# Software Design Document for Live Cricket Score for Inter/Intra University Tournaments

## Group 5

Syed Abrar — CS22BTECH11058

Mohammed Gufran Ali — CS22BTECH11040

Ashwin Kumar — CE21BTECH11008

Paavaneeswar Reddy — CS22BTECH11014

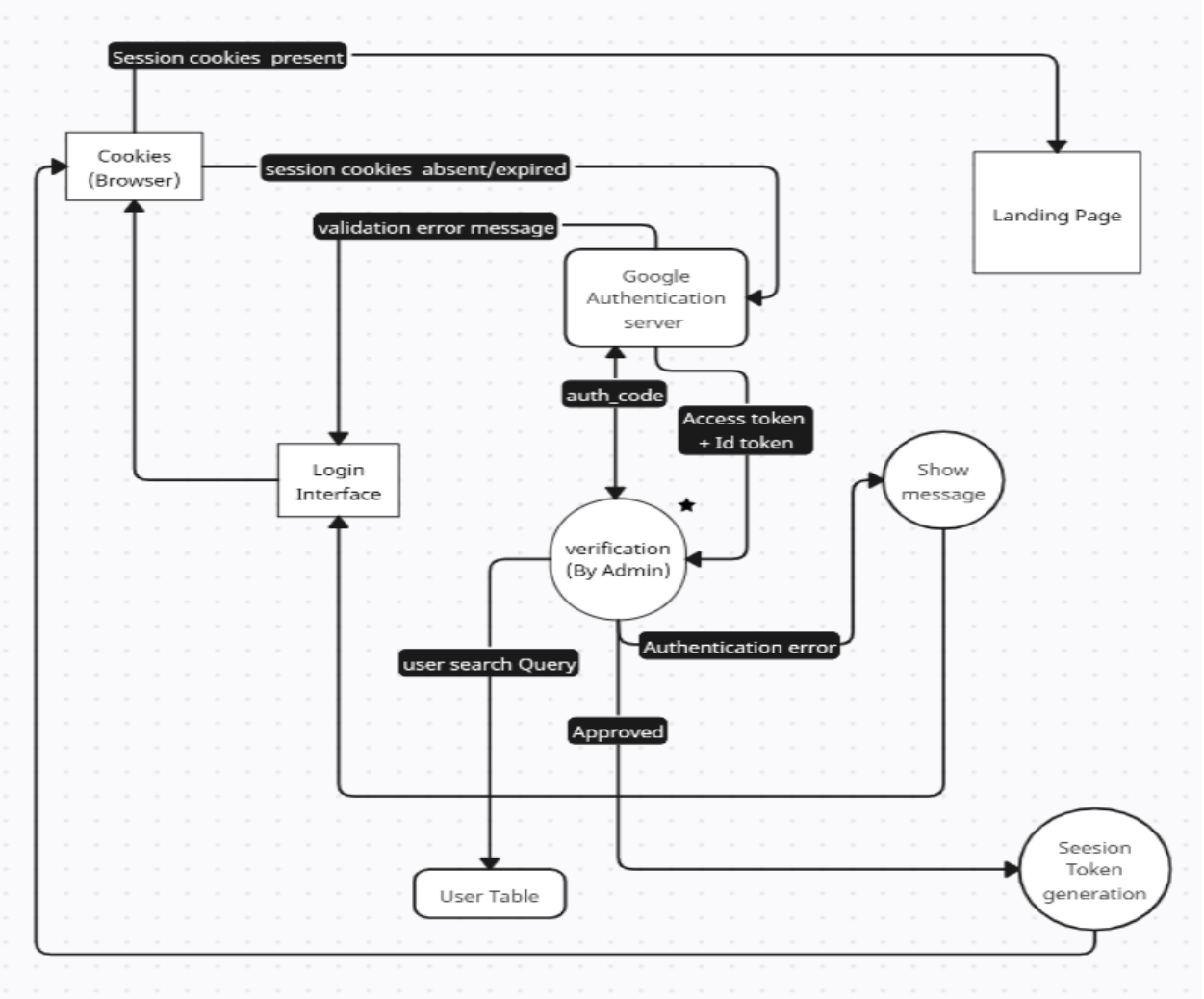
Adil Salfi — CS20BTECH11031

**Overview :**
The Live Cricket Score System serves as a real-time match management platform for inter/intra-university tournaments, facilitating coordination among administrators, match organizers, players, umpires, and fans.

The system's architecture supports live score updates, match details, video streaming, player statistics, and tournament management. This design document presents the system's structure through Data Flow Diagrams (DFD) and Structured charts, detailing first-level factored modules, input, output, and transformation processes.The design analysis section examines module types, cohesion, size, coupling, and complexity, highlighting key modules with high fan-out and fan-in. Additionally, it estimates the total lines of code for an overview of the system's scale.The detailed design specification outlines the final-level factored modules, defining class attributes and methods to ensure a structured and consistent development process.

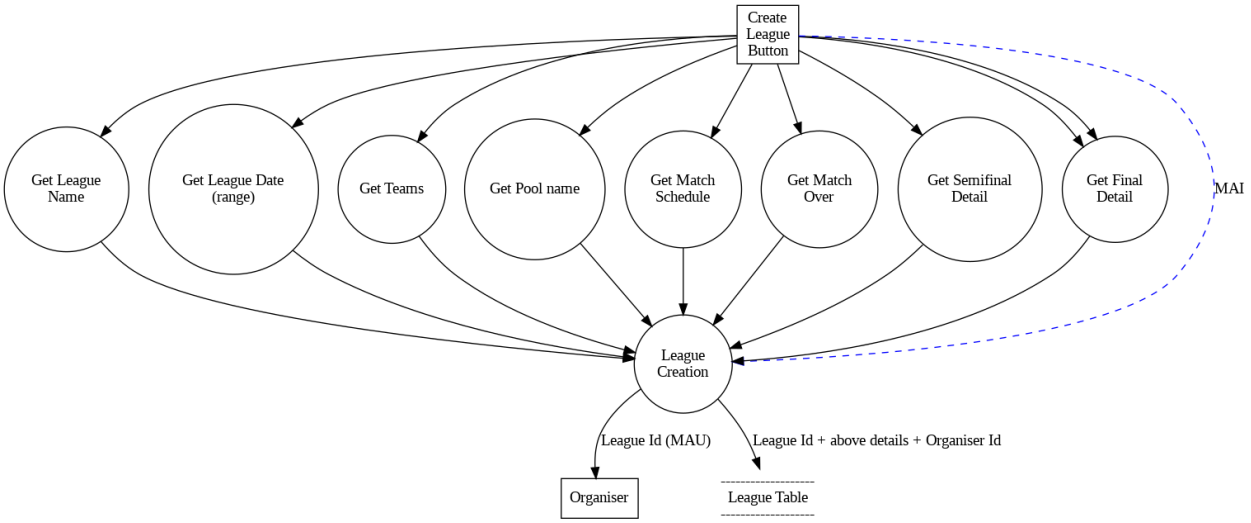
**Data Flow Diagram :**

Login (user/Player/organiser)



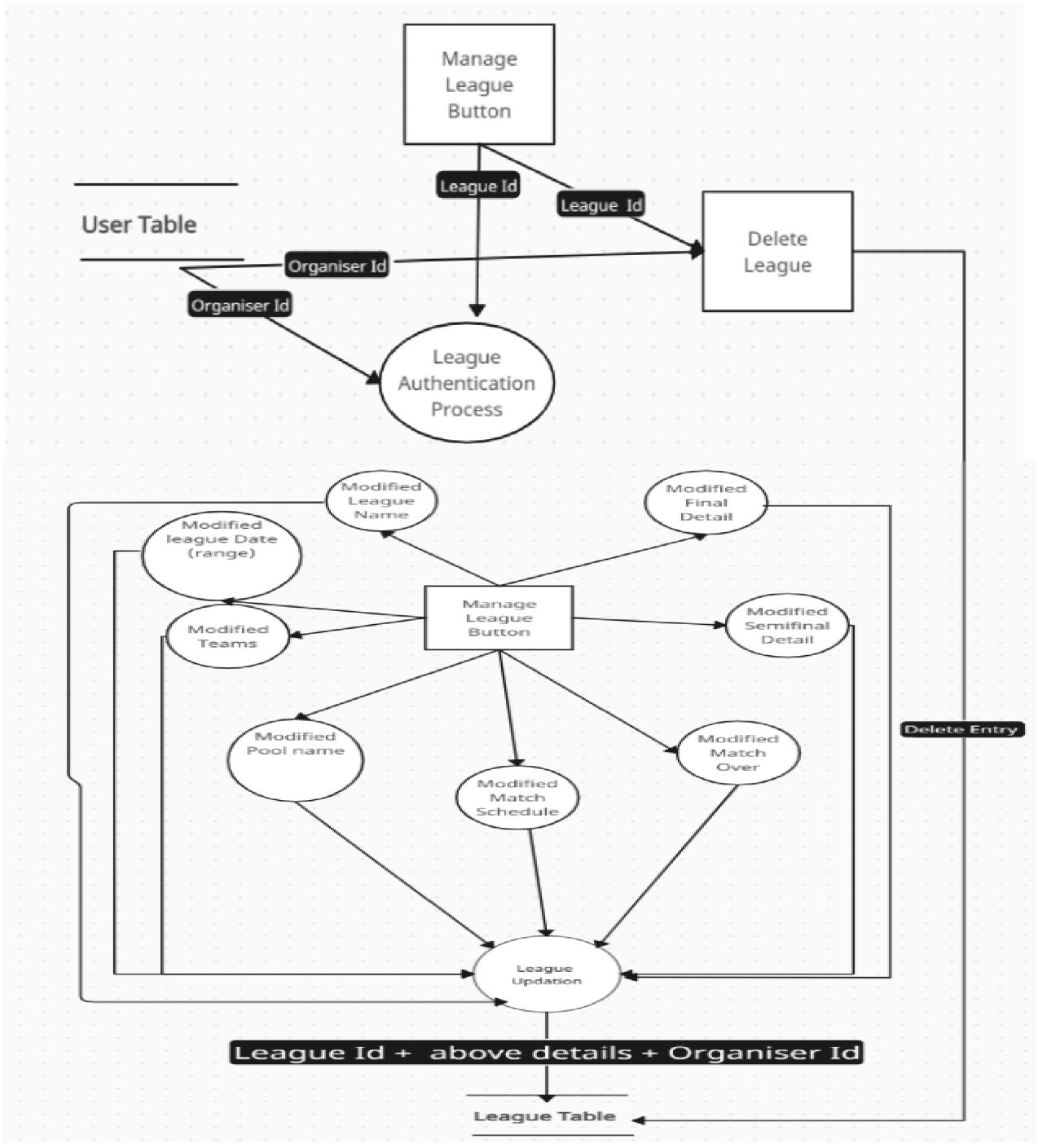

Organiser Process:

Create League :

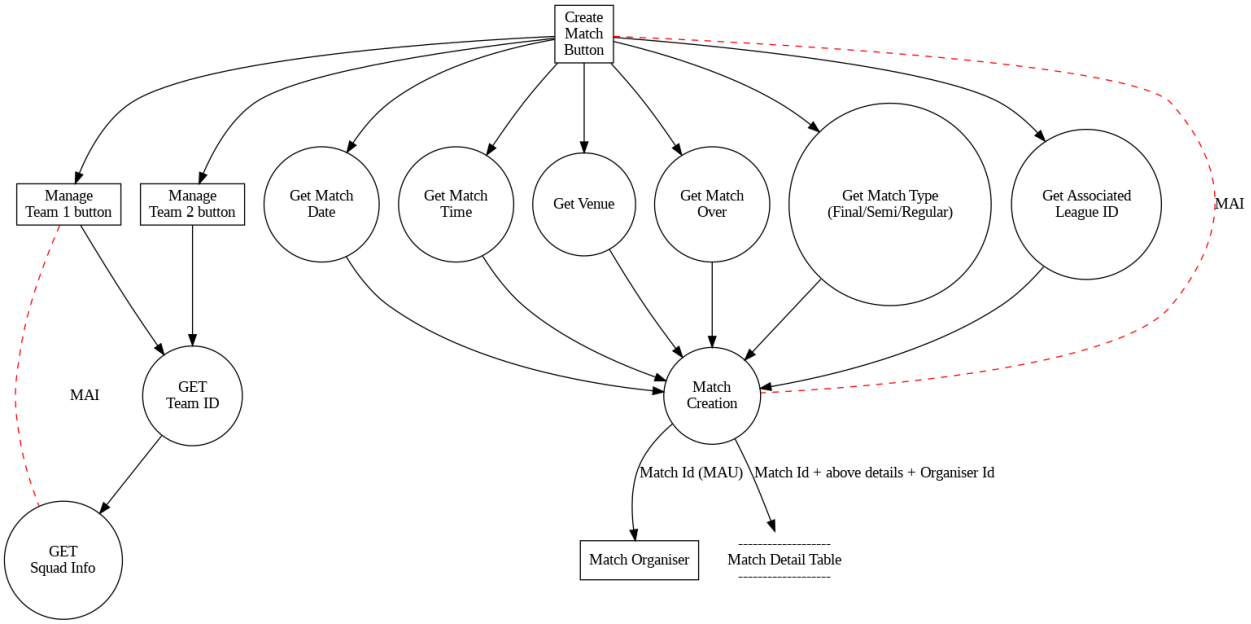



Manage League :

(given two image is combined make google docs if possible )

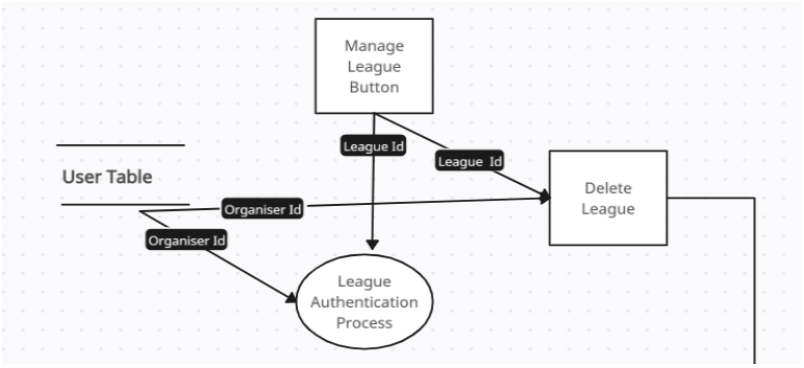


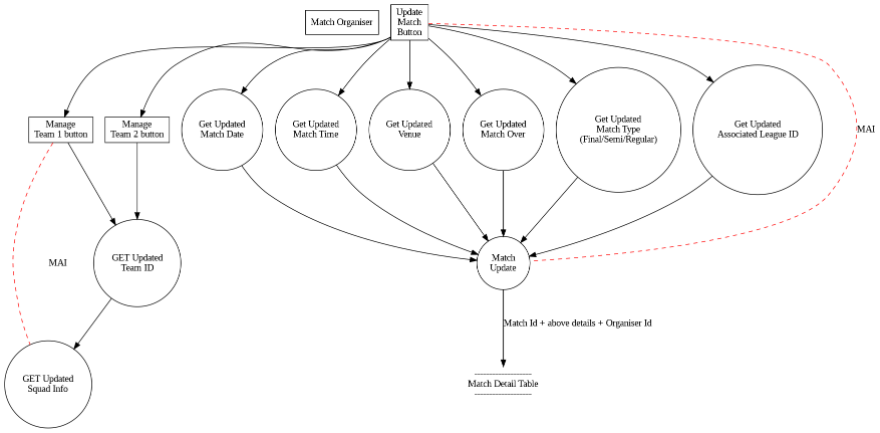
Create Match



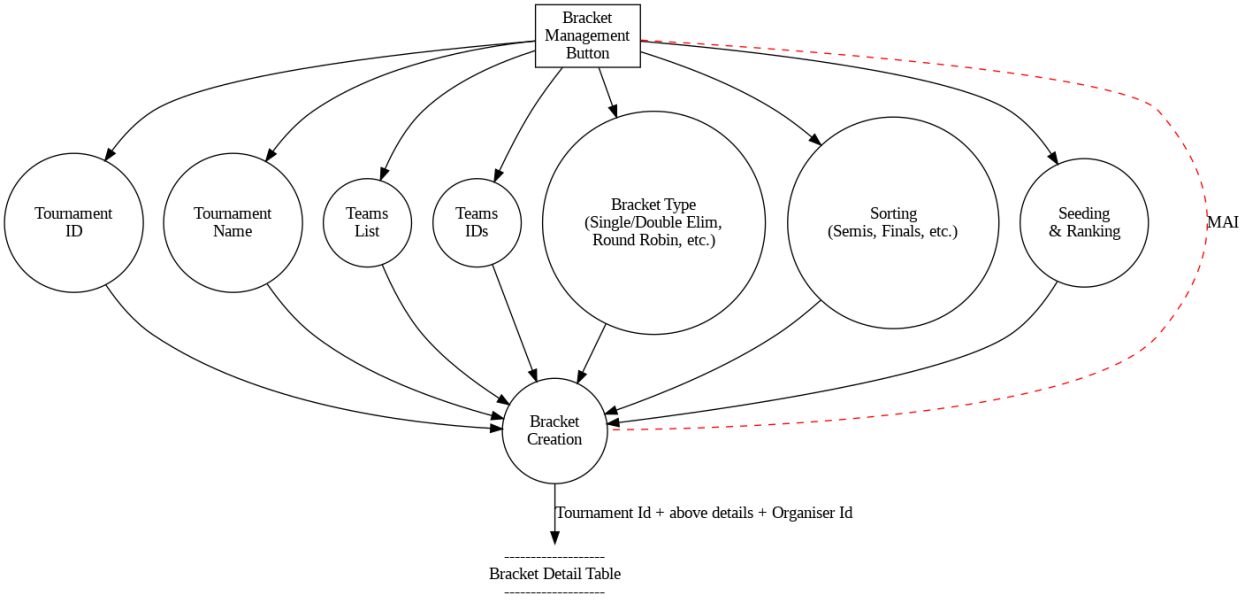

Manage Match :

