BetMate Master Documentation

Your Project

February 20, 2025

Contents

1 Introduction & Motivation

BetMate is a web application for friendly betting on Indian Premier League (IPL) cricket matches. It was inspired by a childhood tradition of coin tosses and daily bets between friends. This project digitizes that experience, allowing users to:

- Sign up and add "Betmates" (friends)
- Initiate a coin toss for an upcoming match
- Choose teams if they win the toss
- Track points automatically based on match outcomes

2 High-Level Overview

2.1 Core Concepts

- **User Accounts:** Secure sign-up/login with JWT-based authentication.
- **Betmates:** Users can send friend requests to others, thereby adding them as "betmates" to place bets.
- **Matches:** Pulled from the Cricbuzz API (via RapidAPI) or test data for the current/future IPL season.
- **Coin Toss:** Randomly decides who chooses a team first for a particular match.
- **Points:** +10 for the winner's chosen team, -10 for the other side.

2.2 Project Goals

1. **Recreate** the excitement of daily coin tosses and bets on IPL matches.
2. **Extend** this so users can bet with multiple friends simultaneously.
3. **Track** points and maintain a friendly, competitive history in one place.

3 Tech Stack & Architecture

3.1 Tech Stack

Frontend:

- React (with React Router)
- Tailwind CSS (styling)
- Axios (API calls)

Backend:

- Node.js / Express
- Prisma (ORM)
- PostgreSQL database

Third-party API: Cricbuzz (via RapidAPI) for real match data (if desired).

3.2 System Architecture

A traditional client-server architecture:

- **Frontend:** A Single-Page React application that manages UI and user navigation.
- **Backend:** Provides RESTful APIs for authentication, friend management, bet logic, and database operations via Prisma.
- **Database:** Stores users, matches, bets, friend requests, etc.

3.3 Folder Structure

Below is a simplified folder layout:

```
BetMate/
 backend/
   app.js
   server.js
   prisma/
   controllers/
   routes/
   middleware/
   ...
 frontend/
    src/
      pages/
      components/
      context/
      ...
    public/
    ...
```

4 Data Model

4.1 Prisma Schema (Condensed)

```
model User {
  id       Int    @id @default(autoincrement())
  username String @unique
  password String
  firstName String
  lastName  String
  isAdmin   Boolean @default(false)

  friendRequestsReceived FriendRequest[] @relation("addressee")
  friendRequestsSent     FriendRequest[] @relation("requester")
  betmateBets            MatchBetmate[]  @relation("BetmateBets")
  bets                   MatchBetmate[]  @relation("UserBets")
  pastBetmatesBetmate    PastBetmate[]   @relation("PastBetmatesBetmate")
  pastBetmatesUser       PastBetmate[]   @relation("PastBetmatesUser")
}

model Match {
```

```
  id               Int    @id @unique
  date             DateTime
  team1            String
  team2            String
  location         String
  matchType        String
  seriesName       String
  matchDescription String
  state            String
  status           String
  winner           String

  matchBetmates    MatchBetmate[]
}

model MatchBetmate {
  id            Int     @id @default(autoincrement())
  userId        Int
  matchId       Int
  betmateId     Int
  isTossed      Boolean @default(false)
  tossWinnerId  Int?
  userChoice    String?
  betmateChoice String?
  status        String  @default("no_bet")
  userScore     Int     @default(0)

  user          User    @relation("UserBets", fields: [userId], references: [id])
  betmate       User    @relation("BetmateBets", fields: [betmateId], references: [id])
  match         Match   @relation(fields: [matchId], references: [id])

  @@unique([userId, matchId, betmateId])
}

model FriendRequest {
  id          Int    @id @default(autoincrement())
  requesterId Int
  addresseeId Int
  status      String

  addressee   User   @relation("addressee", fields: [addresseeId], references: [id])
  requester   User   @relation("requester", fields: [requesterId], references: [id])

  @@unique([requesterId, addresseeId])
}

model PastBetmate {
  id        Int      @id @default(autoincrement())
  userId    Int
  betmateId Int
  removedAt DateTime @default(now())
```

```
  betmate   User     @relation("PastBetmatesBetmate", fields: [betmateId], references: [id])
  user      User     @relation("PastBetmatesUser", fields: [userId], references: [id])

  @@unique([userId, betmateId])
}
```

4.2 Key Entities

- **User:** Stores basic info and authentication details (hashed password).
- **Match:** Represents an IPL match with teams, location, status, and winner.
- **MatchBetmate:** Links a user to a match (plus a specific betmate). Contains toss result, chosen teams, and score.
- **FriendRequest:** Models a friend request cycle (pending, accepted, declined).
- **PastBetmate:** Records betmates who were removed.

5 Main Features & Workflow

5.1 User Registration & Authentication

- **Sign Up:** Users provide `username, firstName, lastName, password`. Passwords are hashed with `bcryptjs`.
- **Sign In:** On success, server sets an HTTP-only JWT cookie.
- **Protected Routes:** `validateToken` middleware verifies the JWT.

5.2 Betmates (Friends) Management

- Global **User Search**.
- **Send Friend Request**.
- **Accept/Decline Requests**.
- **Remove Betmate**, which logs them in `PastBetmate`.

5.3 Matches & Integration

- **Match Storage:** Optionally fetched from Cricbuzz or inserted via a test script.
- **States:** "scheduled", "in-progress", "complete".
- **Updates:** When a match is complete, the backend updates user scores automatically.

