Application Structure

Pages:

- `Index.tsx` : The main landing page, displaying featured categories.
- `CategoryPage.tsx` : Displays details for a specific category.
- `ContactPage.tsx` : Contact form page.
- `CreateRankingPage.tsx` : Page for creating new rankings.
- `FeedPage.tsx` : Displays a feed of user activities.
- `GdprPage.tsx` : GDPR information page.
- `NotFound.tsx` : 404 error page.
- `PrivacyPolicyPage.tsx` : Privacy policy page.
- `PublicProfilePage.tsx` : Public user profile page.
- `QuizPage.tsx` : Quiz functionality page.
- `UserRankingPage.tsx` : Displays a specific user's ranking.
- `UserProfilePage.tsx` : User's private profile page.
- `admin/` : Directory containing admin-related pages.

Core Files:

- `main.tsx` : Entry point of the React application, responsible for rendering the `App` component and setting up `HelmetProvider` for managing document head tags, including Content Security Policy (CSP).
- `App.tsx` : Main application component, likely handles routing and global layout.
- `index.css` : Global CSS styles.
- `App.css` : Application-specific CSS styles.

Components:

- `src/components/` : Contains reusable React components.

Data:

- `src/data/` : Likely contains static data or mock data.

Contexts:

- `src/contexts/` : Contains React Context API providers for global state management.

Hooks:

- `src/hooks/` : Contains custom React hooks for reusable logic.

Lib:

- `src/lib/` : Contains utility functions and third-party library initializations (e.g., `supabase.ts`).

Types:

- `src/types/` : Contains TypeScript type definitions.

Integrations:

- `src/integrations/` : Contains code related to third-party integrations.

Functionalities:

Database (Supabase):

User Authentication and Authorization:

- **Description**: Handles user login, registration, and session management. It also controls access to certain pages based on user roles (e.g., admin pages).
- **Files Involved**:
- `src/contexts/AuthContext.tsx` : Manages authentication state and provides authentication-related functions.
- `src/lib/supabase.ts` : Interacts with Supabase for authentication.
- Pages that require authentication (e.g., `UserProfilePage.tsx` , `CreateRankingPage.tsx` , admin pages).
- **Connection**: `AuthContext` provides the authentication status to various components and pages, enabling or restricting access to features.

Ranking Creation and Management:

- **Description**: Allows users to create new rankings for categories and manage their existing rankings.
- **Files Involved**:
- `src/pages/CreateRankingPage.tsx` : User interface for creating new rankings.

- `src/pages/UserRankingPage.tsx`: Displays a specific user's ranking.
- `src/pages/UserProfilePage.tsx`: Likely lists user's created rankings.
- `src/components/CategoryCard.tsx`: Displays categories, potentially leading to ranking creation.
- `src/lib/supabase.ts`: Interacts with Supabase for ranking data.
- **Connection**: Users select a category (via `CategoryCard` on `Index.tsx` or `CategoryPage.tsx`), then navigate to `CreateRankingPage.tsx` to create a ranking. The created rankings are then viewable on `UserRankingPage.tsx` and `UserProfilePage.tsx`.

Category Browsing and Display:

- **Description**: Enables users to browse different categories and view details within each category.
- **Files Involved**:
- `src/pages/Index.tsx`: Displays featured categories.
- `src/pages/CategoryPage.tsx`: Displays details and rankings within a specific category.
- `src/components/CategoryCard.tsx`: Reusable component for displaying category information.
- `src/lib/supabase.ts`: Fetches category data from Supabase.
- **Connection**: The `Index.tsx` page serves as the entry point to explore categories. Clicking on a `CategoryCard` navigates to `CategoryPage.tsx` to see more details and related rankings.

User Profile Management:

- **Description**: Allows users to view and manage their own profiles, and view public profiles of other users.
- **Files Involved**:
- `src/pages/UserProfilePage.tsx`: User's private profile page.
- `src/pages/PublicProfilePage.tsx`: Displays public information of other users.
- `src/lib/supabase.ts`: Manages user data in Supabase.
- **Connection**: `UserProfilePage.tsx` is accessible to logged-in users to manage their own data, while `PublicProfilePage.tsx` allows viewing other users' public profiles.

Feed Display:

- **Description**: Shows a feed of recent activities within the application, such as new rankings or comments.
- **Files Involved**:
- `src/pages/FeedPage.tsx`: Displays the activity feed.
- `src/lib/supabase.ts`: Fetches feed data from Supabase.
- **Connection**: Provides users with an overview of recent activities and engagements within the platform.

