PM Accelerator, a beacon of guidance for aspiring and experienced PMs alike. We've designed this platform to offer training, education, and job opportunities for Product Managers, creating room for constant improvement and shaping the next generation of PMs. Whether you're a newbie or you've spent years in the industry, you're guaranteed to find a new opportunity with the help of PM Accelerator.

By making industry-leading tools and education available to individuals from all backgrounds, we level the playing field for future PM leaders. This is the PM Accelerator motto, as we grant aspiring and experienced PMs what they need most – Access. We introduce you to industry leaders, surround you with the right PM ecosystem, and discover the new world of AI product management skills.

Project Overview

This project is a weather forecasting application built with Next.js. It integrates with AccuWeather's API to retrieve:

- Current weather conditions (temperature, weather icon/description, date/time).
- A 5-day extended forecast (daily highs, lows, general weather description).
- A 12-hour hourly forecast showing temperatures at specific hours.

It uses React hooks and styled components to keep the UI modular, responsive, and easy to extend.

Key Features

- City-based search for weather data.
- Displays detailed current weather with temperature, condition phrase, feels-like, wind speed, and humidity.
- Shows a 5-day forecast (high/low temperatures, a simple description like "Cloudy," etc.).
- Provides an hourly forecast (up to 12 hours) in a grid.
- Automatic greeting changes (Good Morning, Good Afternoon, etc.) based on current time.

Technology Stack

1. **Next.js**: Provides server-side rendering, file-based routing, and project structure.
2. **React**: For building the UI using customizable, re-usable components.
3. **Styled Components (@emotion/styled)**: CSS-in-JS library used for styling.
4. **Axios**: For making HTTP requests to the AccuWeather API.

File & Directory Structure

```
weather-forecast/
├── app/
│   ├── layout.js        // Root layout, includes metadata & global <head> setup
│   ├── page.js          // Main page that manages state, fetches weather data
│   └── globals.css      // Global CSS resets and styling
│
├── components/
│   ├── CurrentWeather.js // Displays current day's weather
│   ├── DailyForecast.js  // Lists daily weather for 5 days
│   ├── HourlyForecast.js // Shows up to 12 hours of hourly weather
│   ├── SearchForm.js     // Handles user input for city lookups
│   └── WeatherDetails.js // Additional info: wind speed, humidity, feels-like
│
├── lib/
│   └── api.js           // Functions calling AccuWeather endpoints
│
└── ... (configuration, README.md, etc.)
```

layout.js

Sets up HTML metadata and references the "Font Awesome" library. Everything in the Next.js app flows through this root layout.

page.js

Handles the main fetching logic and renders the UI by combining multiple components. Uses React hooks (useState, useEffect) to:

1. Perform a default fetch (e.g., for "Bankura") on mount.
2. Store fetched weather data in state.
3. Render child components that receive the data as props.

globals.css

Defines global CSS rules for consistent styling of elements like body, input, and button.

Components

CurrentWeather.js

Shows the current date, time, city name, temperature, and condition phrase. Uses small helper functions to format date/time strings.

DailyForecast.js

Lists five daily forecasts in horizontally scrollable cards. Each card shows day name ("Today," "Mon," "Tue," etc.), high/low temperatures, and a short description (like "Cloudy" or "Rain").

HourlyForecast.js

Displays up to 6 (out of the 12) hourly forecasts in a grid. Each cell shows an hour label and a temperature.

SearchForm.js

Provides a text input for the user to search by city name. On submission, it calls an onSearch callback that triggers the weather-fetching logic in page.js.

WeatherDetails.js

Presents extra information:

- Greeting based on time of day.
- Wind speed and unit.
- Humidity percentage.
- Feels-like temperature.

API Integration

All requests rely on AccuWeather's API using environment variables:

- **getLocationKey(query)**
  Calls the "locations/v1/search" endpoint to turn a city name into a location key.
- **get5DayForecast(locationKey)**
  Retrieves the 5-day daily forecasts.
- **get24HourForecast(locationKey)**
  Retrieves the 12-hour hourly forecast.
- **getCurrentForecast(locationKey)**
  Returns the live conditions (temperature, humidity, wind, etc.).

All these functions reside in api.js and use Axios to handle the HTTP calls.

Setup & Running

1. **Install Dependencies**
   npm install
2. **Environment Variable**
   Create a file named .env.local with:
   NEXT_PUBLIC_ACCUWEATHER_API_KEY=YOUR_API_KEY_HERE
3. **Development Server**
   npm run dev
   Opens the app at http://localhost:3000
4. **Build & Production**
   npm run build
   npm run start

How It Works (Workflow)

1. User loads the home page → A default city weather request occurs in useEffect (page.js).
2. The Next.js page fetches the location key, then calls the 5-day forecast, 12-hour forecast, and current conditions simultaneously (Promise.all).
3. Weather data is stored in local state (weatherData).
4. Child components (CurrentWeather, DailyForecast, HourlyForecast, WeatherDetails) receive the relevant pieces of weatherData as props.
5. The user can type a new city into SearchForm → triggers updateWeather() → fetches new location key/weather details → re-renders the UI with updated data.

Conclusion

This Next.js app provides a standalone, responsive interface for accessing and displaying AccuWeather data. It shows how a combination of React components, styled components, and external services can produce a clean user experience. Additional cities can be searched on the fly, with data seamlessly re-fetched and re-rendered.