Just change League to Match in the below one and merge both the picture

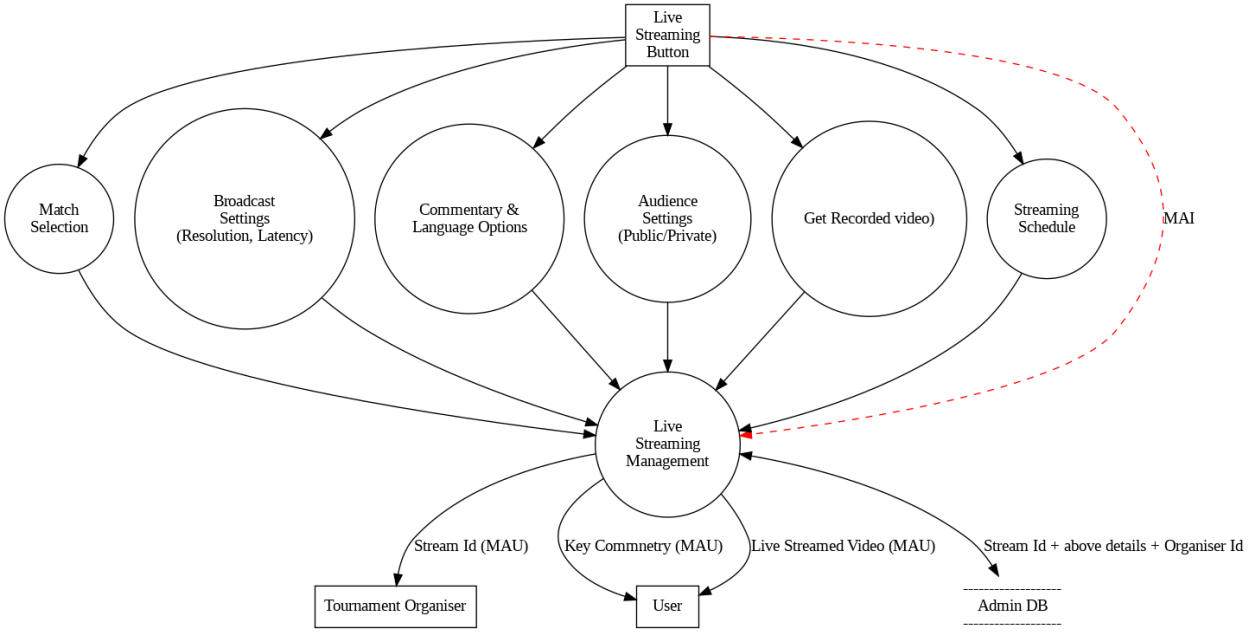


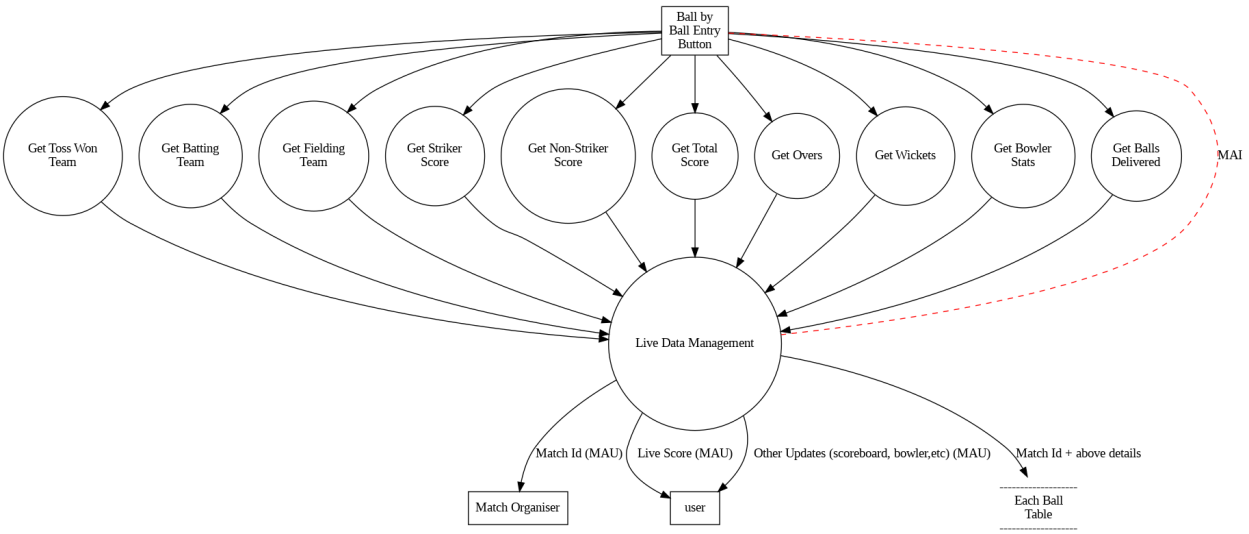

Bracket Management:




Live Streaming Button

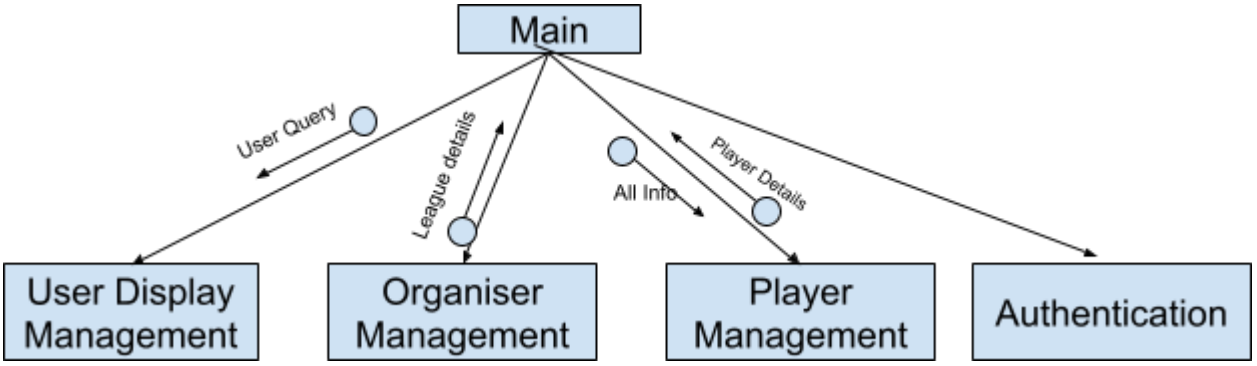



Ball By Ball management :

