# ■ Project Code Export

## ■ Frontend File List:

- .gitignore
- Frontend_Code_Export.pdf
- 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\data\formConfig.json
- 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\styles\index.css
- 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: 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:
```

```
def get_code_files(directory, excluded_files=None, excluded_dirs=None):
    """Fetch all project files except specified exclusions."""
    if excluded_files is None:
        excluded_files = {'package.json', 'package-lock.json'}

    if excluded_dirs is None:
        excluded_dirs = {'node_modules', '.git', '__pycache__', 'build', '.next', 'dist'}

    code_files = {}

    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
```

```
152:     excluded_dirs = {
153:         'node_modules',
154:         '.git',
155:         '__pycache__',
156:         'build',
157:         'dist',
158:         '.next',
159:         'coverage',
160:         '.nyc_output',
161:         '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 } 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 AppContent = () => {
 11:   const [activeTab, setActiveTab] = useState('form');
 12:   const { reportData, generateReport } = useReportGeneration();
 13:
 14:   const handleGenerateReport = (formData) => {
 15:     generateReport(formData);
 16:   };
 17:
 18:   const tabTransition = {
 19:     initial: { opacity: 0, y: 20 },
 20:     animate: { opacity: 1, y: 0 },
 21:     exit: { opacity: 0, y: -20 },
 22:     transition: { duration: 0.3 }
 23:   };
 24:
 25:   return (
 26:     <div className="bg-gray-100 font-sans min-h-screen">
 27:       <TabNavigation activeTab={activeTab} setActiveTab={setActiveTab} />
 28:
 29:       <AnimatePresence mode="wait">
 30:         {activeTab === 'form' && (
 31:           <motion.div key="form" {...tabTransition}>
 32:             <div className="flex flex-col md:flex-row h-screen desktop-layout">
 33:               <InputForm onGenerateReport={handleGenerateReport} reportData={reportData} />
```

```
34:               <ReportOutput reportData={reportData} />
35:             </div>
36:           </motion.div>
37:         )}
38:
39:         {activeTab === 'admin' && (
40:           <motion.div key="admin" {...tabTransition}>
41:             <div className="p-6">
42:               <AdminPanel />
43:             </div>
44:           </motion.div>
45:         )}
46:
47:         {activeTab === 'settings' && (
48:           <motion.div key="settings" {...tabTransition}>
49:             <div className="p-6">
50:               <SettingsPanel />
51:             </div>
52:           </motion.div>
53:         )}
54:       </AnimatePresence>
55:     </div>
56:   );
57: };
58:
59: export default AppContent;
```

---

## ■ File: src\assets\placeholder.svg

==========

---

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

==========

```
  1: import React from "react";
  2: import { useFormConfig } from "../../contexts/FormConfigContext";
  3: import { DragDropContext, Droppable, Draggable } from "@hello-pangea/dnd";
  4: import { toast } from "react-toastify";
  5: import { useState, useEffect } from "react";
  6:
  7: const AdminPanel = () => {
  8:   const { state, dispatch } = useFormConfig();
  9:
 10:   const [localFields, setLocalFields] = useState([]);
 11:
 12:   useEffect(() => {
 13:     // Sync when config loads or changes
 14:     setLocalFields(state.fields);
 15:   }, [state.fields]);
 16:
 17:   const updateLocalField = (index, property, value) => {
 18:     const updated = [...localFields];
 19:     updated[index] = { ...updated[index], [property]: value };
 20:     setLocalFields(updated);
 21:   };
 22:
 23:   const saveField = (index) => {
 24:     const field = localFields[index];
 25:     dispatch({ type: "UPDATE_FIELD", index, property: "full", value: field });
 26:     toast.success(`? Field "${field.label}" saved!`);
 27:   };
 28:
 29:   const categories = state.categories;
 30:
 31:   // ? Add new field
 32:   const addNewField = () => {
 33:     const newField = {
 34:       id: `field_${Date.now()}`,
 35:       label: "New Field",
 36:       category: "",
 37:       high: "",
```

```
      normal: "",
      low: "",
    };
    dispatch({ type: "ADD_FIELD", field: newField });
    toast.success("? Field added!");
  };

  // ?? Delete field
  const deleteField = (index) => {
    if (window.confirm("Delete this field?")) {
      dispatch({ type: "DELETE_FIELD", index });
      toast.info("?? Field deleted");
    }
  };

