Coding Assignment – *please return in 72 hours of receiving. If you need more time, please message me.*

Requirements:

- Must be done in Ruby on Rails
- Accept an address as input
- Retrieve forecast data for the given address. This should include, at minimum, the current temperature (Bonus points - Retrieve high/low and/or extended forecast)
- Display the requested forecast details to the user
- Cache the forecast details for 30 minutes for all subsequent requests by zip codes. Display indicator if result is pulled from cache.

Assumptions:

- This project is open to interpretation
- Functionality is a priority over form
- If you get stuck, complete as much as you can

Submission:

- Use a public source code repository (GitHub, etc) to store your code
- Send us the link to your completed code

Reminders:

Please remember – <u>it's not just whether or not the code works</u> that they will be focused on seeing – <u>it's *all the rest* of what goes into good Senior Software Engineering daily practices for **Enterprise Production Level Code**</u> – such as *specifically*:

- ***Unit Tests*** *(#1 on the list of things people forget to include – so please remember,* ***treat this as if it were true production level code****, do not treat it just as an exercise)*,
- Detailed **Comments/Documentation** within the code, also have a README file
- Include ***Decomposition*** of the Objects in the Documentation
- **Design Patterns** (if/where applicable)
- **Scalability** Considerations (if applicable)
- **Naming Conventions** (name things as you would name them in enterprise-scale production code environments)
- **Encapsulation**, (don't have 1 Method doing 55 things)
- **Code Re-Use**, (don't *over*-engineer the solution, but don't *under*-engineer it either)
- and *any* other **industry Best Practices**.
- Remember to Include the **UI** ***
- No not use **ChatGPT/AI**

*****Tips *****

- Emphasize clean, straightforward code without overcomplication. They have passed on people who overcomplicate it or try to do too much.

- Ensure functional and maintainable code that's easy to understand and follow.
- Avoid unnecessary complexity; strive for a clear path from point A to point B.
- Integrate and handle various data sources robustly, ensuring error handling and format flexibility.
- Implement effective caching strategies for real-world application usage.
- Code should be easy to follow; it should be acceptable for inclusion in the team's codebase.
- Time-efficient: should take about 2-3 hours unless doing something extraordinary.
- Flexibility in UI design; not prescribing specific methods.
- Ensure live deployments are robust and resilient to breakages.