ESCWA Skills Monitor

Technical Documentation

Version 1.0

Last Updated: December 2025

Table of Contents

Architecture Overview

The ESCWA Skills Monitor is a single-page application (SPA) built with React 19, utilizing a modern component-based architecture with client-side routing and data visualization capabilities.

High-Level Architecture

The application follows a layered architecture:

- **Browser (Client)** – React 19 + React Router
- **Layout Layer** – DefaultLayout (Theme Provider)
- **Routing Layer** – AppRoutes (Route Configuration)
- **Page Components** – CountryCitiesMap, GreenOverview, GreenOccupations, GreenInEnergySector, Benchmarking
- **Data Layer** – CSV Files, Excel Files, Streamlit External Service

Design Patterns

- **Component-Based Architecture** – Modular, reusable components
- **Container/Presentational Pattern** – Pages manage state, components handle presentation
- **Layout Pattern** – DefaultLayout wrapper provides consistent UI shell
- **Dynamic Theming** – CSS variables driven by route context
- **Data Fetching** – Client-side fetch with async/await
- **Error Boundaries** – Loading and error states for data components

Technology Stack

Core Framework

Package	Version & Description
React	19.1.1 – Modern UI library with latest features
React DOM	19.1.1 – DOM-specific methods
React Router DOM	7.9.5 – Client-side routing

Build Tools

Tool	Version & Purpose
Vite	rolldown-vite@7.1.14 – Ultra-fast bundler and dev server
@vitejs/plugin-react	5.0.4 – React Fast Refresh support

UI Libraries

- **PrimeReact 10.9.7** – Comprehensive component library (Theme: saga-blue)
- **Tailwind CSS 4.1.16** – Utility-first CSS framework
- **Lucide React 0.554.0** – Icon library

Data Visualization

- **AG Charts React 12.3.1** – Advanced charting library (Community + Enterprise)
- **Chart.js 4.5.1** – Flexible charting library

Data Processing

- **PapaParse 5.5.3** – CSV parsing library
- **XLSX 0.18.5** – Excel file parsing
- **Hyparquet 1.20.2** – Parquet file support

Project Structure

The project follows a standard React application structure:

Root Directory

- **public/** – Static data files (CSV, Excel)
- **src/** – Source code
- **vite.config.js** – Vite configuration
- **package.json** – Dependencies

Source Directory (src/)

- **components/** – Reusable UI components (25+ files)
- **layouts/** – DefaultLayout.jsx
- **pages/** – Route pages (8 files)
- **utils/** – Utility functions
- **App.jsx** – Root component
- **AppRoutes.jsx** – Route configuration
- **main.jsx** – Application entry point

Data Files (public/data/)

Green Jobs Data:

- green_fig1.csv through green_fig9.csv

Benchmarking Data:

- benchmarking_fig1_2.csv, benchmarking_fig3_5.csv, etc.
- 22 country-specific files (benchmarking_fig6_*.csv)

Application Flow

1. Application Initialization

Entry Point: src/main.jsx

1. React creates root element
2. Wraps app in StrictMode for development checks
3. Initializes BrowserRouter for client-side routing
4. Renders App component

2. App Component

File: src/App.jsx

Simple pass-through to routing layer. Redux Provider is commented out for future state management.

3. Routing Layer

File: src/AppRoutes.jsx

All routes wrapped in DefaultLayout. Legacy routes redirect to consolidated pages. 404 redirects to home.

4. Layout System

File: src/layouts/DefaultLayout.jsx

Responsibilities:

- Route-based theming
- Header and navigation
- Content container
- Theme variable injection

Component Architecture

Layout Components

DefaultLayout Features:

- Dynamic theming based on route
- Sticky header with branding
- PrimeReact Menubar integration
- CSS custom properties for theme tokens
- Responsive design

Page Components

Component	Description
CountryCitiesMap	Embeds Streamlit app via iframe for interactive mapping
GreenOverview	Displays GreenFig1 (timeline) and GreenFig2 (overview metrics)
GreenOccupations	Shows GreenFig3 and GreenFig4 occupation breakdowns
GreenInEnergySector	Multi-section energy analysis with GreenFig6-9
Benchmarking	Interactive country comparison with 7 analytical sections

