

■ 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:

```

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:
16:     for root, dirs, files in os.walk(directory):
17:         # Skip excluded directories
18:         dirs[:] = [d for d in dirs if d not in excluded_dirs]
19:
20:         # Skip if current directory is an excluded directory
21:         if any(excluded_dir in root.split(os.sep) for excluded_dir in excluded_dirs):
22:             continue
23:
24:         for file in files:
25:             # Skip excluded files
26:             if file in excluded_files:
27:                 continue
28:
29:             file_path = os.path.join(root, file)
30:
31:             # Get file extension
32:             _, ext = os.path.splitext(file)
33:
34:             try:
35:                 # Try to read as text file first
36:                 if ext.lower() in {'.js', '.jsx', '.ts', '.tsx', '.css', '.scss', '.sass', '.less',
37:                                     '.html', '.htm', '.json', '.md', '.txt', '.xml', '.yaml', '.yml',
38:                                     '.config', '.gitignore', '.env', '.py', '.sh', '.bat', '.cmd',
39:                                     '.svg', '.dockerfile', '.editorconfig', '.eslintrc', '.prettierrc'}:
40:                     with open(file_path, "r", encoding="utf-8", errors="ignore") as f:
41:                         code_files[file_path] = f.readlines()
42:             except:
43:                 # For binary files, just note them as binary
44:                 code_files[file_path] = [f"Binary file - {ext} format"]
45:
46:             except Exception as e:
47:                 print(f"? Error reading {file_path}: {e}")
48:                 code_files[file_path] = [f"Error reading file: {str(e)}"]
49:
50:     return code_files
51:
52:
53: def create_pdf(code_data, output_pdf="Frontend_Code_Export.pdf"):
54:     c = canvas.Canvas(output_pdf, pagesize=A4)
55:     width, height = A4
56:     margin = 20 * mm
57:     line_height = 10
58:     y = height - margin
59:
60:     # Title
61:     c.setFont("Helvetica-Bold", 16)
62:     c.drawString(margin, y, "? Project Code Export")
63:     y -= 2 * line_height
64:     c.setFont("Helvetica-Bold", 12)
65:     c.drawString(margin, y, "? Frontend File List:")
66:     y -= 2 * line_height
67:
68:     file_paths = sorted(list(code_data.keys()))
69:
70:     # 1. File list (original simple format)
71:     c.setFont("Courier", 8)
72:     for path in file_paths:
73:         if y < margin:
74:             c.showPage()
75:             c.setFont("Courier", 8)
76:             y = height - margin
77:
78:         display_path = os.path.relpath(path)

