Rella Project: Frontend Responsibilities and File Structure Guide

Role: Frontend Lead (assigned partner) Collaborator: Backend Lead (you)

🚀Project Context

This document outlines the frontend responsibilities for the Rella fitness tracker app. As the frontend lead, you will be responsible for building the user interface in React, handling user interactions, connecting frontend pages to backend API endpoints, and rendering data dynamically (especially from the workouts and login features).

This work will live inside the `/frontend/` directory of the fullstack app repo.

Frontend Folder Structure

```
/frontend
├── public/
├── src/
│   ├── components/         # Reusable UI elements (buttons, inputs, cards)
│   ├── pages/              # Main pages rendered via routing
│   │   ├── LoginPage.jsx
│   │   ├── Dashboard.jsx
│   │   ├── WorkoutCard.jsx
│   │   └── TrainerDashboard.jsx
│   ├── api/                # Utility functions for calling backend APIs
│   │   ├── auth.js
│   │   ├── workouts.js
│   │   └── users.js
│   ├── contexts/           # (Optional) Context for auth / global user state
│   ├── App.jsx             # Main routing and layout logic
│   ├── index.js            # Entry point
│   └── styles/             # CSS or Tailwind styles
```

🏫Core Responsibilities

1. Authentication (LoginPage.jsx)

- Build login form UI with inputs for username, password, and role selector.
- Connect form to `POST /api/auth/login`.
- On success, store the JWT and user info (e.g. in context or localStorage).
- Redirect to appropriate dashboard based on user role (athlete or trainer).

2. Athlete Dashboard (Dashboard.jsx)

- On page load, call `GET /api/workouts?date=<selected>`.
- Render a calendar view that shows which days have workouts.
- When a day is clicked, route to that day's `WorkoutCard`.

3. Workout Card View (WorkoutCard.jsx)

- Fetch workout details from backend based on `workoutId` in URL.
- Display: Date, Muscle Groups, Warm-ups, Working Sets, Exercises.
- Add UI to mark workout as completed (e.g. call `PATCH /api/workouts/:id/complete`).

4. Trainer Dashboard (TrainerDashboard.jsx)

- Call `GET /api/workouts?trainer=true` or similar.
- Render a list of workouts assigned to all athletes.
- Optionally, allow trainer to create a workout (bonus feature).

Shared Responsibilities

You (backend lead) may help with: - Routing structure setup in `App.jsx` - Auth context boilerplate (if needed) - Styling / layout if time allows - Debugging API call integration

---

🚡Backend Integration Points

Every page must use functions from `/frontend/src/api/`: - `auth.js` → login() - `workouts.js` → fetchWorkoutsByDate(), fetchWorkoutById(), markWorkoutComplete() - `users.js` → fetchAthletes(), etc.

Example API call from frontend:

```
const res = await fetch('/api/workouts?date=2025-11-12', {
  headers: {
    Authorization: `Bearer ${token}`,
  },
});
const workouts = await res.json();
```

🚚Suggested Order of Tasks

1. Build LoginPage UI + auth call logic
2. Build Dashboard with hardcoded calendar view
3. Connect Dashboard to workout API

4. Create WorkoutCard with backend data
5. Build TrainerDashboard view

Let me know if you'd like this exported as a PDF for sharing in the repo!