Requirement Document

Project Overview:

- **Objective:** The objective of the "BallByBall" website is to develop a user-friendly platform for real-time scoring of local cricket matches, aiming to streamline the scoring process and enhance data accuracy.
- **Target Audience:** The target audience for the "BallByBall" website includes cricket enthusiasts, players, umpires, and spectators involved in local cricket matches.

Functional Requirements:

- **Authentication:**
  - Implement user authentication for secure login and registration.
  - Include user profile management features (update, delete, sign-out).
- **Listings:**
  - Users can create new match listings with details such as team names, description, format, etc.
  - Implement CRUD functionality for listings (Create, Read, Update, Delete).
  - Enable users to search for listings based on various criteria.
- **User Interaction:**
  - Users can create their own local matches.
  - Audience can vote for their favourite player.
- **Frontend:**
  - Use React.js for the frontend development.
  - Implement a responsive and user-friendly design.

Non-Functional Requirements:

- **Performance:** Ensure the application is responsive and can handle a reasonable number of concurrent users.
- **Security:** Implement secure authentication and protect user data.

Technology Stack:

- MongoDB for the database
- Express.js as the backend framework
- React.js for the frontend
- Node.js as the runtime environment for the backend
- Mongoose as the ODM (Object Data Modeling) library for MongoDB
- Socket.io for Real Time Updation.
- Tailwind CSS for Styling.

Deployment

- Deploy the application to a hosting service (e.g., Render).

Design Document

System Architecture:

- Client-Server architecture with React.js as the frontend and Express.js/Node.js as the backend.
- Use Socket.io for real time score updation.

Database Schema:

- UserModel: id, username, email, password, etc.
- MatchTable: id, title, description, team name, format, venue, etc.
- PlayerModel: name, achievements, individual score, etc.

Authentication Flow:

- User registration and login with JWT tokens.

User Interface Design:

- Design responsive and intuitive UI using React.js and Tailwind CSS.
- Include pages for current matches, upcoming matches, etc.

API Endpoints

- Define API endpoints for user authentication, CRUD operations on listings, messaging, etc.

Error Handling

- Implement a consistent error-handling mechanism for API requests and form submissions.

Deployment Strategy

Testing:

- Define testing strategies for both frontend and backend components.

Scalability:

- Consider potential future scalability requirements and design the system to handle increased user loads.