```

```

79:         c.drawString(margin, y, f"- {display_path}")
80:         y -= line_height
81:
82:     # Add page break before code content
83:     c.showPage()
84:     y = height - margin
85:
86:     # 2. File contents
87:     for file_path in file_paths:
88:         lines = code_data[file_path]
89:         print(f"? Adding: {file_path}")
90:
91:         if y < margin + 3 * line_height:
92:             c.showPage()
93:             y = height - margin
94:
95:             # File header
96:             rel_path = os.path.relpath(file_path)
97:             c.setFont("Helvetica-Bold", 12)
98:             c.drawString(margin, y, f"? File: {rel_path}")
99:             y -= line_height
100:
101:             # Add separator line
102:             c.setFont("Courier", 8)
103:             c.drawString(margin, y, "=" * 80)
104:             y -= line_height
105:
106:             # File content
107:             for line_num, line in enumerate(lines, 1):
108:                 if y < margin:
109:                     c.showPage()
110:                     c.setFont("Courier", 8)
111:                     y = height - margin
112:
113:                 # Clean and truncate line
114:                 line = line.strip("\n").encode("latin-1", "replace").decode("latin-1")
115:
116:                 # Add line numbers for code files
117:                 if rel_path.endswith((''.js', '.jsx', '.ts', '.tsx', '.css', '.py', '.html', '.json')):
118:                     display_line = f"{line_num:3d}: {line[:280]}"
119:                 else:
120:                     display_line = line[:300]
121:
122:                 c.drawString(margin, y, display_line)
123:                 y -= line_height
124:
125:             # Add spacing between files
126:             y -= line_height
127:             if y > margin:
128:                 c.setFont("Courier", 8)
129:                 c.drawString(margin, y, "-" * 80)
130:                 y -= 2 * line_height
131:
132:     c.save()
133:     print(f"? PDF successfully created: {output_pdf}")
134:     print(f"? Total files processed: {len(code_data)}")
135:
136:
137: def main():
138:     root_dir = os.path.dirname(os.path.abspath(__file__))
139:
140:     # Files to exclude (including package.json as requested)
141:     excluded_files = {
142:         'package.json',
143:         'package-lock.json',
144:         'yarn.lock',
145:         'README.md',
146:         '.DS_Store',
147:         'Thumbs.db',
148:         'Desktop.ini'
149:     }
150:
151:     # 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:
7: function App() {
8:   return (
9:     <ThemeProvider>
10:       <FormConfigProvider>
11:         <AppContent />
12:         <Toasts />
13:       </FormConfigProvider>
14:     </ThemeProvider>
15:   );
16: }
17:
18: 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:
6: const AdminPanel = () => {
7:     const { state, dispatch } = useFormConfig();
8:     const categories = state.categories;
9:     // ? Add new field
10:    const addNewField = () => {
11:        const newField = {
12:            id: `field_${Date.now()}`,
13:            label: "New Field",
14:            category: "",
15:            high: "",
16:            normal: "",
17:            low: "",
18:        };
19:        dispatch({ type: "ADD_FIELD", field: newField });
20:        toast.success("? Field added!");
21:    };
22:
23:    // ?? Delete field
24:    const deleteField = (index) => {
25:        if (window.confirm("Delete this field?")) {
26:            dispatch({ type: "DELETE_FIELD", index });
27:            toast.info("?? Field deleted");
28:        }
29:    };
30:
31:    // ?? Edit field
32:    const updateField = (index, property, value) => {
33:        dispatch({ type: "UPDATE_FIELD", index, property, value });
34:    };
35:
36:    // ? Handle drag-end reorder
37:    const onDragEnd = (result) => {
38:        if (!result.destination) return;
39:        const reordered = Array.from(state.fields);

```



```

40:     const [moved] = reordered.splice(result.source.index, 1);
41:     reordered.splice(result.destination.index, 0, moved);
42:
43:     dispatch({
44:       type: "REORDER_FIELDS",
45:       fields: reordered,
46:     });
47:     toast.success("Field order updated!");
48:   };
49:
50:   return (
51:     <div className="max-w-6xl mx-auto px-4 py-8">
52:       <div className="flex justify-between items-center mb-6">
53:
54:         <h2 className="text-2xl font-bold">Field Management</h2>
55:         <button
56:           onClick={addNewField}
57:           className="bg-green-600 hover:bg-green-700 text-white px-5 py-2 rounded-lg font-semibold"
58:         >
59:           ? Add New Field
60:         </button>
61:       </div>
62:
63:       { /* Drag Context */ }
64:       <DragDropContext onDragEnd={onDragEnd}>
65:         <Draggable draggableId="fields">
66:           {(provided) => (
67:             <div
68:               {...provided.draggableProps}
69:               ref={provided.innerRef}
70:               className="space-y-6"
71:             >
72:               {state.fields.map((field, index) => (
73:                 <Draggable key={field.id} draggableId={field.id} index={index}>
74:                   {(provided, snapshot) => (
75:                     <div
76:                       ref={provided.innerRef}
77:                       {...provided.draggableProps}
78:                       {...provided.dragHandleProps}
79:                       className={`border rounded-lg p-6 bg-white shadow-sm space-y-4 ${
80:                         snapshot.isDragging
81:                           ? "bg-gray-100 ring-2 ring-green-500"
82:                           : ""
83:                       }}
84:                     >
85:                       <div className="flex justify-between items-center">
86:                         <h3 className="text-lg font-semibold">
87:                           {index + 1}. {field.label}
88:                         </h3>
89:                         <button
90:                           onClick={() => deleteField(index)}
91:                           className="text-red-600 hover:text-red-800 font-medium"
92:                         >
93:                           ?? Delete
94:                         </button>
95:                       </div>
96:
97:                       <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
98:                         { /* All input fields: ID, Label, Category, High/Normal/Low */ }
99:                         { /* Reuse existing inputs from your current AdminPanel */ }
100:                        <div>
101:                          <label className="text-sm font-medium">
102:                            Field ID
103:                          </label>
104:                          <input
105:                            type="text"
106:                            className="w-full border rounded px-3 py-2"
107:                            value={field.id}
108:                            onChange={(e) =>
109:                              updateField(index, "id", e.target.value)
110:                            }
111:                          />
112:                        </div>