  // ?? Edit field
  // const updateField = (index, property, value) => {
  //   dispatch({ type: "UPDATE_FIELD", index, property, value });
  // };

  // ? Handle drag-end reorder
  const onDragEnd = (result) => {
    if (!result.destination) return;
    const reordered = Array.from(state.fields);
    const [moved] = reordered.splice(result.source.index, 1);
    reordered.splice(result.destination.index, 0, moved);

    dispatch({
      type: "REORDER_FIELDS",
      fields: reordered,
    });
    toast.success("? Field order updated!");
  };

  return (
    <div className="max-w-6xl mx-auto px-4 py-8 bg-gray-800 dark:bg-gray-900 text-white dark:text-gray-100"
      <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>
        <button
          onClick={() => localFields.forEach((field, i) => saveField(i))}
          className="bg-blue-600 hover:bg-blue-700 text-white px-5 py-2 rounded-lg font-semibold"
        >
          ? Save All
        </button>
      </div>

      {/* Drag Context */}
      <DragDropContext onDragEnd={onDragEnd}>
        <Droppable droppableId="fields">
          {(provided) => (
            <div
              {...provided.droppableProps}
              ref={provided.innerRef}
              className="space-y-6"
            >
              {state.fields.map((field, index) => (
                <Draggable key={field.id} draggableId={field.id} 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>
                        <button
                          onClick={() => deleteField(index)}
                          className="text-red-600 hover:text-red-800 font-medium"
                        >
                          ?? Delete
                        </button>
                        <div className="flex gap-2">
                          <button
                            onClick={() => saveField(index)}
                            className="text-green-600 hover:underline"
                          >
                            ? Save
                          </button>
                        </div>
                      </div>
                      <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                        <div>
                          <label className="text-sm font-medium">
                            Field ID
                          </label>
                          <input
                            type="text"
                            className="w-full border rounded px-3 py-2"
                            value={localFields[index]?.id ?? ""}
                            onChange={(e) =>
                              updateLocalField(index, "id", e.target.value)
                            }
                          />
                        </div>

                        <div>
                          <label className="text-sm font-medium">Label</label>
                          <input
                            type="text"
                            className="w-full border rounded px-3 py-2"
                            value={localFields[index]?.label ?? ""}
                            onChange={(e) =>
                              updateLocalField(index, "label", e.target.value)
                            }
                          />
                        </div>

                        <div>
                          <label className="text-sm font-medium">
                            Category
                          </label>
                          <select
                            className="w-full border rounded px-3 py-2"
                            value={localFields[index]?.category ?? ""}
                            onChange={(e) =>
                              updateLocalField(
                                index,
                                "category",
                                e.target.value
                              )
                            }
                          >
                            <option value="">? None ?</option>
                            {categories.map((cat, i) => (
                              <option key={i} value={cat}>
                                {cat}
                              </option>
                            ))}
                          </select>
                        </div>

                        <div>
                          <label className="text-sm font-medium">
```

```
                        Min Score
                      </label>
                      <input
                        type="number"
                        min="1"
                        max="20"
                        className="w-full border rounded px-3 py-2"
                        value={localFields[index]?.min ?? ""}
                        onChange={(e) =>
                          updateLocalField(
                            index,
                            "min",
                            e.target.value === ""
                              ? undefined
                              : parseInt(e.target.value)
                          )
                        }
                      />
                    </div>

                    <div>
                      <label className="text-sm font-medium">
                        Max Score
                      </label>
                      <input
                        type="number"
                        min="1"
                        max="20"
                        className="w-full border rounded px-3 py-2"
                        value={localFields[index]?.max ?? ""}
                        onChange={(e) =>
                          updateLocalField(
                            index,
                            "max",
                            e.target.value === ""
                              ? undefined
                              : parseInt(e.target.value)
                          )
                        }
                      />
                    </div>

                    <div>
                      <label className="text-sm font-medium">
                        High Recommendation
                      </label>
                      <textarea
                        rows={3}
                        className="w-full border rounded px-3 py-2"
                        value={localFields[index]?.high ?? ""}
                        onChange={(e) =>
                          updateLocalField(index, "high", e.target.value)
                        }
                      />
                    </div>

