From Ashes to Insights:
Building a Cricket AI

A Capstone Project, by Daniel Bacon

Large Language Models (LLMs) are great but…

What are LLM Hallucinations?

Hallucinations are when a LLM gives an answer that sounds confident but is actually wrong, illogical, or made up

This happens because it's trained on huge datasets that contain incorrect information

How do you stop LLMs hallucinating?

Ground the model by connecting it to trusted, verified data

This limits the scope of questions but massively improves accuracy

Objectives

1. **Build a reliable cricket data foundation:**

Create a structured dataset rich enough to support detailed, context-aware performance evaluation.

2. **Incorporate contextual adjustments:**

Account for opponent strength, venue difficulty, and match conditions so players' performances are evaluated fairly.

3. **Develop an application layer that prevents hallucinations:**

Instead of letting the model access everything, it can only query prebuilt, verified functions that return factual data.

4. **Provide flexible analytics:**

Enable filtering by format, era, country, venue, or player role to support a wide range of grounded insights.

Why Cricket?

High-granularity data
(ball-by-ball statistics)

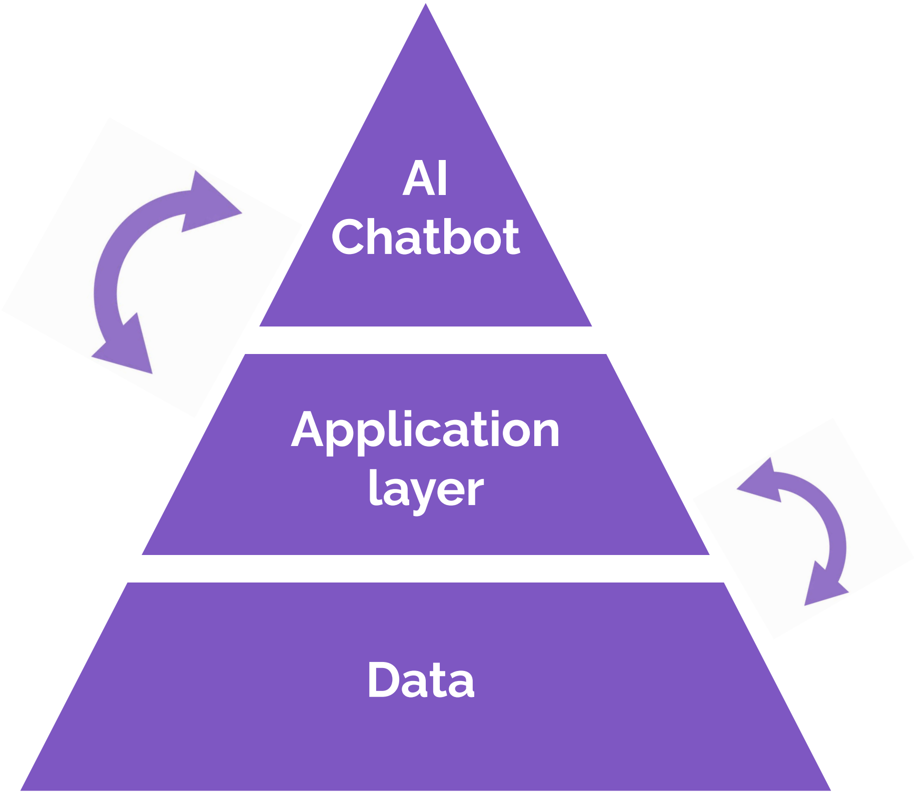
Cricket is a team sport where context matters enormously:

- **Opponent strength** influences how impressive a performance is
- **Venue & pitch conditions** can drastically affect batting and bowling difficulty

Cricket Terminology

3 Formats of the game;

1. **T20**
   - Shortest format
   - Fastest pace
   - Strike rate & Economy more important
1. **ODI** (One Day International)
   - Middle ground
2. **Test match**
   - Longest format
   - Lowest rate of scoring



Methodology

Section 1: Data Foundation

- Loading & formatting the data
- Calculate player impact, venue & opposition metrics

Section 2: Application Layer

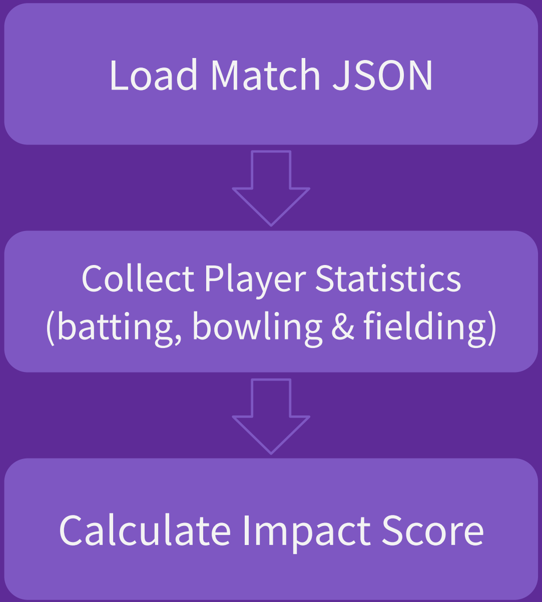
- Create utility functions that manipulate the data to reduce AI usage (and cost)