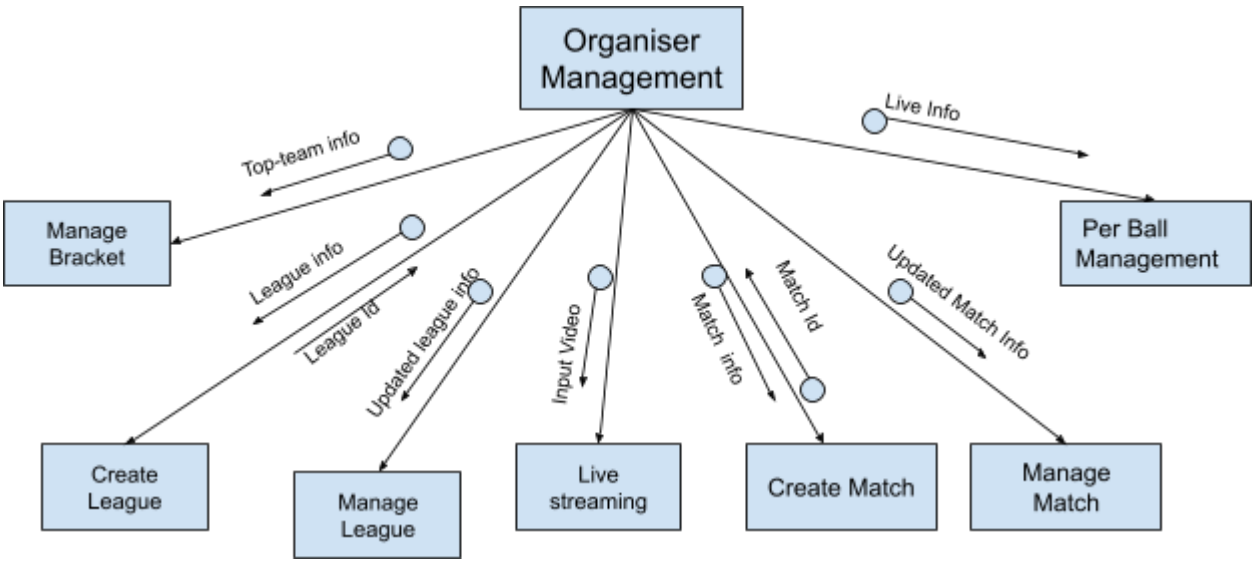



## Structure Charts

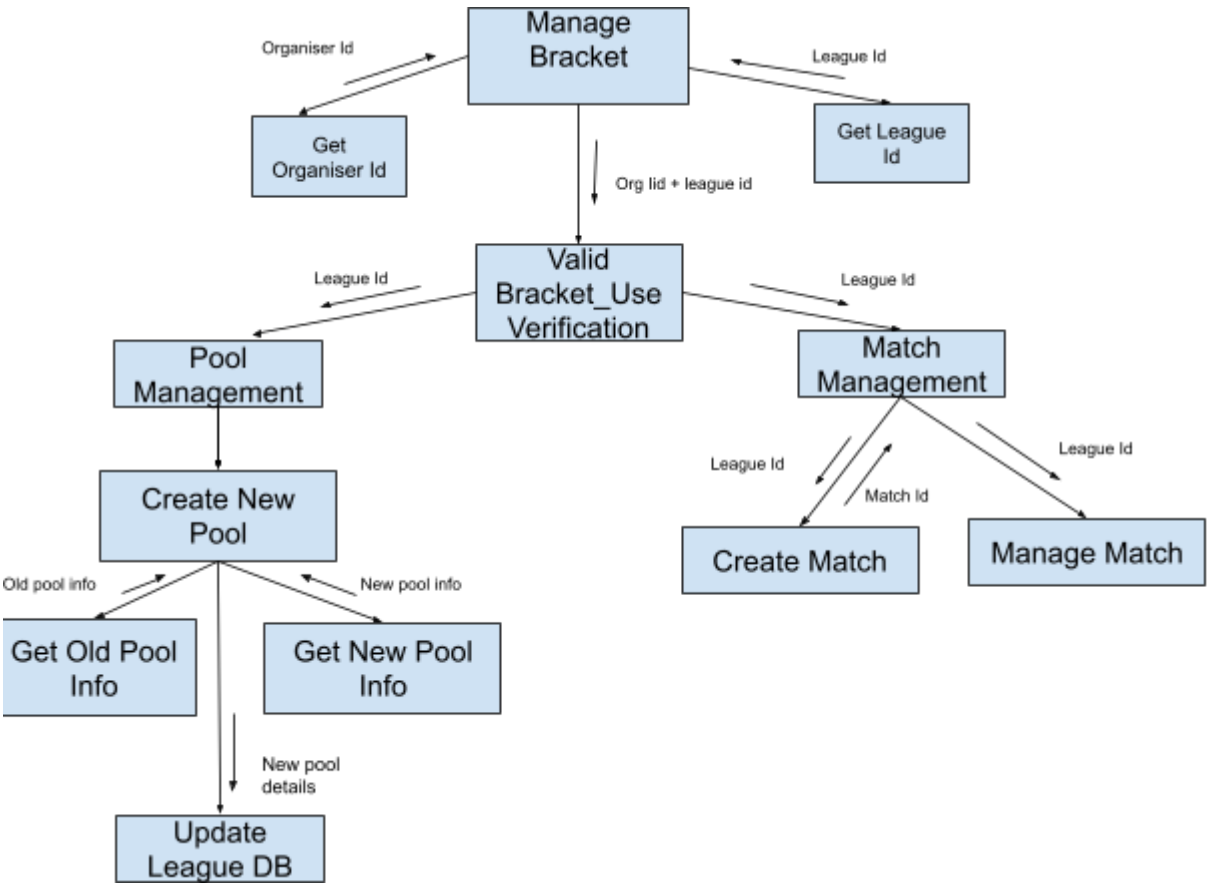
First Level Factoring



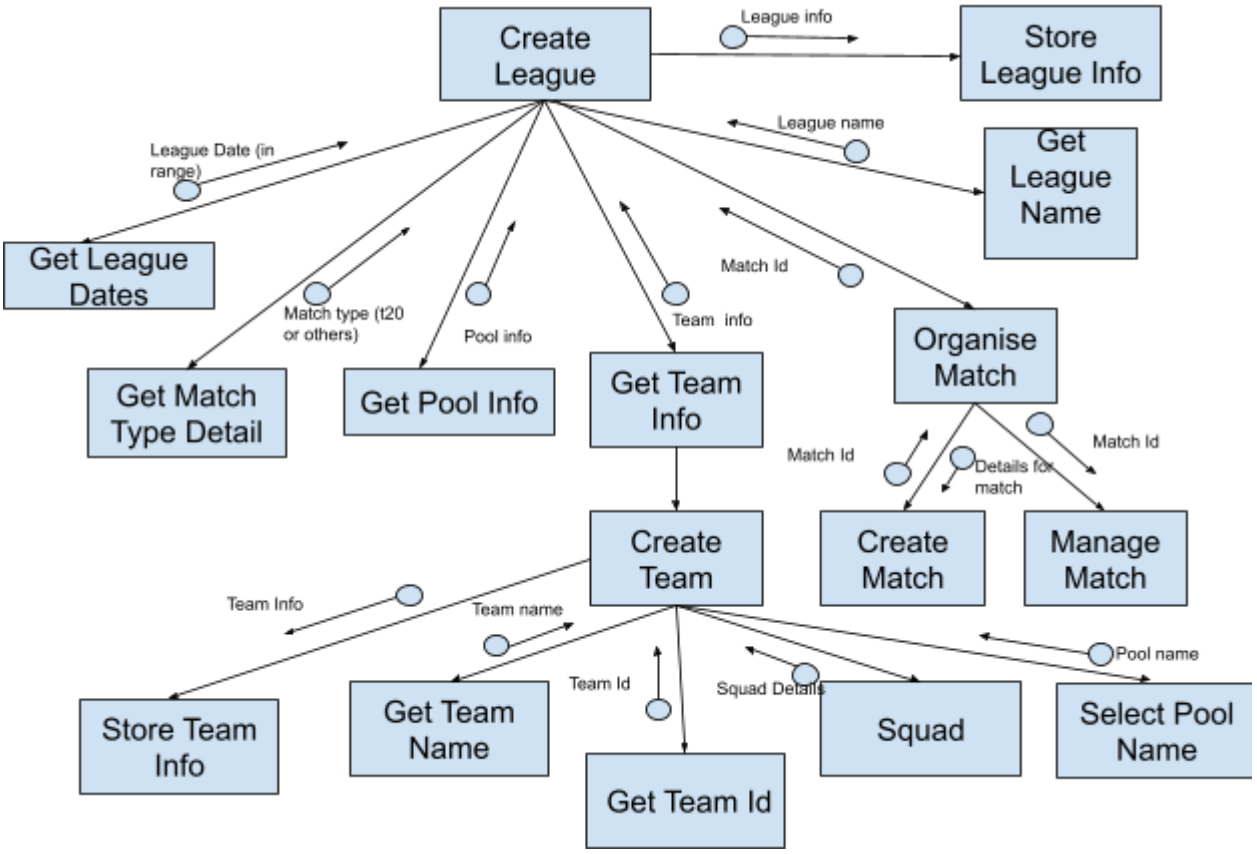

**Organiser Management:-**

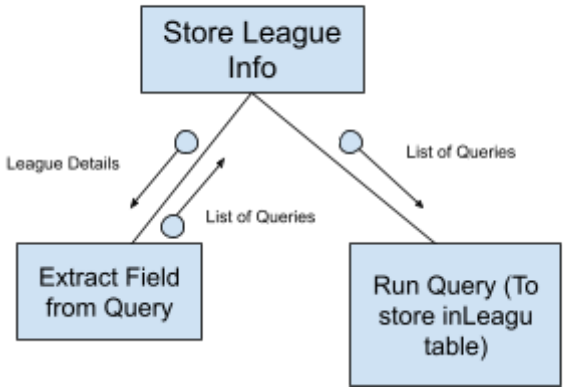



A .Manage Bracket :