                    <div>
                      <label className="text-sm font-medium">
                        Normal Recommendation
                      </label>
                      <textarea
                        rows={3}
                        className="w-full border rounded px-3 py-2"
                        value={localFields[index]?.normal ?? ""}
                        onChange={(e) =>
                          updateLocalField(index, "normal", e.target.value)
                        }
                      />
                    </div>

                    <div>
                      <label className="text-sm font-medium">
                        Low Recommendation
```

```
257:                           </label>
258:                           <textarea
259:                             rows={3}
260:                             className="w-full border rounded px-3 py-2"
261:                             value={localFields[index]?.low ?? ""}
262:                             onChange={(e) =>
263:                               updateLocalField(index, "low", e.target.value)
264:                             }
265:                           />
266:                         </div>
267:                       </div>
268:                     </div>
269:                   )}
270:                 </Draggable>
271:               ))}
272:               {provided.placeholder}
273:             </div>
274:           )}
275:         </Droppable>
276:       </DragDropContext>
277:     </div>
278:   );
279: };
280:
281: 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:           <div className="flex gap-2">
 17:             {tabs.map((tab) => (
 18:               <button
 19:                 key={tab.id}
 20:                 className={`px-4 py-2 rounded-lg font-medium transition-all ${
 21:                   activeTab === tab.id
 22:                     ? "bg-green-600 text-white"
 23:                     : "bg-gray-200 text-gray-800 dark:bg-gray-700 dark:text-gray-100"
 24:                 }`}
 25:                 onClick={() => setActiveTab(tab.id)}
 26:               >
 27:                 {tab.label}
 28:               </button>
 29:             ))}
 30:           </div>
 31:
 32:           {/* Theme Toggle */}
 33:           <button
 34:             onClick={toggleTheme}
 35:             className="text-sm px-3 py-2 bg-gray-100 dark:bg-gray-700 dark:text-white rounded"
 36:           >
 37:             {theme === "dark" ? "?? Light" : "? Dark"}
 38:           </button>
 39:         </div>
 40:       </div>
 41:     </div>
 42:   );
 43: };
 44: export default TabNavigation;

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

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

```
================================================================================
  1: import { ToastContainer } from "react-toastify";
  2: import "react-toastify/dist/ReactToastify.css";
  3:
  4: const Toasts = () => {
  5:   return (
  6:     <ToastContainer
  7:       position="top-right"
  8:       autoClose={3000}
  9:       hideProgressBar={false}
 10:       newestOnTop={false}
 11:       closeOnClick
 12:       pauseOnFocusLoss
 13:       draggable
 14:       pauseOnHover
 15:       theme="colored"
 16:     />
 17:   );
 18: };
 19:
 20: 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()) {
 60:       onGenerateReport(formData);
 61:       toast.success("? Report generated and saved!");
 62:     } else {
 63:       toast.error("? Please fix errors before submitting.");
 64:     }
 65:   };
 66:
```

```
  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"
            onClick={handleClearForm}
            className="bg-gray-500 hover:bg-gray-600 text-white px-6 py-3 rounded-lg font-semibold"
          >
            Clear Input
          </button>
          <button
            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}`}
 45:                 className="bg-gray-200 px-4 py-2 font-bold text-sm text-gray-700 border-l-4 border-gray-400
 46:               >
 47:                 {currentCategory}
 48:               </div>
 49:             );
 50:           }
 51:
```

```
          // 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 */}
```

```
125:                 <div className="w-full md:w-1/3 md:pl-2">
126:                   <div
127:                     className={`text-xs font-bold mb-1 ${getTextStyle(
128:                       showLow
129:                     )}`}
130:                   >
131:                     LOW
132:                   </div>
133:                   <div
134:                     className={`text-xs leading-tight ${getTextStyle(showLow)}`}
135:                   >
136:                     {field.low.split("\n").map((line, i, arr) => (
137:                       <React.Fragment key={i}>
138:                         {line}
139:                         {i < arr.length - 1 && <br />}
140:                       </React.Fragment>
141:                     ))}
142:                   </div>
143:                 </div>
144:               </div>
145:             </div>
146:           );
147:
148:           return elements;
149:         })}
150:       </div>
151:     </div>
152:   );
153: };
154:
155: export default ReportOutput;
```