Data Management

Data Sources

1. CSV Files (Static)

Location: /public/data/*.csv

Format: Tab-delimited or comma-separated

2. Excel Files (XLSX)

- grouped_country_with_desc_updated_aggregated_clustered.xlsx – Geographic centers
- cities_skills_with_desc_updated.xlsx – Skills by city

3. External Services

Streamlit App: https://cities-skills-v2.streamlit.app/

Data Parsing Utilities

File: src/utils/ParseUtils.jsx

- **safeParseInt(val)** – Convert to integer with fallback to 0
- **safeParseFloat(val)** – Convert to float with fallback to 0

Data Transformation

- Job counts normalized to "per 1,000" or "per 100" for fair comparison
- Percentage calculations for green job shares
- Year filtering (2021-2025 range)

Routing System

Route Configuration

Path	Component
/	CountryCitiesMap
/greenOverview	GreenOverview
/greenOccupations	GreenOccupations
/greenInEnergySector	GreenInEnergySector
/benchmarking	Benchmarking
*	Navigate to /

Navigation Menu Structure

1. **Jobs Across ESCWA Countries** – navigates to /
2. **Green Jobs** – submenu with Overview, Green Occupations, Energy Sector
3. **Benchmarking** – navigates to /benchmarking

Theming System

CSS Custom Properties

Global styles defined in src/index.css use CSS custom properties for theming:

- --theme-accent
- --theme-accent-strong
- --theme-accent-soft
- --theme-ink
- --theme-muted
- --theme-surface
- --theme-shadow-soft

Theme Tokens

Theme	Accent Color	Usage
Purple	#8b5cf6	Jobs/Home section
Green	#0ea36d	Green Jobs section
Blue	#0ea5e9	Benchmarking section

Theme Detection Logic

Theme is determined based on route path: routes containing "green" use green theme, routes containing "bench" use blue theme, all others use purple theme.

Chart Components

Common Pattern

All chart components follow a consistent pattern:

1. State management for chartData, chartOptions, loading, and error
2. useEffect hook for data fetching on mount
3. Try/catch block for error handling
4. Loading and error state rendering
5. Chart rendering with configuration

Chart Libraries

- **Chart.js (via PrimeReact)** – Used in GreenFig1, GreenFig2
- **AG Charts** – Used in most benchmarking components

Chart Types

- **Radar Charts** – Benchmarking skill distributions
- **Line Charts** – Time series (green jobs trends)
- **Bar Charts** – Comparative metrics (country comparisons)
- **Heatmaps** – Skill similarity matrices

Development Setup

Prerequisites

- **Node.js** – v18+ recommended
- **npm** – v9+ or equivalent package manager
- **Git** – For version control

Installation

1. Clone repository: git clone <repository-url>
2. Navigate to directory: cd skillsmonitoradditional
3. Install dependencies: npm install
4. Start development server: npm run dev

Available Scripts

Command	Description
npm run dev	Start development server with HMR
npm run build	Create production build
npm run lint	Run ESLint
npm run preview	Preview production build locally

Build & Deployment

Production Build

Command: npm run build

Output: dist/ directory

Build Process:

1. Vite bundles all JavaScript/JSX
2. Tailwind CSS processes utility classes
3. Assets optimized and hashed
4. HTML entry point generated

Build Optimizations

- **Code Splitting** – Automatic by Vite
- **Tree Shaking** – Dead code elimination
- **Minification** – JavaScript and CSS
- **Asset Optimization** – Images, fonts

Deployment Platforms

Static hosting compatible:

- Vercel
- Netlify
- GitHub Pages
- AWS S3 + CloudFront
- Azure Static Web Apps

Configuration Required: SPA fallback to index.html for all routes. CORS headers for Streamlit embed if needed.

