Project Title

***MiniBasketball***

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.

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

**List of schema modules for the project titled *MiniBasketball***

**1. User Module**

* name : string
* email : string
* password : string
* role :: [‘registered’, ‘dataentry’, ‘admin’]
* dataEntryTasks: [‘live’, ‘fixture’, ‘standings’, ‘players’, ‘teams’]
* profileImage : string
* isActive: boolean

**2. League Module**

* leagueName : string
* createdBy: Schema.Types.ObjectId
* updatedBy: Schema.Types.ObjectId

**3. Season Module**

* seasonName : string
* createdBy: Schema.Types.ObjectId

**4. Team Module**

* teamName : string
* logoImageUrl : string
* homeCity : string
* season:[ {seasonYear, coachName, players(Schema.Types.ObjectId)}]
* createdBy: Schema.Types.ObjectId
* updatedBy: Schema.Types.ObjectId

**5. Player Module**

* firstName : string
* lastName : string
* position : string
* jerseyNumber: Number
* height: String
* weight: String
* nationality: String
* birthdate: Date
* profileImageUrl: string
* isActive: Boolean
* createdBy: : Schema.Types.ObjectId
* updatedBy: Schema.Types.ObjectId
* season:[{seasonYear,team}]

**6. Schedule Module**

* gameId : Number
* seasonYear :String (ex 2024-2025)
* matchDate: Date
* homeTeam: Schema.Types.ObjectId
* awayTeam: Schema.Types.ObjectId
* venue: String
* status: [‘pre-game’,’in-progress’,’cancled’, ‘final’, ‘postponed’]
* homeTeamScore: Number
* awayTeamScore: Number
* attendance: Number
* createdBy: : Schema.Types.ObjectId
* updatedBy: Schema.Types.ObjectId

**7. Line-up Module**

* gameId: : Schema.Types.ObjectId
* teamHome: Schema.Types.ObjectId
* starters: [{{player},{positon}}]
* substitutions:[{{ playerIn},{playerOut},{time}}]
* teamAway: Schema.Types.ObjectId
* starters: [{{player},{positon}}]
* substitutions:[{{ playerIn},{playerOut},{time}}]
* createdBy: : Schema.Types.ObjectId
* updatedBy: Schema.Types.ObjectId

**8. Match stats Module**

* gameId: Schema.Types.ObjectId
* teamId: Schema.Types.ObjectId
* playerId: Schema.Types.ObjectId
* stats:{ points, rebounds, assists, steals, blocks, fouls, minutesPlayed }
* createdBy: : Schema.Types.ObjectId
* updatedBy: Schema.Types.ObjectId

**Below can be ignored**

**7. Standings Module**

* season : string
* team: Schema.Types.ObjectId
* conference :String
* wins : Number
* losses: Number
* winPercentage: Number
* gamesBehind: Number
* streak: String(W,L)
* streakCount: Number
* lastTenWins: Number
* lastTenLosses: Number
* points (Number)

**8. Live Module**

* gameId : Schema.Types.ObjectId
* currentQuarter :Number
* isOvertime: Boolean
* overtimeCount: Number
* timeRemaining: Number
* homeTeamScore: Number
* awayTeamScore: number
* status: [‘pre-game’,’in-progress’,’cancled’, ‘final’, ‘postponed’]
* createdBy: : Schema.Types.ObjectId
* updatedBy: Schema.Types.ObjectId

/backend

├── config/

│ └── db.js # MongoDB connection

├── controllers/

│ ├── userController.js

│ ├── teamController.js

│ ├── playerController.js

│ ├── scheduleController.js

│ ├── lineupController.js

│ ├── matchStatsController.js

│ ├── leagueController.js

│ ├── seasonController.js

├── models/

│ ├── User.js

│ ├── Team.js

│ ├── Player.js

│ ├── Schedule.js

│ ├── GameLineup.js

│ ├── MatchStats.js

│ ├── League.js

│ └── Season.js

├── routes/

│ ├── userRoutes.js

│ ├── teamRoutes.js

│ ├── playerRoutes.js

│ ├── scheduleRoutes.js

│ ├── lineupRoutes.js

│ ├── matchStatsRoutes.js

│ ├── leagueRoutes.js

│ └── seasonRoutes.js

├── middlewares/

│ ├── authMiddleware.js

│ ├── roleMiddleware.js

│ └── errorMiddleware.js

├── utils/

│ └── validators.js # Optional, for input validation

├── server.js

└── .env

#### 1. models/User.js

const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({

name: String,

email: { type: String, unique: true },

password: String,

role: { type: String, enum: ['registered', 'dataentry', 'admin'], default: 'registered' },

dataEntryTasks: [String],

profileImage: String,

isActive: { type: Boolean, default: true }

}, { timestamps: true });

module.exports = mongoose.model('User', userSchema);

controllers/userController.js

const User = require('../models/User');

const bcrypt = require('bcryptjs');

// Register user

exports.register = async (req, res) => {

const { name, email, password, role } = req.body;

const hashedPassword = await bcrypt.hash(password, 10);

const user = new User({ name, email, password: hashedPassword, role });

await user.save();

res.status(201).json({ message: 'User registered successfully' });

};

routes/userRoutes.js

const express = require('express');

const router = express.Router();

const userController = require('../controllers/userController');

router.post('/register', userController.register);

// router.post('/login', userController.login); (you can add later)

module.exports = router;

server.js

const express = require('express');

const mongoose = require('mongoose');

const dotenv = require('dotenv');

const userRoutes = require('./routes/userRoutes');

// (other route imports)

dotenv.config();

const app = express();

app.use(express.json());

// DB connection

mongoose.connect(process.env.MONGO\_URI)

.then(() => console.log('MongoDB connected'))

.catch(err => console.log(err));

// Routes

app.use('/api/users', userRoutes);

// app.use('/api/teams', teamRoutes); etc.

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));

### Repeat Modules For:

| **Feature** | **Model Filename** | **Route Prefix** |
| --- | --- | --- |
| User | User.js | /api/users |
| League | League.js | /api/leagues |
| Season | Season.js | /api/seasons |
| Team | Team.js | /api/teams |
| Player | Player.js | /api/players |
| Schedule | Schedule.js | /api/schedules |
| Line-up | GameLineup.js | /api/lineups |
| Match Stats | MatchStats.js | /api/match-stats |