# Cricket 24 – Bowling Strategy Lab

Requirements Specification (JSON-only roster)

## 1. Purpose

Provide a simulation environment that generates realistic, wicket-taking bowling plans in the style of Cricket 24. Bowler roster data is loaded from JSON only. Delivery sequences are produced by code (trap logic + biases), not from CSV/JSON.

## 2. Scope

Users can select a bowler, overs, match phase, and pitch type to generate a unique, tactical over plan. The goal is strategy formation rather than statistical replication.

## 3. Functional Requirements

**3.1 Bowling Logic**

- Deliveries are generated in code; roster JSON provides only bowler info (name, type, strengths).

- Each ball across selected overs must be unique.

- Logic factors: bowler type (pace/spin), strengths, match phase, and pitch type.

- Subtle 'after-touch' variations (e.g., angle wider, late release).

**3.2 Trap & Wicket-Taking Logic**

- Outswing setup → inswing to stumps.

- Hard-length/cross-seam → fuller nip-backer.

- Death: yorkers/wide yorkers with surprise bouncer.

- Spin: tease outside off + flight → arm ball at pads/wood.

- Traps vary by phase and bowler archetype.

**3.3 Terminology (Cricket 24)**

Delivery Types: Standard, Slower, Outswing, Inswing, Cross seam, Off cutter, Leg cutter, Bouncer.

Lengths: Full, Good, Short, Yorker, Back-of-length.

Lines: off stump, 4th stump, 5th stump, outside off, middle, leg stump, at body.

**3.4 Inputs & Outputs**

Inputs: JSON roster; user selections (overs, phase, pitch).

Outputs: Over plan rows (over, type, length, line, purpose). Export: CSV and JSON; open plan in new tab.

## 4. Pitch & Phase Model

- Pitch types: Normal, Green, Dusty, Dry.

- Phase: Powerplay, Middle, Death (with tailored biases and guaranteed patterns).

## 5. Non-Functional Requirements

- Strict type safety using shared unions to avoid drift (e.g., 'Cross seam', lowercase 'middle').

- Deterministic RNG for reproducibility.

- Lightweight, Vercel-friendly front-end.

## 6. JSON Roster Schema

{

"id": "string",

"name": "string",

"country": "string",

"iplTeam": "string",

"formats": ["ODI"|"T20I"|"Test"],

"arm": "R"|"L",

"type": "pace|seam|fast|off-spin|leg-spin|...",

"paceKph": "string",

"strengths": ["outswing","inswing","cutter","bouncer","yorker", "..."],

"strategies": ["free text"],

"isLegend": true|false

}

Note: roster JSON can be an array of the above objects.

## 7. Module Responsibilities (JSON-only)

- types/cricket.ts: shared types.

- lib/jsonLoader.ts: parse and validate JSON roster (replaces csvLoader.ts).

- lib/strategyMap.ts: strengths/phase/pitch weighting helpers.

- lib/deliveryLogic.ts: trap logic and unique delivery generation.

- lib/overLogic.ts: assemble plan for N overs.

- app/page.tsx: UI to load JSON roster, select inputs, display/export.

- app/layout.tsx: app shell.

- lib/random.ts: RNG utilities.

## 8. File Structure (Target)

/src  
 /app  
 /layout.tsx  
 /page.tsx  
 /lib  
 /jsonLoader.ts # NEW: loads JSON roster only  
 /strategyMap.ts  
 /deliveryLogic.ts  
 /overLogic.ts  
 /random.ts  
 /types  
 /cricket.ts  
/public  
 /assets  
 /diagrams/delivery\_generation\_flow.png

## 9. Change Plan

1) Replace CSV loader with JSON loader: create lib/jsonLoader.ts and remove lib/csvLoader.ts usage.

2) Update page.tsx to fetch a JSON roster URL (or accept pasted JSON) and call jsonLoader.

3) Keep delivery generation purely algorithmic in deliveryLogic.ts; no delivery data in JSON.

4) Ensure PitchType uses 'Dry' (not 'Flat') throughout.

5) Maintain string literal unions for DeliveryType/Length/Line; enforce via imports in strategyMap.ts.

6) Optional: add JSON export alongside CSV export for plans.

## 10. Flow Diagram

Delivery generation pipeline for JSON-only configuration:

