Freelancer Job Posting

□ JOB TITLE

Full-Stack Web Developer – Bilingual Personal Trainer Portfolio Website with Admin Dashboard (React + Node.js + PostgreSQL)

🏷 JOB DESCRIPTION

We are looking for an experienced **Full-Stack Web Developer** to build a **premium, bilingual (English & Arabic) portfolio/landing website** for a personal trainer, PE teacher, and medical osteopathy specialist. The website must include a **fully-featured admin dashboard** for managing content, contacts, reviews, and client transformations.

□ Project Overview

Build a complete, production-ready website for a fitness professional named **Eslam Mohamed** that serves as both a **public-facing portfolio/marketing site** and a **private admin panel** for business management. The site must support **English and Arabic (RTL)** with a language switcher, and must be deployed to **Railway** with a **Neon PostgreSQL** database.

🏗 Scope of Work

A. Public-Facing Landing Page (Single-Page Layout)

The landing page must be a **modern, premium-looking, single-page scrollable design** with the following sections (in order):

1. **Header / Navigation Bar**
    - Sticky top navigation
    - Logo / Brand name
    - Navigation links (smooth scroll to sections)
    - Language switch button (EN ↔ AR)
    - "Book a Session" CTA button
2. **Hero Section**
    - Full-width hero with bold headline, subtitle, and description text
    - "Book a Session" and "WhatsApp" CTA buttons
    - Tagline: "Online & in-person programs — No excuses. Just results."

- Animated scroll-down indicator

3. **Training Section**
    - Showcase real training session images/videos
    - Subtitle: "Real moments from intense training sessions."
4. **About Section ("Meet Your Coach")**
    - Two bio paragraphs about the trainer
    - Credential badges (Certified Trainer, PE Degree, Osteopathy Specialist, Youth Coach)
    - Side card: "Work with Eslam" listing 5 bullet benefits
    - "Start Your Journey" CTA button
5. **Services Section ("Services for Every Goal")**
    - 4 service cards, each with:
        - Title, description, and a "What's Included" list (4 items each)
    - Services:
        - Personal Training
        - PE & Kids Training
        - Online Coaching
        - Injury Rehabilitation (Medical Osteopathy)
6. **Certifications Section ("Certified Excellence")**
    - Tab-based filtering: All / Master's / Bachelor's / Doctor / Soon
    - Featured academic credentials (Master's & Bachelor's degrees with institution details)
    - Grid of 25+ certification cards with title & institution
    - "In Progress" section for Doctorate (Coming Soon badge)
7. **Experience Section ("Battle-Tested Experience")**
    - Timeline-style layout with 3 positions:
        - Medical Osteopathy at Egypt Stars FC (2021 – Present)
        - PE Teacher at International Academy of Cairo (2019 – 2021)
        - Youth Athletic Trainer – Private Practice (2017 – 2019)
    - Each entry has 3 bullet points
8. **Testimonials Section ("What Warriors Say")**
    - Display client reviews fetched from the database (dynamic)
    - Star ratings (1–5)

- Client name, title/role, and review text
- "View All Testimonials" link

9. **Footer Section**
    - Brand description
    - Quick links (Services, About, Certifications, Testimonials)
    - "Get in Touch" with contact information
    - Social media links
    - Copyright notice
10. **Contact Modal (Popup Form)**
    - Triggered by "Book a Session" buttons throughout the page
    - Form fields:
        - Full Name (required)
        - Email (required)
        - Phone Number (optional)
        - Service Interested In (dropdown: Personal Training, PE Teaching & Kids Training, Online Coaching, Injury Rehabilitation)
        - Message (optional, textarea)
    - Form submits to backend API and stores in database
    - Success/error toast notifications
11. **Floating Elements**
    - WhatsApp floating button (mobile only, bottom-right) linking to `wa.me/96894408664`
    - Back-to-top button (bottom-left, appears after scrolling 400px)
    - Scroll progress bar at top of page

B. Admin Dashboard (Protected Area)

A secure admin panel accessible via `/admin/login` with **JWT-based authentication** (access + refresh tokens). Role-based access control with 3 roles: `super_admin`, `admin`, `viewer`.

