Individual Contribution Report

SwingAI - AI-Powered Cricket Performance Analysis Platform

Alex Thuruthel | 2022101028

Role & Responsibilities

As the **Core Infrastructure & Batting System Lead**, I was responsible for establishing the foundational application architecture, navigation system, and the complete batting analysis module. This included end-to-end development of video uploading, AI-powered swing analysis, and performance progress tracking.

Key Contributions

1. Application Architecture & Routing

Component	File	Contribution
App Entry Point	`src/App.tsx`	Configured React Router (10 routes), implemented responsive layout structure.
Navigation	`src/Sidebar.tsx`	Built collapsible sidebar with 8 items and Framer Motion animations.
Main App	`src/main.tsx`	Set up React 19 root, Bootstrap integration, and theme system.

2. Dashboard Development

- **Main Dashboard:** Developed unified view with batting/bowling toggles and 4 dynamic stat cards.
- **Coach View:** Implemented team-wide analytics, injury alerts, and player progress tables.
- **Quick Actions:** Built context-aware action cards that adapt to the selected view mode.
- **Visualizations:** Integrated Recharts for interactive performance trend analysis.

3. Batting Analysis Module

- **Swing Analysis Interface:** Drag-and-drop video upload with URL-based processing.
- **AI Simulation:** Created 3-second analysis loading states with real-time progress feedback.
- **Metrics Visuals:** Developed an 8-metric radar chart tracking Bat Speed, Timing, Balance, etc.
- **View Modes:** Implemented Original, Overlay, and Skeleton (SVG) tracking visualizations.

4. Batting Progress Tracking

- **Performance Charts:** Combined area/line charts for 6-month speed and consistency trends.
- **Shot Type Analysis:** Table-based breakdown of performance across different shot categories.

5. Settings & Configuration

- **Panel Design:** Developed 7-tab configuration (Profile, Goals, Privacy, Appearance, etc.).
- **Appearance:** Built a theme selector supporting Dark/Light/Auto modes and 6 accent colors.
- **Data Management:** Implemented storage usage displays and data export/deletion functionality.

Technologies Used

– **React 19 & TypeScript:** For component architecture and strict type safety.
– **Framer Motion:** For fluid page transitions and UI animations.
– **Recharts:** For data-heavy visualization (Line, Bar, Area, and Radar).
– **React Bootstrap:** For responsive grid systems and layout containers.

Problem Solving

Challenge: Video Timeline Scrubbing
The custom video player required frame-by-frame precision. I implemented an HTML range input synchronized with the `videoRef` using a `handleSeek` function to map range values directly to `currentTime`, ensuring smooth scrubbing.

Impact & Verification

- **User Experience:** Achieved fast initial loads (<3s) and smooth 60fps animations.
- **Code Quality:** 100% TypeScript type coverage and zero ESLint errors.
- **Verification:** Successfully passed `tsc -b`, `npm run lint`, and production build tests.