

Name: Mohammad Sajeel Hasan Seat Number: EB24210106051

Couse Name: Software Design and Architecture Course Number: SE 454

Teacher Name: Mr. Hussain Saleem

Domain Analysis Reports

**Champions Trophy 2025 Player Stats & Data Analyzer**

## **1. Domain Overview**

The **Champions Trophy 2025 Player Stats & Data Analyzer** is a system designed to collect, process, and visualize cricket statistics for the Champions Trophy 2025.  
It allows **viewers**, **analysts**, and **admins** to interact with **real-time player and team data** through a **Flutter-based mobile application**.

The backend ensures that all data is efficiently collected, cleaned, analyzed, and presented visually through an interactive user interface.  
The system aims to simplify complex statistical data into clear, dynamic dashboards for improved user engagement and analytical decision-making.

## **2. Key Domain Concepts**

| **Concept** | **Description** |
| --- | --- |
| **Player** | Includes attributes such as name, team, matches played, runs, wickets, strike rate, and economy rate. |
| **Team** | Represents a squad participating in the tournament; includes team name, players, performance stats, and rankings. |
| **Match** | Contains information such as match date, teams involved, venue, score, and outcome. |
| **Tournament** | Defines the overall event, including the schedule, match details, and tournament phases. |
| **Analytics** | Refers to processed insights including leaderboards, comparisons, and performance trends. |
| **Dashboard** | The graphical user interface that displays charts, tables, and insights for players and teams. |
| **User** | General viewer who explores and compares stats through the app. |
| **Admin** | Manages data ingestion, cleaning, and transformation processes. |

## **3. Domain Entities and Relationships**

* **Player** belongs to a **Team** and participates in one or more **Matches**.
* **Match** belongs to a **Tournament** and links multiple **Players** and **Teams**.
* **Analytics** is generated from **Player**, **Team**, and **Match** data.
* **Users** interact with the **Dashboard** to view and analyze insights.

## **4. Functional Requirements**

1. Ingest tournament data from APIs or manual input.
2. Clean and transform raw data into structured formats.
3. Generate analytical dashboards displaying stats and trends.
4. Allow users to explore and compare player or team performance.
5. Provide real-time updates and refresh after every match.
6. Enable admins to monitor data accuracy and integrity.
7. Support filtering and searching of specific data points.

## **5. Non-Functional Requirements**

* **Performance:** Must efficiently handle large datasets and display results quickly.
* **Accuracy:** Data must remain reliable and updated in real time.
* **Security:** Access control for admin and analyst operations.
* **Scalability:** Should support future tournaments or additional teams.
* **Responsiveness:** Flutter UI must adapt to multiple device sizes.
* **Availability:** The system should remain accessible with minimal downtime.

## **6. Domain Constraints**

* Data sources (APIs or files) may contain missing or inconsistent records.
* Player and match statistics must be verified from reliable sources.
* Dashboards must refresh automatically after each data cycle (ETL).
* The app must remain performant even with large volumes of data.
* Only authorized users (Admin/Analyst) can alter or upload data.

## **7. Domain Glossary**

| **Term** | **Meaning** |
| --- | --- |
| **ETL** | Extract, Transform, Load — the process of preparing data for analytics. |
| **Dashboard** | A visual display of analytics such as charts, graphs, and leaderboards. |
| **API** | Application Programming Interface — source for fetching live tournament data. |
| **Analytics** | Statistical summaries and comparisons derived from processed data. |
| **Flutter** | The frontend framework used to build the mobile user interface. |
| **Firestore** | Cloud database used for storing structured and processed data. |

## **8. System Architecture Design: Champions Trophy 2025 Player Stats & Data Analyzer**

This section describes the overall architecture and internal design of the system, outlining its **components, layers, and data flow**.

### **8.1 Layered Modular Design**

The system follows a **layered modular architecture** to maintain flexibility, scalability, and reusability across the application.

#### **a. Data Collection Layer**

* Fetches **tournament**, **match**, **player**, and **team data** via APIs, CSV files, or manual uploads.
* Validates and stores raw data temporarily before processing.

#### **b. Data Processing Layer**

* Cleans, validates, and normalizes data (removes duplicates, fixes formats).
* Computes **derived statistics** such as batting averages, strike rates, and economy rates.
* Transforms raw data into structured datasets for analytics.

#### **c. Analytics & Computation Layer**

* Generates analytical insights: leaderboards, comparison charts, and trends.
* Summarizes and aggregates data for each player, team, and tournament stage.

#### **d. Storage Layer**

* Uses a secure database (MongoDB).
* Stores structured, cleaned, and processed datasets.
* Supports **indexing** for faster retrieval and reporting.

#### **e. Presentation Layer (Flutter UI)**

* Displays player and team stats interactively on mobile.
* Allows users to compare players, view team insights, and apply filters.
* Provides charts, graphs, and tables for better visualization.

#### **f. Admin & Monitoring Layer**

* Enables **admins** to upload, clean, and manage data sources.
* Monitors system performance, analytics accuracy, and app usage.

### **8.2 Data Flow**

1. **Input:** Match, team, and player data collected from external sources (APIs or files).
2. **Processing:** Data is cleaned, validated, and transformed into a usable format.
3. **Storage:** Processed data is stored in a MongoDB
4. **Analytics:** System computes metrics and trends for visualization.
5. **Output:** Flutter UI displays updated dashboards with real-time statistics.

### **8.3 Design Approach**

* The system follows an **N-tier architecture**, separating frontend, backend, and database operations.
* Inter-layer communication occurs through **REST APIs** and **data models**.
* The design promotes **modularity**, **scalability**, and **ease of maintenance**.
* Real-time updates are achieved using **Firebase Cloud Functions** and listeners.
* The design ensures **data consistency** and **fast response times** for users.

### **8.4 Key Design Considerations**

| **Aspect** | **Description** |
| --- | --- |
| **Performance** | Ensures efficient data retrieval and quick analytics rendering. |
| **Scalability** | Supports new tournaments, players, or teams without redesign. |
| **Security** | Implements Firebase Authentication for secure access. |
| **Usability** | Provides an intuitive, interactive Flutter UI for all users. |
| **Accuracy** | All statistics are verified and recalculated after every match. |
| **Reliability** | Designed to maintain functionality during API or network issues. |

### **8.5 Tools and Technologies**

| **Component** | **Technology Used** |
| --- | --- |
| **Frontend (App)** | Flutter |
| **Backend (Logic / API)** | Node.js |
| **Database** | MongoBD |
| **Analytics Engine** | Custom Aggregation and Computation Logic |
| **Storage** | Firebase Collections |
| **Deployment** | Firebase Hosting / Cloud Storage |
| **Visualization** | Flutter Charts, Graphs, and Widgets |

# 









