Khel-Connect: Architecture & Design Document

1. System Architecture Overview

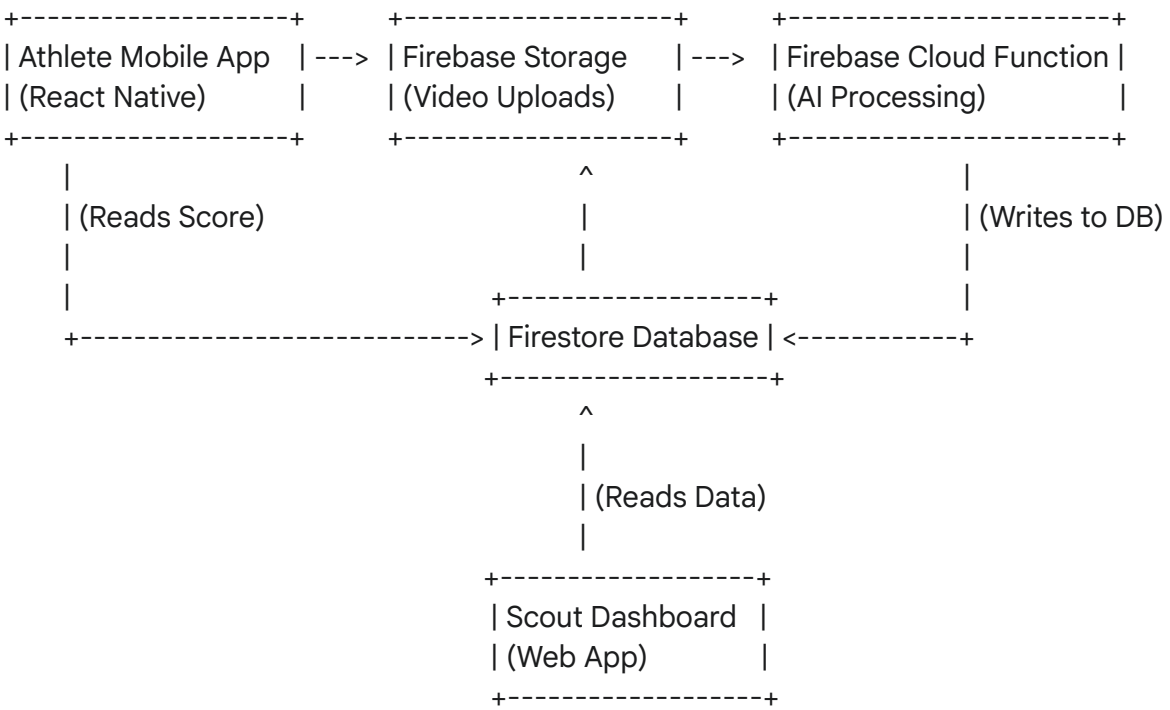
Khel-Connect is built on a modern, serverless, and event-driven architecture designed for scalability, cost-effectiveness, and rapid development. The system's core is a **Firebase-centric backend**, which leverages Cloud Functions to perform the computationally intensive AI analysis without requiring a dedicated server.

The architecture follows a clear division of responsibilities:

- **Athlete App**: A cross-platform mobile application for user interaction, video capture, and profile management.
- **Scout Dashboard**: A web portal for authenticated sports authorities to access and visualize player data.
- **AI Backend**: A serverless pipeline that triggers on video uploads, processes the video with computer vision models, and updates the database with the results.

2. High-Level Architectural Flow

The following diagram illustrates the flow of data from a user's device to the scout's dashboard.



3. Component Breakdown

3.1. Frontend Components

- **Athlete App (khel-connect-app):**
  - **Technology**: React Native, chosen for its ability to create a single codebase for both iOS and Android.
  - **Purpose**: Handles user authentication, profile management, video selection/upload, and displaying the AI-generated "Skill Score" and leaderboards.
  - **Key APIs**: Communicates directly with Firebase Authentication, Firebase Storage, and Firestore.
- **Scout Dashboard (khel-connect-scout):**
  - **Technology**: A web framework (e.g., React, Vue.js, or simple HTML/JS) that provides a secure, read-only interface.
  - **Purpose**: Allows sports authorities to securely log in, view player profiles, and access filtered leaderboards and detailed AI analysis metrics.
  - **Key APIs**: Authenticates users via Firebase Auth and reads data directly from the Firestore database.

3.2. Backend & AI Components

- **Firebase Authentication**: Manages user accounts for both athletes and sports authorities, ensuring a secure login process.
- **Firebase Storage**: A scalable cloud storage solution that hosts all the raw video clips uploaded by the athletes.
- **Firestore Database**: A NoSQL document database that serves as the single source of truth for the entire application. It stores:
  - User profiles (name, location, sport).
  - Video metadata (timestamp, link to video in Storage).
  - The results of the AI analysis, including the Skill Score and specific metrics (e.g., arm angle, swing speed).
- **Firebase Cloud Functions**: The core of the AI backend. A specific function is triggered whenever a new video file is uploaded to Firebase Storage. This function performs the following actions:
  1. **Video Ingestion**: Reads the newly uploaded video file from Firebase Storage.
  2. **Computer Vision Analysis**:
     - The function uses a **MediaPipe** model to perform **pose estimation** on the video.
     - It extracts key joint coordinates (e.g., shoulder, elbow, wrist) frame-by-frame.
  3. **Custom Logic**: The function runs a custom algorithm on the extracted joint coordinates to determine the "Skill Score." This logic will be tailored to the specific biomechanics of a cricket bowl.
  4. **Database Update**: The calculated Skill Score and any other relevant metrics are then

written back to the corresponding Firestore document for that video.

4. API and Data Model

4.1. Firestore Data Model

The Firestore database will be structured to support the core features, with separate collections for users and videos.

- **users Collection**:
  - **Document ID**: uid (from Firebase Authentication)
  - **Fields**:
    - name: string
    - email: string
    - location: string
    - sport: string (initially "Cricket")
    - role: string ("athlete" or "scout")
- **videos Collection**:
  - **Document ID**: Auto-generated
  - **Fields**:
    - userId: string (reference to the user document)
    - videoUrl: string (public URL to the video in Firebase Storage)
    - uploadTimestamp: timestamp
    - status: string ("processing", "complete", "failed")
    - skillScore: number (populated by the AI function)
    - analysisMetrics: map/object (e.g., { "armAngle": 90, "swingSpeed": 75 })

4.2. API Endpoints (via Firebase SDK)

The application does not use traditional REST APIs. Instead, it relies on the Firebase SDK and triggers.

- **Authentication**: firebase.auth().signInWithEmailAndPassword()
- **Data Upload**: firebase.storage().ref().put()
- **Data Retrieval**: firebase.firestore().collection().get() or onSnapshot() for real-time updates.

This architecture is designed to be both powerful and flexible, allowing you to easily scale the backend processing and add new sports or AI models in the future.