Global Wellness Perspectives - Implementation Guide

This guide provides instructions for implementing the Global Wellness Perspectives application for both human developers and Cursor AI assistant.

Project Overview

Global Wellness Perspectives is a Next.js application that visualizes wellness metrics across different countries and societal structures. The goal is to help users expand their understanding of alternative social systems through objective data rather than cultural stereotypes.

Project Structure

Copy

```
global-wellness-perspectives/
├── pages/
│   ├── index.js                 # Main application page
│   ├── api/
│   │   └── wellness-data.js     # API endpoint for data
├── components/
│   ├── WellnessComparison.js    # Visualization component
│   ├── CountrySelector.js       # Country selection UI
│   ├── MetricSelector.js        # Metric selection UI
│   ├── WellnessInsights.js      # Data analysis component
│   ├── SocietalStructuresContext.js  # Educational component
│   ├── EducationalResources.js  # Resources list component
│   ├── FAQ.js                   # FAQ component
├── public/
│   └── favicon.ico              # Site favicon
├── styles/
│   └── globals.css              # Global styles
├── package.json
├── next.config.js
└── tailwind.config.js
```

Implementation Steps for Human Developers

1. Set Up the Next.js Project

bash

Copy

```bash
# Create a new Next.js project with Tailwind CSS
npx create-next-app global-wellness-perspectives
cd global-wellness-perspectives
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p

# Install required dependencies
npm install recharts axios
```

2. Configure Tailwind CSS

Update `tailwind.config.js`:

javascript

Copy

```javascript
module.exports = {
  content: [
    './pages/**/*.{js,ts,jsx,tsx}',
    './components/**/*.{js,ts,jsx,tsx}',
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

Update `styles/globals.css`:

css

Copy

```css
@tailwind base;
@tailwind components;
@tailwind utilities;
```

3. Create the File Structure

Create folders and files according to the project structure above.

4. Implement the Components

Copy the provided code for each component into its corresponding file:

- Copy the main page code into `pages/index.js`
- Copy the API endpoint code into `pages/api/wellness-data.js`
- Create each component in the components directory

5. Data Sourcing Strategy

For a production implementation, you'll need real data. Consider these options:

1. **Static Data Approach**:
   - Manually compile data from various sources (OECD, WHO, UN, World Happiness Report)
   - Store as JSON files in the project or in a simple database
   - Update annually when new reports are released
2. **API Integration Approach**:
   - Integrate with existing data APIs where available (Our World In Data, World Bank, etc.)
   - Create a backend service to aggregate and normalize data from various sources
   - Implement a caching strategy to minimize API calls
3. **Hybrid Approach**:
   - Start with static data for the MVP
   - Gradually add API integrations for metrics that have reliable, accessible APIs
   - Create an admin interface for manual data updates

6. Launch and Testing

1. Test the application locally:

bash Copy

```
npm run dev
```

2. Review all components for functionality and accuracy
3. Test responsive behavior on different screen sizes
4. Verify data visualization accuracy
5. Build for production:

bash

Copy

```
npm run build
npm run start
```

Implementation Guide for Cursor

Cursor can assist with this project in several ways:

Code Generation and Completion

1. **Component Implementation**:
   - Request Cursor to implement specific components based on the provided code
   - Ask Cursor to suggest improvements or optimizations for the visualization components
   - Have Cursor generate additional metric definitions or country data
2. **Debugging**:
   - When encountering errors, share the error message with Cursor for troubleshooting
   - Ask Cursor to review code for potential issues or performance bottlenecks
3. **Feature Extensions**:
   - Request Cursor to help implement additional features, such as:
     - User accounts to save favorite comparisons
     - More advanced data visualizations
     - Data export capabilities
     - Mobile-specific optimizations

Data Processing

1. **Data Transformation**:
   - Ask Cursor to write data processing utilities for normalization and transformation
   - Request help with statistical analysis for insights generation
   - Get assistance with data validation functions
2. **API Integration**:
   - Request code for integrating with external data APIs
   - Ask for help implementing caching strategies
   - Get assistance with error handling for API requests

Data Schema

The application expects data in the following format from the API:

javascript Copy

```javascript
{
  countries: ["USA", "Finland", "Japan", "Germany", "Costa Rica"],
  metrics: {
    happiness: {
      "USA": 70.5,
      "Finland": 92.3,
      "Japan": 83.1,
      "Germany": 82.4,
      "Costa Rica": 87.2
    },
    healthcare: {
      "USA": 60.2,
      "Finland": 88.4,
      "Japan": 89.5,
      "Germany": 87.3,
      "Costa Rica": 69.1
    },
    // Additional metrics...
  }
}
```

Future Development Considerations

1. **Localization**: Add support for multiple languages to make the tool more accessible globally
2. **Data Customization**: Allow users to upload their own datasets for comparison
3. **Time Series Analysis**: Add historical data to show how metrics have changed over time
4. **Interactive Learning Modules**: Create guided tours through the data to highlight specific patterns or insights
5. **Community Features**: Add discussion capabilities for users to share observations and insights

---

This guide should provide both human developers and Cursor with the information needed to implement and extend the Global Wellness Perspectives application.