Infinite scroll - Weather Forecast Web Application

ASSIGNMENT HAS BEEN UPDATED WITH MORE DESCRIPTION TO ANSWER SOME FAQs

Description:

For this assignment, you will create an optimized cities table and a weather forecast web application using React, TypeScript.

Core Requirements:

Display Cities in a table:

- Show all cities in a table format with **infinite scroll** (use this api to get the city data: https://public.opendatasoft.com/explore/dataset/geonames-all-cities-with-a-population-1000/api/?disjunctive.cou_name_en&sort=name).
- The table should have columns city name, country, timezone etc.
- Implement search as you type for this table. Utilize an autocomplete feature to suggest possible locations as users type.
- Implement filter and sorting for each column
- Clicking on the city name, should take to to the weather page for this city
- Right clicking on the city name and opening in the new tab should also open the weather page for the city but in the new tab.

Weather page:

- You can reach this page by clicking on the city name in the table section as mentioned in the previous section.
- Use https://openweathermap.org free

(temperature, weather description, humidity, wind speed, and atmospheric pressure), forecast (temperature highs and lows, weather descriptions, and precipitation chances) etc.

- Optional: You can get innovative and do stuff like display location on map, show options for changing unit etc
- Once weather data is loaded on this page, display basics like day high/low, for the city, on the cities table page as well.

Other requirements:

Styling:

- Use appropriate color pallets & styled components to design layouts and different pages.
- Implement dynamic backgrounds based on the current weather conditions.
- Use appropriate images or animations to represent different weather conditions (e.g., sunny, rainy, cloudy).

Responsive Design:

- Ensure that the application is responsive and works well on different screen sizes.
- Implement media queries or responsive design techniques.

Error Handling:

- Handle errors gracefully in case of failed API requests or invalid search queries.
- Display error messages to users when necessary.

down necessary data as props.

Type Safety:

- Utilize TypeScript for type safety throughout the application.
- Define interfaces or types for weather objects and API responses.

Optional Features:

- Utilize the preferred stack, such as Next.js for the frontend framework, MST (MobX State Tree) for state management, and Tailwind CSS for styling.
- Implement geolocation to automatically detect and display the weather for the user's current location.
- Maintain history of viewed locations' weather.
- Provide options to switch between different units of measurement (e.g., Celsius/Fahrenheit, metric/imperial).
- Add a feature to save favorite locations for quick access.

Deployment:

- Deploy your project with any hosting provider.
- Choose a hosting provider that best fits your needs:
  - Netlify
  - Vercel
  - GitHub Pages
  - AWS Amplify
  - Firebase Hosting

Submission Guidelines:

- Create a GitHub repository for your project.
- Document any setup instructions or prerequisites in a README.md file.
- Ensure that your code is well-documented and follows best practices.
- Test your application thoroughly to ensure it works as expected.
- Deploy your application on a hosting server.

your skills and creativity.
- Consider user experience and accessibility while designing the application.
- Make sure to handle edge cases and error scenarios gracefully.

APIs:

- To get locations use open public API. https://public.opendatasoft.com/explore/dataset/geonames-all-cities-with-a-population-1000/table/?disjunctive.cou_name_en&sort=name
- Use a free account to generate an API key to access OpenWeatherMap current and 5-day forecast weather APIs. (https://openweathermap.org)

Submission Link:
https://forms.gle/yPeaPWegqv2Tab829