5.4 Bets, Coin Toss, and Team Selection

1. **Initiate Toss:** Random (50/50). Stores `tossWinnerId` in `MatchBetmate`.
2. **Choose Team:** The winner picks a team; the other user is assigned the opponent.
3. **Status:** Can be `no_bet, toss_won, toss_lose, team_chosen, won, lost`.

5.5 Points & Scoring System

- +**10** for the user whose chosen team won, **-10** for the other.
- **No result** leads to 0 points assigned.
- `updateUserScores` runs whenever a match's `state` is "complete".

5.6 Viewing Bets & Points

- **MyPoints:** Summarizes total score vs. a selected betmate.
- **CurrentBets:** Lists all bets for the logged-in user.
- **Home:** Filter matches by "active", "upcoming", or "completed" for a chosen betmate.

6 API Endpoints (Backend)

6.1 Authentication Routes

Method	Endpoint	Description
POST	`/api/auth/register`	Register a new user.
POST	`/api/auth/login`	Log in user & set JWT cookie.
GET	`/api/auth/logout`	Clear JWT cookie (logout).
GET	`/api/auth/validate`	Validate JWT & return user info.

6.2 Match Routes

Method	Endpoint	Description
GET	`/api/matches`	Fetch stored matches (IPL 2024).
GET	`/api/matches/{id}`	Fetch specific match by ID.
POST	`/api/matches/fetch-updates`	Trigger match fetch from Cricbuzz.

6.3 Bet Routes

Method	Endpoint	Description
GET	`/api/bets/user/{userId}`	Fetch bets for a given user.
POST	`/api/bets/initiateToss`	Initiate a coin toss.
POST	`/api/bets/chooseTeam`	Winner chooses a team.
GET	`/api/bets/user/{userId}/totalScore`	Summarize points vs. a betmate.

Method	Endpoint	Description
POST	`/api/friends/sendRequest`	Send a friend request.
GET	`/api/friends/pendingRequests`	Fetch user's pending requests.
PUT	`/api/friends/acceptRequest/{requestId}`	Accept friend request.
PUT	`/api/friends/declineRequest/{requestId}`	Decline friend request.
GET	`/api/friends/list/{userId}`	List accepted betmates.
DELETE	`/api/friends/removeBetmate/{betMateId}`	Remove betmate (moves to PastBetmate).

Method	Endpoint	Description
GET	`/api/users/search`	Search for users by name/username.
GET	`/api/users/{userId}`	Fetch user data or a user-betmate score.

6.4 Friend Routes

6.5 User Routes

7 Frontend Overview

7.1 Key Pages

- **LandingPage:** Public homepage with a hero section.
- **SignIn & SignUp:** Authentication forms.
- **HomePage:** Once logged in, select betmate, view matches by status (active, upcoming, completed).
- **BetMates:** Manage friend requests, see or remove existing betmates.
- **CoinFlip:** Start or see coin toss results. If you win, pick a team.
- **CurrentBets:** Shows all bets for the logged-in user.
- **MyPoints:** View total score vs. a selected betmate, with match details.

7.2 Notable Components

- **MatchCard:** Renders team logos, time, location, plus flip or choose-team buttons.
- **CurrentBetsCard:** Shows user-chosen team, betmate-chosen team, status/toss results.
- **TeamSelectionModal:** Popup for picking teams if user won the toss.

7.3 Contexts & State Management

- **AuthContext:** Manages user login state, logs out user, verifies JWT on startup.
- **MatchContext:** Stores currently selected match plus chosen betmate, shared across pages.

8 Local Development Setup

8.1 Prerequisites

- Node.js (v14+ recommended)
- npm or yarn
- PostgreSQL (local or hosted)
- `.env` file in the backend with DB connection string, JWT secret, etc.

8.2 Installation & Running Locally

1. **Clone the Repo:**

   ```
   git clone <your-repo-url>
   cd BetMate
   ```

2. **Backend Setup:**

   ```
   cd backend
   npm install
   npx prisma migrate dev
   npm start
   ```

   Optionally run your seeding scripts (e.g., insertTestMatches.js) if desired.

3. **Frontend Setup:**

   ```
   cd ../frontend
   npm install
   npm run dev
   ```

   Typically, the frontend is accessible at http://localhost:5173.

9 Environment Variables

9.1 Backend (`.env` Example)

```
DATABASE_URL=postgresql://user:password@localhost:5432/betmate
JWT_SECRET=some_super_secret_key
X_RAPIDAPI_KEY=your_rapidapi_key
NODE_ENV=development
PORT=3000
```

9.2 Frontend

Usually set in a `.env` file under `VITE_BACKEND_API_URL`:

```
VITE_BACKEND_API_URL=http://localhost:3000
```

10 Future Enhancements & Next Steps

1. **Real-Time Updates:** Use websockets to notify when opponent picks a team, or when a match completes.
2. **Leaderboards:** A global or group-level ranking by total points.
3. **Push Notifications:** Browser/mobile push for match completion or toss results.
4. **Multi-Sport Expansion:** Support other tournaments or sports beyond IPL.
5. **Admin Panel:** For administrative tasks (managing matches, user moderation, etc.).

11 License & Disclaimer

Friendly Betting Only: BetMate is for fun and does not involve real-money transactions. No warranties or liability. If you plan on distributing or open-sourcing widely, add an appropriate open-source license (e.g., MIT, Apache) in the repository.

End of BetMate Master Documentation.