Quiz Functionality:

- **Description**: Provides interactive quizzes for users.
- **Files Involved**:
- `src/pages/QuizPage.tsx`: User interface for taking quizzes.
- `src/pages/admin/CreateQuizPage.tsx`: Admin interface for creating new quizzes.
- `src/lib/supabase.ts`: Manages quiz data in Supabase.
- **Connection**: Admins create quizzes via `CreateQuizPage.tsx`, and users can then take these quizzes on `QuizPage.tsx`.

Contact and Legal Pages:

- **Description**: Provides static pages for contact information, GDPR details, and privacy policy.
- **Files Involved**:
- `src/pages/ContactPage.tsx`: Contact form.
- `src/pages/GdprPage.tsx`: GDPR information.
- `src/pages/PrivacyPolicyPage.tsx`: Privacy policy.
- **Connection**: These are standalone informational pages, primarily for legal compliance and user support.

Admin Management:

- **Description**: Provides administrative interfaces for managing users, comments, and quizzes.
- **Files Involved**:
- `src/pages/admin/UserManagementPage.tsx`: Admin interface for managing users.
- `src/pages/admin/CommentManagementPage.tsx`: Admin interface for managing comments.

- `src/pages/admin/CreateQuizPage.tsx`: Admin interface for creating quizzes.
- `src/lib/supabase.ts`: Interacts with Supabase for admin-related data.
- **Connection**: These pages are restricted to admin users and allow for moderation and content management within the application.

Global UI Components and Utilities:

- **Description**: Reusable UI components and utility functions used across the application.
- **Files Involved**:
- `src/components/Navbar.tsx`: Navigation bar.
- `src/components/Footer.tsx`: Application footer.
- `src/components/ui/sonner.tsx`: Toast notification system.
- `src/lib/supabase.ts`: Supabase client initialization and configuration.
- `src/hooks/`: Custom React hooks.
- `src/types/`: TypeScript type definitions.
- **Connection**: These files provide foundational elements and shared logic, ensuring consistency and reusability throughout the application.

Database (Supabase):

Tables:

`profiles`

- **Purpose**: Stores public user profile information.
- **Columns**:
- `id` (uuid): Primary key, references `auth.users.id`. Unique identifier for the user profile.
- `full_name` (TEXT): User's full name.
- `avatar_url` (TEXT): URL to the user's avatar image.
- `updated_at` (TIMESTAMPTZ): Timestamp of the last update to the profile.
- `country` (TEXT): User's country.
- `favorite_sports` (TEXT[]): Array of user's favorite sports.
- **Interactions with Pages**:
- `UserProfilePage.tsx`: Users can view and update their own profiles.
- `PublicProfilePage.tsx`: Displays public profile information of other users.
- **Usage**: Used to display user-specific information across the application.

`user_roles`

- **Purpose**: Assigns roles (e.g., 'admin', 'user') to users for authorization purposes.
- **Columns**:
- `id` (uuid): Primary key, unique identifier for the role assignment.
- `user_id` (uuid): Foreign key, references `auth.users.id`. The ID of the user to whom the role is assigned.
- `role` (public.app_role ENUM): The assigned role, either 'admin' or 'user'.
- **Interactions with Pages**:
- `UserManagementPage.tsx` (admin): Admins can manage user roles.
- All pages with restricted access (e.g., admin pages) use this table for authorization checks.
- **Usage**: Controls access to specific features and pages based on the user's assigned role.

`categories`

- **Purpose**: Stores information about different categories for ranking, including parent-child relationships.
- **Columns**:
- `id` (UUID): Primary key, unique identifier for the category.
- `name` (TEXT): Name of the category (e.g., "GOAT Footballer").
- `description` (TEXT): Description of the category.
- `parent_id` (UUID): Foreign key, references `public.categories.id`. Used to establish hierarchical relationships between categories (e.g., "GOAT Footballer" is a subcategory of "GOAT").
- `created_at` (TIMESTAMP WITH TIME ZONE): Timestamp of category creation.
- `image_url` (TEXT): URL to the category's image.
- **Interactions with Pages**:
- `Index.tsx` : Displays featured categories.
- `CategoryPage.tsx` : Displays details and subcategories for a specific category.
- `CreateRankingPage.tsx` : Users select categories to create rankings.
- **Usage**: Organizes content into logical groups, allowing users to explore and create rankings within specific areas of interest.