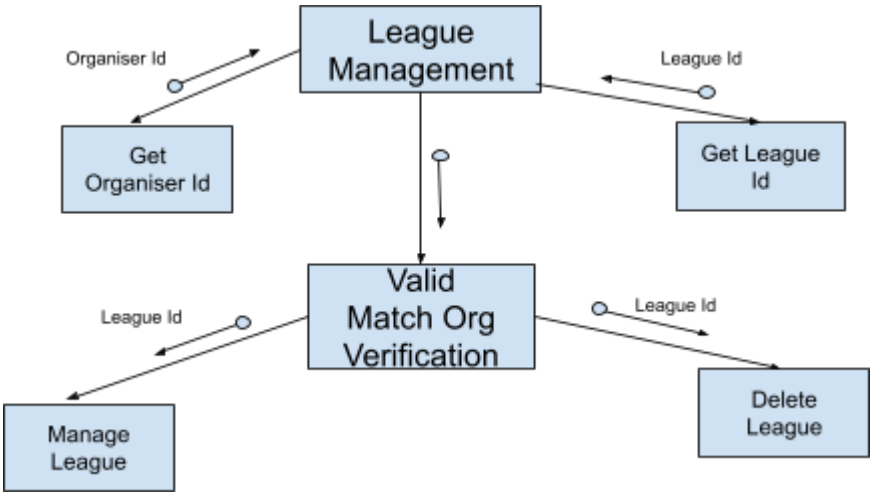


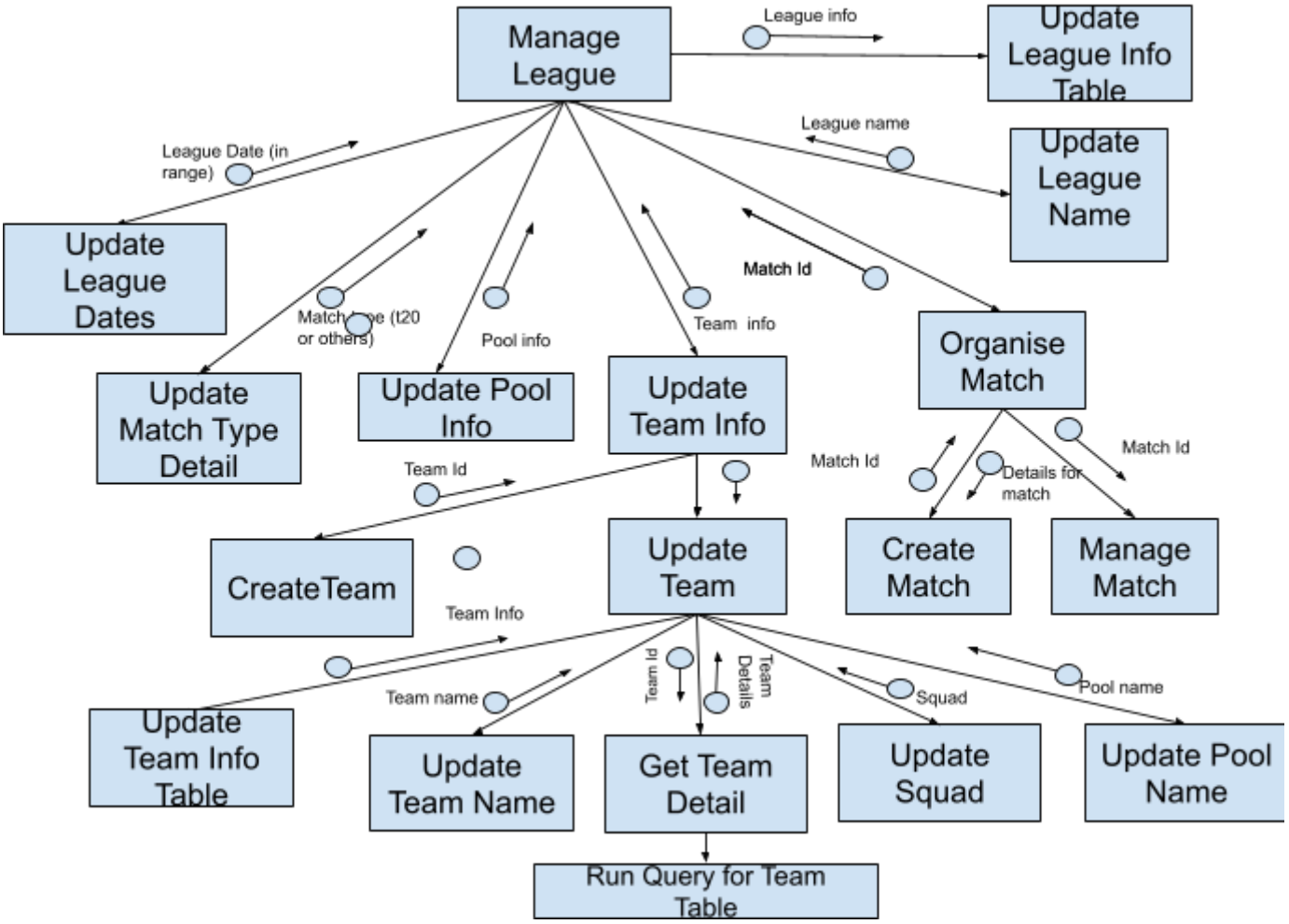

B.Create League :



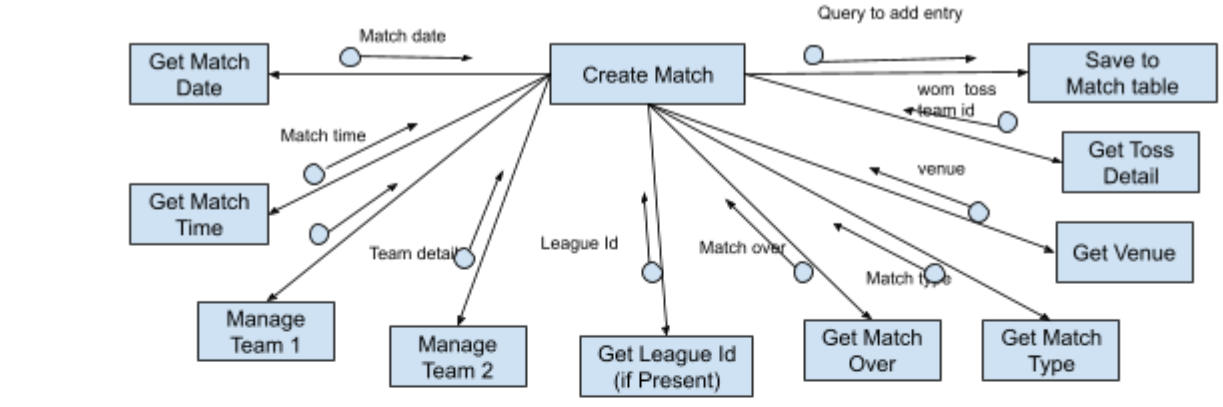



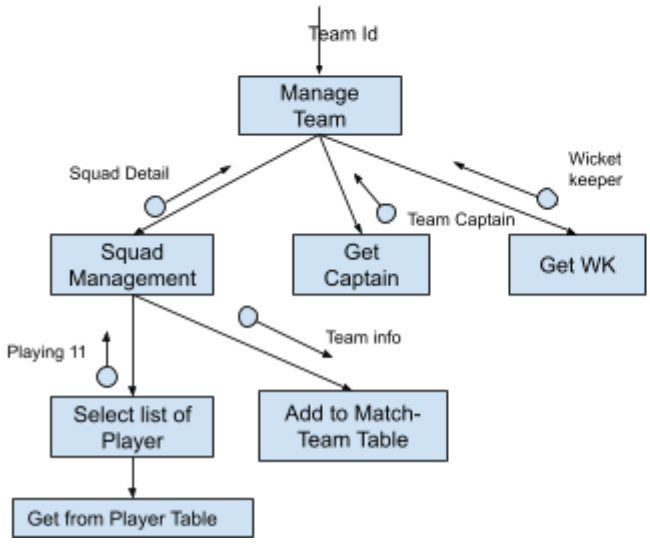

C.Manage League :



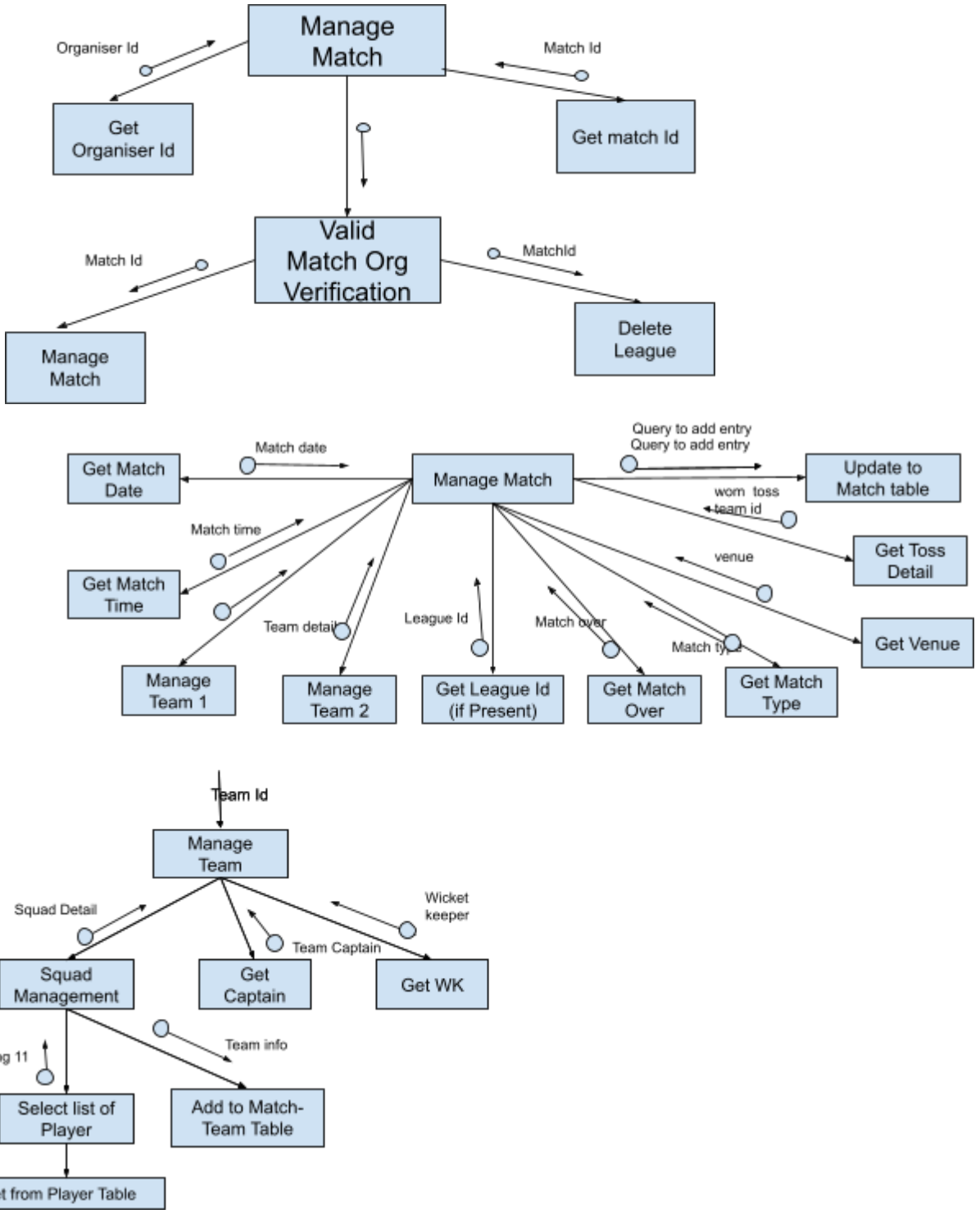



D.Create Match :

