Project Title

KhelStats

Project Description

This project will showcase and manage sports data for basketball. The platform enables users to view match schedules, explore team and player statistics, and stay informed about league events. It also includes an admin panel for managing data entry, users, and advanced features such as AI-powered search and payment integration for premium services.

Why did you choose this project?

I chose this project because I currently work as a Support Engineer at a sports data company, where I deal with live sports data, match stats, and client requests. This experience gave me a better understanding of how sports platforms work. I wanted to use my technical skills and build something similar using the MERN stack. This project helps me apply what I’ve learned at work and also shows my full-stack development skills in a real-world sports use case.

What are the benefits for the users of this app?

The app allows users to explore upcoming and completed matches, team and player stats with league-specific filtering for easy access, while registered users can follow favourite teams or unlock premium content, and admins or data entry can manage structured data through a user-friendly backend.

How can you generate revenue with this application?

Premium features include push notifications for favourite teams, player comparison tools, downloadable reports, and access to historical data, while businesses or fantasy apps can subscribe to access real-time, match data through APIs.

Additional feature - AI integration - registered user can search player statistics by simple provide player names and match names

List of Schema modules:

User Schema Module:

**1. User Module**

**Purpose:**  
Manage users, authentication, and roles.

**Key Features:**

* Register a new user
* Login user
* Update user profile
* Admin: View all users
* Admin: Edit/Delete user roles

**API Endpoints:**

* POST /api/users/register
* POST /api/users/login
* GET /api/users/profile
* PUT /api/users/profile
* GET /api/users (Admin only)
* PUT /api/users/:id (Admin only)
* DELETE /api/users/:id (Admin only)

**Database Model Fields:**

* name (String, required)
* email (String, required, unique)
* password (String, required)
* role (String: 'user', 'registered', 'admin')
* profileImage (String, optional)

**Frontend Components:**

* Register Page
* Login Page
* Profile Page
* Admin User List Page
* Admin Edit User Page

**Special Notes:**

* Password hashed using bcrypt
* JWT-based authentication

**2. Team Module**

**Purpose:**  
Manage basketball teams.

**Key Features:**

* Add team (Admin)
* View all teams
* View single team
* Edit/Delete team (Admin)

**API Endpoints:**

* POST /api/teams
* GET /api/teams
* GET /api/teams/:id
* PUT /api/teams/:id
* DELETE /api/teams/:id

**Database Model Fields:**

* teamName (String, required)
* logoImageUrl (String)
* coachName (String)
* homeCity (String)
* foundedYear (Number)
* players (Array of Player IDs)

**Frontend Components:**

* Team List Page
* Team Detail Page
* Team Create/Edit Form

**Special Notes:**

* Logo upload functionality

**3. Player Module**

**Purpose:**  
Manage player profiles.

**Key Features:**

* Add player (Admin)
* View all players
* View single player
* Edit/Delete player (Admin)

**API Endpoints:**

* POST /api/players
* GET /api/players
* GET /api/players/:id
* PUT /api/players/:id
* DELETE /api/players/:id

**Database Model Fields:**

* playerName (String, required)
* age (Number)
* position (String)
* team (Team ID reference)
* stats (Object: points, assists, rebounds, etc.)

**Frontend Components:**

* Player List Page
* Player Detail Page
* Player Create/Edit Form

**Special Notes:**

* Players are linked to teams

**4. Schedule Module**

**Purpose:**  
Manage upcoming matches and events.

**Key Features:**

* Add new matches (Admin)
* View upcoming matches
* Edit match details

**API Endpoints:**

* POST /api/schedule
* GET /api/schedule
* GET /api/schedule/:id
* PUT /api/schedule/:id
* DELETE /api/schedule/:id

**Database Model Fields:**

* homeTeam (Team ID reference)
* awayTeam (Team ID reference)
* matchDate (DateTime)
* venue (String)

**Frontend Components:**

* Schedule List Page
* Schedule Detail Page
* Schedule Create/Edit Form

**5. Live Module**

**Purpose:**  
Manage live matches and live updates.

**Key Features:**

* View live matches
* Admin: Start/Stop live matches
* Push live updates (score, event)

**API Endpoints:**

* POST /api/live
* GET /api/live
* PUT /api/live/:id
* DELETE /api/live/:id

**Database Model Fields:**

* matchId (Schedule ID reference)
* liveStatus (Boolean)
* currentScore (Object: homeScore, awayScore)
* events (Array: event type, timestamp)

**Frontend Components:**

* Live Matches Page
* Live Match Detail Page

**Special Notes:**

* WebSocket integration optional for real-time updates

**6. Standings Module**

**Purpose:**  
Track team rankings based on match results.

**Key Features:**

* Auto-update standings after each match
* View team standings

**API Endpoints:**

* GET /api/standings
* POST /api/standings/update (Admin triggers after match)

**Database Model Fields:**

* teamId (Team ID reference)
* matchesPlayed (Number)
* wins (Number)
* losses (Number)
* points (Number)

**Frontend Components:**

* Standings Page

**Special Notes:**

* Auto-calculate based on match results

**✅ Summary**

| **Module** | **Key Features** | **Access Control** |
| --- | --- | --- |
| User | Login, register, edit profile | Users/Admins |
| Team | CRUD teams | Admin only for create/update/delete |
| Player | CRUD players | Admin only for create/update/delete |
| Schedule | Manage matches | Admin only for create/update/delete |
| Live | Live match updates | Admin for update, users for view |
| Standings | View rankings | Public view |

Would you like me to also suggest a **folder structure** for organizing these modules inside your project? 📁 (It’ll make your code even cleaner.) 🚀