you feel happy, keeps you healthy, does not make you
  6:     "headerColor": "#16a34a",
  7:     "colors": {
  8:       "low": "#16a34a",
  9:       "medium": "#f59e0b",
 10:       "high": "#dc2626"
 11:     },
 12:     "highThreshold": 6
 13:   },
 14:   "categories": [
 15:     {
 16:       "id": 1,
 17:       "name": "MACRONUTRIENTS"
 18:     },
 19:     {
 20:       "id": 2,
 21:       "name": "MEAL PATTERN"
 22:     },
 23:     {
 24:       "id": 3,
 25:       "name": "FOOD SENSITIVITIES"
 26:     }
 27:   ],
 28:   "fields": [
 29:     {
 30:       "id": "carb",
 31:       "label": "Carbohydrate Sensitivit",
 32:       "category": "MACRONUTRIENTS",
 33:       "min": 1,
 34:       "max": 10,
 35:       "high": "Maintain carb intake <45%\nFor obesity & IR control",
 36:       "normal": "Maintain carb intake <50%\nBalanced recommendation",
 37:       "low": "Maintain carb intake <60%\nFor obesity & IR control",
 38:       "_uuid": "cad62c59-d101-4eb5-9e60-8cbd3c17f195",
 39:       "_originalId": "carb"
 40:     },
 41:     {
 42:       "id": "fat",
 43:       "label": "Fat Sensitivity",
 44:       "category": "MACRONUTRIENTS",
 45:       "min": 1,
 46:       "max": 10,
 47:       "high": "Fat intake not to exceed\n15% of total calories",
 48:       "normal": "Fat intake should be\n20% of total calories",
 49:       "low": "Fat intake advised up to\n25% of total calories"
 50:     },
 51:     {
 52:       "id": "protein",
 53:       "label": "Protein Requirement",
 54:       "category": "MACRONUTRIENTS",
 55:       "min": 1,
 56:       "max": 15,
 57:       "high": "Protein supplements\nneeded along with dietary\nsource",
 58:       "normal": "Maintain adequate protein\nthrough balanced diet and\noccasional supplements",
 59:       "low": "Protein supplements not\nneeded, intake through\ndiet is enough"
```

```
    },
    {
      "id": "meal",
      "label": "Meal Frequency",
      "category": "MEAL PATTERN",
      "min": 1,
      "max": 10,
      "high": "4-5 small meals suggested\nin a day",
      "normal": "3-4 balanced meals\nrecommended per day",
      "low": "Less frequent meals, 2-3\nmeals are enough in a day"
    },
    {
      "id": "alcohol",
      "label": "Alcohol Sensitivity",
      "category": "",
      "min": 1,
      "max": 6,
      "high": "High sensitivity, avoid\nalcohol if possible,\nespecially the types of\nbeverages that trigg
      "normal": "Moderate sensitivity,\nlimit alcohol consumption\nto 1-2 drinks per day,\nmonitor for any
      "low": "Low sensitivity, know\nyour alcohol intake limits,\nconsult doctor for\nknowing your upper li
    },
    {
      "id": "caffeine",
      "label": "Caffeine Sensitivity",
      "category": "",
      "min": 1,
      "max": 5,
      "high": "High sensitivity, do not\nconsume >4 cups/day",
      "normal": "Moderate sensitivity,\nlimit caffeine to 4-5\ncups per day",
      "low": "Caffeine up to 5 cups a\nday can be consumed"
    },
    {
      "id": "gluten",
      "label": "Gluten Sensitivity",
      "category": "FOOD SENSITIVITIES",
      "min": 1,
      "max": 10,
      "high": "Gluten intake needs to be\nreduced/stopped",
      "normal": "Monitor gluten intake,\nreduce if experiencing\ndigestive issues",
      "low": "Gluten to be avoided in\ncases of gastric distress"
    },
    {
      "id": "lactose",
      "label": "Lactose Sensitivity",
      "category": "FOOD SENSITIVITIES",
      "min": 1,
      "max": 10,
      "high": "Milk & milk products need\nto be avoided",
      "normal": "Limit milk & milk products,\nconsider lactose-free\nalternatives if needed",
      "low": "Milk or milk products to\nbe avoided in gastric\ndistress"
    },
    {
      "id": "salt",
      "label": "Salt Sensitivity",
      "category": "",
      "min": 1,
      "max": 8,
      "high": "Try to reduce overall salt\nintake to up to 3-5 gm per\nday",
      "normal": "Maintain salt intake\naround 5 gm per day",
      "low": "Consumption of salt up to\n5 gm/day can be done"
    },
    {
      "id": "field_1751974814401",
      "label": "New Field",
      "category": "",
      "min": 1,
      "max": 10,
      "high": "",
      "normal": "",
      "low": ""
    }
  ]
}
```

■ File: postcss.config.js

```
  1: module.exports = {
  2:   plugins: {
  3:     tailwindcss: {},
  4:     autoprefixer: {},
  5:   },
  6: }
```

■ File: public\favicon.ico

[Binary file - .ico format]

■ File: public\index.html

```
  1: <!DOCTYPE html>
  2: <html lang="en">
  3:   <head>
  4:     <meta charset="utf-8" />
  5:     <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
  6:     <meta name="viewport" content="width=device-width, initial-scale=1" />
  7:     <meta name="theme-color" content="#000000" />
  8:     <meta
  9:       name="description"
 10:       content="Web site created using create-react-app"
 11:     />
 12:     <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
 13:     <!--
 14:       manifest.json provides metadata used when your web app is installed on a
 15:       user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
 16:     -->
 17:     <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
 18:     <!--
 19:       Notice the use of %PUBLIC_URL% in the tags above.
 20:       It will be replaced with the URL of the `public` folder during the build.
 21:       Only files inside the `public` folder can be referenced from the HTML.
 22:
 23:       Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
 24:       work correctly both with client-side routing and a non-root public URL.
 25:       Learn how to configure a non-root public URL by running `npm run build`.
 26:     -->
 27:     <title>React App</title>
 28:   </head>
 29:   <body>
 30:     <noscript>You need to enable JavaScript to run this app.</noscript>
 31:     <div id="root"></div>
 32:     <!--
 33:       This HTML file is a template.
 34:       If you open it directly in the browser, you will see an empty page.
 35:
 36:       You can add webfonts, meta tags, or analytics to this file.
 37:       The build step will place the bundled scripts into the <body> tag.
 38:
 39:       To begin the development, run `npm start` or `yarn start`.
 40:       To create a production bundle, use `npm run build` or `yarn build`.
 41:     -->
 42:   </body>
 43: </html>
```

■ File: public\left.png

[Binary file - .png format]

■ File: public\logo192.png

```
================================================================================
[Binary file - .png format]

--------------------------------------------------------------------------------
```

■ File: public\logo512.png

```
================================================================================
[Binary file - .png format]

--------------------------------------------------------------------------------
```

■ File: public\manifest.json

```
================================================================================
  1: {
  2:   "short_name": "React App",
  3:   "name": "Create React App Sample",
  4:   "icons": [
  5:     {
  6:       "src": "favicon.ico",
  7:       "sizes": "64x64 32x32 24x24 16x16",
  8:       "type": "image/x-icon"
  9:     },
 10:     {
 11:       "src": "logo192.png",
 12:       "type": "image/png",
 13:       "sizes": "192x192"
 14:     },
 15:     {
 16:       "src": "logo512.png",
 17:       "type": "image/png",
 18:       "sizes": "512x512"
 19:     }
 20:   ],
 21:   "start_url": ".",
 22:   "display": "standalone",
 23:   "theme_color": "#000000",
 24:   "background_color": "#ffffff"
 25: }

--------------------------------------------------------------------------------
```

■ File: public\right.png

```
================================================================================
[Binary file - .png format]

--------------------------------------------------------------------------------
```

■ File: public\robots.txt

```
================================================================================
# https://www.robotstxt.org/robotstxt.html
User-agent: *
Disallow:

--------------------------------------------------------------------------------
```

■ File: script.py

```
================================================================================
  1: import os
  2: from reportlab.lib.pagesizes import A4
  3: from reportlab.lib.units import mm
  4: from reportlab.pdfgen import canvas
  5:
  6: def get_code_files(directory, excluded_files=None, excluded_dirs=None):
  7:     """Fetch all project files except specified exclusions."""
  8:     if excluded_files is None:
  9:         excluded_files = {'package.json', 'package-lock.json'}
 10:
 11:     if excluded_dirs is None:
 12:         excluded_dirs = {'node_modules', '.git', '__pycache__', 'build', '.next', 'dist'}
 13:
 14:     code_files = {}
 15:
```

```
    for root, dirs, files in os.walk(directory):
        # Skip excluded directories
        dirs[:] = [d for d in dirs if d not in excluded_dirs]

        # Skip if current directory is an excluded directory
        if any(excluded_dir in root.split(os.sep) for excluded_dir in excluded_dirs):
            continue

        for file in files:
            # Skip excluded files
            if file in excluded_files:
                continue

            file_path = os.path.join(root, file)