`category_likes`

- **Purpose**: Records user likes for specific categories.
- **Columns**:
- `id` (BIGINT): Primary key.

- `user_id` (UUID): Foreign key, references `auth.users.id`. The ID of the user who liked the category.
- `category_id` (UUID): Foreign key, references `public.categories.id`. The ID of the liked category.
- `created_at` (TIMESTAMP WITH TIME ZONE): Timestamp of when the like was recorded.
- **Interactions with Pages**:
- `UserProfilePage.tsx`: Likely displays categories liked by the user.
- **Usage**: Tracks user engagement with categories, potentially influencing content recommendations or popular category listings.

`category_comments`

- **Purpose**: Stores comments made on categories, supporting threaded discussions.
- **Columns**:
- `id` (UUID): Primary key, unique identifier for the comment.
- `user_id` (UUID): Foreign key, references `public.profiles.id`. The ID of the user who made the comment.
- `category_id` (UUID): Foreign key, references `public.categories.id`. The ID of the category the comment belongs to.
- `parent_comment_id` (UUID): Foreign key, references `public.category_comments.id`. Used for replies to establish a thread.
- `comment` (TEXT): The content of the comment.
- `created_at` (TIMESTAMPTZ): Timestamp of when the comment was created.
- **Interactions with Pages**:
- `CategoryPage.tsx`: Displays comments related to a specific category.
- `CommentManagementPage.tsx` (admin): Admins can manage comments.
- **Usage**: Facilitates user interaction and discussion within categories.

`notifications`

- **Purpose**: Stores notifications for users, such as new comment replies or new categories.
- **Columns**:
- `id` (UUID): Primary key, unique identifier for the notification.
- `user_id` (UUID): Foreign key, references `auth.users.id`. The ID of the user receiving the notification.
- `type` (public.notification_type ENUM): The type of notification (e.g., `new_comment_reply`, `new_category`).
- `data` (JSONB): JSON data containing additional information relevant to the notification (e.g., category ID, comment ID, replying user name).

- `is_read` (BOOLEAN): Indicates whether the notification has been read by the user.
- `created_at` (TIMESTAMPTZ): Timestamp of when the notification was created.
- **Interactions with Pages**:
- Likely a notification display component in the UI (e.g., in the Navbar).
- **Usage**: Informs users about relevant activities and updates within the application.

`friendships`

- **Purpose**: Manages friend requests and connections between users.
- **Columns**:
- `id` (UUID): Primary key, unique identifier for the friendship.
- `requester_id` (UUID): Foreign key, references `public.profiles.id`. The ID of the user who sent the friend request.
- `receiver_id` (UUID): Foreign key, references `public.profiles.id`. The ID of the user who received the friend request.
- `status` (public.friendship_status ENUM): Current status of the friendship (`pending`, `accepted`, `declined`, `blocked`).
- `created_at` (TIMESTAMPTZ): Timestamp of when the friendship record was created.
- `updated_at` (TIMESTAMPTZ): Timestamp of the last update to the friendship record.
- **Interactions with Pages**:
- `UserProfilePage.tsx`: Users can manage their friend requests and friends list.
- `PublicProfilePage.tsx`: Users can send friend requests to other users.
- **Usage**: Enables social features within the application, allowing users to connect with each other.

`feed_items`

- **Purpose**: Stores items for the activity feed, such as new user registrations, new comments, and accepted friendships.
- **Columns**:
- `id` (UUID): Primary key, unique identifier for the feed item.
- `created_at` (TIMESTAMPTZ): Timestamp of when the feed item was created.
- `type` (public.feed_item_type ENUM): The type of activity (`new_user`, `new_comment`, `accepted_friendship`, `new_ranking`).
- `data` (JSONB): JSON data containing details specific to the feed item type (e.g., user ID, comment text, involved profiles, and for `new_ranking` type, the ranked athletes data and ranking description).
- **Interactions with Pages**:

- `FeedPage.tsx` : Displays the activity feed to users.
- **Usage**: Provides a dynamic overview of recent activities within the application, enhancing user engagement.

`user_rankings`

- **Purpose**: Stores user-created rankings for specific categories.
- **Columns**:
- `id` (UUID): Primary key, unique identifier for the user ranking.
- `user_id` (UUID): Foreign key, references `auth.users.id` . The ID of the user who created the ranking.
- `category_id` (UUID): Foreign key, references `public.categories.id` . The ID of the category this ranking belongs to.
- `title` (TEXT): Title of the user-created ranking.
- `description` (TEXT): Description of the user-created ranking.
- `created_at` (TIMESTAMPTZ): Timestamp of when the ranking was created.
- **Interactions with Pages**:
- `CreateRankingPage.tsx` : Users create new rankings.
- `UserRankingPage.tsx` : Displays a specific user-created ranking.
- `UserProfilePage.tsx` : Likely lists user's created rankings.
- **Usage**: Allows users to create and share their own ordered lists of athletes within a category.

`ranking_athletes`

- **Purpose**: Stores the athletes included in a user-created ranking, along with their position and points.
- **Columns**:
- `id` (BIGINT): Primary key.
- `ranking_id` (UUID): Foreign key, references `public.user_rankings.id` . The ID of the ranking this athlete belongs to.
- `athlete_id` (TEXT): Identifier for the athlete (e.g., name or external ID).
- `position` (INT): The athlete's position in the ranking.
- `points` (INT): Points assigned to the athlete in the ranking.
- `created_at` (TIMESTAMPTZ): Timestamp of when the athlete was added to the ranking.
- **Interactions with Pages**:
- `CreateRankingPage.tsx` : Users add athletes to their rankings.
- `UserRankingPage.tsx` : Displays the athletes within a specific ranking.

- **Usage**: Details the specific athletes and their order/score within a user's ranking.

- **Constraints**:
- `user_rankings_user_id_category_id_key`: Ensures a user can only have one ranking per category.

`quizzes`

- **Purpose**: Stores information about quizzes available in the application.
- **Columns**:
- `id` (UUID): Primary key, unique identifier for the quiz.
- `title` (TEXT): Title of the quiz.
- `topic` (TEXT): Topic or subject of the quiz.
- `active_date` (DATE): The date on which the quiz becomes active and available.
- `created_at` (TIMESTAMPTZ): Timestamp of when the quiz was created.
- **Interactions with Pages**:
- `QuizPage.tsx`: Users can view and take quizzes.
- `CreateQuizPage.tsx` (admin): Admins create new quizzes.
- **Usage**: Defines the quizzes that users can participate in.

`quiz_questions`

- **Purpose**: Stores individual questions for each quiz.
- **Columns**:
- `id` (UUID): Primary key, unique identifier for the question.
- `quiz_id` (UUID): Foreign key, references `public.quizzes.id`. The quiz this question belongs to.
- `question_text` (TEXT): The text of the question.
- `order` (INTEGER): The order of the question within the quiz.
- `created_at` (TIMESTAMPTZ): Timestamp of when the question was created.
- **Interactions with Pages**:
- `QuizPage.tsx`: Displays questions to users.
- `CreateQuizPage.tsx` (admin): Admins add questions to quizzes.
- **Usage**: Defines the questions that make up each quiz.

`quiz_answers`

- **Purpose**: Stores possible answers for each quiz question, indicating which one is correct.
- **Columns**:
- `id` (UUID): Primary key, unique identifier for the answer.
- `question_id` (UUID): Foreign key, references `public.quiz_questions.id`. The question this answer belongs to.

- `answer_text` (TEXT): The text of the answer.
- `is_correct` (BOOLEAN): Indicates if this answer is the correct one for the question.
- `created_at` (TIMESTAMPTZ): Timestamp of when the answer was created.
- **Interactions with Pages**:
- `QuizPage.tsx` : Displays answer options to users.
- `CreateQuizPage.tsx` (admin): Admins define answers for questions.
- **Usage**: Provides answer choices for quiz questions and identifies the correct solution.

`quiz_attempts`

- **Purpose**: Records user attempts on quizzes, including their score and selected answers.
- **Columns**:
- `id` (UUID): Primary key, unique identifier for the quiz attempt.
- `user_id` (UUID): Foreign key, references `public.profiles.id` . The user who made the attempt.
- `quiz_id` (UUID): Foreign key, references `public.quizzes.id` . The quiz that was attempted.
- `score` (INTEGER): The score achieved by the user in this attempt.
- `started_at` (TIMESTAMPTZ): Timestamp of when the quiz attempt started.
- `completed_at` (TIMESTAMPTZ): Timestamp of when the quiz attempt was completed.
- `answers` (JSONB): JSON array storing the user's selected answers for each question (e.g., `[{"question_id": "...", "answer_id": "..."}]`).
- **Interactions with Pages**:
- `QuizPage.tsx` : Records user attempts and displays results.
- `UserProfilePage.tsx` : Potentially displays user's quiz history or scores.
- **Usage**: Tracks user performance and participation in quizzes.