Event Platform - Project Documentation

1. Project Overview

This repository hosts a full-stack Event Management Platform designed to handle the complete lifecycle of events—from creation and registration to participation and evaluation. It supports various event types, including quizzes and coding competitions.

Key Capabilities

- **User Management:** Authentication via Supabase (Sign up, Login, Password Reset).
- **Role-Based Access:** Distinct portals for **Participants** (browse, register, compete) and **Admins** (create events, manage users, view analytics).
- **Event Logic:** Support for standard events, quizzes with automated scoring, and coding problem scopes.
- **Payments:** Integrated with Razorpay for paid event registrations.
- **Content Management:** Gallery management, form builders, and submission handling.

2. Technology Stack

Frontend

- **Framework:** Next.js 14+ (App Router structure)
- **Language:** JavaScript / JSX
- **Styling:** Tailwind CSS
- **UI Components:** Shadcn/UI (Radix UI primitives) + Lucide React icons
- **State Management:** React Context (AuthContext) + Hooks

Backend & Database

- **BaaS (Backend as a Service):** Supabase
- **Database:** PostgreSQL
- **Authentication:** Supabase Auth
- **Storage:** Supabase Storage (for event banners, gallery images)
- **Server Logic:** Next.js API Routes (app/api/) + Supabase Edge Functions (e.g., send-email)

External Integrations

- **Payments:** Razorpay
- **Email:** SMTP / Supabase Edge Functions

3. Project Structure

The project follows the standard Next.js App Router directory structure:

/app (Core Application)

- **(main)**: Main layout group containing public and authenticated pages.
  - **admin/**: Admin dashboard routes (Event creation, analytics, gallery).
  - **events/**: Public event listing and specific event details.
  - **profile/**: User profile settings.
  - **auth/**: Authentication pages.
- **api/**: Server-side API endpoints (Database interactions, Payment verification).

/components (UI Building Blocks)

- **ui/**: Reusable atomic components (Buttons, Inputs, Cards) powered by Shadcn.
- **admin/**: Admin-specific components (e.g., EventGalleryManager).
- **Functional Components:** EventCard.js, EventForm.js, FormBuilder.js, Navbar.js.

/lib (Utilities)

- **supabase/**: Client and Server initialization for Supabase.
- **utils.js**: Helper functions (class merging, formatting).
- **email.js**: Email handling logic.

4. Database Schema (Inferred)

The system relies on several core tables in PostgreSQL (Supabase):

- **profiles**: User details linked to auth.users.
- **events**: Stores event metadata (title, date, description, price).
- **registrations**: Links profiles to events (payment status, registration date).
- **quizzes & quiz_attempts**: Stores questions and user attempts/scores.
- **problems & submissions**: Coding or text-based problem definitions and user answers.
- **gallery**: Stores references to event images.

5. Key Features Breakdown

A. Authentication & Security

- Uses AuthContext to manage session state globally.
- ProtectedRoute.js wrapper ensures sensitive routes (like Admin pages) are accessible only to authorized personnel.
- Supabase Row Level Security (RLS) policies likely protect the database layer.

B. Event Administration

- **Dashboard:** View participant counts and revenue.
- **Builders:** Drag-and-drop style builders for Registration Forms and Quizzes.
- **Management:** Edit event details, manage gallery images, and view submissions.

C. Participant Experience

- **Discovery:** Grid view of events with filtering (implied).
- **Registration:** Dynamic forms based on event requirements + Razorpay checkout.
- **Participation:** Real-time quiz interface and problem submission forms.