```

```

113:
114:     <div>
115:         <label className="text-sm font-medium">Label</label>
116:         <input
117:             type="text"
118:             className="w-full border rounded px-3 py-2"
119:             value={field.label}
120:             onChange={(e) =>
121:                 updateField(index, "label", e.target.value)
122:             }
123:         />
124:     </div>
125:
126:     <div>
127:         <label className="text-sm font-medium">
128:             Category
129:         </label>
130:         <select
131:             className="w-full border rounded px-3 py-2"
132:             value={field.category} // the current selected category (string)
133:             onChange={(e) =>
134:                 updateField(index, "category", e.target.value)
135:             }
136:         >
137:             <option value="">? None ?</option>
138:             {categories.map((cat, i) => (
139:                 <option key={i} value={cat}>
140:                     {cat}
141:                 </option>
142:             ))}
143:         </select>
144:     </div>
145:
146:     <div>
147:         <label className="text-sm font-medium">
148:             Min Score
149:         </label>
150:         <input
151:             type="number"
152:             min="1"
153:             max="20"
154:             className="w-full border rounded px-3 py-2"
155:             value={field.min ?? ""}
156:             onChange={(e) =>
157:                 updateField(
158:                     index,
159:                     "min",
160:                     e.target.value === ""
161:                         ? undefined
162:                         : parseInt(e.target.value)
163:                 )
164:             }
165:         />
166:     </div>
167:
168:     <div>
169:         <label className="text-sm font-medium">
170:             Max Score
171:         </label>
172:         <input
173:             type="number"
174:             min="1"
175:             max="20"
176:             className="w-full border rounded px-3 py-2"
177:             value={field.max ?? ""}
178:             onChange={(e) =>
179:                 updateField(
180:                     index,
181:                     "max",
182:                     e.target.value === ""
183:                         ? undefined
184:                         : parseInt(e.target.value)
185:                 )

```

```

186:         }
187:     />
188: </div>
189:
190: <div>
191:     <label className="text-sm font-medium">
192:         High Recommendation
193:     </label>
194:     <textarea
195:         rows={3}
196:         className="w-full border rounded px-3 py-2"
197:         value={field.high}
198:         onChange={(e) =>
199:             updateField(index, "high", e.target.value)
200:         }
201:     />
202: </div>
203:
204: <div>
205:     <label className="text-sm font-medium">
206:         Normal Recommendation
207:     </label>
208:     <textarea
209:         rows={3}
210:         className="w-full border rounded px-3 py-2"
211:         value={field.normal}
212:         onChange={(e) =>
213:             updateField(index, "normal", e.target.value)
214:         }
215:     />
216: </div>
217:
218: <div>
219:     <label className="text-sm font-medium">
220:         Low Recommendation
221:     </label>
222:     <textarea
223:         rows={3}
224:         className="w-full border rounded px-3 py-2"
225:         value={field.low}
226:         onChange={(e) =>
227:             updateField(index, "low", e.target.value)
228:         }
229:     />
230: </div>
231: </div>
232: </div>
233:     )}
234: </Draggable>
235:   )}
236:   {provided.placeholder}
237: </div>
238:   )}
239: </Droppable>
240: </DragDropContext>
241: </div>
242:   );
243: };
244:
245: 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:
67: const handleDownloadPDF = () => {
68:   if (!reportData || reportData.length === 0) {
69:     toast.warning("Please generate a report first.");
70:     return;
71:   }
72:   generatePDF(reportData);
73: };
74:
75: const handleClearForm = () => {
76:   if (window.confirm("Clear all form scores and reset saved state?")) {
77:     localStorage.removeItem(LOCAL_STORAGE_KEY);
78:     setFormData({});
79:     toast.info("Cleared saved input.");
80:   }
81: };
82:
83: return (
84:   <div className="w-full md:w-1/2 bg-white p-4 md:p-8 overflow-y-auto mobile-section">
85:     { /* ... header & description remain the same */ }
86:
87:     <form onSubmit={handleSubmit} className="space-y-3 md:space-y-4">
88:       <div className="grid grid-cols-1 gap-4">
89:         {state.fields.map((field, index) => (
90:           <div
91:             key={field.id}
92:             className="flex flex-col md:flex-row md:items-center"
93:           >
94:             <label className="w-full md:w-48 text-sm font-semibold mb-1 md:mb-0">
95:               {field.label}
96:             </label>
97:             <input
98:               type="number"
99:               min="1"
100:               max="10"
101:               value={formData[field.id] || ""}
102:               onChange={(e) => handleInputChange(field.id, e.target.value)}
103:               className={`border p-2 w-full md:w-20 text-center rounded
104:                 ${errors[field.id] ? "border-red-500" : "border-gray-300"}`}