            # Get file extension
            _, ext = os.path.splitext(file)

            try:
                # Try to read as text file first
                if ext.lower() in {'.js', '.jsx', '.ts', '.tsx', '.css', '.scss', '.sass', '.less',
                                   '.html', '.htm', '.json', '.md', '.txt', '.xml', '.yaml', '.yml',
                                   '.config', '.gitignore', '.env', '.py', '.sh', '.bat', '.cmd',
                                   '.svg', '.dockerfile', '.editorconfig', '.eslintrc', '.prettierrc'}:
                    with open(file_path, "r", encoding="utf-8", errors="ignore") as f:
                        code_files[file_path] = f.readlines()
                else:
                    # For binary files, just note them as binary
                    code_files[file_path] = [f"[Binary file - {ext} format]"]

            except Exception as e:
                print(f"? Error reading {file_path}: {e}")
                code_files[file_path] = [f"[Error reading file: {str(e)}]"]

    return code_files


def create_pdf(code_data, output_pdf="Frontend_Code_Export.pdf"):
    c = canvas.Canvas(output_pdf, pagesize=A4)
    width, height = A4
    margin = 20 * mm
    line_height = 10
    y = height - margin

    # Title
    c.setFont("Helvetica-Bold", 16)
    c.drawString(margin, y, "? Project Code Export")
    y -= 2 * line_height
    c.setFont("Helvetica-Bold", 12)
    c.drawString(margin, y, "? Frontend File List:")
    y -= 2 * line_height

    file_paths = sorted(list(code_data.keys()))

    # 1. File list (original simple format)
    c.setFont("Courier", 8)
    for path in file_paths:
        if y < margin:
            c.showPage()
            c.setFont("Courier", 8)
            y = height - margin

        display_path = os.path.relpath(path)
        c.drawString(margin, y, f"- {display_path}")
        y -= line_height

    # Add page break before code content
    c.showPage()
    y = height - margin

    # 2. File contents
    for file_path in file_paths:
        lines = code_data[file_path]
```

```
        print(f"? Adding: {file_path}")

        if y < margin + 3 * line_height:
            c.showPage()
            y = height - margin

        # File header
        rel_path = os.path.relpath(file_path)
        c.setFont("Helvetica-Bold", 12)
        c.drawString(margin, y, f"? File: {rel_path}")
        y -= line_height

        # Add separator line
        c.setFont("Courier", 8)
        c.drawString(margin, y, "=" * 80)
        y -= line_height

        # File content
        for line_num, line in enumerate(lines, 1):
            if y < margin:
                c.showPage()
                c.setFont("Courier", 8)
                y = height - margin

            # Clean and truncate line
            line = line.strip("\n").encode("latin-1", "replace").decode("latin-1")

            # Add line numbers for code files
            if rel_path.endswith(('.js', '.jsx', '.ts', '.tsx', '.css', '.py', '.html', '.json')):
                display_line = f"{line_num:3d}: {line[:280]}"
            else:
                display_line = line[:300]

            c.drawString(margin, y, display_line)
            y -= line_height

        # Add spacing between files
        y -= line_height
        if y > margin:
            c.setFont("Courier", 8)
            c.drawString(margin, y, "-" * 80)
            y -= 2 * line_height

    c.save()
    print(f"? PDF successfully created: {output_pdf}")
    print(f"? Total files processed: {len(code_data)}")


def main():
    root_dir = os.path.dirname(os.path.abspath(__file__))

    # Files to exclude (including package.json as requested)
    excluded_files = {
        'package.json',
        'package-lock.json',
        'yarn.lock',
        'README.md',
        '.DS_Store',
        'Thumbs.db',
        'Desktop.ini'
    }

    # Directories to exclude
    excluded_dirs = {
        'node_modules',
        '.git',
        '__pycache__',
        'build',
        'dist',
        '.next',
        'coverage',
        '.nyc_output',
        'logs',
```

```
162:         '*.log'
163:     }
164: 
165:     print("? Scanning project files...")
166:     code_files = get_code_files(root_dir, excluded_files, excluded_dirs)
167: 
168:     if not code_files:
169:         print("? No files found to process!")
170:         return
171: 
172:     print(f"? Found {len(code_files)} files to include in PDF")
173:     create_pdf(code_files)
174: 
175: 
176: if __name__ == "__main__":
177:     main()
```

■ File: src\App.css

```
  1: .App {
  2:   text-align: center;
  3: }
  4: 
  5: .App-logo {
  6:   height: 40vmin;
  7:   pointer-events: none;
  8: }
  9: 
 10: @media (prefers-reduced-motion: no-preference) {
 11:   .App-logo {
 12:     animation: App-logo-spin infinite 20s linear;
 13:   }
 14: }
 15: 
 16: .App-header {
 17:   background-color: #282c34;
 18:   min-height: 100vh;
 19:   display: flex;
 20:   flex-direction: column;
 21:   align-items: center;
 22:   justify-content: center;
 23:   font-size: calc(10px + 2vmin);
 24:   color: white;
 25: }
 26: 
 27: .App-link {
 28:   color: #61dafb;
 29: }
 30: 
 31: @keyframes App-logo-spin {
 32:   from {
 33:     transform: rotate(0deg);
 34:   }
 35:   to {
 36:     transform: rotate(360deg);
 37:   }
 38: }
```

■ File: src\App.js

```
  1: import React from 'react';
  2: import { FormConfigProvider } from './contexts/FormConfigContext';
  3: import { ThemeProvider } from './contexts/ThemeContext';
  4: import AppContent from './AppContent';
  5: import Toasts from './components/common/Toasts';
  6: // import { FormConfigProvider } from "./contexts/FormConfigContext";
  7: 
  8: 
  9: function App() {
```

```
 10:   return (
 11:     <ThemeProvider>
 12:       <FormConfigProvider>
 13:         <AppContent />
 14:         <Toasts />
 15:       </FormConfigProvider>
 16:     </ThemeProvider>
 17:   );
 18: }
 19:
 20: export default App;
```

■ File: src\App.test.js

```
  1: import { render, screen } from '@testing-library/react';
  2: import App from './App';
  3:
  4: test('renders learn react link', () => {
  5:   render(<App />);
  6:   const linkElement = screen.getByText(/learn react/i);
  7:   expect(linkElement).toBeInTheDocument();
  8: });
```

■ File: src\AppContent.js

```
  1: import React, { useState, useEffect } from "react";
  2: import TabNavigation from "./components/common/TabNavigation";
  3: import InputForm from "./components/form/InputForm";
  4: import ReportOutput from "./components/report/ReportOutput";
  5: import AdminPanel from "./components/admin/AdminPanel";
  6: import SettingsPanel from "./components/settings/SettingsPanel";
  7: import { useReportGeneration } from "./hooks/useReportGeneration";
  8: import { AnimatePresence, motion } from "framer-motion";
  9:
 10: const LOCAL_TAB_KEY = "activeTab";
 11:
 12: const AppContent = () => {
 13:   // ? Load initial tab from localStorage if available
 14:   const [activeTab, setActiveTab] = useState(() => {
 15:     return localStorage.getItem(LOCAL_TAB_KEY) || "form";
 16:   });
 17:
 18:   const { reportData, generateReport } = useReportGeneration();
 19:
 20:   // ? Sync tab state to localStorage on every change
 21:   useEffect(() => {
 22:     localStorage.setItem(LOCAL_TAB_KEY, activeTab);
 23:   }, [activeTab]);
 24:
 25:   const handleGenerateReport = (formData) => {
 26:     generateReport(formData);
 27:   };
 28:
 29:   const tabTransition = {
 30:     initial: { opacity: 0, y: 20 },
 31:     animate: { opacity: 1, y: 0 },
 32:     exit: { opacity: 0, y: -20 },
 33:     transition: { duration: 0.3 },
 34:   };
 35:
 36:   return (
 37:     <div className="bg-gray-100 font-sans min-h-screen dark:bg-gray-900">
 38:       <TabNavigation activeTab={activeTab} setActiveTab={setActiveTab} />
 39:
 40:       <AnimatePresence mode="wait">
 41:         {activeTab === "form" && (
 42:           <motion.div key="form" {...tabTransition}>
 43:             <div className="flex flex-col md:flex-row h-screen desktop-layout">
 44:               <InputForm
```

```
45:                 onGenerateReport={handleGenerateReport}
46:                 reportData={reportData}
47:               />
48:               <ReportOutput reportData={reportData} />
49:             </div>
50:           </motion.div>
51:         )}
52:
53:         {activeTab === "admin" && (
54:           <motion.div key="admin" {...tabTransition}>
55:             <div className="p-6">
56:               <AdminPanel />
57:             </div>
58:           </motion.div>
59:         )}
60:
61:         {activeTab === "settings" && (
62:           <motion.div key="settings" {...tabTransition}>
63:             <div className="p-6">
64:               <SettingsPanel />
65:             </div>
66:           </motion.div>
67:         )}
68:       </AnimatePresence>
69:     </div>
70:   );
71: };
72:
73: export default AppContent;
```

■ File: src\assets\placeholder.svg

==========

■ File: src\components\admin\AdminPanel.js