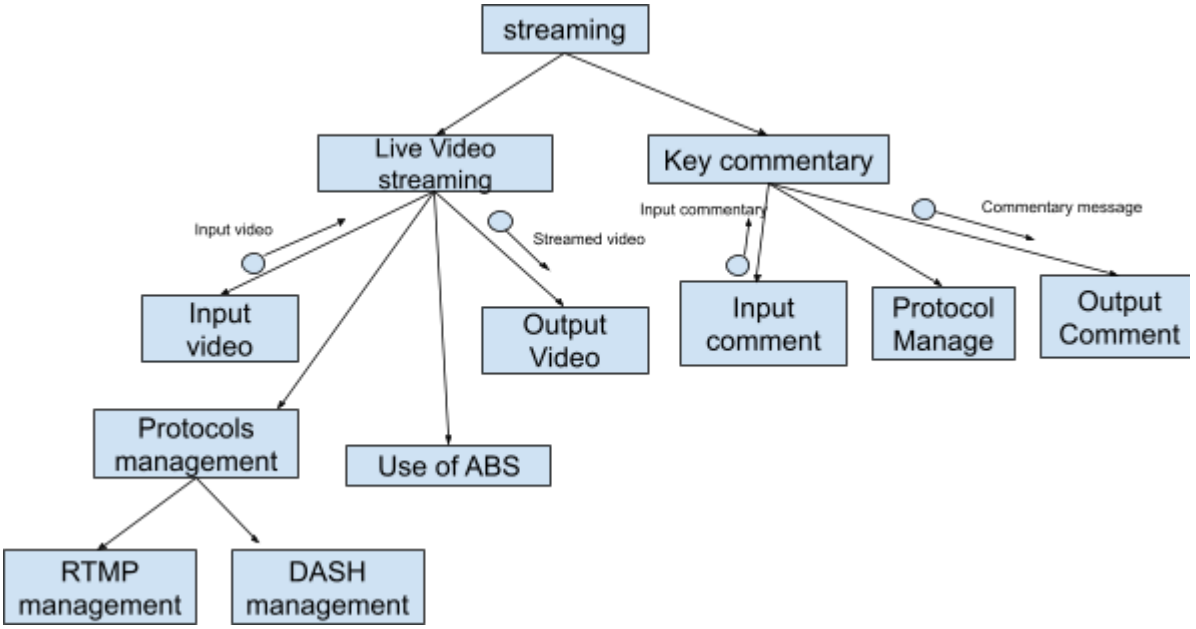


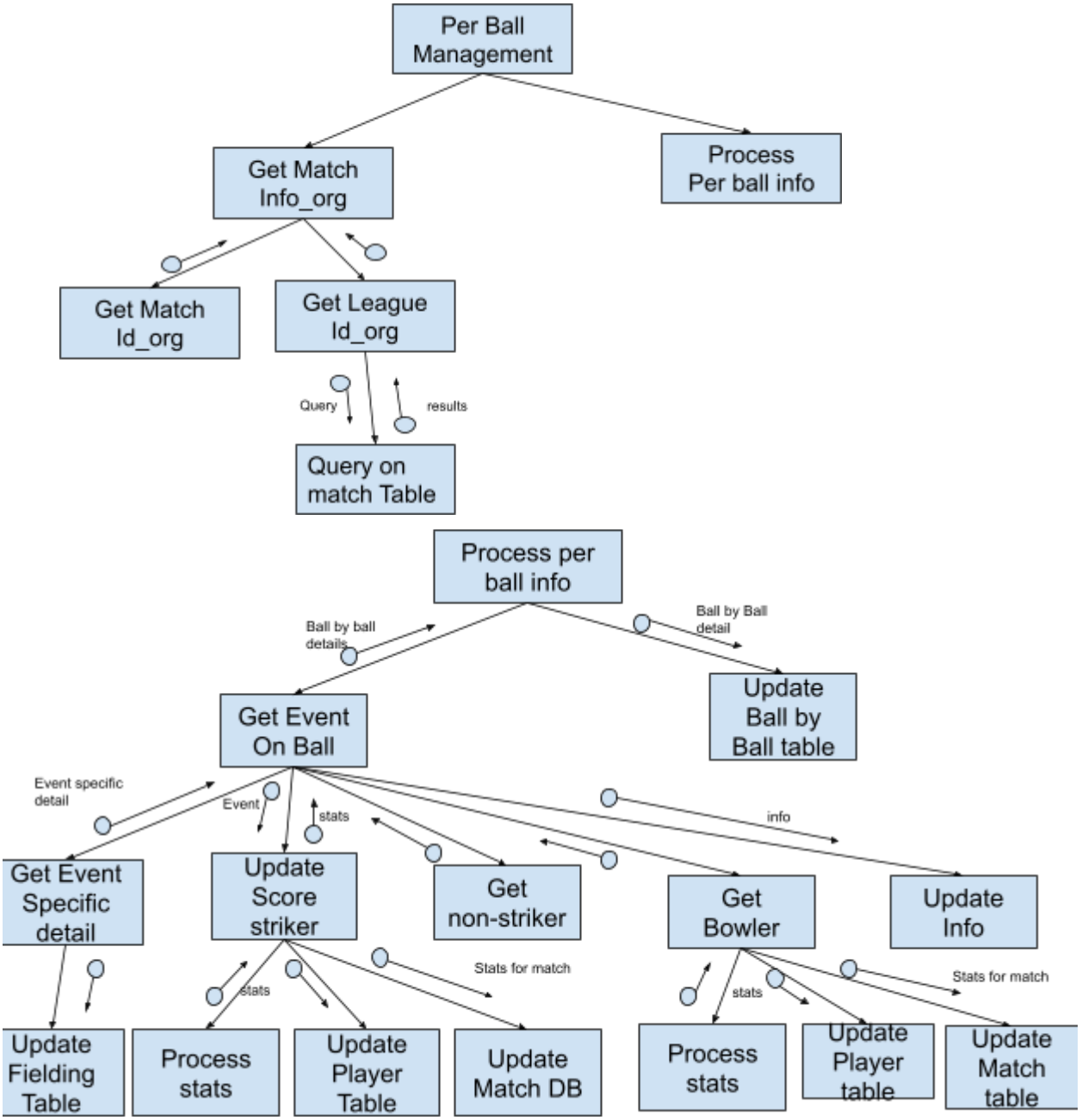
E.Manage Match :




(f) Streaming
Live Stream (Video & Comment) :
RTMP : Real time message Protocol
DASH : Dynamic Adaptive streaming over HTTP
ABS: Adaptive Bitrate streaming

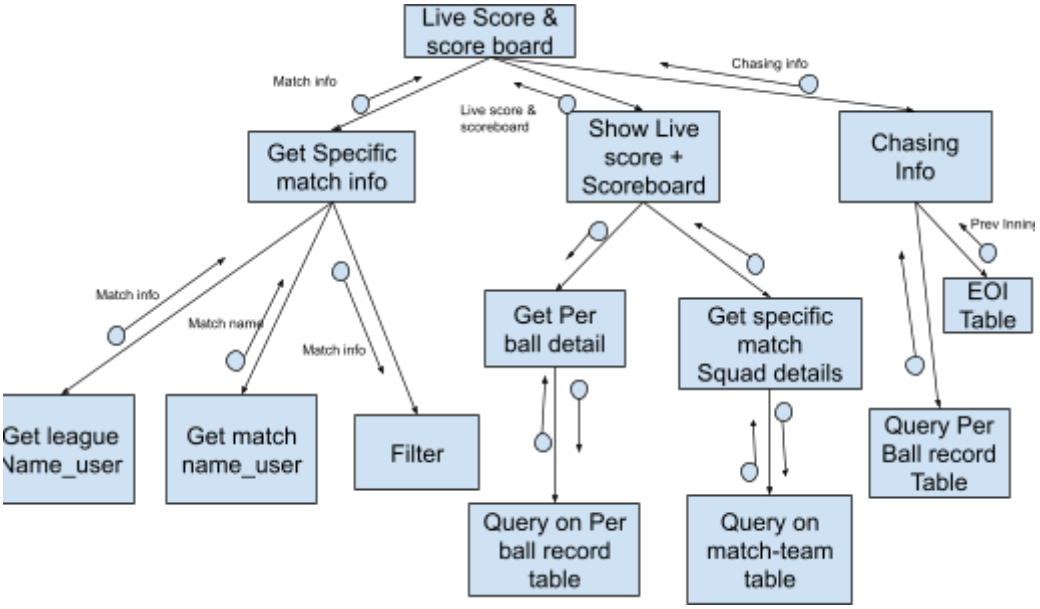



(g) Per Ball Management:

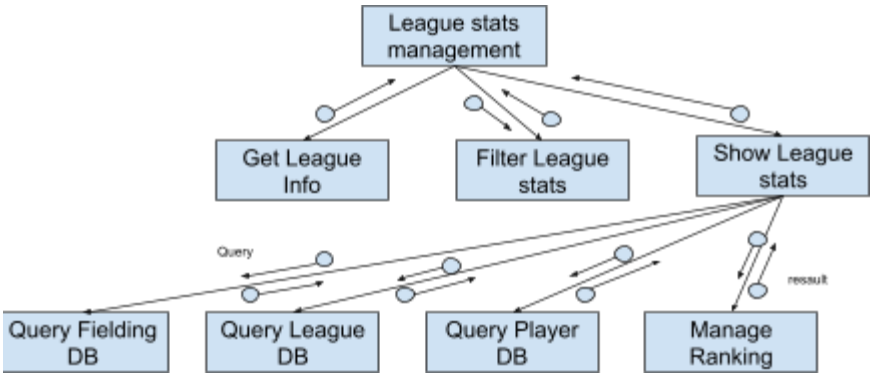



**User Display Management**

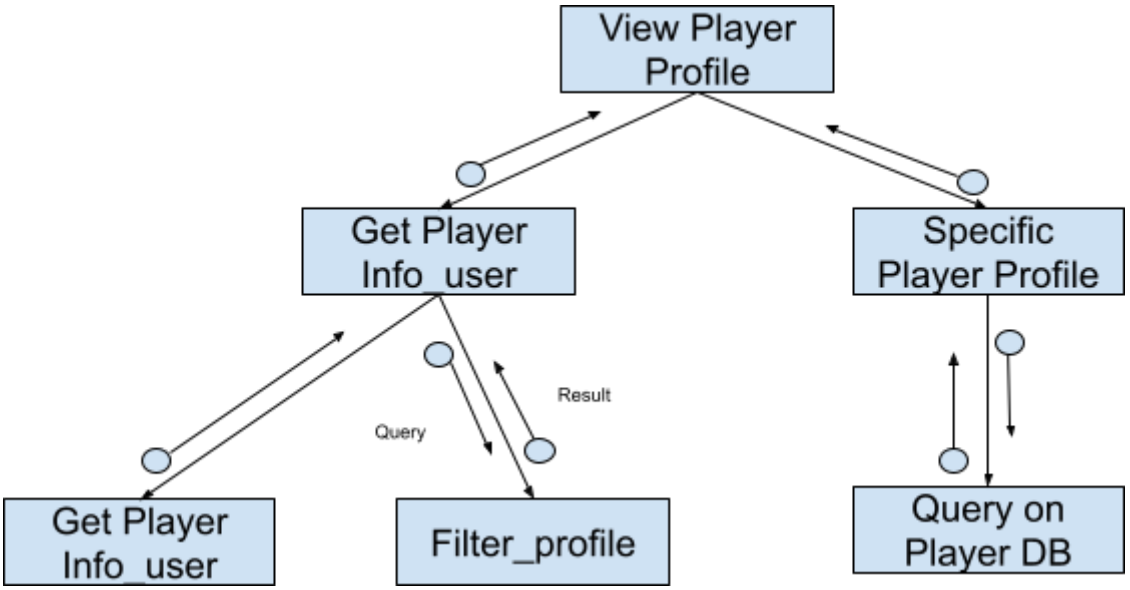
(a)Get Live score:



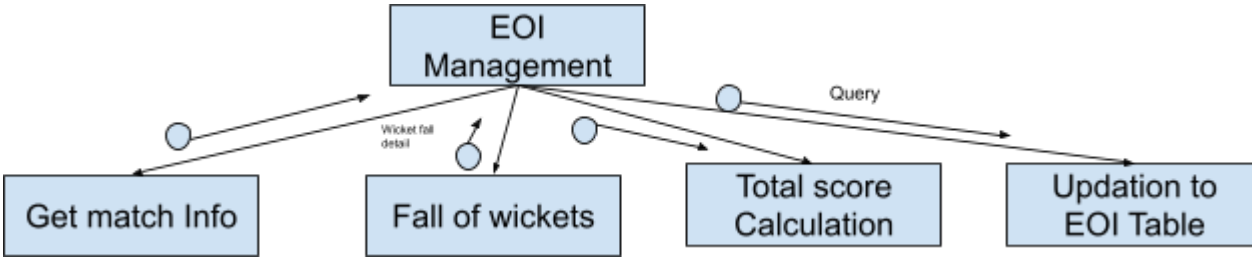

(b)Stats Management:



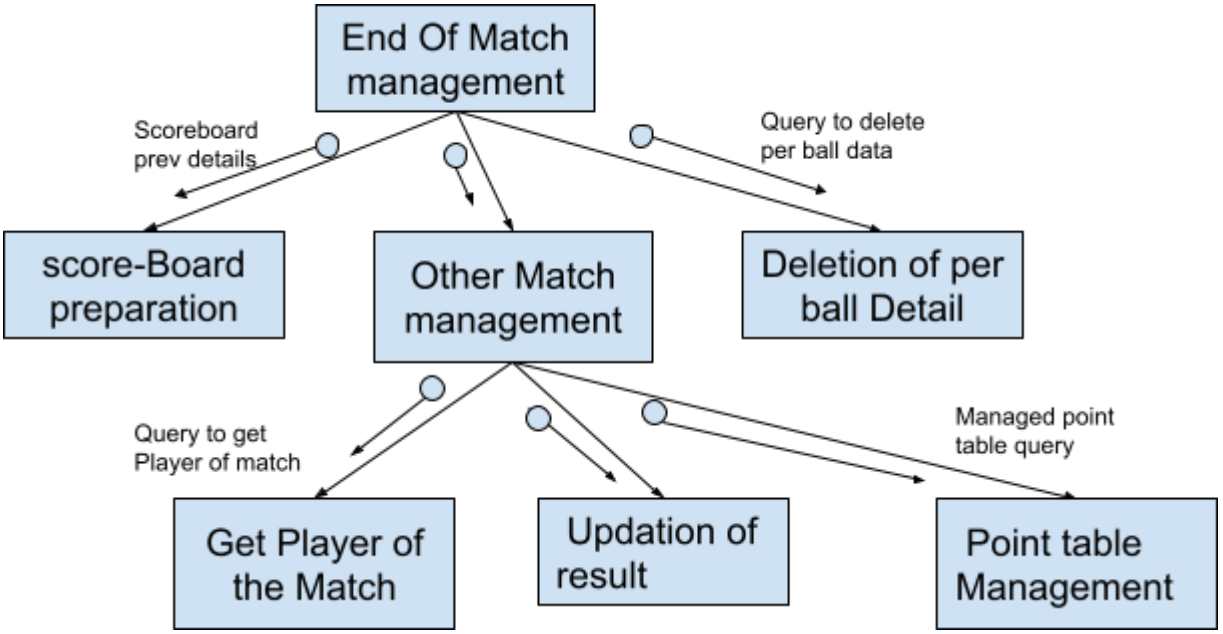

(c) View Profile :




(D) End Of Innings:




(E) End of Match:




**Assumptions: -**

**Note** : some arrow labels are not named because they are obvious from the name of module.

**Note**: We have created modules based upon the user for better flow and in design analysis we have mentioned module type (input /output/transform).

(Many Previous SD docs share by sir follow this pattern )

**Note**: Regarding "Final structure chart showing all the modules" it's getting too large to be accommodated in one image and very low diagram quality is coming, So we have not mentioned that but the pathway we mentioned of modules can be easily traced by naming.

**Note** : Some modules that follow the standard are not mentioned. As our authentication is taking place via Google and clearly mentioned in DFD above so we have not added those modules.

Also For signup standard procedure is followed that can be directly referred.

# Design Analysis

## 4.1 Final Factored Module

| **Module Name** | **Type** | **Cohesion Type** | **Estimated Size (LOC)** |
|---|---|---|---|
| Main | Coordinate | - | 200 |
| Login (Google Auth) | Input | Functional | 50 |
| Signup | Input | Functional | 70 |
| Logout | Output | Functional | 30 |
| Organiser Management | Coordinate | Functional | 200 |
| Manage Bracket | Transform | Sequential | 100 |
| Get Organiser ID | Input | Functional | 10 |
| Get League ID | Input | Functional | 10 |

*Continued on next page*

*Continued from previous page*

| **Module Name** | **Type** | **Cohesion Type** | **Estimated Size (LOC)** |
|---|---|---|---|
| Valid Bracket Use Verification | Transform | Functional | 50 |
| Pool Management | Composite | Communication | 150 |
| Create New Pool | Input | Functional | 60 |
| Get Old Pool Info | Input | Functional | 40 |
| Update League DB | Transform | Sequential | 40 |
| Get New Pool Info | Input | Functional | 40 |
| Match Management | Composite | Functional | 250 |
| Create Match | Input | Functional | 80 |
| Manage Match | Transform | Communication | 70 |
| League Management | Composite | Functional | 350 |
| Valid Match Org Verification | Transform | Functional | 50 |
| Manage League | Composite | Functional | 150 |
| Delete League | Transform | Temporal | 50 |
| Update League Dates | Transform | Functional | 30 |
| Update Match Type Detail | Transform | Communication | 30 |
| Update Pool Info | Transform | Communication | 30 |
| Get Team Info | Output | Communication | 60 |
| Update Team Info | Transform | Fucntional | 60 |
| Update League Info | Transform | Functional | 60 |
| Extract Field from Query | Transform | Functional | 40 |
| Run Query (To store in DB) | Transform | Functional | 50 |
| Store in League Table | Output | Functional | 40 |
| Store in Team Info Table | Output | Functional | 40 |
| Team Management | Composite | Functional | 300 |
| Update Team | Composite | Functional | 120 |
| Update Team Name | Transform | Functional | 50 |
| Get Team Detail | Input | Functional | 50 |
| Run Query for Team Table | Transform | Functional | 50 |
| Update Squad | Transform | Communication | 50 |
| Create Team | Input | Functional | 80 |
| Store Team Info | Output | Functional | 50 |
| Get Team Name | Input | Functional | 10 |
| Get Team ID | Input | Functional | 40 |
| Squad Management | Composite | Communication | 100 |
| Select List of Players | Input | Functional | 50 |
| Get from Player Table | Input | Functional | 40 |
| Add to Match-Team Table | Output | Functional | 40 |
| Get Captain | Input | Functional | 10 |
| Get WK (Wicketkeeper) | Input | Functional | 10 |
| Per Ball Management | Composite | Communication | 150 |
| Create Match | Input | Functional | 60 |
| Get Match Date | Input | Functional | 10 |
| Get Match Time | Input | Functional | 10 |
| Manage Team 1 | Transform | Communication | 50 |
| Manage Team 2 | Transform | Communication | 50 |
| Get League Id (if Present) | Input | Functional | 10 |