```

```

105:         />
106:         {errors[field.id] && (
107:             <p className="text-red-600 text-xs mt-1 md:ml-4">
108:                 {errors[field.id]}
109:             </p>
110:         )}
111:     </div>
112:   )})
113: </div>
114:
115:   <div className="flex flex-col md:flex-row gap-2 mt-6">
116:     <button
117:       type="submit"
118:       className="bg-green-600 hover:bg-green-700 text-white px-6 py-3 rounded-lg font-semibold"
119:     >
120:       Generate Report
121:     </button>
122:
123:     <button
124:       type="button"
125:       onClick={handleDownloadPDF}
126:       className="bg-blue-600 hover:bg-blue-700 text-white px-6 py-3 rounded-lg font-semibold"
127:     >
128:       Download PDF
129:     </button>
130:
131:     <button
132:       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}`}
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:
52:           // Add field row
53:           elements.push(
54:             <div
55:               key={field.id}
56:               className="flex flex-col md:flex-row items-stretch border-b border-gray-200"
57:             >
58:               {/* Field Label */}
59:               <div className="w-full md:w-48 px-3 py-3 text-center md:text-right bg-gray-100 md:bg-white">
60:                 <div className="text-xs font-bold text-gray-700 uppercase leading-tight">
61:                   {field.label}
62:                 </div>
63:               </div>
64:
65:               {/* Score */}
66:               <div className="w-full md:w-16 flex justify-center items-center py-3">
67:                 <div
68:                   className={`w-10 h-10 ${getScoreColor(
69:                     score
70:                   )} text-white font-bold text-lg flex items-center justify-center rounded-full`}
71:                 >
72:                   {score}
73:                 </div>
74:               </div>
75:
76:               {/* Recommendations */}
77:               <div className="flex-1 px-3 py-3 flex flex-col md:flex-row">
78:                 {/* High */}
79:                 <div className="w-full md:w-1/3 md:pr-2 mb-3 md:mb-0">
80:                   <div
81:                     className={`text-xs font-bold mb-1 ${getTextStyle(
82:                       showHigh
83:                     )}`}
84:                   >
85:                     HIGH
86:                   </div>
87:                   <div
88:                     className={`text-xs leading-tight ${getTextStyle(
89:                       showHigh

```



```

90:         )}`}`
91:     >
92:     {field.high.split("\n").map((line, i, arr) => (
93:         <React.Fragment key={i}>
94:             {line}
95:             {i < arr.length - 1 && <br />}
96:         </React.Fragment>
97:     )})
98: </div>
99: </div>
100:
101: {/* Normal */}
102: <div className="w-full md:w-1/3 md:px-2 mb-3 md:mb-0">
103:     <div
104:         className={`text-xs font-bold mb-1 ${getTextStyle(
105:             showNormal
106:         )}`}
107:     >
108:         NORMAL
109:     </div>
110:     <div
111:         className={`text-xs leading-tight ${getTextStyle(
112:             showNormal
113:         )}`}
114:     >
115:         {field.normal?.split("\n").map((line, i, arr) => (
116:             <React.Fragment key={i}>
117:                 {line}
118:                 {i < arr.length - 1 && <br />}
119:             </React.Fragment>
120:         )})
121:     </div>
122: </div>
123:
124: {/* 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";
4: import { toast } from "react-toastify";
5: import ResetAll from "../common/ResetAll";
6: import CategoryManager from "../CategoryManager";
7:
8: const SettingsPanel = () => {
9:     const { state, dispatch } = useFormConfig();
10:
11:     // Internal editable state
12:     const [settings, setSettings] = useState({
13:         title: state.title,
14:         quote: state.quote,
15:         description: state.description,
16:         headerColor: state.headerColor,
17:         highThreshold: state.highThreshold,
18:         colors: {
19:             low: state.colors.low,
20:             medium: state.colors.medium,
21:             high: state.colors.high,
22:         },
23:     });
24:
25:     const fileInputRef = useRef(null);
26:     const { exportConfig, importConfig } = useConfigImportExport();
27:
28:     const handleImportClick = () => {
29:         fileInputRef.current?.click();
30:     };
31:
32:     const handleFileSelected = (e) => {
33:         const file = e.target.files?.[0];
34:         if (!file || !file.name.endsWith(".json")) {
35:             toast.error("Please upload a valid JSON file.");
36:             return;
37:         }
38:         importConfig(file);
39:         e.target.value = ""; // reset input
40:     };
41:

```

```

42: // Change handlers
43: const handleChange = (key, value) => {
44:   setSettings((prev) => ({
45:     ...prev,
46:     [key]: value,
47:   }));
48: };
49:
50: const handleColorChange = (level, value) => {
51:   setSettings((prev) => ({
52:     ...prev,
53:     colors: {
54:       ...prev.colors,
55:       [level]: value,
56:     },
57:   }));
58: };
59:
60: // Apply changes to context
61: const applySettings = () => {
62:   dispatch({ type: "UPDATE_SETTINGS", settings });
63:   alert("? Settings applied successfully!");
64: };
65:
66: // Reset to initial/default config
67: const resetSettings = () => {
68:   if (
69:     window.confirm("Are you sure you want to reset all settings to default?")
70:   ) {
71:     window.location.reload(); // simplest way to reload JSON state
72:   }
73: };
74:
75: return (
76:   <div className="max-w-4xl mx-auto px-4 py-8">
77:     <h2 className="text-2xl font-bold mb-6">?? Form Customization</h2>
78:
79:     <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
80:       /* Left Section */
81:       <div className="bg-white rounded-lg shadow-sm p-6 space-y-4">
82:         <h3 className="text-lg font-semibold">? Header Info</h3>
83:
84:         <div>
85:           <label className="block text-sm font-medium mb-1">Main Title</label>
86:           <input
87:             type="text"
88:             value={settings.title}
89:             onChange={(e) => handleChange("title", e.target.value)}
90:             className="w-full border rounded px-3 py-2"
91:           />
92:         </div>
93:
94:         <div>
95:           <label className="block text-sm font-medium mb-1">Quote</label>
96:           <textarea
97:             rows={2}
98:             value={settings.quote}
99:             onChange={(e) => handleChange("quote", e.target.value)}
100:             className="w-full border rounded px-3 py-2"
101:           />
102:         </div>
103:
104:         <div>
105:           <label className="block text-sm font-medium mb-1">
106:             Description
107:           </label>
108:           <textarea
109:             rows={4}
110:             value={settings.description}
111:             onChange={(e) => handleChange("description", e.target.value)}
112:             className="w-full border rounded px-3 py-2"
113:           />
114:         </div>

```

```

115:
116:     <div>
117:         <label className="block text-sm font-medium mb-1">
118:             Header Background Color
119:         </label>
120:         <input
121:             type="color"
122:             value={settings.headerColor}
123:             onChange={(e) => handleChange("headerColor", e.target.value)}
124:             className="h-10 w-full border rounded"
125:         />
126:     </div>
127: </div>
128:
129: { /* Right Section */ }
130: <div className="bg-white rounded-lg shadow-sm p-6 space-y-4">
131:     <h3 className="text-lg font-semibold">? Score Logic</h3>
132:
133:     <div>
134:         <label className="block text-sm font-medium mb-1">
135:             High Score Threshold (?)
136:         </label>
137:         <input
138:             type="number"
139:             min="1"
140:             max="10"
141:             value={settings.highThreshold}
142:             onChange={(e) =>
143:                 handleChange("highThreshold", parseInt(e.target.value))
144:             }
145:             className="w-full border rounded px-3 py-2"
146:         />
147:     </div>
148:
149:     <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:     />
223: </div>
224: <div className="mt-6">
225:     <CategoryManager />
226: </div>
227: </div>
228: );
229: };
230:
231: export default SettingsPanel;

```

File: src\contexts\FormConfigContext.js

```

1: import React, { createContext, useContext, useReducer } from "react";
2: import initialConfig from "../data/formConfig.json";
3:
4: // --- Reducer Function ---
5: const formConfigReducer = (state, action) => {
6:     switch (action.type) {
7:         case "UPDATE_FIELD":
8:             return {
9:                 ...state,
10:                 fields: state.fields.map((field, i) =>
11:                     i === action.index
12:                     ? { ...field, [action.property]: action.value }
13:                     : field
14:                 ),
15:             };
16:
17:         case "ADD_FIELD":
18:             return {
19:                 ...state,
20:                 fields: [...state.fields, action.field],
21:             };
22:
23:         case "DELETE_FIELD":
24:             return {