```
  1: import React, { useEffect, useState } from "react";
  2: import { DragDropContext, Droppable, Draggable } from "@hello-pangea/dnd";
  3: import { toast } from "react-toastify";
  4: import { v4 as uuidv4 } from "uuid";
  5:
  6: const API_URL = "http://localhost:5000";
  7:
  8: const AdminPanel = () => {
  9:   const [localFields, setLocalFields] = useState([]);
 10:   const [categories, setCategories] = useState([]);
 11:
 12:   // ? Fetch all fields
 13:   const fetchFields = async () => {
 14:     try {
 15:       const res = await fetch(`${API_URL}/fields`);
 16:       let data = await res.json();
 17:
 18:       data = data.map((f) => ({
 19:         ...f,
 20:         _uuid: uuidv4(),
 21:         _originalId: f.id,
 22:       }));
 23:
 24:       setLocalFields(data);
 25:     } catch (err) {
 26:       toast.error("? Failed to load fields");
 27:     }
 28:   };
 29:
 30:   // ? Fetch all categories
 31:   const fetchCategories = async () => {
 32:     try {
 33:       const res = await fetch(`${API_URL}/categories`);
 34:       const data = await res.json();
```

```
      // Convert to string list if needed
      const names = data.map((cat) =>
        typeof cat === "string" ? cat : cat.name
      );
      setCategories(names);
    } catch (err) {
      toast.error("? Failed to load categories");
    }
  };

  useEffect(() => {
    fetchFields();
    fetchCategories();
  }, []);

  const addNewField = async () => {
    const newField = {
      _uuid: uuidv4(),
      _originalId: null,
      id: `field_${Date.now()}`,
      label: "New Field",
      category: "",
      min: 1,
      max: 10,
      high: "",
      normal: "",
      low: "",
    };

    const res = await fetch(`${API_URL}/fields`, {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(newField),
    });

    if (res.ok) {
      toast.success("? Field added!");
      fetchFields();
    } else {
      toast.error("? Failed to add field");
    }
  };

  const saveField = async (index) => {
    const field = localFields[index];
    const targetId = field._originalId ?? field.id;

    const res = await fetch(`${API_URL}/fields/${targetId}`, {
      method: "PUT",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ ...field }),
    });

    if (res.ok) {
      toast.success("? Field saved!");
      fetchFields();
    } else {
      toast.error("? Failed to save field");
    }
  };

  const deleteField = async (id) => {
    if (!window.confirm("Are you sure?")) return;

    await fetch(`${API_URL}/fields/${id}`, { method: "DELETE" });
    toast.info("?? Field deleted");
    fetchFields();
  };

  const updateLocalField = (index, key, value) => {
    const updated = [...localFields];

    if (key === "id") {
```

```
      const isDuplicate = localFields.some(
        (f, i) => i !== index && f.id.trim() === value.trim()
      );
      if (isDuplicate) {
        toast.error("? Field ID must be unique");
        return;
      }
    }

    updated[index] = { ...updated[index], [key]: value };
    setLocalFields(updated);
  };

  const onDragEnd = (result) => {
    if (!result.destination) return;

    const reordered = [...localFields];
    const [moved] = reordered.splice(result.source.index, 1);
    reordered.splice(result.destination.index, 0, moved);
    setLocalFields(reordered);
    toast.info("? Field order updated (not saved yet)");
  };

  return (
    <div className="max-w-6xl mx-auto px-4 py-8">
      <div className="flex justify-between items-center mb-6">
        <h2 className="text-2xl font-bold">?? Field Management</h2>
        <button
          onClick={addNewField}
          className="bg-green-600 hover:bg-green-700 text-white px-5 py-2 rounded-lg font-semibold"
        >
          ? Add New Field
        </button>
      </div>

      <DragDropContext onDragEnd={onDragEnd}>
        <Droppable droppableId="fields">
          {(provided) => (
            <div ref={provided.innerRef} {...provided.droppableProps}>
              {localFields.map((field, index) => (
                <Draggable
                  key={field._uuid}
                  draggableId={field._uuid}
                  index={index}
                >
                  {(provided, snapshot) => (
                    <div
                      ref={provided.innerRef}
                      {...provided.draggableProps}
                      {...provided.dragHandleProps}
                      className={`border rounded-lg p-6 bg-white shadow-sm space-y-4 ${
                        snapshot.isDragging
                          ? "bg-gray-100 ring-2 ring-green-500"
                          : ""
                      }`}
                    >
                      <div className="flex justify-between items-center">
                        <h3 className="text-lg font-semibold">
                          {index + 1}. {field.label}
                        </h3>
                        <div className="flex gap-2">
                          <button
                            onClick={() => saveField(index)}
                            className="text-blue-600 hover:underline"
                          >
                            ? Save
                          </button>
                          <button
                            onClick={() =>
                              deleteField(field._originalId ?? field.id)
                            }
                            className="text-red-600 hover:underline"
                          >
```

```
                        ?? Delete
                      </button>
                    </div>
                  </div>

                  <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                    <FieldInput
                      label="Field ID"
                      value={field.id}
                      onChange={(val) =>
                        updateLocalField(index, "id", val.trim())
                      }
                    />
                    <FieldInput
                      label="Label"
                      value={field.label}
                      onChange={(val) =>
                        updateLocalField(index, "label", val)
                      }
                    />
                    <FieldSelect
                      label="Category"
                      value={field.category}
                      options={categories}
                      onChange={(val) =>
                        updateLocalField(index, "category", val)
                      }
                    />
                    <FieldInput
                      label="Min"
                      type="number"
                      value={field.min}
                      onChange={(val) =>
                        updateLocalField(
                          index,
                          "min",
                          val ? parseInt(val) : undefined
                        )
                      }
                    />
                    <FieldInput
                      label="Max"
                      type="number"
                      value={field.max}
                      onChange={(val) =>
                        updateLocalField(
                          index,
                          "max",
                          val ? parseInt(val) : undefined
                        )
                      }
                    />
                    <FieldTextarea
                      label="High Recommendation"
                      value={field.high}
                      onChange={(val) =>
                        updateLocalField(index, "high", val)
                      }
                    />
                    <FieldTextarea
                      label="Normal Recommendation"
                      value={field.normal}
                      onChange={(val) =>
                        updateLocalField(index, "normal", val)
                      }
                    />
                    <FieldTextarea
                      label="Low Recommendation"
                      value={field.low}
                      onChange={(val) =>
                        updateLocalField(index, "low", val)
                      }
                    />
```

```
                      </div>
                    </div>
                  )}
                </Draggable>
              ))}
              {provided.placeholder}
            </div>
          )}
        </Droppable>
      </DragDropContext>
    </div>
  );
};

// ? Reusable components
const FieldInput = ({
  label,
  value,
  onChange,
  type = "text",
  disabled = false,
}) => (
  <div>
    <label className="text-sm font-medium">{label}</label>
    <input
      type={type}
      disabled={disabled}
      className="w-full border rounded px-3 py-2"
      value={value ?? ""}
      onChange={(e) => onChange?.(e.target.value)}
    />
  </div>
);

const FieldTextarea = ({ label, value, onChange }) => (
  <div>
    <label className="text-sm font-medium">{label}</label>
    <textarea
      rows={3}
      className="w-full border rounded px-3 py-2"
      value={value ?? ""}
      onChange={(e) => onChange?.(e.target.value)}
    />
  </div>
);

const FieldSelect = ({ label, value, options, onChange }) => (
  <div>
    <label className="text-sm font-medium">{label}</label>
    <select
      className="w-full border rounded px-3 py-2"
      value={value ?? ""}
      onChange={(e) => onChange?.(e.target.value)}
    >
      <option value="">? None ?</option>
      {options.map((opt, i) => (
        <option key={i} value={opt}>
          {opt}
        </option>
      ))}
    </select>
  </div>
);

export default AdminPanel;
```

■ File: src\components\common\LoadingSpinner.js

==========

■ File: src\components\common\ResetAll.js

```
================================================================================
```

```
  1: import React from "react";
  2: import { toast } from "react-toastify";
  3:
  4: const ResetAll = () => {
  5:   const handleReset = () => {
  6:     const confirmReset = window.confirm(
  7:       "This will erase all unsaved changes and restore the app to its default state. Continue?"
  8:     );
  9:
 10:     if (!confirmReset) return;
 11:
 12:     // Clear form input, config, settings
 13:     localStorage.removeItem("genomics_form_data");
 14:     // Optionally remove other keys if added later (e.g., config, theme)
 15:     // localStorage.removeItem('genomics_config');
 16:
 17:     toast.success("App state reset. Reloading...");
 18:
 19:     setTimeout(() => {
 20:       window.location.reload(); // reload default from formConfig.json
 21:     }, 1000);
 22:   };
 23:
 24:   return (
 25:     <button
 26:       onClick={handleReset}
 27:       className="bg-red-600 hover:bg-red-700 text-white px-6 py-3 rounded-lg font-semibold"
 28:     >
 29:       ?? Reset All
 30:     </button>
 31:   );
 32: };
 33:
 34: export default ResetAll;
```

```
--------------------------------------------------------------------------------
```

■ File: src\components\common\TabNavigation.js

```
================================================================================
```

