Framework vs Vanilla choice

Sportradar Coding Academy F…
PDF

I have a task to do to get an internship position, the submission is due in two weeks and i am not quite sure which framework I should use or if I should go with plain vanilla html, css + js

Perfect — I've read through the **Sportradar Coding Academy FE 2026** exercise.

Here's a breakdown to help you decide **which framework or stack** to use and how to approach it:

🧩 What They're Looking For

The exercise evaluates:

- Frontend fundamentals (DOM manipulation, state management)
- Responsiveness
- Clean, understandable code
- Basic UX/navigation
- Git/version control use
- Responsible AI use (they'll ask about it in interviews)

They explicitly say:

> *"You may use plain HTML, CSS, and JavaScript, or any frontend framework/library you are comfortable with (e.g., React, Vue.js, Angular)."*

So **you have freedom** — but the key is: pick something you can finish confidently in 2 weeks.

⚖️ Framework vs. Vanilla

Option	Pros	Cons	Recommended for
Vanilla HTML, CSS, JS	Lightweight, no setup, shows strong fundamentals	More manual DOM manipulation; managing event data and navigation will be harder	If you’re a beginner and want full control
React	Component-based UI, great for state handling (calendar, event details, adding events)	Requires setup (Vite or CRA), some learning curve if new	✅ **Best balance** — widely used, easy routing (React Router), easy to show competence
Vue	Simpler reactivity, clean syntax	Slightly less common in corporate FE tests	If you’re comfortable with Vue
Angular	Full framework with routing and forms	Heavy and overkill for this scope	Not recommended unless you’re already fluent

---

✅ Recommendation (Most Efficient Choice)

Use **React + Vite + Tailwind CSS**.

Why:

- Quick to set up (`npm create vite@latest` → choose React).
- Tailwind makes responsiveness easy.
- React Router simplifies navigation (Calendar ↔ Add Event ↔ Details).
- Easy to show event state updates dynamically.
- You can host it easily on GitHub Pages or Netlify.

📘 Suggested Project Structure

```
/src
 ├── components/
 │   ├── Calendar.jsx
 │   ├── EventCard.jsx
 │   └── AddEventForm.jsx
 ├── pages/
```

```
│   ├── CalendarPage.jsx
│   ├── EventDetailPage.jsx
│   └── AddEventPage.jsx
├── data/
│   └── events.json  ← (provided)
├── App.jsx
├── main.jsx
├── App.css
```

🧠 Implementation Plan

1. **Week 1**
   - Setup project (Vite + React + Tailwind)
   - Implement the Calendar grid
   - Load events from JSON
   - Show markers on days with events
   - Implement responsive layout
2. **Week 2**
   - Add navigation (React Router)
   - Implement “Add Event” form (update in-memory state)
   - Implement “Event Detail” page
   - Polish UI, test responsiveness
   - Write README + AI_Reflection.txt

🧾 AI_Reflection.txt Tip

They’ll expect you to note things like:

> *“I used ChatGPT to help structure the component layout and for debugging form validation logic, but I wrote the event handling and state management code myself.”*

Would you like me to create a **starter React + Tailwind project structure (with Calendar + routing skeleton)** for you to build upon? It would save you setup time and give you a clean foundation for the assignment.

yes please and include the npm comamnds for iniations

Perfect 👍 — here's a clean **React + Vite + Tailwind** starter setup tailored exactly for your Sportradar Coding Academy assignment, including all **npm commands**, **file structure**, and **sample code skeletons** to get you started fast.

🧱 Project Setup Instructions

1️⃣ Initialize Vite + React project

```
# Create project
npm create vite@latest sportradar-fe --template react

# Move into project
cd sportradar-fe

# Install dependencies
npm install
```

2️⃣ Install Tailwind CSS

```
# Install Tailwind and required plugins
npm install -D tailwindcss postcss autoprefixer

# Initialize Tailwind config
npx tailwindcss init -p
```

Then modify your `` `tailwind.config.js` ``:

```
/** @type {import('tailwindcss').Config} */
export default {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

And replace `src/index.css` with:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

3️⃣ Add React Router

```
npm install react-router-dom
```

📁 Recommended File Structure

```
sportradar-fe/
├── public/
│   └── events.json              # mock data file
├── src/
│   ├── components/
│   │   ├── Calendar.jsx
│   │   ├── EventCard.jsx
│   │   └── AddEventForm.jsx
│   ├── pages/
│   │   ├── CalendarPage.jsx
│   │   ├── EventDetailPage.jsx
│   │   └── AddEventPage.jsx
│   ├── App.jsx
│   ├── main.jsx
│   ├── index.css
│   └── utils/
│       └── dateUtils.js
├── tailwind.config.js
├── package.json
└── README.md
```

🧩 Base Code Skeletons

`src/main.jsx`