```

```

25:     ...state,
26:     fields: state.fields.filter((_, i) => i !== action.index),
27:   };
28:
29:   case "UPDATE_SETTINGS":
30:     return {
31:       ...state,
32:       ...action.settings,
33:     };
34:
35:   case "IMPORT_CONFIG":
36:     return {
37:       ...state,
38:       ...action.config,
39:     };
40:
41:   case "REORDER_FIELDS":
42:     return {
43:       ...state,
44:       fields: action.fields,
45:     };
46:   case "ADD_CATEGORY":
47:     return {
48:       ...state,
49:       categories: [...state.categories, action.name],
50:     };
51:
52:   case "UPDATE_CATEGORY":
53:     return {
54:       ...state,
55:       categories: state.categories.map((cat, i) =>
56:         i === action.index ? action.newName : cat
57:       ),
58:       fields: state.fields.map((field) =>
59:         field.category === state.categories[action.index]
60:         ? { ...field, category: action.newName }
61:         : field
62:       ),
63:     };
64:
65:   case "DELETE_CATEGORY":
66:     return {
67:       ...state,
68:       categories: state.categories.filter((_, i) => i !== action.index),
69:       fields: state.fields.map((field) =>
70:         field.category === state.categories[action.index]
71:         ? { ...field, category: "" }
72:         : field
73:       ),
74:     };
75:
76:   default:
77:     return state;
78: }
79: };
80:
81: // --- Context Creation ---
82: const FormConfigContext = createContext(null);
83:
84: // --- Provider ---
85: export const FormConfigProvider = ({ children }) => {
86:   const [state, dispatch] = useReducer(formConfigReducer, initialConfig);
87:
88:   return (
89:     <FormConfigContext.Provider value={{ state, dispatch }}>
90:       {children}
91:     </FormConfigContext.Provider>
92:   );
93: };
94:
95: // --- Custom Hook ---
96: export const useFormConfig = () => {
97:   const context = useContext(FormConfigContext);

```

```

98:   if (!context) {
99:     throw new Error("useFormConfig must be used within a FormConfigProvider");
100:   }
101:   return context;
102: };

```

■ 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 trigger",
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 limit"
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",
72:     "low": "Caffeine up to 5 cups a\nday can be consumed"
73: },
74: {
75:     "id": "gluten",
76:     "label": "Gluten Sensitivity",
77:     "category": "FOOD SENSITIVITIES",
78:     "min": 1,
79:     "max": 10,
80:     "high": "Gluten intake needs to be\nreduced/stopped",
81:     "normal": "Monitor gluten intake,\nreduce if experiencing\ndigestive issues",
82:     "low": "Gluten to be avoided in\ncases of gastric distress"
83: },
84: {
85:     "id": "lactose",
86:     "label": "Lactose Sensitivity",
87:     "category": "FOOD SENSITIVITIES",
88:     "min": 1,
89:     "max": 10,
90:     "high": "Milk & milk products need\nto be avoided",
91:     "normal": "Limit milk & milk products,\nconsider lactose-free\nalternatives if needed",
92:     "low": "Milk or milk products to\nbe avoided in gastric\ndistress"
93: },
94: {
95:     "id": "salt",
96:     "label": "Salt Sensitivity",
97:     "category": "",
98:     "min": 1,
99:     "max": 8,
100:     "high": "Try to reduce overall salt\nintake to up to 3-5 gm per\nday",
101:     "normal": "Maintain salt intake\naround 5 gm per day",

```

```

102:         "low": "Consumption of salt up to\n5 gm/day can be done"
103:     }
104: ]
105: }

```

■ File: src\hooks\useConfigImportExport.js