```
  1: import { useTheme } from "../../contexts/ThemeContext";
  2:
  3: const TabNavigation = ({ activeTab, setActiveTab }) => {
  4:   const { theme, toggleTheme } = useTheme();
  5:
  6:   const tabs = [
  7:     { id: "form", label: "? Form View" },
  8:     { id: "admin", label: "?? Admin Panel" },
  9:     { id: "settings", label: "?? Form Settings" },
 10:   ];
 11:
 12:   return (
 13:     <div className="bg-white dark:bg-gray-800 shadow-sm border-b sticky top-0 z-10">
 14:       <div className="container mx-auto px-4">
 15:         <div className="flex justify-between items-center py-4">
 16:           {/* Tabs */}
 17:           <div className="flex gap-2">
 18:             {tabs.map((tab) => (
 19:               <button
 20:                 key={tab.id}
 21:                 role="tab"
 22:                 aria-selected={activeTab === tab.id}
 23:                 className={`px-4 py-2 rounded-lg font-medium transition-all ${
 24:                   activeTab === tab.id
 25:                     ? "bg-green-600 text-white"
 26:                     : "bg-gray-200 text-gray-800 hover:bg-gray-300 dark:bg-gray-700 dark:text-gray-100 dark
 27:                 }`}
 28:                 onClick={() => setActiveTab(tab.id)}
 29:               >
 30:                 {tab.label}
 31:               </button>
 32:             ))}
```

```
          </div>

          {/* Theme Toggle */}
          <button
            onClick={toggleTheme}
            className="text-sm px-3 py-2 bg-gray-100 dark:bg-gray-700 dark:text-white rounded"
          >
            {theme === "dark" ? "?? Light" : "? Dark"}
          </button>
        </div>
      </div>
    </div>
  );
};

export default TabNavigation;
```

■ File: src\components\common\Toasts.js

```
import { ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";

const Toasts = () => {
  return (
    <>
      <ToastContainer
        position="top-center"
        autoClose={2000}
        hideProgressBar={false}
        newestOnTop={true}
        closeOnClick
        pauseOnFocusLoss
        draggable
        pauseOnHover
        closeButton={false}
        theme="light"
        limit={3}
        toastClassName="!bg-white/95 !backdrop-blur-md !rounded-xl !shadow-lg !shadow-black/10 !border !bor
        bodyClassName="!p-0 !m-0"
        progressClassName="!bg-gradient-to-r !from-blue-500 !to-cyan-500 !h-1"
        style={{
          top: "20px",
          left: "50%",
          transform: "translateX(-50%)",
          width: "400px",
          maxWidth: "90vw",
        }}
      />

      <style jsx global>{`
        .Toastify__toast-container {
          @apply font-sans;
        }

        .Toastify__toast--success {
          @apply !bg-green-50/95 !border-l-4 !border-l-green-500 !text-green-800;
        }

        .Toastify__toast--error {
          @apply !bg-red-50/95 !border-l-4 !border-l-red-500 !text-red-800;
        }

        .Toastify__toast--warning {
          @apply !bg-yellow-50/95 !border-l-4 !border-l-yellow-500 !text-yellow-800;
        }

        .Toastify__toast--info {
          @apply !bg-blue-50/95 !border-l-4 !border-l-blue-500 !text-blue-800;
        }
      `}</style>
    </>
```

```
 53:   );
 54: };
 55:
 56: export default Toasts;
```

■ File: src\components\form\FieldInputRow.js

==========

■ File: src\components\form\InputForm.js

==========

```
  1: import React, { useState, useEffect } from "react";
  2: import { useExcelExport } from "../../hooks/useExcelExport";
  3: import { useFormConfig } from "../../contexts/FormConfigContext";
  4: import { usePDFGeneration } from "../../hooks/usePDFGeneration";
  5: import { isValidScore } from "../../utils/helpers";
  6: import { toast } from "react-toastify";
  7:
  8: const LOCAL_STORAGE_KEY = "genomics_form_data";
  9:
 10: const InputForm = ({ onGenerateReport, reportData }) => {
 11:   const { state } = useFormConfig();
 12:   const { generatePDF } = usePDFGeneration();
 13:
 14:   // Try restoring from localStorage
 15:   const [formData, setFormData] = useState(() => {
 16:     try {
 17:       const stored = localStorage.getItem(LOCAL_STORAGE_KEY);
 18:       return stored ? JSON.parse(stored) : {};
 19:     } catch (e) {
 20:       return {};
 21:     }
 22:   });
 23:
 24:   const { exportToExcel } = useExcelExport();
 25:
 26:   const [errors, setErrors] = useState({});
 27:
 28:   // ? Save to localStorage on formData change
 29:   useEffect(() => {
 30:     localStorage.setItem(LOCAL_STORAGE_KEY, JSON.stringify(formData));
 31:   }, [formData]);
 32:
 33:   const handleInputChange = (fieldId, value) => {
 34:     const updatedValue = value.replace(/\D/g, "");
 35:     setFormData((prev) => ({
 36:       ...prev,
 37:       [fieldId]: updatedValue,
 38:     }));
 39:     setErrors((prev) => ({
 40:       ...prev,
 41:       [fieldId]: null,
 42:     }));
 43:   };
 44:
 45:   const validateForm = () => {
 46:     const newErrors = {};
 47:     state.fields.forEach((field) => {
 48:       const value = formData[field.id];
 49:       if (!isValidScore(value)) {
 50:         newErrors[field.id] = "Score must be between 1 and 10";
 51:       }
 52:     });
 53:     setErrors(newErrors);
 54:     return Object.keys(newErrors).length === 0;
 55:   };
 56:
 57:   const handleSubmit = (e) => {
 58:     e.preventDefault();
 59:     if (validateForm()) {
```

```
      onGenerateReport(formData);
      toast.success("Report generated and saved!");
    } else {
      toast.error("Please fix errors before submitting.");
    }
  };

  const handleDownloadPDF = () => {
    if (!reportData || reportData.length === 0) {
      toast.warning("Please generate a report first.");
      return;
    }
    generatePDF(reportData);
  };

  const handleClearForm = () => {
    if (window.confirm("Clear all form scores and reset saved state?")) {
      localStorage.removeItem(LOCAL_STORAGE_KEY);
      setFormData({});
      toast.info("?? Cleared saved input.");
    }
  };

  return (
    <div className="w-full md:w-1/2 bg-white p-4 md:p-8 overflow-y-auto mobile-section">
      {/* ... header & description remain the same */}

      <form onSubmit={handleSubmit} className="space-y-3 md:space-y-4">
        <div className="grid grid-cols-1 gap-4">
          {state.fields.map((field, index) => (
            <div
              key={field.id}
              className="flex flex-col md:flex-row md:items-center"
            >
              <label className="w-full md:w-48 text-sm font-semibold mb-1 md:mb-0">
                {field.label}
              </label>
              <input
                type="number"
                min="1"
                max="10"
                value={formData[field.id] || ""}
                onChange={(e) => handleInputChange(field.id, e.target.value)}
                className={`border p-2 w-full md:w-20 text-center rounded
                  ${errors[field.id] ? "border-red-500" : "border-gray-300"}`}
              />
              {errors[field.id] && (
                <p className="text-red-600 text-xs mt-1 md:ml-4">
                  {errors[field.id]}
                </p>
              )}
            </div>
          ))}
        </div>

        <div className="flex flex-col md:flex-row gap-2 mt-6">
          <button
            type="submit"
            className="bg-green-600 hover:bg-green-700 text-white px-6 py-3 rounded-lg font-semibold"
          >
            Generate Report
          </button>

          <button
            type="button"
            onClick={handleDownloadPDF}
            className="bg-blue-600 hover:bg-blue-700 text-white px-6 py-3 rounded-lg font-semibold"
          >
            Download PDF
          </button>

          <button
            type="button"
```

```
133:             onClick={handleClearForm}
134:             className="bg-gray-500 hover:bg-gray-600 text-white px-6 py-3 rounded-lg font-semibold"
135:           >
136:             Clear Input
137:           </button>
138:           <button
139:             type="button"
140:             onClick={() => exportToExcel(reportData)}
141:             className="bg-yellow-500 hover:bg-yellow-600 text-white px-6 py-3 rounded-lg font-semibold"
142:           >
143:             ? Export Excel
144:           </button>
145:         </div>
146:       </form>
147:     </div>
148:   );
149: };
150:
151: export default InputForm;
```

■ File: src\components\report\PDFPreview.js

==========

■ File: src\components\report\ReportOutput.js

==========