Admin Pages:

1. **Admin Login Page** (`/admin/login`)
    - Email + password login form
    - JWT token pair returned on success (access token + refresh token)
    - Auto-redirect to dashboard on valid session

2. **Dashboard** (`/admin/dashboard`)
    - Overview of contact submissions (paginated table)
    - Search & filter contacts
    - Mark contacts as "contacted" (toggle)
    - Delete contacts
    - View count / totals
    - Accessible by: super_admin, admin, viewer
3. **Transformations Management** (`/admin/transformations`)
    - CRUD for client transformation entries (before/after photos)
    - Fields: client name, before image URL, after image URL, description, time period, active/inactive toggle, sort order
    - Image upload endpoint (stores in `/uploads/` directory)
    - Accessible by: super_admin, admin
4. **Reviews Management** (`/admin/reviews`)
    - CRUD for client reviews/testimonials
    - Fields: client name, rating (1–5 stars), review text, client avatar URL, client title, featured flag, active/inactive toggle, sort order
    - Reviews marked "active" are shown on the public landing page
    - Accessible by: super_admin, admin
5. **Audit Logs** (`/admin/audit-logs`)
    - Read-only paginated table of all admin actions
    - Logs: login, logout, create/update/delete operations, image uploads
    - Includes admin ID, action type, target resource ID, IP address, timestamp
    - Accessible by: super_admin only

C. Backend API (RESTful)

Build a **Node.js/Express** REST API with the following endpoints:

Method	Endpoint	Auth	Description
`GET`	`/api/health`	Public	Health check (database connectivity)
`POST`	`/api/contact`	Public	Submit contact form
`POST`	`/api/admin/login`	Public	Admin login (returns JWT tokens)

Method	Endpoint	Auth	Description
POST	/api/admin/refresh	Public	Refresh access token
POST	/api/admin/logout	Auth	Logout (invalidate refresh token)
GET	/api/contacts	Auth	Get contacts (paginated)
PATCH	/api/contacts/:id	Auth	Update contact status (mark contacted)
DELETE	/api/contacts/:id	Auth	Delete contact
GET	/api/transformations	Public	Get active transformations
GET	/api/admin/transformations	Auth	Get all transformations (incl. inactive)
POST	/api/admin/transformations	Auth	Create transformation
PATCH	/api/admin/transformations/:id	Auth	Update transformation
DELETE	/api/admin/transformations/:id	Auth	Delete transformation
GET	/api/reviews	Public	Get active reviews
GET	/api/admin/reviews	Auth	Get all reviews (incl. inactive)
POST	/api/admin/reviews	Auth	Create review
PATCH	/api/admin/reviews/:id	Auth	Update review
DELETE	/api/admin/reviews/:id	Auth	Delete review
GET	/api/admin/audit-logs	Super Admin	Get audit logs (paginated)
POST	/api/admin/upload-image	Auth	Upload image file

D. Database Schema (PostgreSQL)

Design and implement the following tables using **Drizzle ORM**:

1. **`users`** – id, username, password

2. **`admins`** – id, email, password_hash, role (super_admin/admin/viewer), created_at
3. **`admin_audit_logs`** – id, admin_id (FK → admins), action, target_id, ip_address, created_at
4. **`contact_submissions`** – id, name, email, phone, service, message, contacted (boolean), created_at
5. **`clients`** – id, full_name, email, phone, gender, age, notes, created_at, updated_at
6. **`transformations`** – id, client_name, before_image_url, after_image_url, description, time_period, is_active, sort_order, created_at, updated_at
7. **`bookings`** – id, client_id (FK → clients), trainer_id (FK → admins), start_at, end_at, status (pending/approved/cancelled/completed), notes, created_at, updated_at
8. **`payments`** – id, booking_id (FK → bookings), amount, currency, status (unpaid/paid/refunded), provider, external_id, metadata (JSONB), created_at, updated_at
9. **`reviews`** – id, client_name, rating, review_text, client_avatar, client_title, featured, is_active, sort_order, created_at, updated_at

All IDs use UUID (`gen_random_uuid()`).