Performance Considerations

Bundle Size Optimization

Recommendations:

1. **Code Splitting** – Use lazy() for route-based splitting
2. **Tree Shaking** – Import only used components
3. **Dynamic Imports** – Load charts on demand

Data Loading Optimization

Current Issues:

- Multiple CSV fetches
- No caching between route changes

Improvements:

1. **Data Caching** – Implement Map-based cache
2. **Prefetching** – Load data before route transition

3. **Service Worker** – Cache CSV files offline

Render Optimization

- **React.memo** – Prevent unnecessary re-renders
- **Virtual Scrolling** – For large data tables
- **Debouncing** – Country selection changes

Security Considerations

Content Security Policy (CSP)

Currently not implemented. Recommended to add CSP headers for default-src, script-src, style-src, font-src, frame-src, and connect-src directives.

Data Validation

ParseUtils.jsx provides safe parsing with fallbacks. Additional checks should validate CSV structure and data ranges.

Third-Party Dependencies

Audit regularly using:

- npm audit
- npm audit fix

Iframe Security

Streamlit embed currently uses basic iframe. Recommend re-enabling sandbox attribute with minimal permissions: allow-same-origin, allow-scripts, allow-forms, allow-popups.

API Reference

Utility Functions

Function	Description
safeParseInt(val)	Convert to integer with fallback to 0 if NaN
safeParseFloat(val)	Convert to float with fallback to 0 if NaN
haversine(lat1, lon1, lat2, lon2)	Calculate distance between coordinates in km

Component Props

All benchmarking chart components accept a **selectedCountries** prop (array of country names). Example: BenchmarkingFig1Fig2, BenchmarkingFig3, BenchmarkingFig4, etc.

Troubleshooting

Common Issues

1. CSV Loading Errors

Symptom: "Error loading chart data: HTTP error! status: 404"

Cause: CSV file not found in /public/data/

Solution: Verify file exists and name matches exactly (case-sensitive on Linux)

2. Chart Not Rendering

Symptom: Blank chart area, no errors

Possible Causes: Data parsing failed silently, CSV delimiter mismatch, invalid data format

Solution: Add console.log statements to debug raw CSV, parsed data, and chart options

3. Streamlit Embed Not Loading

Symptom: Blank iframe or CORS errors

Causes: Streamlit app down, CORS policy blocking, network firewall

Solution: Test Streamlit URL directly in browser, update URL if moved

4. Build Failures

Symptom: npm run build fails

Common Causes: ESLint errors, missing dependencies, syntax errors

Solution: Run npm run lint, clean install (rm -rf node_modules && npm install), clear Vite cache

5. Slow Performance

Symptoms: Slow route transitions, chart loading delays, high memory usage

Solutions: Implement data caching, add loading skeletons, use React.memo, reduce bundle size

Browser Compatibility

Supported: Chrome 90+, Firefox 88+, Safari 14+, Edge 90+

Unsupported: Internet Explorer (ES6+ required)

Appendix

Country List

22 countries supported in benchmarking:

United States (locked baseline), Algeria, Bahrain, Djibouti, Egypt, Iraq, Jordan, Kuwait, Lebanon, Libya, Mauritania, Morocco, Oman, Palestine, Qatar, Saudi Arabia, Somalia, Sudan, Syria, Tunisia, United Arab Emirates, Yemen.

External Dependencies

- **PrimeReact Theme:** saga-blue
- **Font:** Space Grotesk (Google Fonts)
- **Streamlit App:** https://cities-skills-v2.streamlit.app/

Roadmap

Planned Features:

1. Authentication – User login and role-based access
2. Data Export – Download charts as PNG/CSV
3. Comparison Tool – Save and compare country selections
4. Real-time Updates – WebSocket integration
5. Mobile Optimization – Enhanced mobile UI
6. Offline Mode – Service worker support
7. Analytics – User behavior tracking
8. Multi-language – i18n support (Arabic, English, French)

Technical Debt:

- Redux Integration cleanup
- Legacy route removal
- Global data cache implementation
- TypeScript migration
- Unit and integration tests
- WCAG 2.1 AA accessibility compliance

Version Information

Document Version: 1.0

Application Version: 0.0.0

Last Updated: December 2025

Maintained By: Development Team