```
  1: import React from "react";
  2: import { useFormConfig } from "../../contexts/FormConfigContext";
  3:
  4: const ReportOutput = ({ reportData }) => {
  5:   const { state } = useFormConfig();
  6:
  7:   // Assign color to score
  8:   const getScoreColor = (score) => {
  9:     if (score >= state.highThreshold) return "bg-red-600";
 10:     if (score >= 4) return "bg-yellow-500";
 11:     return "bg-green-600";
 12:   };
 13:
 14:   // Style active/inactive text
 15:   const getTextStyle = (isActive) =>
 16:     isActive ? "text-gray-900 font-semibold" : "text-gray-400";
 17:
 18:   if (!reportData || reportData.length === 0) {
 19:     return (
 20:       <div className="w-full md:w-1/2 p-4 md:p-8 mobile-section bg-gray-50">
 21:         <div className="bg-white border border-gray-300 rounded-lg p-8 text-center text-gray-500">
 22:           Fill in the form and generate report to view results here.
 23:         </div>
 24:       </div>
 25:     );
 26:   }
 27:
 28:   let currentCategory = null;
 29:
 30:   return (
 31:     <div className="w-full md:w-1/2 p-4 md:p-8 mobile-section bg-gray-50 overflow-y-auto">
 32:       <div className="bg-white border border-gray-300 rounded-lg shadow-sm overflow-hidden">
 33:         {reportData.map((item, index) => {
 34:           const { field, score, showHigh, showNormal, showLow } = item;
 35:           const isNewCategory =
 36:             field.category && field.category !== currentCategory;
 37:           const elements = [];
 38:
 39:           // Add category header if changed
 40:           if (isNewCategory) {
 41:             currentCategory = field.category;
 42:             elements.push(
 43:               <div
 44:                 key={`category-${field.category}`}
```

```
                className="bg-gray-200 px-4 py-2 font-bold text-sm text-gray-700 border-l-4 border-gray-400
              >
                {currentCategory}
              </div>
            );
          }

          // Add field row
          elements.push(
            <div
              key={field.id}
              className="flex flex-col md:flex-row items-stretch border-b border-gray-200"
            >
              {/* Field Label */}
              <div className="w-full md:w-48 px-3 py-3 text-center md:text-right bg-gray-100 md:bg-white">
                <div className="text-xs font-bold text-gray-700 uppercase leading-tight">
                  {field.label}
                </div>
              </div>

              {/* Score */}
              <div className="w-full md:w-16 flex justify-center items-center py-3">
                <div
                  className={`w-10 h-10 ${getScoreColor(
                    score
                  )} text-white font-bold text-lg flex items-center justify-center rounded-full`}
                >
                  {score}
                </div>
              </div>

              {/* Recommendations */}
              <div className="flex-1 px-3 py-3 flex flex-col md:flex-row">
                {/* High */}
                <div className="w-full md:w-1/3 md:pr-2 mb-3 md:mb-0">
                  <div
                    className={`text-xs font-bold mb-1 ${getTextStyle(
                      showHigh
                    )}`}
                  >
                    HIGH
                  </div>
                  <div
                    className={`text-xs leading-tight ${getTextStyle(
                      showHigh
                    )}`}
                  >
                    {field.high.split("\n").map((line, i, arr) => (
                      <React.Fragment key={i}>
                        {line}
                        {i < arr.length - 1 && <br />}
                      </React.Fragment>
                    ))}
                  </div>
                </div>

                {/* Normal */}
                <div className="w-full md:w-1/3 md:px-2 mb-3 md:mb-0">
                  <div
                    className={`text-xs font-bold mb-1 ${getTextStyle(
                      showNormal
                    )}`}
                  >
                    NORMAL
                  </div>
                  <div
                    className={`text-xs leading-tight ${getTextStyle(
                      showNormal
                    )}`}
                  >
                    {field.normal?.split("\n").map((line, i, arr) => (
                      <React.Fragment key={i}>
                        {line}
```

```
                        {i < arr.length - 1 && <br />}
                      </React.Fragment>
                    ))}
                  </div>
                </div>

                {/* Low */}
                <div className="w-full md:w-1/3 md:pl-2">
                  <div
                    className={`text-xs font-bold mb-1 ${getTextStyle(
                      showLow
                    )}`}
                  >
                    LOW
                  </div>
                  <div
                    className={`text-xs leading-tight ${getTextStyle(showLow)}`}
                  >
                    {field.low.split("\n").map((line, i, arr) => (
                      <React.Fragment key={i}>
                        {line}
                        {i < arr.length - 1 && <br />}
                      </React.Fragment>
                    ))}
                  </div>
                </div>
              </div>
            </div>
          );

          return elements;
        })}
      </div>
    </div>
  );
};

export default ReportOutput;
```

■ File: src\components\settings\CategoryManager.js

```
import { useState, useEffect } from "react";
import { toast } from "react-toastify";

const API_URL = "http://localhost:5000";

const CategoryManager = () => {
  const [categories, setCategories] = useState([]);
  const [newCategory, setNewCategory] = useState("");
  const [editIndex, setEditIndex] = useState(null);
  const [editedName, setEditedName] = useState("");

  // ? Fetch from server
  const fetchCategories = async () => {
    try {
      const res = await fetch(`${API_URL}/categories`);
      const data = await res.json();
      setCategories(data);
    } catch (err) {
      toast.error("? Failed to load categories");
    }
  };

  useEffect(() => {
    fetchCategories();
  }, []);

  // ? Add
  const addCategory = async () => {
    const trimmed = newCategory.trim();
    if (!trimmed) return;
```

```

    const exists = categories.some((c) => c.name === trimmed);
    if (exists) {
      toast.error("? Category already exists");
      return;
    }

    const res = await fetch(`${API_URL}/categories`, {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ name: trimmed }),
    });

    if (res.ok) {
      toast.success("? Category added");
      setNewCategory("");
      fetchCategories(); // ? refresh list
    }
  };

  // ?? Start editing
  const startEdit = (i, name) => {
    setEditIndex(i);
    setEditedName(name);
  };

  // ? Confirm edit
  const confirmEdit = async () => {
    if (!editedName.trim()) return;

    const cat = categories[editIndex];

    const res = await fetch(`${API_URL}/categories/${cat.id}`, {
      method: "PUT",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ ...cat, name: editedName.trim() }),
    });

    if (res.ok) {
      toast.success("?? Category updated");
      setEditIndex(null);
      setEditedName("");
      fetchCategories(); // ? refresh
    } else {
      toast.error("? Failed to update category");
    }
  };

  // ?? Delete
  const deleteCategory = async (index) => {
    const cat = categories[index];

    if (!window.confirm(`Delete category "${cat.name}"?`)) return;

    const res = await fetch(`${API_URL}/categories/${cat.id}`, {
      method: "DELETE",
    });

    if (res.ok) {
      toast.info("?? Category deleted");
      fetchCategories(); // ? refresh
    } else {
      toast.error("? Failed to delete category");
    }
  };

  return (
    <div className="mt-10 border-t pt-6">
      <h3 className="text-xl font-semibold mb-4">?? Category Manager</h3>

      <div className="flex gap-2 mb-4">
        <input
          type="text"
```

```
104:           className="border px-3 py-2 rounded w-full"
105:           placeholder="New category name"
106:           value={newCategory}
107:           onChange={(e) => setNewCategory(e.target.value)}
108:         />
109:         <button
110:           onClick={addCategory}
111:           className="bg-green-600 hover:bg-green-700 text-white px-4 py-2 rounded"
112:         >
113:           ? Add
114:         </button>
115:       </div>
116:
117:       <div className="space-y-3">
118:         {categories.map((cat, i) => (
119:           <div
120:             key={cat.id}
121:             className="flex items-center justify-between bg-white border p-3 rounded"
122:           >
123:             {editIndex === i ? (
124:               <>
125:                 <input
126:                   type="text"
127:                   className="border px-2 py-1 rounded w-full mr-2"
128:                   value={editedName}
129:                   onChange={(e) => setEditedName(e.target.value)}
130:                 />
131:                 <button
132:                   onClick={confirmEdit}
133:                   className="bg-blue-600 text-white px-3 py-1 rounded"
134:                 >
135:                   ?
136:                 </button>
137:               </>
138:             ) : (
139:               <>
140:                 <span>{cat.name}</span>
141:                 <div className="flex gap-2">
142:                   <button
143:                     onClick={() => startEdit(i, cat.name)}
144:                     className="text-blue-600 hover:underline"
145:                   >
146:                     ?? Edit
147:                   </button>
148:                   <button
149:                     onClick={() => deleteCategory(i)}
150:                     className="text-red-600 hover:underline"
151:                   >
152:                     ?? Delete
153:                   </button>
154:                 </div>
155:               </>
156:             )}
157:           </div>
158:         ))}
159:       </div>
160:     </div>
161:   );
162: };
163:
164: export default CategoryManager;
```

■ File: src\components\settings\SettingsPanel.js

```
  1: import React, { useState, useRef } from "react";
  2: import { useFormConfig } from "../../contexts/FormConfigContext";
  3: import { useConfigImportExport } from "../../hooks/useConfigImportExport";
  4: import { toast } from "react-toastify";
  5: import ResetAll from "../common/ResetAll";
  6: import CategoryManager from "./CategoryManager";
  7:
```

```
const API_URL = "http://localhost:5000";

const SettingsPanel = () => {
  const { state, dispatch } = useFormConfig();

  const [settings, setSettings] = useState({
    title: state.title,
    quote: state.quote,
    description: state.description,
    headerColor: state.headerColor,
    highThreshold: state.highThreshold,
    colors: {
      low: state.colors.low,
      medium: state.colors.medium,
      high: state.colors.high,
    },
  });

  const fileInputRef = useRef(null);
  const { exportConfig, importConfig } = useConfigImportExport();

  const handleImportClick = () => {
    fileInputRef.current?.click();
  };

  const handleFileSelected = (e) => {
    const file = e.target.files?.[0];
    if (!file || !file.name.endsWith(".json")) {
      toast.error("Please upload a valid JSON file.");
      return;
    }
    importConfig(file);
    e.target.value = ""; // reset input
  };

