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:

import mongoose from 'mongoose';

const userSchema = new mongoose.Schema(

{

name: {

type: String,

required: true,

trim: true,

},

email: {

type: String,

required: true,

unique: true,

lowercase: true,

trim: true,

},

password: {

type: String,

required: true,

},

role: {

type: String,

enum: ['registered', 'premium', 'dataentry', 'admin'],

default: 'registered',

},

favorites: {

type: [String],

default: [],

},

subscriptionStatus: {

type: Boolean,

default: false,

},

dataEntryTasks: {

type: [String],

enum: ['live', 'fixture', 'standings', 'players', 'teams'],

default: [],

},

},

profileImage:{

type: image,

default:””

},

isActive:{

type: Boolean,

default: true

}

{ timestamps: true }

);

const User = mongoose.model('User', userSchema);

export default User;

**Team Schema Model**

import mongoose from 'mongoose';

const teamSchema = new mongoose.Schema(

{

teamName: {

type: String,

required: true,

trim: true,

unique: true,

},

logoImageUrl: {

type: String,

default: '',

},

homeCity: {

type: String,

default: '',

},

foundedYear: {

type: Number,

},

seasons: [

{

seasonYear: {

type: String, // Example: "2024-2025"

required: true,

},

coachName: {

type: String,

default: '',

},

players: [

{

type: mongoose.Schema.Types.ObjectId,

ref: 'Player',

}

],

conference: {

type: String,

enum: ['Eastern', 'Western'],

},

}

],

createdBy: {

type: mongoose.Schema.Types.ObjectId,

ref: 'User',

required: true,

},

updatedBy: {

type: mongoose.Schema.Types.ObjectId,

ref: 'User',

},

},

{ timestamps: true }

);

const Team = mongoose.model('Team', teamSchema);

export default Team;

**Player Schema Model**

const playerSchema = new mongoose.Schema(

{

firstName: { type: String, required: true, trim: true },

lastName: { type: String, required: true, trim: true },

position: {

type: String,

enum: ['PG', 'SG', 'SF', 'PF', 'C'],

required: true

},

jerseyNumber: { type: Number },

height: {

ft: { type: Number, required: true },

in: { type: Number, required: true },

},

weight: { type: Number },

nationality: { type: String },

birthDate: { type: Date },

profileImageUrl: { type: String, default: '' },

isActive: { type: Boolean, default: true },

seasons: [

{

seasonYear: { type: String, required: true }, // "2024-2025"

team: {

type: mongoose.Schema.Types.ObjectId,

ref: 'Team',

required: true

},

}

],

createdBy: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },

updatedBy: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },

},

{ timestamps: true }

);

**Schedule Schema Model**

const scheduleSchema = new mongoose.Schema(

{

gameId: {

type: String,

required: true,

unique: true,

trim: true,

},

seasonYear: {

type: String, // Example: "2024-2025"

required: true,

},

matchDate: {

type: Date,

required: true,

},

homeTeam: {

type: mongoose.Schema.Types.ObjectId,

ref: 'Team',

required: true,

},

awayTeam: {

type: mongoose.Schema.Types.ObjectId,

ref: 'Team',

required: true,

},

location: {

type: String,

default: '',

},

status: {

type: String,

enum: ['upcoming', 'completed', 'postponed'],

default: 'upcoming',

},

homeTeamScore: {

type: Number,

default: 0,

},

awayTeamScore: {

type: Number,

default: 0,

},

attendance: {

type: Number,

default: 0,

},

createdBy: {

type: mongoose.Schema.Types.ObjectId,

ref: 'User',

required: true,

},

updatedBy: {

type: mongoose.Schema.Types.ObjectId,

ref: 'User',

},

},

{ timestamps: true }

);

const Schedule = mongoose.model(‘Schedule’, scheduleSchema);

**Live Schema Model**

const liveSchema = new mongoose.Schema(

{

gameId: {

type: mongoose.Schema.Types.ObjectId,

ref: 'Schedule',

required: true,

unique: true,

},

currentQuarter: {

type: Number,

default: 1,

},

isOvertime: {

type: Boolean,

default: false,

},

overtimeCount: {

type: Number,

default: 0, // 0 = no overtime, 1 = first overtime, etc.