```
import React from "react";
import ReactDOM from "react-dom/client";
import { BrowserRouter } from "react-router-dom";
import App from "./App";
import "./index.css";

ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);
```

`src/App.jsx`

```
import { Routes, Route, Link } from "react-router-dom";
import CalendarPage from "./pages/CalendarPage";
import AddEventPage from "./pages/AddEventPage";
import EventDetailPage from "./pages/EventDetailPage";

export default function App() {
  return (
    <div className="min-h-screen bg-gray-50 text-gray-800">
      <nav className="bg-blue-600 text-white p-4 flex justify-between items-
center shadow">
        <h1 className="text-lg font-semibold">🏟️ Sportradar Events</h1>
        <div className="space-x-4">
          <Link to="/" className="hover:underline">Calendar</Link>
          <Link to="/add" className="hover:underline">Add Event</Link>
        </div>
      </nav>

      <main className="p-4">
        <Routes>
          <Route path="/" element={<CalendarPage />} />
          <Route path="/add" element={<AddEventPage />} />
          <Route path="/event/:id" element={<EventDetailPage />} />
        </Routes>
      </main>
    </div>
  );
}
```

`src/pages/CalendarPage.jsx`

```
import { useState, useEffect } from "react";
import Calendar from "../components/Calendar";

export default function CalendarPage() {
  const [events, setEvents] = useState([]);

  useEffect(() => {
    fetch("/events.json")
      .then(res => res.json())
      .then(data => setEvents(data))
      .catch(err => console.error("Error loading events:", err));
  }, []);

  return (
    <div>
      <h2 className="text-xl font-semibold mb-4">Sports Event Calendar</h2>
      <Calendar events={events} />
    </div>
  );
}
```

`src/components/Calendar.jsx`

```
import { Link } from "react-router-dom";

export default function Calendar({ events }) {
  // Placeholder logic — later replace with dynamic month grid
  return (
    <div className="grid grid-cols-7 gap-2">
      {[...Array(30)].map((_, i) => (
        <div
          key={i}
          className="border p-2 h-24 relative bg-white hover:bg-gray-100 rounded"
        >
          <p className="text-sm font-medium">{i + 1}</p>
          {events
            .filter(e => new Date(e.date).getDate() === i + 1)
            .map(e => (
              <Link
                key={e.id}
                to={`/event/${e.id}`}
                className="absolute bottom-1 left-1 text-xs text-blue-600 underline"
              >
                {e.sport}
              </Link>
            ))}
        </div>
      ))}
    </div>
  );
}
```

`src/pages/AddEventPage.jsx`

```
import AddEventForm from "../components/AddEventForm";

export default function AddEventPage() {
  return (
    <div>
      <h2 className="text-xl font-semibold mb-4">Add New Event</h2>
      <AddEventForm />
    </div>
  );
}
```

`src/components/AddEventForm.jsx`

```
import { useState } from "react";

export default function AddEventForm() {
  const [form, setForm] = useState({
    date: "",
    time: "",
    sport: "",
    teams: ""
  });

  const handleChange = (e) => {
    setForm({ ...form, [e.target.name]: e.target.value });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    console.log("New Event:", form);
    alert("Event added (in-memory only)");
  };

  return (
    <form onSubmit={handleSubmit} className="space-y-4 max-w-md">
      {["date", "time", "sport", "teams"].map((field) => (
        <div key={field}>
          <label className="block capitalize mb-1">{field}</label>
          <input
            type={field === "date" || field === "time" ? field : "text"}
            name={field}
            value={form[field]}
            onChange={handleChange}
            required
            className="border p-2 w-full rounded"
          />
        </div>
      ))}
      <button
        type="submit"
        className="bg-blue-600 text-white px-4 py-2 rounded hover:bg-blue-700"
      >
        Add Event
      </button>
    </form>
  );
}
```

`src/pages/EventDetailPage.jsx`

```
import { useParams } from "react-router-dom";

export default function EventDetailPage() {
  const { id } = useParams();
  // You’ll later fetch or find event by ID here
  return (
    <div>
      <h2 className="text-xl font-semibold mb-4">Event Details</h2>
      <p>Event ID: {id}</p>
      {/* Add full event details here */}
    </div>
  );
}
```

🚀 Run the App

```
npm run dev
```

Then open the provided localhost link — usually:

```
http://localhost:5173
```

Would you like me to include a **sample `events.json`** (with 2–3 mock events matching the example in the PDF) so you can drop it straight into **`/public`**?

create me a pdf of this conversation

Your PDF guide has been created successfully. You can download it below:

📄 Download Sportradar_FE_Starter_Guide.pdf ‹/›