  // ? Update local state as user types
  const handleChange = (key, value) => {
    setSettings((prev) => ({
      ...prev,
      [key]: value,
    }));
  };

  const handleColorChange = (level, value) => {
    setSettings((prev) => ({
      ...prev,
      colors: {
        ...prev.colors,
        [level]: value,
      },
    }));
  };

  // ? Save to backend
  const applySettings = async () => {
    try {
      const res = await fetch(`${API_URL}/settings`, {
        method: "PUT",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(settings),
      });

      if (!res.ok) throw new Error("Failed to save settings");

      dispatch({ type: "UPDATE_SETTINGS", settings }); // update UI too
      toast.success("? Settings saved to server!");
    } catch (error) {
      console.error(error);
      toast.error("? Failed to save settings");
    }
  };

  const resetSettings = () => {
```

```
    if (
      window.confirm("Are you sure you want to reset all settings to default?")
    ) {
      window.location.reload(); // simplest way
    }
  };

  return (
    <div className="max-w-4xl mx-auto px-4 py-8">
      <h2 className=" bg-white text-2xl font-bold mb-6">
        ?? Form Customization
      </h2>

      <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
        {/* Left Section */}
        <div className="bg-white rounded-lg shadow-sm p-6 space-y-4">
          <h3 className="text-lg font-semibold">? Header Info</h3>

          <div>
            <label className="block text-sm font-medium mb-1">Main Title</label>
            <input
              type="text"
              value={settings.title}
              onChange={(e) => handleChange("title", e.target.value)}
              className="w-full border rounded px-3 py-2"
            />
          </div>

          <div>
            <label className="block text-sm font-medium mb-1">Quote</label>
            <textarea
              rows={2}
              value={settings.quote}
              onChange={(e) => handleChange("quote", e.target.value)}
              className="w-full border rounded px-3 py-2"
            />
          </div>

          <div>
            <label className="block text-sm font-medium mb-1">
              Description
            </label>
            <textarea
              rows={4}
              value={settings.description}
              onChange={(e) => handleChange("description", e.target.value)}
              className="w-full border rounded px-3 py-2"
            />
          </div>

          <div>
            <label className="block text-sm font-medium mb-1">
              Header Background Color
            </label>
            <input
              type="color"
              value={settings.headerColor}
              onChange={(e) => handleChange("headerColor", e.target.value)}
              className="h-10 w-full border rounded"
            />
          </div>
        </div>

        {/* Right Section */}
        <div className="bg-white rounded-lg shadow-sm p-6 space-y-4">
          <h3 className="text-lg font-semibold">? Score Logic</h3>

          <div>
            <label className="block text-sm font-medium mb-1">
              High Score Threshold (?)
            </label>
            <input
              type="number"
```

```
154:               min="1"
155:               max="10"
156:               value={settings.highThreshold}
157:               onChange={(e) =>
158:                 handleChange("highThreshold", parseInt(e.target.value))
159:               }
160:               className="w-full border rounded px-3 py-2"
161:             />
162:           </div>
163:
164:           <h3 className="text-lg font-semibold mt-6">? Score Colors</h3>
165:
166:           <div className="space-y-2">
167:             <ColorInput
168:               label="High"
169:               value={settings.colors.high}
170:               onChange={(val) => handleColorChange("high", val)}
171:             />
172:             <ColorInput
173:               label="Medium"
174:               value={settings.colors.medium}
175:               onChange={(val) => handleColorChange("medium", val)}
176:             />
177:             <ColorInput
178:               label="Low"
179:               value={settings.colors.low}
180:               onChange={(val) => handleColorChange("low", val)}
181:             />
182:           </div>
183:         </div>
184:       </div>
185:
186:       {/* Buttons */}
187:       <div className="mt-6 flex flex-wrap gap-4">
188:         <button
189:           onClick={applySettings}
190:           className="bg-green-600 hover:bg-green-700 text-white px-6 py-3 rounded-lg font-semibold"
191:         >
192:           ? Apply Settings
193:         </button>
194:         <button
195:           onClick={resetSettings}
196:           className="bg-gray-600 hover:bg-gray-700 text-white px-6 py-3 rounded-lg font-semibold"
197:         >
198:           ?? Reset to Default
199:         </button>
200:         <button
201:           onClick={exportConfig}
202:           className="bg-blue-600 hover:bg-blue-700 text-white px-6 py-3 rounded-lg font-semibold"
203:         >
204:           ? Export Config (.json)
205:         </button>
206:
207:         <button
208:           onClick={handleImportClick}
209:           className="bg-purple-600 hover:bg-purple-700 text-white px-6 py-3 rounded-lg font-semibold"
210:         >
211:           ? Import Config (.json)
212:         </button>
213:
214:         <div className="mt-2">
215:           <ResetAll />
216:         </div>
217:
218:         <input
219:           type="file"
220:           accept=".json"
221:           ref={fileInputRef}
222:           className="hidden"
223:           onChange={handleFileSelected}
224:         />
225:       </div>
226:
```

```
      {/* ?? Category management (uses API now) */}
      <div className="mt-6">
        <CategoryManager />
      </div>
    </div>
  );
};

const ColorInput = ({ label, value, onChange }) => (
  <div>
    <label className="block text-sm font-medium mb-1">{label}</label>
    <input
      type="color"
      value={value}
      onChange={(e) => onChange(e.target.value)}
      className="h-10 w-full border rounded"
    />
  </div>
);

export default SettingsPanel;
```

■ File: src\contexts\FormConfigContext.js

```
import React, { createContext, useContext, useReducer, useEffect } from "react";

// Base URL of your json-server
const API_URL = "http://localhost:5000";

const formConfigReducer = (state, action) => {
  switch (action.type) {
    case "IMPORT_CONFIG":
      return { ...action.config };

    case "ADD_FIELD":
      return { ...state, fields: [...state.fields, action.field] };

    case "UPDATE_FIELD":
      if (action.property === "full") {
        return {
          ...state,
          fields: state.fields.map((field, i) =>
            i === action.index ? action.value : field
          ),
        };
      }
      return {
        ...state,
        fields: state.fields.map((field, i) =>
          i === action.index
            ? { ...field, [action.property]: action.value }
            : field
        ),
      };

    case "DELETE_FIELD":
      return {
        ...state,
        fields: state.fields.filter((_, i) => i !== action.index),
      };

    case "REORDER_FIELDS":
      return { ...state, fields: action.fields };

    case "UPDATE_SETTINGS":
      return { ...state, ...action.settings };

    case "ADD_CATEGORY":
      return {
        ...state,
        categories: [
```

```
          ...state.categories,
          { id: Date.now(), name: action.name },
        ],
      };

    case "UPDATE_CATEGORY":
      return {
        ...state,
        categories: state.categories.map((cat, i) =>
          i === action.index ? { ...cat, name: action.newName } : cat
        ),
        fields: state.fields.map((field) =>
          field.category === state.categories[action.index]?.name
            ? { ...field, category: action.newName }
            : field
        ),
      };

    case "DELETE_CATEGORY":
      const catName = state.categories[action.index]?.name;
      return {
        ...state,
        categories: state.categories.filter((_, i) => i !== action.index),
        fields: state.fields.map((field) =>
          field.category === catName ? { ...field, category: "" } : field
        ),
      };

    default:
      return state;
  }
};

const FormConfigContext = createContext(null);

export const FormConfigProvider = ({ children }) => {
  const [state, dispatch] = useReducer(formConfigReducer, {
    title: "",
    quote: "",
    description: "",
    headerColor: "",
    colors: { low: "", medium: "", high: "" },
    highThreshold: 6,
    categories: [],
    fields: [],
  });

  // ? Load config from json-server at startup
  useEffect(() => {
    const loadFromServer = async () => {
      try {
        const [settingsRes, categoriesRes, fieldsRes] = await Promise.all([
          fetch(`${API_URL}/settings`),
          fetch(`${API_URL}/categories`),
          fetch(`${API_URL}/fields`),
        ]);

        const settings = await settingsRes.json();
        const categories = await categoriesRes.json();
        const fields = await fieldsRes.json();

        dispatch({
          type: "IMPORT_CONFIG",
          config: {
            ...settings,
            categories, // array of { id, name }
            fields, // array of full field objects
          },
        });
      } catch (err) {
        console.error("Failed to fetch config from API", err);
      }
    };
```

```
121:
122:     loadFromServer();
123:   }, []);
124:
125:   return (
126:     <FormConfigContext.Provider value={{ state, dispatch }}>
127:       {children}
128:     </FormConfigContext.Provider>
129:   );
130: };
131:
132: export const useFormConfig = () => {
133:   const context = useContext(FormConfigContext);
134:   if (!context)
135:     throw new Error("useFormConfig must be used within a FormConfigProvider");
136:   return context;
137: };
```

■ File: src\contexts\ThemeContext.js