},

timeRemaining: {

type: String, // Example: "03:21"

default: "12:00",

},

homeTeamScore: {

type: Number,

default: 0,

},

awayTeamScore: {

type: Number,

default: 0,

},

status: {

type: String,

enum: ['scheduled', 'live', 'completed'],

default: 'scheduled',

},

createdBy: {

type: mongoose.Schema.Types.ObjectId,

ref: 'User',

required: true,

},

updatedBy: {

type: mongoose.Schema.Types.ObjectId,

ref: 'User',

},

},

{ timestamps: true }

);

**Standings Schema Model**

const standingsSchema = new mongoose.Schema(

{

season: {

type: String,

required: true,

},

team: {

type: mongoose.Schema.Types.ObjectId,

ref: 'Team',

required: true,

unique: true, // 1 record per team per season

},

conference: {

type: String,

enum: ['Eastern', 'Western'],

required: true,

},

wins: {

type: Number,

default: 0,

},

losses: {

type: Number,

default: 0,

},

winPercentage: {

type: Number,

default: 0.0,

},

gamesBehind: {

type: Number,

default: 0.0,

},

streak: {

type: String,

enum: ['W', 'L'], // Win streak or Lose streak

default: 'W',

},

streakCount: {

type: Number,

default: 0,

},

lastTenWins: {

type: Number,

default: 0,

},

lastTenLosses: {

type: Number,

default: 0,

}

},

{ timestamps: true }

);

**Line-up Schema Model**

const gameLineupSchema = new mongoose.Schema({

gameId: {

type: mongoose.Schema.Types.ObjectId,

ref: 'Schedule', // or your Game/Match model

required: true,

},

team: {

type: mongoose.Schema.Types.ObjectId,

ref: 'Team',

required: true,

},

starters: [

{

player: {

type: mongoose.Schema.Types.ObjectId,

ref: 'Player',

required: true,

},

position: {

type: String,

enum: ['PG', 'SG', 'SF', 'PF', 'C'],

required: true,

}

}

],

substitutions: [

{

playerIn: {

type: mongoose.Schema.Types.ObjectId,

ref: 'Player',

required: true,

},

playerOut: {

type: mongoose.Schema.Types.ObjectId,

ref: 'Player',

required: true,

},

time: {

type: String, // E.g., "Q2 04:25" or exact timestamp

required: true,

}

}

],

createdBy: {

type: mongoose.Schema.Types.ObjectId,

ref: 'User',

required: true,

},

updatedBy: {

type: mongoose.Schema.Types.ObjectId,

ref: 'User',

},

}, { timestamps: true });

**Match Stats Schema Model**

const matchStatSchema = new mongoose.Schema({

gameId: {

type: mongoose.Schema.Types.ObjectId,

ref: 'Schedule',

required: true,

},

teamId: {

type: mongoose.Schema.Types.ObjectId,

ref: 'Team',

required: true,

},

playerId: {

type: mongoose.Schema.Types.ObjectId,

ref: 'Player',

required: true,

},

stats: {

points: { type: Number, default: 0 },

rebounds: { type: Number, default: 0 },

assists: { type: Number, default: 0 },

steals: { type: Number, default: 0 },

blocks: { type: Number, default: 0 },

fouls: { type: Number, default: 0 },

minutesPlayed: { type: Number, default: 0 }

// You can add turnovers, 3-pointers, etc.

},

createdBy: {

type: mongoose.Schema.Types.ObjectId,

ref: 'User',

required: true,

},

updatedBy: {

type: mongoose.Schema.Types.ObjectId,

ref: 'User',

}

}, { timestamps: true });

**-hold this Schema -- Player Stats Schema Model**

const playerStatsSchema = new mongoose.Schema(

{

gameId: {

type: mongoose.Schema.Types.ObjectId,

ref: 'Schedule', // or 'Game', depending on your model name

required: true,

},

playerId: {

type: mongoose.Schema.Types.ObjectId,

ref: 'Player',

required: true,

},

teamId: {

type: mongoose.Schema.Types.ObjectId,

ref: 'Team',

required: true,

},

points: {

type: Number,

default: 0,

},

assists: {

type: Number,

default: 0,

},

rebounds: {

type: Number,

default: 0,

},

steals: {

type: Number,

default: 0,

},

blocks: {

type: Number,

default: 0,

},

minutesPlayed: {

type: Number,

default: 0,

},

fieldGoalPercentage: {

type: Number,

default: 0,

},

threePointPercentage: {

type: Number,

default: 0,

},

freeThrowPercentage: {

type: Number,

default: 0,

},

turnovers: {

type: Number,

default: 0,

},

personalFouls: {

type: Number,

default: 0,

},

createdBy: {

type: mongoose.Schema.Types.ObjectId,

ref: 'User', // data entry user who entered stats

required: true,

},

updatedBy: {

type: mongoose.Schema.Types.ObjectId,

ref: 'User', // if someone edited stats later

},

},

{ timestamps: true }

);