---

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

```
  1: import { useFormConfig } from '../../contexts/FormConfigContext';
  2: import { useState } from 'react';
  3:
  4: const CategoryManager = () => {
  5:   const { state, dispatch } = useFormConfig();
  6:   const [newCategory, setNewCategory] = useState('');
  7:   const [editIndex, setEditIndex] = useState(null);
  8:   const [editedName, setEditedName] = useState('');
  9:
 10:   const addCategory = () => {
 11:     if (!newCategory.trim()) return;
 12:     if (state.categories.includes(newCategory)) return;
 13:     dispatch({ type: 'ADD_CATEGORY', name: newCategory.trim() });
 14:     setNewCategory('');
 15:   };
 16:
 17:   const startEdit = (i, name) => {
 18:     setEditIndex(i);
 19:     setEditedName(name);
 20:   };
 21:
 22:   const confirmEdit = () => {
 23:     if (!editedName.trim()) return;
 24:     dispatch({ type: 'UPDATE_CATEGORY', index: editIndex, newName: editedName.trim() });
 25:     setEditIndex(null);
 26:     setEditedName('');
 27:   };
 28:
 29:   const deleteCategory = (i) => {
 30:     if (window.confirm('Remove this category?')) {
 31:       dispatch({ type: 'DELETE_CATEGORY', index: i });
 32:     }
 33:   };
 34:
 35:   return (
 36:     <div className="mt-10 border-t pt-6">
 37:       <h3 className="text-xl font-semibold mb-4">?? Category Manager</h3>
```

```
 38:
 39:       <div className="flex gap-2 mb-4">
 40:         <input
 41:           type="text"
 42:           className="border px-3 py-2 rounded w-full"
 43:           placeholder="New category name"
 44:           value={newCategory}
 45:           onChange={(e) => setNewCategory(e.target.value)}
 46:         />
 47:         <button
 48:           onClick={addCategory}
 49:           className="bg-green-600 hover:bg-green-700 text-white px-4 py-2 rounded"
 50:         >
 51:           ? Add
 52:         </button>
 53:       </div>
 54:
 55:       <div className="space-y-3">
 56:         {state.categories.map((cat, i) => (
 57:           <div
 58:             key={i}
 59:             className="flex items-center justify-between bg-white border p-3 rounded"
 60:           >
 61:             {editIndex === i ? (
 62:               <>
 63:                 <input
 64:                   type="text"
 65:                   className="border px-2 py-1 rounded w-full mr-2"
 66:                   value={editedName}
 67:                   onChange={(e) => setEditedName(e.target.value)}
 68:                 />
 69:                 <button
 70:                   onClick={confirmEdit}
 71:                   className="bg-blue-600 text-white px-3 py-1 rounded"
 72:                 >
 73:                   ?
 74:                 </button>
 75:               </>
 76:             ) : (
 77:               <>
 78:                 <span>{cat}</span>
 79:                 <div className="flex gap-2">
 80:                   <button
 81:                     onClick={() => startEdit(i, cat)}
 82:                     className="text-blue-600 hover:underline"
 83:                   >
 84:                     ?? Edit
 85:                   </button>
 86:                   <button
 87:                     onClick={() => deleteCategory(i)}
 88:                     className="text-red-600 hover:underline"
 89:                   >
 90:                     ?? Delete
 91:                   </button>
 92:                 </div>
 93:               </>
 94:             )}
 95:           </div>
 96:         ))}
 97:       </div>
 98:     </div>
 99:   );
100: };
101:
102: 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";
```

```
import { toast } from "react-toastify";
import ResetAll from "../common/ResetAll";
import CategoryManager from "./CategoryManager";

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

  // Internal editable state
  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
  };

  // Change handlers
  const handleChange = (key, value) => {
    setSettings((prev) => ({
      ...prev,
      [key]: value,
    }));
  };

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

  // Apply changes to context
  const applySettings = () => {
    dispatch({ type: "UPDATE_SETTINGS", settings });
    alert("? Settings applied successfully!");
  };

  // Reset to initial/default config
  const resetSettings = () => {
    if (
      window.confirm("Are you sure you want to reset all settings to default?")
    ) {
      window.location.reload(); // simplest way to reload JSON state
    }
  };

  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"
              min="1"
              max="10"
              value={settings.highThreshold}
              onChange={(e) =>
                handleChange("highThreshold", parseInt(e.target.value))
              }
              className="w-full border rounded px-3 py-2"
            />
          </div>

          <h3 className="text-lg font-semibold mt-6">? Score Colors</h3>
```

```
150:
151:           <div className="space-y-2">
152:             <div>
153:               <label className="block text-sm font-medium mb-1">High</label>
154:               <input
155:                 type="color"
156:                 value={settings.colors.high}
157:                 onChange={(e) => handleColorChange("high", e.target.value)}
158:                 className="h-10 w-full border rounded"
159:               />
160:             </div>
161:
162:             <div>
163:               <label className="block text-sm font-medium mb-1">Medium</label>
164:               <input
165:                 type="color"
166:                 value={settings.colors.medium}
167:                 onChange={(e) => handleColorChange("medium", e.target.value)}
168:                 className="h-10 w-full border rounded"
169:               />
170:             </div>
171:
172:             <div>
173:               <label className="block text-sm font-medium mb-1">Low</label>
174:               <input
175:                 type="color"
176:                 value={settings.colors.low}
177:                 onChange={(e) => handleColorChange("low", e.target.value)}
178:                 className="h-10 w-full border rounded"
179:               />
180:             </div>
181:           </div>
182:         </div>
183:       </div>
184:
185:       {/* Buttons */}
186:       <div className="mt-6 flex gap-4">
187:         <button
188:           onClick={applySettings}
189:           className="bg-green-600 hover:bg-green-700 text-white px-6 py-3 rounded-lg font-semibold"
190:         >
191:           ? Apply Settings
192:         </button>
193:         <button
194:           onClick={resetSettings}
195:           className="bg-gray-600 hover:bg-gray-700 text-white px-6 py-3 rounded-lg font-semibold"
196:         >
197:           ?? Reset to Default
198:         </button>
199:         <button
200:           onClick={exportConfig}
201:           className="bg-blue-600 hover:bg-blue-700 text-white px-6 py-3 rounded-lg font-semibold"
202:         >
203:           ? Export Config (.json)
204:         </button>
205:
206:         <button
207:           onClick={handleImportClick}
208:           className="bg-purple-600 hover:bg-purple-700 text-white px-6 py-3 rounded-lg font-semibold"
209:         >
210:           ? Import Config (.json)
211:         </button>
212:         <div className="mt-6">
213:           <ResetAll />
214:         </div>
215:
216:         <input
217:           type="file"
218:           accept=".json"
219:           ref={fileInputRef}
220:           className="hidden"
221:           onChange={handleFileSelected}
222:         />
```

```
      </div>
      <div className="mt-6">
        <CategoryManager />
      </div>
    </div>
  );
};

export default SettingsPanel;
```

