Senior Web Developer Coding Exercise

We would like you to solve the following coding challenge designed specifically for senior-level developers.

Please note this challenge is solely for the purpose of interviewing and all code will be discarded post interview.

Please read ALL this document before you begin.

Weather Alert

Assessment Criteria

We analyse your coding challenge submission using several criteria, particularly emphasizing senior-level competencies:

Core Technical Skills:

- Code quality and architecture
- Readability and maintainability
- Relevant technology selection and implementation
- Pragmatic design choices
- Comprehensive testing practices
- Consistent coding techniques
- Evidence of functional testing and robust deliverable
- Long-term supportability
- Breadth of skills demonstrated

Senior-Level Evaluation Points:

- **Architectural decisions**: How you structure and organize the codebase
- **Performance optimization**: Efficient API usage, caching strategies, memoization
- **Security considerations**: API key management, input validation, error handling
- **Scalability planning**: Code that could handle growth in users and features
- **Production readiness**: Error boundaries, logging, monitoring considerations
- **Documentation quality**: Clear README, architectural decisions, code comments

It is expected that the challenge would take **6-8 hours to complete**. We value quality over speed - focus on demonstrating clean architecture and solid engineering practices.

Please consider that you may be required to reason about your solution, justify implementation and design choices you have made, and demo your solution during your interview.

The App: Weather Alert

Implement a React/NodeJS website that will allow the user to check the wind conditions and forecast at one or more locations around the world.

Technical Requirements

Stack Requirements:

- TypeScript strongly preferred
- Include at least basic unit and integration tests
- Implement proper error boundaries and comprehensive error handling
- Include API rate limiting/caching strategy to handle the 2000 requests/hour limit
- Demonstrate responsive design principles
- Support for current versions of modern browsers

User Interface

The design of the UI is totally down to you. We expect you to choose a sensible user interface design and implement controls, views and layout in a way that will offer users an excellent experience. We expect the use of modern technologies and patterns in the app.

Requirements

Executive Summary

We require a website that allows a user to monitor current and/or forecast wind information from selected locations around the world.

The Data Source

Use the OpenWeather API for this challenge. The current API documentation can be found at: https://openweathermap.org/api

All functionality within the requirements can be achieved using the free account service. The app can use the JSON API (XML support is deprecated).

Important API Note: The OpenWeather API is rate-limited to prevent abuse. More than 2000 API requests in one hour can result in throttling or temporary bans. **Your implementation must demonstrate awareness of this constraint** through appropriate caching, request optimization, or user experience design.

The Main View – Favourite Locations

This is the main view and home page of the website. Users can:

- See previously added 'favourite' locations
- Add new 'favourite' locations for wind monitoring
- View current or forecast wind information summary

Implementation Considerations:

- How you persist user favorites (database, localStorage, etc.) is your choice - justify your decision
- Consider user experience for both first-time visitors and returning users
- Think about what data you can practically display without overwhelming the interface
- No limit on favorite locations, though you may implement reasonable constraints

State Management: On access, the site should check if the user has already chosen locations in previous sessions and present them in this main view.

Empty State: When a user accesses the site for the first time, or when no favourite locations exist, the interface should be clear and intuitive with prominent "add location" functionality.

Getting Locations

Read the API documentation to determine how to search for cities effectively. Consider implementing:

- Search autocomplete or suggestions
- Handling of ambiguous city names
- User-friendly location selection process

The Wind Forecast - Detail View

For any favourite location, users should be able to access detailed forecast information.

UI Design Considerations:

- Information must be clear with intuitive navigation
- Consider data visualization for wind direction, speed, and patterns
- Wind direction concepts: refer to cardinal directions if needed
- Focus on wind-specific data while making practical use of available weather information

Submission Requirements

Your submission must include:

- Source code with complete git history
- Comprehensive README with:

    - Setup and installation instructions
    - Architecture overview (1-2 pages)
    - Technology choices and justifications
    - Known limitations or assumptions
    - Future improvement suggestions
- Test results and coverage information
- Any deployment configurations or scripts

Interview Discussion Points

Be prepared to discuss:

- Your architectural decisions and trade-offs
- How you would scale this application
- Security considerations you implemented
- Performance optimizations you made
- How you would monitor this application in production
- Team collaboration and maintenance considerations

Submission Instructions

Please send us the solution as:

- **Preferred**: Link to a source repository (GitHub, GitLab, BitBucket)
- **Alternative**: Zip file with complete source code and git history (include .git folder)

Host zip files on any file sharing service (Dropbox, WeTransfer, etc.).

Questions or Issues

If you have problems or questions about the challenge, please contact:
darren.murphy@homegroup.org.uk

Remember: This exercise is designed to showcase senior-level thinking, not just implementation ability. Focus on demonstrating the depth of experience and architectural judgment we'd expect from a senior team member.