```
  1: import React, { createContext, useEffect, useState, useContext } from "react";
  2:
  3: const ThemeContext = createContext();
  4:
  5: export const ThemeProvider = ({ children }) => {
  6:   const [theme, setTheme] = useState("light");
  7:
  8:   useEffect(() => {
  9:     const stored = localStorage.getItem("theme");
 10:     if (stored === "dark") {
 11:       document.documentElement.classList.add("dark");
 12:       setTheme("dark");
 13:     }
 14:   }, []);
 15:
 16:   const toggleTheme = () => {
 17:     const nextTheme = theme === "dark" ? "light" : "dark";
 18:     setTheme(nextTheme);
 19:     localStorage.setItem("theme", nextTheme);
 20:     document.documentElement.classList.toggle("dark");
 21:   };
 22:
 23:   return (
 24:     <ThemeContext.Provider value={{ theme, toggleTheme }}>
 25:       {children}
 26:     </ThemeContext.Provider>
 27:   );
 28: };
 29:
 30: export const useTheme = () => useContext(ThemeContext);
```

■ File: src\hooks\useConfigImportExport.js

```
  1: import { useCallback } from "react";
  2: import { useFormConfig } from "../contexts/FormConfigContext";
  3:
  4: const API_URL = "http://localhost:5000";
  5:
  6: export const useConfigImportExport = () => {
  7:   const { state, dispatch } = useFormConfig();
  8:
  9:   // ?? Export config from current state to file
 10:   const exportConfig = useCallback(() => {
 11:     const dataStr = JSON.stringify(state, null, 2);
 12:     const blob = new Blob([dataStr], { type: "application/json" });
 13:     const url = URL.createObjectURL(blob);
 14:
 15:     const link = document.createElement("a");
 16:     link.href = url;
```

```
    link.download = "genomics-form-config.json";
    link.click();
    URL.revokeObjectURL(url);
  }, [state]);

  // ?? Import config and write to all backend endpoints
  const importConfig = useCallback(
    async (file) => {
      const reader = new FileReader();

      reader.onload = async (e) => {
        try {
          const parsed = JSON.parse(e.target.result);

          // Validate structure
          if (
            !parsed.fields ||
            !parsed.categories ||
            !parsed.title ||
            !parsed.colors
          ) {
            alert("Invalid configuration file.");
            return;
          }

          // ?? Overwrite ALL current data via PUT/DELETE/POST
          await Promise.all([
            // Clear old fields
            fetch(`${API_URL}/fields`)
              .then((res) => res.json())
              .then((existing) =>
                Promise.all(
                  existing.map((f) =>
                    fetch(`${API_URL}/fields/${f.id}`, { method: "DELETE" })
                  )
                )
              ),

            // Clear old categories
            fetch(`${API_URL}/categories`)
              .then((res) => res.json())
              .then((existing) =>
                Promise.all(
                  existing.map((c) =>
                    fetch(`${API_URL}/categories/${c.id}`, { method: "DELETE" })
                  )
                )
              ),
          ]);

          // ? Upload settings
          await fetch(`${API_URL}/settings`, {
            method: "PUT",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify({
              title: parsed.title,
              quote: parsed.quote,
              description: parsed.description,
              headerColor: parsed.headerColor,
              colors: parsed.colors,
              highThreshold: parsed.highThreshold,
            }),
          });

          // ? Upload categories
          for (const cat of parsed.categories) {
            await fetch(`${API_URL}/categories`, {
              method: "POST",
              headers: { "Content-Type": "application/json" },
              body: JSON.stringify(
                typeof cat === "string" ? { name: cat } : cat
              ),
            });
```

```
 90:           }
 91:
 92:           // ? Upload fields
 93:           for (const field of parsed.fields) {
 94:             await fetch(`${API_URL}/fields`, {
 95:               method: "POST",
 96:               headers: { "Content-Type": "application/json" },
 97:               body: JSON.stringify(field),
 98:             });
 99:           }
100:
101:           // ? Dispatch to update UI immediately
102:           dispatch({ type: "IMPORT_CONFIG", config: parsed });
103:           alert("? Configuration imported and saved to server.");
104:         } catch (err) {
105:           console.error(err);
106:           alert("? Error importing config: " + err.message);
107:         }
108:       };
109:
110:       reader.readAsText(file);
111:     },
112:     [dispatch]
113:   );
114:
115:   return { exportConfig, importConfig };
116: };
```

■ File: src\hooks\useExcelExport.js

```
  1: import * as XLSX from "xlsx";
  2:
  3: export const useExcelExport = () => {
  4:   const exportToExcel = (reportData) => {
  5:     if (!reportData || reportData.length === 0) {
  6:       alert("No report data to export.");
  7:       return;
  8:     }
  9:
 10:     const exportRows = reportData.map((item) => {
 11:       const { field, score, showHigh, showNormal, showLow } = item;
 12:
 13:       let recommendation = "";
 14:       if (showHigh) recommendation = field.high;
 15:       else if (showNormal) recommendation = field.normal;
 16:       else if (showLow) recommendation = field.low;
 17:
 18:       return {
 19:         Field: field.label,
 20:         Category: field.category || "Uncategorized",
 21:         Score: score,
 22:         Recommendation: recommendation.replace(/\n/g, " "),
 23:       };
 24:     });
 25:
 26:     const worksheet = XLSX.utils.json_to_sheet(exportRows);
 27:     const workbook = XLSX.utils.book_new();
 28:     XLSX.utils.book_append_sheet(workbook, worksheet, "Genomics Report");
 29:
 30:     XLSX.writeFile(workbook, "genomics_diet_report.xlsx");
 31:   };
 32:
 33:   return { exportToExcel };
 34: };
```

■ File: src\hooks\usePDFGeneration.js

```
  1: import { useCallback } from "react";
  2: import { jsPDF } from "jspdf";
```

```
import { useFormConfig } from "../contexts/FormConfigContext";
import { hexToRgb } from "../utils/helpers";

export const usePDFGeneration = () => {
  const { state } = useFormConfig();

  // Base64 logo (optional)
  const leftLogoUrl = "/left.png";
  const rightLogoUrl = "/right.png";

  const generatePDF = useCallback(
    (reportData) => {
      if (!reportData || reportData.length === 0) {
        alert("No report data found.");
        return;
      }

      const doc = new jsPDF();
      const pageHeight = doc.internal.pageSize.height;
      const pageWidth = doc.internal.pageSize.width;
      const margin = 10;
      let y = 20;

      // ----------------------------
      // Header section
      // ----------------------------
      doc.setFillColor(...hexToRgb(state.headerColor));
      doc.rect(margin, y, pageWidth - margin * 2, 20, "F");
      doc.setTextColor(255, 255, 255);
      doc.setFontSize(16);
      doc.setFont(undefined, "bold");
      doc.text(state.title, pageWidth / 2, y + 13, { align: "center" });
      y += 30;

      // Quote
      doc.setTextColor(0, 0, 0);
      doc.setFontSize(11);
      doc.setFont(undefined, "bold");
      doc.text(state.quote, pageWidth / 2, y, { align: "center" });
      y += 10;

      // Description
      doc.setFont(undefined, "normal");
      doc.setFontSize(10);
      const descLines = doc.splitTextToSize(
        state.description,
        pageWidth - 2 * margin
      );
      doc.text(descLines, margin, y);
      y += descLines.length * 5 + 5;

      let currentCategory = null;

      reportData.forEach((item, index) => {
        const { field, score, showHigh, showNormal, showLow } = item;

        // Insert page break if needed
        const estimatedFieldHeight = 40; // Rough estimate
        if (y + estimatedFieldHeight > pageHeight - 20) {
          doc.addPage();
          y = 20;
        }

        // Render category title if needed
        if (field.category && field.category !== currentCategory) {
          currentCategory = field.category;
          doc.setFontSize(12);
          doc.setFont(undefined, "bold");
          doc.setTextColor(50, 50, 50);
          doc.text(currentCategory, margin, y);
          y += 8;
        }

```

```
        // Field Label
        doc.setFontSize(10);
        doc.setFont(undefined, "bold");
        doc.setTextColor(0, 0, 0);
        doc.text(field.label, margin, y);
        y += 6;

        // Score Circle
        const circleX = margin + 5;
        doc.setDrawColor(0);
        const rgb =
          score >= state.highThreshold
            ? hexToRgb(state.colors.high)
            : score >= 4
            ? hexToRgb(state.colors.medium)
            : hexToRgb(state.colors.low);

        doc.setFillColor(...rgb);
        doc.circle(circleX, y + 5, 4, "FD");
        doc.setTextColor(255, 255, 255);
        doc.setFontSize(8);
        doc.text(String(score), circleX, y + 6, { align: "center" });

        y += 12;