Section 3: Grounding the AI

- Connecting the AI to the application layer

Section 1: Data Foundation

Phase 1:
Impact Calculations



For each match in the dataset:

- Collected batting, bowling and fielding statistics
- Scores were normalised per match
- Combine into Match Impact Score
- Add man of the match bonus

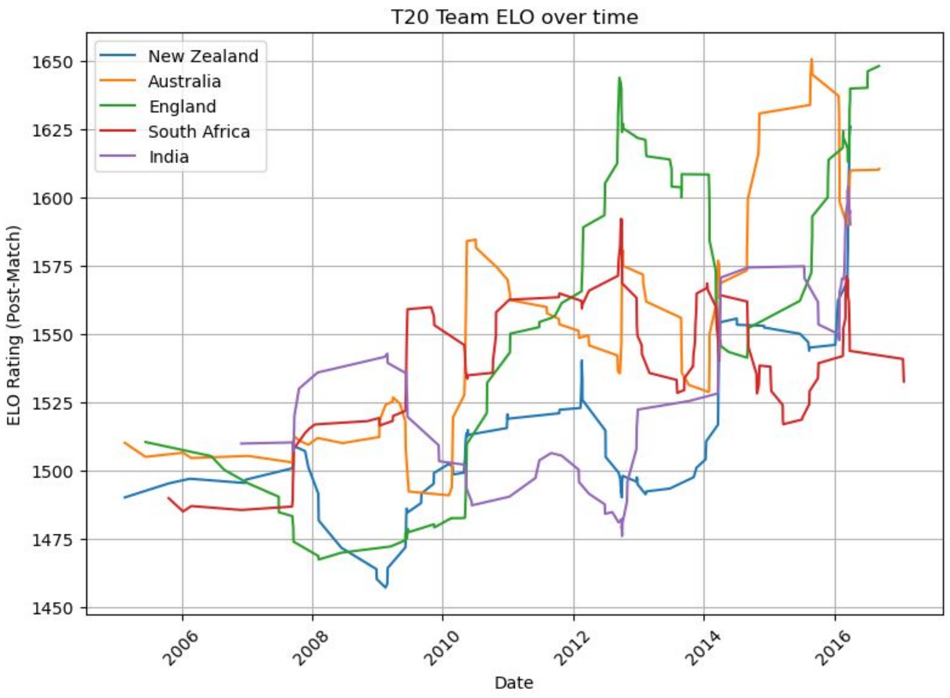
Iterate over every match to get

Player	*Metadata*	*Match Stats*	*Impact Score*

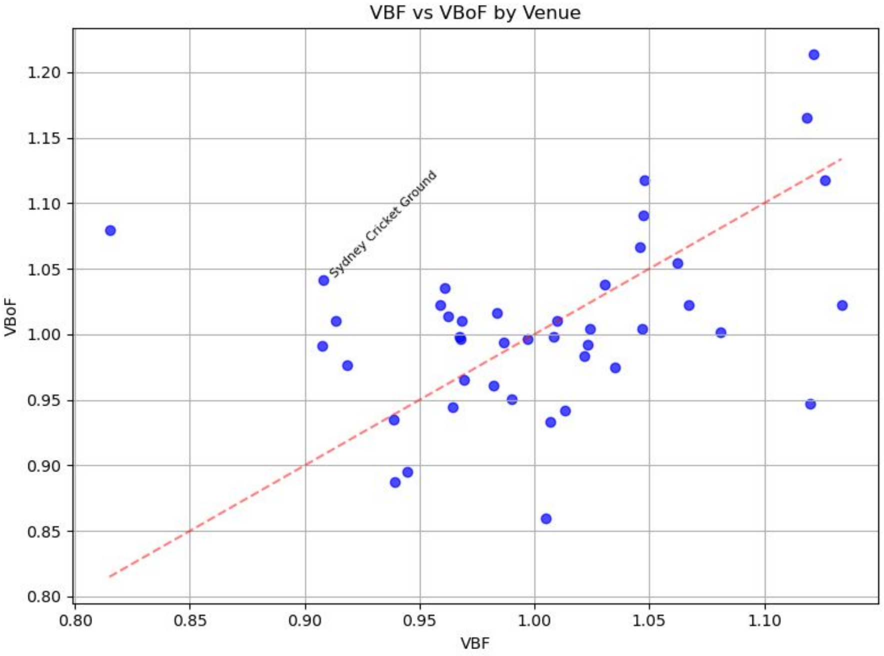
Phase 2: Team ELO Ratings

Goal: Adjust player impact by looking at Opponent Strength

1. Evaluate team strength over time
2. Use the ELO system to convert Wins/Losses to a relative strength
3. Adjust player performances based on oppositions strength



Phase 3: Venue Factors



Goal: Adjust player impact to account for venue difficulty

To account for venue factors I calculated:

- **Venue Batting Factor** (VBF)
- **Venue Bowling Factor** (VBoF)

For each venue, based on how its averages compare to all venue

Higher factor = harder to perform that skill

Phase 4: Combining All Layers

1. **Match Impact**: Calculated per player per match (batting, bowling, fielding, MoM).
2. **Opposition Strength Adjustment**: Scales impact by opposition strength.
3. **Venue Adjustment**: Scales impact by venue difficulty (VBF & VBoF).

$$
\begin{gathered}
WeightedImpactScore = \\
[(Batting \times VBF) + (Bowling \times VBoF) + Fielding + MOM\_Bonus] \\
\times (Opponent\_ELO / Team\_ELO)
\end{gathered}
$$

Result: Venue and Opponent-adjusted Match Impact Scores.

