**Student PID:** dhruvaddanki
**Assignment:** Homework 1 - Temperature Converter (React + Vite)

---

**1) Model–View–Controller (MVC) Explanation**

This React application is organized to follow the Model–View–Controller pattern as required.

Model (State / Data)

The **Model** is implemented in **App.jsx** using React state (useState). All application state is stored at the top level in App.jsx, including:

- selectorScale — the selected input temperature scale ("C" or "F")
- selectorTemp — the user's input temperature (stored as a string)
- displayScale — the selected output temperature scale ("C" or "F")
- displayTemp — the converted output temperature (stored as a string)

Keeping the full state in App.jsx ensures a single source of truth and makes the rest of the components purely presentational.

View (User Interface)

The **View** is composed of the four helper components, each responsible for rendering UI and receiving values via props:

- **Header.jsx**: renders the page title text using props and displays it with an <h1> element.
- **Selector.jsx**: renders the input scale dropdown and editable input temperature field (controlled inputs via props).
- **Display.jsx**: renders the output scale dropdown and the read-only converted temperature field (controlled via props).
- **Convert.jsx**: renders the Convert button.

These components do not own conversion logic or app-wide state; they display what App.jsx provides and call callbacks passed down from App.jsx.

Controller (Logic / Event Handling)

The **Controller** logic is implemented in **App.jsx**. App.jsx contains:

- Event handlers to update state when the user changes:
    1. Selector scale dropdown
    2. Selector input temperature
    3. Display scale dropdown

- A handleConvert() function (triggered only by the Convert button) that:
    1. Reads the input temperature and both scales from state
    2. Validates the input (blank/whitespace/non-numeric → invalid)
    3. Performs the appropriate conversion based on selected scales
    4. Updates the output temperature state
    5. If invalid, sets the output field to exactly: **Error!**

This structure cleanly separates the UI (View), the state (Model), and the logic (Controller), while still using idiomatic React.

---

**2) Temperature Conversion & Error Handling**

- Conversion occurs **only when the Convert button is pressed** (no auto-convert during typing).
- The Selector input is validated before conversion:
    - If the input is blank, whitespace-only, or not a finite number, the Display output becomes **Error!**.
- Conversions implemented:
    - Celsius → Fahrenheit: F = C * 9/5 + 32
    - Fahrenheit → Celsius: C = (F - 32) * 5/9
    - Same scale: output equals input (still requires numeric validation)

---

**3) Styling Requirements**

The CSS follows the assignment requirements:

- body background color is **orange**
- Header uses <h1> and text color is **maroon**
- Button uses **maroon** background, **white** text, and **bold** font weight
- Dropdown menus (select) use **maroon** background and **white** text
- Text inputs (input) are **right-aligned**