---

## ■ File: src\contexts\FormConfigContext.js

```
import React, { createContext, useContext, useReducer, useEffect } from "react";
import initialConfig from "../data/formConfig.json";

const LOCAL_STORAGE_KEY = "genomics_config";

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

    case "ADD_CATEGORY":
      return { ...state, categories: [...state.categories, action.name] };

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

    case "DELETE_CATEGORY":
      return {
        ...state,
        categories: state.categories.filter((_, i) => i !== action.index),
```

```
 60:         fields: state.fields.map((field) =>
 61:           field.category === state.categories[action.index]
 62:             ? { ...field, category: "" }
 63:             : field
 64:         ),
 65:       };
 66:
 67:     default:
 68:       return state;
 69:   }
 70: };
 71:
 72: const FormConfigContext = createContext(null);
 73:
 74: export const FormConfigProvider = ({ children }) => {
 75:   const stored = localStorage.getItem(LOCAL_STORAGE_KEY);
 76:   const parsed = stored ? JSON.parse(stored) : initialConfig;
 77:
 78:   const [state, dispatch] = useReducer(formConfigReducer, parsed);
 79:
 80:   useEffect(() => {
 81:     localStorage.setItem(LOCAL_STORAGE_KEY, JSON.stringify(state));
 82:   }, [state]);
 83:
 84:   return (
 85:     <FormConfigContext.Provider value={{ state, dispatch }}>
 86:       {children}
 87:     </FormConfigContext.Provider>
 88:   );
 89: };
 90:
 91: export const useFormConfig = () => {
 92:   const context = useContext(FormConfigContext);
 93:   if (!context) throw new Error("useFormConfig must be used within a FormConfigProvider");
 94:   return context;
 95: };
```