Finishing Touches

Player Role Classification

Main 4 roles:

- Batsman
- Bowler
- Allrounder
- Wicketkeeper

Infer role from career statistics based on the ruleset

- Batsman: Above batting average
- Bowler: High wickets taken
- Allrounder: Both of the above
- Wicketkeeper: Stumpings/Catches

Career Profile

After building match-level records, I aggregated them into **career-level profiles**

The Career Impact Score uses:

- Total career Stats
- Total adjusted impact
- Format-specific weightings

The result is a contextual, fair measure of a player's overall contribution and skill

Section 2: Application Layer

Utility Functions

LLM uses hardcoded utility functions so it never gets the full dataset

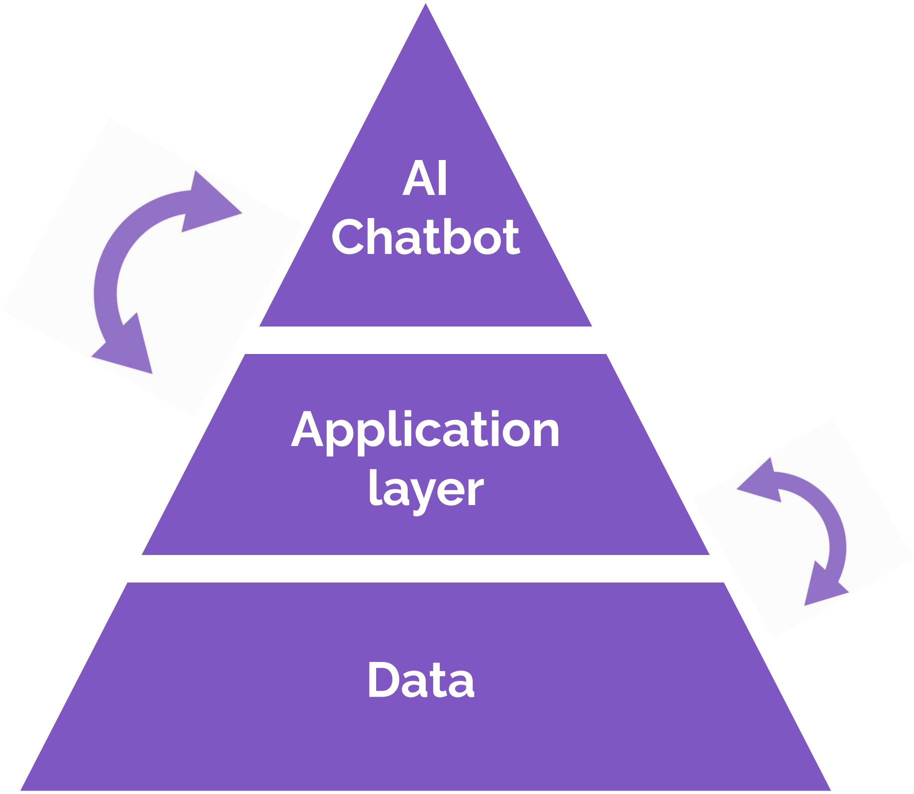
Only require **3 functions**:

1. "*find_player()*" - look up career stats for a player & return all match data
2. "*select_best_XI()*" - select optimal playing XI based on scores
3. "*match_scorecard()"* - Generates full match summary for a given game

For all of these functions, you can pass a time period, country, format, etc

eg. ask for Best ODI XI made up of Indian players

Section 3: Grounding the AI



How the AI Chatbot works

1. **User Query**
   - eg. Who is Joe Root
2. **LLM Interprets Intent**
   - Chooses correct function
3. **Function Call**
   - Utility function returns selected stats from data
4. **LLM Formats the Answer**
   - AI become reasoning layer

Time for a Demo

Limitations

Historical Gaps: Ball-by-ball data only from 2001, missing earlier players and matches.

Scope of Metrics: Only quantifiable statistics are used; qualitative factors like leadership, clutch performance, and captaincy are not captured.

Modeling & Adjustments: Opponent strength (ELO), venue difficulty, and player roles rely on assumptions and simplified rules, which may not fully reflect real-world context.

Grounding Limits: Chatbot can only answer questions within the dataset and predefined functions; it cannot handle queries outside these bounds.

By combining the **data foundation**, **contextual scoring**, and **grounded AI layer,** I have built a chatbot that gives **accurate**, **fact-based** cricket insights **without hallucinating**.

Thank you for listening

Any Questions?