Cricket Fantasy League App Detailed Proposal

This document outlines the detailed explanation, system architecture, and feature implementations for the Cricket Fantasy League App.
It includes the task breakdown, database schema, relations, API logic, and implementation steps for each feature.

User Registration & Profile

The user registration system allows users to sign up via email, phone number (OTP), or social media accounts.

It stores user preferences and tracks their prediction performance. The profile also includes login credentials and historical prediction data.

Database Schema:

users

- user_id (PK)
- email (unique)
- phone_number
- username
- favorite_team
- profile_picture
- registration_date
- last_login

API Logic:

POST /register: Registers a new user with email or phone.

POST /login: Logs in the user with JWT or OAuth.

GET /profile: Retrieves the user profile and prediction data.

Season & League Selection

This feature allows users to select a season (e.g., IPL, T20 World Cup) and choose a league to participate in.

The league can be either public or private, and users can join or create leagues with custom rules.

Database Schema:

seasons

- season_id (PK)
- season_name
- start_date
- end_date
- status (active, completed)

leagues

- league_id (PK)
- league_name
- season_id (FK)

- country

- match_format (T20, ODI, etc.)

API Logic:

POST /create-league: Creates a new league with custom rules.

GET /leagues: Lists all available leagues for the user to join.

GET /seasons: Retrieves available seasons for user selection.

Player Prediction System

The player prediction system allows users to predict top performers for each match or season.

Points are awarded based on player stats like runs, wickets, and catches.

The system integrates with cricket data APIs to fetch real-time match and player performance data.

Database Schema:

predictions

- prediction_id (PK)

- user_id (FK)

- league_id (FK)

- match_id (FK)

- player_id (FK)

- predicted_stat (runs, wickets)

- prediction_type (match/season)

- points_awarded (after match)

API Logic:

POST /predict: Allows users to submit their predictions.

GET /predictions/{user_id}: Retrieves all predictions made by a user.

GET /match/{match_id}/results: Fetch match results and calculate prediction points.

Leaderboard & Rankings

The leaderboard displays user rankings based on their prediction accuracy.

Leaderboards can be global or limited to specific leagues. Rankings update after each match.

Database Schema:

leaderboards

- leaderboard_id (PK)
- user_id (FK)
- league_id (FK)
- points
- rank
- season_id (FK)

API Logic:

GET /leaderboard/{league_id}: Fetches the leaderboard for a specific league.

GET /leaderboard: Fetches global leaderboard for all leagues.

Social Features & Community Engagement

This feature allows users to interact with each other through chat, join private leagues, and collaborate on predictions in teams.

Users can invite friends, discuss predictions, and share their achievements on social media.

Database Schema:

friends

- friend_id (PK)
- user_id (FK)
- friend_user_id (FK)
- status (pending, accepted)

chats

- chat_id (PK)
- user_id (FK)
- message
- timestamp

API Logic:

POST /invite-friend: Allows users to invite friends to join leagues.

POST /chat: Allows users to send messages in the chat.

GET /chats/{user_id}: Retrieves all chats for a user.

In-App Rewards & Monetization

Users earn virtual coins and badges based on their prediction performance. Premium features, such as advanced stats, can be unlocked through a subscription model.

Database Schema:

rewards

- reward_id (PK)
- user_id (FK)
- reward_type
- points
- timestamp

subscriptions

- subscription_id (PK)
- user_id (FK)
- subscription_plan
- start_date
- end_date

API Logic:

POST /redeem-reward: Allows users to redeem rewards for coins.

POST /subscribe: Allows users to subscribe to premium features.

GET /rewards/{user_id}: Retrieves all rewards earned by the user.

AI-Powered Prediction Model

The app uses machine learning models to provide AI-based suggestions for optimal player picks.

The AI model considers historical performance, player form, injuries, and other factors.

API Logic:

POST /train-predictor: Trains the AI prediction model based on player data.

GET /predict/{user_id}: Fetches AI-powered recommendations for the user.

Advanced Stats & Analytics

This feature provides in-depth analysis of player and team performance through visual insights such as graphs, charts, and heatmaps.

It allows users to make informed predictions based on data.

API Logic:

GET /player-stats/{player_id}: Retrieves detailed stats for a specific player.

GET /team-stats/{team_id}: Retrieves detailed stats for a specific team.

GET /analytics/{user_id}: Fetches advanced analytics for users' prediction strategies.