---

## ■ 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\data\formConfig.json

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

```
  1: {
  2:   "title": "GENOMICS & DIET",
  3:   "quote": "\"YOU ARE WHAT YOU EAT\" - Victor Lindlahr",
  4:   "description": "A right diet is the one that makes you feel happy, keeps you healthy, does not make you f
  5:   "headerColor": "#16a34a",
  6:   "colors": {
  7:     "low": "#16a34a",
  8:     "medium": "#f59e0b",
  9:     "high": "#dc2626"
 10:   },
 11:   "highThreshold": 6,
 12:   "categories": ["MACRONUTRIENTS", "MEAL PATTERN", "FOOD SENSITIVITIES"],
 13:   "fields": [
 14:     {
 15:       "id": "carb",
 16:       "label": "Carbohydrate Sensitivity",
 17:       "category": "MACRONUTRIENTS",
 18:       "min": 1,
 19:       "max": 10,
 20:       "high": "Maintain carb intake <45%\nFor obesity & IR control",
 21:       "normal": "Maintain carb intake <50%\nBalanced recommendation",
 22:       "low": "Maintain carb intake <60%\nFor obesity & IR control"
 23:     },
 24:     {
 25:       "id": "fat",
 26:       "label": "Fat Sensitivity",
 27:       "category": "MACRONUTRIENTS",
 28:       "min": 1,
 29:       "max": 10,
 30:       "high": "Fat intake not to exceed\n15% of total calories",
 31:       "normal": "Fat intake should be\n20% of total calories",
 32:       "low": "Fat intake advised up to\n25% of total calories"
 33:     },
 34:     {
 35:       "id": "protein",
 36:       "label": "Protein Requirement",
 37:       "category": "MACRONUTRIENTS",
 38:       "min": 1,
 39:       "max": 15,
 40:       "high": "Protein supplements\nneeded along with dietary\nsource",
 41:       "normal": "Maintain adequate protein\nthrough balanced diet and\noccasional supplements",
 42:       "low": "Protein supplements not\nneeded, intake through\ndiet is enough"
 43:     },
 44:     {
 45:       "id": "meal",
 46:       "label": "Meal Frequency",
 47:       "category": "MEAL PATTERN",
 48:       "min": 1,
 49:       "max": 10,
 50:       "high": "4-5 small meals suggested\nin a day",
 51:       "normal": "3-4 balanced meals\nrecommended per day",
 52:       "low": "Less frequent meals, 2-3\nmeals are enough in a day"
 53:     },
 54:     {
 55:       "id": "alcohol",
 56:       "label": "Alcohol Sensitivity",
 57:       "category": "",
 58:       "min": 1,
 59:       "max": 6,
 60:       "high": "High sensitivity, avoid\nalcohol if possible,\nespecially the types of\nbeverages that trigg
 61:       "normal": "Moderate sensitivity,\nlimit alcohol consumption\nto 1-2 drinks per day,\nmonitor for any 
 62:       "low": "Low sensitivity, know\nyour alcohol intake limits,\nconsult doctor for\nknowing your upper li
 63:     },
 64:     {
 65:       "id": "caffeine",
 66:       "label": "Caffeine Sensitivity",
 67:       "category": "",
 68:       "min": 1,
 69:       "max": 5,
 70:       "high": "High sensitivity, do not\nconsume >4 cups/day",
 71:       "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"
    }
  ]
}
```

---

## ■ File: src\hooks\useConfigImportExport.js

```
import { useCallback } from "react";
import { useFormConfig } from "../contexts/FormConfigContext";

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

  const exportConfig = useCallback(() => {
    const dataStr = JSON.stringify(state, null, 2);
    const blob = new Blob([dataStr], { type: "application/json" });
    const url = URL.createObjectURL(blob);

    const link = document.createElement("a");
    link.href = url;
    link.download = "genomics-form-config.json";
    link.click();
    URL.revokeObjectURL(url);
  }, [state]);

  const importConfig = useCallback(
    (file) => {
      const reader = new FileReader();
      reader.onload = (e) => {
        try {
          const parsed = JSON.parse(e.target.result);
          dispatch({ type: "IMPORT_CONFIG", config: parsed });
          alert("Configuration imported successfully.");
        } catch (err) {
          alert("Error importing config: " + err.message);
        }
      };
      reader.readAsText(file);
    },
    [dispatch]
  );
```

```
 35:
 36:   return { exportConfig, importConfig };
 37: };
```