```

=====
1: import { useCallback } from "react";
2: import { useFormConfig } from "../contexts/FormConfigContext";
3:
4: export const useConfigImportExport = () => {
5:   const { state, dispatch } = useFormConfig();
6:
7:   const exportConfig = useCallback(() => {
8:     const dataStr = JSON.stringify(state, null, 2);
9:     const blob = new Blob([dataStr], { type: "application/json" });
10:    const url = URL.createObjectURL(blob);
11:
12:    const link = document.createElement("a");
13:    link.href = url;
14:    link.download = "genomics-form-config.json";
15:    link.click();
16:    URL.revokeObjectURL(url);
17:  }, [state]);
18:
19:   const importConfig = useCallback(
20:     (file) => {
21:       const reader = new FileReader();
22:       reader.onload = (e) => {
23:         try {
24:           const parsed = JSON.parse(e.target.result);
25:           dispatch({ type: "IMPORT_CONFIG", config: parsed });
26:           alert("Configuration imported successfully.");
27:         } catch (err) {
28:           alert("Error importing config: " + err.message);
29:         }
30:       };
31:       reader.readAsText(file);
32:     },
33:     [dispatch]
34:   );
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:      // -----
27:      // Header section
28:      // -----
29:      doc.setFillColor(...hexToRgb(state.headerColor));
30:      doc.rect(margin, y, pageWidth - margin * 2, 20, "F");
31:      doc.setTextColor(255, 255, 255);
32:      doc.setFontSize(16);
33:      doc.setFont(undefined, "bold");
34:      doc.text(state.title, pageWidth / 2, y + 13, { align: "center" });
35:      y += 30;
36:
37:      // Quote
38:      doc.setTextColor(0, 0, 0);
39:      doc.setFontSize(11);
40:      doc.setFont(undefined, "bold");
41:      doc.text(state.quote, pageWidth / 2, y, { align: "center" });
42:      y += 10;
43:
44:      // Description
45:      doc.setFont(undefined, "normal");
46:      doc.setFontSize(10);
47:      const descLines = doc.splitTextToSize(
48:        state.description,
49:        pageWidth - 2 * margin
50:      );
51:      doc.text(descLines, margin, y);
52:      y += descLines.length * 5 + 5;
53:
54:      let currentCategory = null;
55:
56:      reportData.forEach((item, index) => {

```

```

57:     const { field, score, showHigh, showNormal, showLow } = item;
58:
59:     // Insert page break if needed
60:     const estimatedFieldHeight = 40; // Rough estimate
61:     if (y + estimatedFieldHeight > pageHeight - 20) {
62:         doc.addPage();
63:         y = 20;
64:     }
65:
66:     // Render category title if needed
67:     if (field.category && field.category !== currentCategory) {
68:         currentCategory = field.category;
69:         doc.setFontSize(12);
70:         doc.setFont(undefined, "bold");
71:         doc.setTextColor(50, 50, 50);
72:         doc.text(currentCategory, margin, y);
73:         y += 8;
74:     }
75:
76:     // Field Label
77:     doc.setFontSize(10);
78:     doc.setFont(undefined, "bold");
79:     doc.setTextColor(0, 0, 0);
80:     doc.text(field.label, margin, y);
81:     y += 6;
82:
83:     // Score Circle
84:     const circleX = margin + 5;
85:     doc.setDrawColor(0);
86:     const rgb =
87:         score >= state.highThreshold
88:         ? hexToRgb(state.colors.high)
89:         : score >= 4
90:         ? hexToRgb(state.colors.medium)
91:         : hexToRgb(state.colors.low);
92:
93:     doc.setFillColor(...rgb);
94:     doc.circle(circleX, y + 5, 4, "FD");
95:     doc.setTextColor(255, 255, 255);
96:     doc.setFontSize(8);
97:     doc.text(String(score), circleX, y + 6, { align: "center" });
98:
99:     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 {
9:     transition: background-color 0.3s ease;
10:  }
11:
12:   body {
13:     @apply bg-white text-gray-900 dark:bg-gray-900 dark:text-white;
14:  }
15:
16:   /* Optional: Customize link styles, buttons, etc. */
17:   a {
18:     @apply text-blue-600 dark:text-blue-400;
19:  }
20:
21:   button {
22:     @apply bg-gray-300 dark:bg-gray-700 text-gray-800 dark:text-gray-100;
23:  }
24:
25:   /* Global font and smoothing */
26:   body {
27:     margin: 0;
28:     font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Oxygen",
29:       "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue", sans-serif;
30:     -webkit-font-smoothing: antialiased;
31:     -moz-osx-font-smoothing: grayscale;
32:  }
33:
34:   code {
35:     font-family: source-code-pro, Menlo, Monaco, Consolas, "Courier New", monospace;
36:  }
37: }

```

■ 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: };
-----
```