E. Internationalization (i18n)

- Full **bilingual support**: English (LTR) and Arabic (RTL)
- Custom i18n system using React Context (no external i18n library)
- Language toggle in header; persists selection
- All landing page text must be translated: hero, about, services, certifications, experience, testimonials, footer, contact form, stats, error/success messages
- Arabic text uses **Cairo** Google Font
- Proper RTL layout support (CSS `direction: rtl`, logical properties like `start`/`end`)

F. Design Requirements

- **Color palette**: Dark theme with orange accent (`#FF6A00` to `#FF4500` gradient)
- **Typography**: Google Fonts – primary: Inter/DM Sans, Arabic: Cairo
- **UI Library**: Radix UI primitives + shadcn/ui components
- **Animations**: Framer Motion for scroll animations and micro-interactions
- **Responsive**: Fully responsive across mobile, tablet, and desktop
- **Premium feel**: Glassmorphism effects, gradient accents, smooth transitions, hover effects
- **Scroll progress bar** at top of page

- **WhatsApp floating CTA** on mobile

G. Security Requirements

- JWT-based authentication with access + refresh token rotation
- Password hashing with bcrypt
- Role-based access control middleware
- Input validation with Zod on all endpoints
- Helmet.js for HTTP security headers
- Express rate limiting
- IP allowlist middleware for admin routes
- Audit logging of all admin actions with IP tracking

🛠 Required Tech Stack

Layer	Technology
Frontend	React 18, TypeScript, Vite 5
Routing	Wouter
State/Data	TanStack React Query
UI Components	Radix UI + shadcn/ui
Styling	Tailwind CSS 3 + tailwindcss-animate
Animations	Framer Motion
Forms	React Hook Form + Zod resolver
Charts	Recharts (admin dashboard)
Backend	Node.js, Express 4, TypeScript
ORM	Drizzle ORM + drizzle-zod
Database	PostgreSQL (Neon serverless)
Auth	JWT (jsonwebtoken), bcrypt/bcryptjs
File Upload	Multer
Deployment	Railway (Nixpacks builder)
Build	esbuild (server), Vite (client)

📦 Deliverables

1. Complete source code in a single repository (monorepo structure: `client/`, `server/`, `shared/`)

2. Working PostgreSQL database schema with Drizzle migrations
3. Fully functional bilingual landing page (EN + AR with RTL support)
4. Fully functional admin dashboard with JWT auth and role-based access
5. All 21+ REST API endpoints working and tested
6. Image upload system for transformations
7. Deployed and running on Railway with Neon PostgreSQL
8. Environment variable documentation (`.env.example`)
9. Railway deployment configuration (`railway.json`)
10. Seed script for default admin credentials

□ Project Structure

```
/
├── client/
│   ├── index.html
│   ├── public/
│   └── src/
│       ├── App.tsx
│       ├── main.tsx
│       ├── index.css
│       ├── components/
│       │   ├── Header.tsx
│       │   ├── LanguageSwitch.tsx
│       │   ├── landing/          (11 section components)
│       │   └── ui/               (shadcn/ui components)
│       ├── hooks/
│       ├── i18n/
│       │   ├── LanguageContext.tsx
│       │   ├── en.ts
│       │   ├── ar.ts
│       │   ├── index.ts
│       │   └── types.ts
│       ├── lib/
│       └── pages/
│           ├── LandingPage.tsx
│           ├── AdminLogin.tsx
│           ├── AdminDashboard.tsx
│           ├── AdminTransformations.tsx
│           ├── AdminReviews.tsx
│           ├── AdminAuditLogs.tsx
│           └── not-found.tsx
├── server/
│   ├── index.ts
│   ├── routes.ts
│   ├── storage.ts
│   ├── seed.ts
│   ├── static.ts
│   ├── vite.ts
│   └── middleware/
│       ├── auth.ts
│       ├── ipAllowlist.ts
│       └── requireAdmin.ts
├── shared/
```

```
│   └── schema.ts
├── package.json
├── tsconfig.json
├── vite.config.ts
├── tailwind.config.ts
├── drizzle.config.ts
├── railway.json
└── .env
```

🎓 Required Skills

- React 18 + TypeScript (advanced)
- Node.js / Express (intermediate-advanced)
- PostgreSQL + Drizzle ORM
- JWT authentication & security best practices
- Tailwind CSS + Radix UI / shadcn/ui
- Framer Motion animations
- Bilingual / RTL web development (Arabic)
- Railway / cloud deployment
- Responsive web design
- REST API design

□ Estimated Timeline

3-5 days for a single experienced full-stack developer.
