# Goal

Build a simple weather forecast application based on the OpenWeatherMap API (https://www.openweathermap.org/api). You can sign up for a free account and API key at https://www.openweathermap.org/appid.

# Tasks

## 1.

Create a "Current Temperature" page that fetches and displays the current temperature at the user's current location. See https://www.openweathermap.org/current for relevant documentation.

## 2.

Add a "5 Day Temperatures" page to your application that fetches and displays the 5 day forecast for the user's current location. Display all of the 3-hourly forecasts within this 5 day period. See https://www.openweathermap.org/forecast5 for relevant documentation.

The user should be able to access this "5 Day Forecast" page from the "Current Temperature" page. You may choose the navigation pattern used. For example, both pages could be top-level tabs within your application.

Each row in the list should display:

- The forecast date and time,
- The forecast temperature in Fahrenheit, and
- The OpenWeatherMap icon that represents the forecast weather conditions. See https://openweathermap.org/weather-conditions for relevant documentation.

# FAQ

### How long should I/may I spend on this exercise?

Please spend no more than 3 hours total on this exercise; simply complete as much of it as you can within that time.

### Which languages/frameworks may I use?

Our current technology stack at Detroit Labs includes the languages and frameworks listed below. These languages and frameworks are preferred for completing this exercise.

Languages:

- JavaScript
- Java
- Kotlin
- Objective-C
- Swift

Frameworks:

- React
- Angular
- Node.js
- Android
- iOS

If you normally use a project template when creating new applications, feel free to use it for this exercise.

### How will my submission be evaluated?

At Detroit Labs, we care about:

- **Code quality, style, and readability**. Code should be of excellent quality and should follow standard style guides for the language/platform of choice. We value clarity, simplicity, and maintainability. You should be proud of the code you send us for this exercise.
- **Architecture and design**. The implementation mechanics, libraries, databases, and architecture(s) used should be suitable for this exercise. Avoid using unnecessary tools that add more complexity than value.
- **Error and edge case handling**. Identify and handle any errors and edge cases as you would for a production application.
- **Appropriate documentation**. The application should contain the level of documentation and commentary you would use in a production application. Your commit history should be clean, with each commit representing a solid piece of work in your application.
- **Completeness and correctness of features**. The features implemented should be complete, and accurate/correct in line with the asks of the exercise.

- **Platform familiarity**. Your solution should use components and patterns in line with platform and/or framework guidelines.
- **UX & Design**. We're assessing your development skills, not your design skills, so don't spend time making fancy graphics. We only ask that the user interface be clean and clear and follow relevant platform guidelines.

# Submitting

Services applicants:

- Bring your code with you to your interview.

OnSite applicants:

- Create an archive file (.ZIP, .RAR, etc) that contains all your code as well as a copy of this PDF.
- Email this archive file back to your contact at Detroit Labs. (If your email is rejected because of the .ZIP attachment, change the archive file extension to .PIZ to avoid the filtering!)
- Make sure any services that your app relies on are deployed so we can run and test your app.
- Once we have reviewed your submission we will be in contact regarding the outcome and next steps.

Thank you for applying to Detroit Labs!