Project 10. Checklist

Project rejected without review

- Errors occur when building or running the project.
- The markup has not been ported into JSX.
- Part of the functionality has not been implemented: more than three components have not been coded.
- There are questions addressed to the reviewers in the project.
- Critical mistakes discovered during a previous review (or reviews) have not been fixed.

Project requirements

General

- Infrastructural project files are created using CRA.
- The project is built and run without errors.
- HTML, CSS, and JS files are stored in the `src` folder using one of the following approaches:
    - JS and CSS files are located together inside folders, grouped by component.
    - File are grouped by type (`components`, `blocks`).
- Stylesheets are connected.
- The project contains the following:
    - The `index.css` and `index.js` files
    - The `index.html` file, stored in the public folder
    - An `images` folder
    - CSS files with styles for corresponding components

  - A `components` directory, which contains the following components: `App.js`, `Footer.js`, `Header.js`, `ItemCard.js`, `ItemModal.js`, `Main.js`, `ModalWithForm.js`, and `WeatherCard.js`
  - A `vendor` directory with `normalize.css`, `fonts.css` files and a `fonts` directory stored inside it
  - The `utils` directory, containing the files described in the brief
  - A `README.md` file
  - A `.prettierignore` file, which tells Prettier to ignore `normalize.css`
  - The `.gitignore` file, which tells Git to ignore `node_modules`, `dist`, and `.DS_Store`.
- The project complies with the following code style requirements:
  - camelCase is used for function and variable names.
  - Only nouns are used as variable names.
  - Variable names clearly describe what is stored in them. If the project has several variables with similar data, then those variables have unique but descriptive names.
  - Descriptive names are used for functions which reflect what they do.
  - Function names start with a verb.
  - JS classes and functional components are named using nouns and start with a capital letter.
  - Names must not include inappropriate or unclear abbreviations.
- The `README.md` file contains the following:
  - The project's name
  - A description of the project and its functionality
  - A description of the technologies and techniques used
  - Pictures, GIFs, or screenshots that detail project features (highly recommended)
  - A demo video of your project (highly recommended)
  - A link to GitHub Pages (optional)

- The code is well-formatted using the Prettier.

React

- The markup has been ported to JSX and:
  - is included within `( )`.
  - has been moved to the corresponding components.
- Components:
  - Hooks are not used inside conditional statements.
  - Hooks are called in a component's main function.
  - For class components, the effects are described inside the component lifecycle methods.
- All state variables from the brief have been created and defined within the components specified in the brief.
- The initial state of state variables contains the correct data type.

All of the features listed in the brief have been implemented and are functioning properly:

- Across all components:
  - Contain all the items listed in the brief
  - Accept required props mentioned in the brief
- The `App` component
  - Includes `Header`, `Main`, `Footer`, `ModalWithForm`, and `ItemModal` components.
  - Makes an API request for the weather data (only once — on mounting).
  - Saves default clothing items in the state.
- The `Header` component calculates the current date.
- The `Main` component includes `WeatherCard`, `ItemCard` components.
- The `WeatherCard` component displays temperature in Fahrenheit.
- The `ItemCard` component renders an image and title of a clothing item.

- The `ModalWithForm` component renders a modal with a form.
- The `ItemModal` component renders the item image and title.
- The utils files contain
  - Default clothing items
  - Coordinates object with `latitude` and `longitude` fields
  - Weather API fetch and filter methods
  - The API key for the Weather API
  - Logic for defining temperature