---

## ■ 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";
  3: import { useFormConfig } from "../contexts/FormConfigContext";
  4: import { hexToRgb } from "../utils/helpers";
  5:
  6: export const usePDFGeneration = () => {
  7:   const { state } = useFormConfig();
  8:
  9:   // Base64 logo (optional)
 10:   const leftLogoUrl = "/left.png";
 11:   const rightLogoUrl = "/right.png";
 12:
 13:   const generatePDF = useCallback(
 14:     (reportData) => {
 15:       if (!reportData || reportData.length === 0) {
 16:         alert("No report data found.");
 17:         return;
 18:       }
 19:
 20:       const doc = new jsPDF();
 21:       const pageHeight = doc.internal.pageSize.height;
 22:       const pageWidth = doc.internal.pageSize.width;
 23:       const margin = 10;
 24:       let y = 20;
 25:
 26:       // ----------------------------
```

```
      // 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;
```

```
100:
101:         // Render matching text
102:         const renderTextBlock = (label, text, active) => {
103:           if (!text) return;
104:           doc.setFontSize(9);
105:           doc.setFont(undefined, "bold");
106:           doc.setTextColor(
107:             active ? 0 : 180,
108:             active ? 0 : 180,
109:             active ? 0 : 180
110:           );
111:           doc.text(`${label}:`, margin, y);
112:           y += 5;
113:
114:           doc.setFont(undefined, "normal");
115:           const lines = doc.splitTextToSize(text, pageWidth - 2 * margin);
116:           lines.forEach((line) => {
117:             if (y + 6 > pageHeight - 15) {
118:               doc.addPage();
119:               y = 20;
120:             }
121:             doc.text(line, margin, y);
122:             y += 5;
123:           });
124:           y += 2;
125:         };
126:
127:         renderTextBlock("HIGH", field.high, showHigh);
128:         renderTextBlock("NORMAL", field.normal, showNormal);
129:         renderTextBlock("LOW", field.low, showLow);
130:
131:         y += 4;
132:       });
133:
134:       // Footer (optional logos)
135:       const addLogos = () => {
136:         try {
137:           doc.addImage(leftLogoUrl, "PNG", 10, pageHeight - 20, 30, 10);
138:           doc.addImage(
139:             rightLogoUrl,
140:             "PNG",
141:             pageWidth - 40,
142:             pageHeight - 20,
143:             30,
144:             10
145:           );
146:         } catch (e) {
147:           console.warn("Logo failed to load. Skipping...");
148:         }
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 */
```

```
}

/* Background color for any white-background elements */
.bg-white {
  @apply dark:bg-gray-900; /* Switches background color to dark mode */
}

/* Text color for general text */
.text-gray-900 {
  @apply dark:text-white; /* Changes text color to white in dark mode */
}

/* Form elements: inputs, textareas, and selects */
input,
textarea,
select {
  @apply bg-white dark:bg-gray-800 text-black dark:text-white border dark:border-gray-700;
}

/* Focused state of inputs, textareas, and selects */
input:focus,
textarea:focus,
select:focus {
  @apply ring-2 ring-blue-500 dark:ring-blue-300;
}

/* Global font and smoothing */
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto",
    "Oxygen", "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans",
    "Helvetica Neue", sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

/* Code styles for code blocks */
code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, "Courier New",
    monospace;
}

/* Global styling for text input elements */
input[type="text"],
input[type="number"],
input[type="email"],
textarea,
select {
  @apply border-2 rounded-lg p-2 dark:border-gray-600;
}

/* Button hover and focus states */
button:hover {
  @apply bg-gray-200 dark:bg-gray-700; /* Darker button on hover */
}

/* Ensuring links are visible in dark mode */
.text-blue-600 {
  @apply dark:text-blue-400; /* Light blue in normal mode, changes to darker in dark mode */
}

/* Customize card elements for dark mode */
.card {
  @apply bg-white dark:bg-gray-800 text-black dark:text-white border dark:border-gray-700;
}

/* Customize borders */
.border-gray-300 {
  @apply dark:border-gray-600; /* Change border color in dark mode */
}

/* Ensure smooth transitions when toggling dark/light mode */
.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\styles\index.css

---

## ■ 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: };
```

---