        // Render matching text
        const renderTextBlock = (label, text, active) => {
          if (!text) return;
          doc.setFontSize(9);
          doc.setFont(undefined, "bold");
          doc.setTextColor(
            active ? 0 : 180,
            active ? 0 : 180,
            active ? 0 : 180
          );
          doc.text(`${label}:`, margin, y);
          y += 5;

          doc.setFont(undefined, "normal");
          const lines = doc.splitTextToSize(text, pageWidth - 2 * margin);
          lines.forEach((line) => {
            if (y + 6 > pageHeight - 15) {
              doc.addPage();
              y = 20;
            }
            doc.text(line, margin, y);
            y += 5;
          });
          y += 2;
        };

        renderTextBlock("HIGH", field.high, showHigh);
        renderTextBlock("NORMAL", field.normal, showNormal);
        renderTextBlock("LOW", field.low, showLow);

        y += 4;
      });

      // Footer (optional logos)
      const addLogos = () => {
        try {
          doc.addImage(leftLogoUrl, "PNG", 10, pageHeight - 20, 30, 10);
          doc.addImage(
            rightLogoUrl,
            "PNG",
            pageWidth - 40,
            pageHeight - 20,
            30,
            10
          );
        } catch (e) {
          console.warn("Logo failed to load. Skipping...");
        }
```

```
149:       };
150:       addLogos();
151:
152:       // Save file
153:       doc.save("genomics-diet-report.pdf");
154:     },
155:     [state]
156:   );
157:
158:   return { generatePDF };
159: };
```

■ File: src\hooks\useReportGeneration.js

```
================================================================================
  1: import { useState, useCallback } from 'react';
  2: import { useFormConfig } from '../contexts/FormConfigContext';
  3: import { isValidScore } from '../utils/helpers';
  4:
  5: /**
  6:  * Hook: useReportGeneration
  7:  * Transforms form input into structured report data based on score thresholds
  8:  */
  9: export const useReportGeneration = () => {
 10:   const { state } = useFormConfig(); // Get field config and settings from context
 11:   const [reportData, setReportData] = useState([]);
 12:
 13:   /**
 14:    * generateReport
 15:    * @param {Object} formData - { fieldId: score }
 16:    */
 17:   const generateReport = useCallback((formData) => {
 18:     const processedData = [];
 19:
 20:     // Loop through all fields from config
 21:     state.fields.forEach((field) => {
 22:       const rawValue = formData[field.id];
 23:       const score = parseInt(rawValue);
 24:
 25:       if (!isValidScore(score)) {
 26:         return; // Skip invalid scores
 27:       }
 28:
 29:       // Determine logic: high / normal / low
 30:       const isHigh = score >= state.highThreshold;
 31:       const isNormal = score >= 4 && score < state.highThreshold;
 32:       const isLow = score < 4;
 33:
 34:       processedData.push({
 35:         field,       // full field config
 36:         score,       // numeric score
 37:         showHigh: isHigh,
 38:         showNormal: isNormal,
 39:         showLow: isLow,
 40:       });
 41:     });
 42:
 43:     // Update state
 44:     setReportData(processedData);
 45:
 46:     // Return for immediate use
 47:     return processedData;
 48:   }, [state.fields, state.highThreshold]);
 49:
 50:   return { reportData, generateReport };
 51: };
```

■ File: src\index.css

```
================================================================================
  1: /* Tailwind's base styles */
  2: @tailwind base;
  3: @tailwind components;
  4: @tailwind utilities;
  5:
  6: /* Custom global styles */
  7: @layer base {
  8:   /* HTML transition for smooth dark mode switching */
  9:   html {
 10:     transition: background-color 0.3s ease, color 0.3s ease;
 11:   }
 12:
 13:   /* Body settings for light and dark mode */
 14:   body {
 15:     @apply bg-white text-gray-900 dark:bg-gray-900 dark:text-white; /* Global body colors */
 16:   }
 17:
 18:   /* Customizations for links */
 19:   a {
 20:     @apply text-blue-600 dark:text-blue-400; /* Links: light mode blue, dark mode lighter blue */
 21:   }
 22:
 23:   /* Buttons with default light/dark mode background */
 24:   button {
 25:     @apply bg-gray-300 dark:bg-gray-700 text-gray-800 dark:text-gray-100; /* Default button styles */
 26:   }
 27:
 28:   /* Background color for any white-background elements */
 29:   .bg-white {
 30:     @apply dark:bg-gray-900; /* Switches background color to dark mode */
 31:   }
 32:
 33:   /* Text color for general text */
 34:   .text-gray-900 {
 35:     @apply dark:text-white; /* Changes text color to white in dark mode */
 36:   }
 37:
 38:   /* Form elements: inputs, textareas, and selects */
 39:   input,
 40:   textarea,
 41:   select {
 42:     @apply bg-white dark:bg-gray-800 text-black dark:text-white border dark:border-gray-700;
 43:   }
 44:
 45:   /* Focused state of inputs, textareas, and selects */
 46:   input:focus,
 47:   textarea:focus,
 48:   select:focus {
 49:     @apply ring-2 ring-blue-500 dark:ring-blue-300;
 50:   }
 51:
 52:   /* Global font and smoothing */
 53:   body {
 54:     margin: 0;
 55:     font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto",
 56:       "Oxygen", "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans",
 57:       "Helvetica Neue", sans-serif;
 58:     -webkit-font-smoothing: antialiased;
 59:     -moz-osx-font-smoothing: grayscale;
 60:   }
 61:
 62:   /* Code styles for code blocks */
 63:   code {
 64:     font-family: source-code-pro, Menlo, Monaco, Consolas, "Courier New",
 65:       monospace;
 66:   }
 67:
 68:   /* Global styling for text input elements */
 69:   input[type="text"],
 70:   input[type="number"],
 71:   input[type="email"],
```

```
  72:   textarea,
  73:   select {
  74:     @apply border-2 rounded-lg p-2 dark:border-gray-600;
  75:   }
  76:
  77:   /* Button hover and focus states */
  78:   button:hover {
  79:     @apply bg-gray-200 dark:bg-gray-700; /* Darker button on hover */
  80:   }
  81:
  82:   /* Ensuring links are visible in dark mode */
  83:   .text-blue-600 {
  84:     @apply dark:text-blue-400; /* Light blue in normal mode, changes to darker in dark mode */
  85:   }
  86:
  87:   /* Customize card elements for dark mode */
  88:   .card {
  89:     @apply bg-white dark:bg-gray-800 text-black dark:text-white border dark:border-gray-700;
  90:   }
  91:
  92:   /* Customize borders */
  93:   .border-gray-300 {
  94:     @apply dark:border-gray-600; /* Change border color in dark mode */
  95:   }
  96:
  97:   /* Ensure smooth transitions when toggling dark/light mode */
  98:   .transition-all {
  99:     transition: all 0.3s ease; /* Apply smooth transitions */
 100:   }
 101: }
```

■ File: src\index.js

```
  1: import React from 'react';
  2: import ReactDOM from 'react-dom/client';
  3: import './index.css';
  4: import App from './App';
  5: import reportWebVitals from './reportWebVitals';
  6:
  7: const root = ReactDOM.createRoot(document.getElementById('root'));
  8: root.render(
  9:   <React.StrictMode>
 10:     <App />
 11:   </React.StrictMode>
 12: );
 13:
 14: // If you want to start measuring performance in your app, pass a function
 15: // to log results (for example: reportWebVitals(console.log))
 16: // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
 17: reportWebVitals();
```

■ File: src\logo.svg

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 841.9 595.3"><g fill="#61DAFB"><path d="M666.3 296.5c0-32.5
```

■ File: src\reportWebVitals.js

```
  1: const reportWebVitals = onPerfEntry => {
  2:   if (onPerfEntry && onPerfEntry instanceof Function) {
  3:     import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => {
  4:       getCLS(onPerfEntry);
  5:       getFID(onPerfEntry);
  6:       getFCP(onPerfEntry);
  7:       getLCP(onPerfEntry);
  8:       getTTFB(onPerfEntry);
  9:     });
 10:   }
```

```
 11: };
 12:
 13: export default reportWebVitals;
```

■ File: src\setupTests.js

```
  1: // jest-dom adds custom jest matchers for asserting on DOM nodes.
  2: // allows you to do things like:
  3: // expect(element).toHaveTextContent(/react/i)
  4: // learn more: https://github.com/testing-library/jest-dom
  5: import '@testing-library/jest-dom';
```

■ File: src\utils\constants.js

■ File: src\utils\helpers.js

```
  1: // Convert HEX color to RGB array for jsPDF
  2: export function hexToRgb(hex) {
  3:   const result = /^#?([a-f\d]{2})([a-f\d]{2})([a-f\d]{2})$/i.exec(hex);
  4:   return result
  5:     ? [
  6:         parseInt(result[1], 16),
  7:         parseInt(result[2], 16),
  8:         parseInt(result[3], 16)
  9:       ]
 10:     : [0, 0, 0];
 11: }
 12:
 13: // Simple validation helper
 14: export const isValidScore = (value) => {
 15:   const num = parseInt(value);
 16:   return !isNaN(num) && num >= 1 && num <= 10;
 17: };
```

■ File: tailwind.config.js

```
  1: /** @type {import('tailwindcss').Config} */
  2: module.exports = {
  3:   darkMode: "class", // ? enables class-based dark mode
  4:   content: ["./src/**/*.{js,jsx}"],
  5:   theme: {
  6:     extend: {},
  7:   },
  8:   plugins: [],
  9: };
```