*Continued on next page*

*Continued from previous page*

| **Module Name** | **Type** | **Cohesion Type** | **Estimated Size (LOC)** |
|---|---|---|---|
| Get Match Over | Input | Functional | 10 |
| Get Match Type | Input | Functional | 10 |
| Get Venue | Input | Functional | 40 |
| Get Toss Detail | Input | Functional | 10 |
| Save to Match Table (Query to add entry) | Transform | Functional | 50 |
| Streaming Management | Composite | Communication | 300 |
| Live Video Streaming | Output | Functional | 100 |
| Protocols Management | Transform | Functional | 70 |
| RTMP Management | Transform | Functional | 50 |
| DASH Management | Transform | Functional | 50 |
| Use of ABS | Transform | Functional | 50 |
| Output Video | Output | Functional | 50 |
| Key Commentary | Output | Functional | 70 |
| Live Score and Scoreboard | Composite | Communication | 400 |
| Get Specific Match Info | Composite | Functional | 200 |
| Get League Name (User) | input | Functional | 100 |
| Get Match Name (User) | input | Functional | 100 |
| Filter | Transform | Functional | 100 |
| Show Live Score + Scoreboard | Composite | Communication | 250 |
| Get Per Ball Detail | input | Functional | 120 |
| Query on Per Ball Record Table | Transform | Functional | 100 |
| Get Specific Match Squad Details | input | Functional | 120 |
| Query on Match-Team Table | Transform | Functional | 100 |
| Chasing Info | Composite | Communication | 200 |
| EOI Table | Output | Functional | 100 |
| Query Per Ball Record Table | Transform | Functional | 100 |
| League Stats Management | Composite | Functional | 500 |
| Get League Info | Output | Functional | 150 |
| Query Fielding DB | Output | Functional | 100 |
| Query League DB | Output | Functional | 100 |
| Query Player DB | Output | Functional | 100 |
| Filter League Stats | Transform | Functional | 120 |
| Show League Stats | Output | Functional | 150 |
| Manage Ranking | Transform | Functional | 120 |
| Player Profile Management | Composite | Functional | 200 |
| View Player Profile | Output | Communication | 100 |
| Get Player Info_user | Input | Functional | 50 |
| Filter_profile | Transform | Functional | 50 |
| Specific Player Profile | Output | Functional | 50 |
| EOI Management | Composite | Functional | 150 |
| Get Match Info | Input | Functional | 50 |
| Fall of Wickets | Output | Functional | 50 |
| Total Score Calculation | Transform | Functional | 50 |
| Updation to EOI Table | Output | Functional | 50 |
| End of Match Management | Composite | Functional | 200 |

*Continued on next page*

*Continued from previous page*

| Module Name | Type | Cohesion Type | Estimated Size (LOC) |
|---|---|---|---|
| Score-Board Preparation | Output | Functional | 70 |
| Other Match Management | Composite | Communication | 100 |
| Get Player of the Match | Input | Functional | 50 |
| Updation of Result | Output | Functional | 50 |
| Point Table Management | Output | Functional | 50 |
| Deletion of Per Ball Detail | Transform | Temporal | 80 |

# Cohesion and Coupling

**Main Coordinate**

- **Cohesion:** Coordinate This module coordinates the overall system workflow, ensuring that all other modules work together seamlessly. It acts as the central hub for managing interactions between different parts of the system.
- **Coupling:** Weak It interacts with other modules through well-defined interfaces, minimizing dependencies and ensuring modularity.

**Login (Google Auth)**

- **Cohesion:** Input Functional This module handles user authentication using Google Auth. It focuses solely on verifying user credentials and granting access.
- **Coupling:** Loosely coupled It depends on the Google Auth API but has minimal interaction with other modules, making it independent and easy to maintain.

**Signup**

- **Cohesion:** Input Functional This module manages user registration by collecting and storing user details in the database. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It interacts only with the database and does not depend on other modules, ensuring minimal dependencies.

**Logout**

- **Cohesion:** Output Functional This module handles user logout by terminating the user session and clearing relevant data. It performs a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to manage.

**Organiser Management**

- **Cohesion:** Coordinate Functional This module manages organiser-related operations, such as creating, updating, and deleting organisers. It ensures smooth coordination of organiser data.
- **Coupling:** Moderate It interacts with multiple modules for organiser data, creating some dependencies but maintaining a clear structure.

**Manage Bracket**

- **Cohesion:** Transform Sequential This module processes and manages bracket data in a sequential manner. It ensures that bracket-related operations are executed in a specific order.
- **Coupling:** Moderate It depends on input from other modules for bracket data, creating some dependencies but maintaining a logical flow.

**Get Organiser ID**

- **Cohesion:** Input Functional This module retrieves the organiser ID from the database. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Get League ID**

- **Cohesion:** Input Functional This module retrieves the league ID from the database. It focuses on a single, specific task.

- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Valid Bracket Use**

- **Cohesion:** Verification Transform This module validates the usage of brackets by checking their correctness and relevance. It ensures that brackets are used appropriately.
- **Coupling:** Moderate It depends on bracket and league data from other modules, creating some dependencies but maintaining a clear purpose.

**Pool Management**

- **Cohesion:** Composite Communication This module manages pool-related operations, such as creating, updating, and retrieving pool data. It combines multiple tasks related to pools.
- **Coupling:** High It interacts with multiple modules for pool data, creating significant dependencies but ensuring comprehensive functionality.

**Create New Pool**

- **Cohesion:** Input Functional This module creates a new pool by collecting and storing pool details in the database. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Get Old Pool Info**

- **Cohesion:** Input Functional This module retrieves old pool information from the database. It focuses on a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Update League DB**

- **Cohesion:** Transform Sequential This module updates the league database in a sequential manner. It ensures that updates are executed in a specific order.
- **Coupling:** Moderate It depends on league data from other modules, creating some dependencies but maintaining a logical flow.

**Get New Pool Info**

- **Cohesion:** Input Functional This module retrieves new pool information from the database. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Match Management**

- **Cohesion:** Composite Functional This module manages match-related operations, such as creating, updating, and retrieving match data. It combines multiple tasks related to matches.
- **Coupling:** High It interacts with multiple modules for match data, creating significant dependencies but ensuring comprehensive functionality.

**Create Match**

- **Cohesion:** Input Functional This module creates a new match by collecting and storing match details in the database. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Manage Match**

- **Cohesion:** Transform Communication This module manages match data and communication between teams and organisers. It ensures smooth coordination of match-related activities.
- **Coupling:** Moderate It depends on match and team data from other modules, creating some dependencies but maintaining a clear purpose.

**League Management**

- **Cohesion:** Composite Functional This module manages league-related operations, such as creating, updating, and retrieving league data. It combines multiple tasks related to leagues.

- **Coupling:** High It interacts with multiple modules for league data, creating significant dependencies but ensuring comprehensive functionality.

**Valid Match Org**

- **Cohesion:** Verification Transform This module validates the match organiser by checking their credentials and permissions. It ensures that only authorised organisers can manage matches.
- **Coupling:** Moderate It depends on organiser and match data from other modules, creating some dependencies but maintaining a clear purpose.

**Manage League**

- **Cohesion:** Composite Functional This module manages league operations, such as updating league details and managing teams. It combines multiple tasks related to leagues.
- **Coupling:** High It interacts with multiple modules for league data, creating significant dependencies but ensuring comprehensive functionality.

**Delete League**

- **Cohesion:** Transform Temporal This module deletes league data from the database. It ensures that deletions are executed at the appropriate time.
- **Coupling:** Moderate It depends on league data from other modules, creating some dependencies but maintaining a logical flow.

**Update League Dates**

- **Cohesion:** Transform Temporal This module updates league dates (will be range) in the database. It ensures that updates are executed at the appropriate time.
- **Coupling:** Moderate It depends on league data from other modules, creating some dependencies but maintaining a logical flow.

**Update Match Type Detail**

- **Cohesion:** Transform Communication This module updates match type (one-day,t-20,test or others) details in the database. It ensures that updates are communicated to relevant modules.
- **Coupling:** Moderate It depends on match data from other modules, creating some dependencies but maintaining a clear purpose.

**Update Pool Info**

- **Cohesion:** Transform Communication This module updates pool information in the database. It ensures that updates are communicated to relevant modules.
- **Coupling:** Moderate It depends on pool data from other modules, creating some dependencies but maintaining a clear purpose.

**Update Team Info**

- **Cohesion:** Transform Communication This module updates team information in the database. It ensures that updates are communicated to relevant modules.
- **Coupling:** Moderate It depends on team data from other modules, creating some dependencies but maintaining a clear purpose.

**Update League Info**

- **Cohesion:** Transform Communication This module updates league information in the database. It ensures that updates are communicated to relevant modules.
- **Coupling:** Moderate It depends on league data from other modules, creating some dependencies but maintaining a clear purpose.

**Extract Field from Query**

- **Cohesion:** Transform Functional This module extracts specific fields from database queries. It focuses on a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Run Query (To store in DB)**

- **Cohesion:** Transform Functional This module runs queries to store data in the database. It performs a single, specific task.

- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Store in League Table**

- **Cohesion:** Output Functional This module stores data in the league table. It focuses on a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Store in Team Info Table**

- **Cohesion:** Output Functional This module stores data in the team info table. It performs a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Team Management**

- **Cohesion:** Composite Functional This module manages team-related operations, such as creating, updating, and retrieving team data. It combines multiple tasks related to teams.
- **Coupling:** High It interacts with multiple modules for team data, creating significant dependencies but ensuring comprehensive functionality.

**Update Team**

- **Cohesion:** Composite Functional This module updates team information, such as team names and squad details. It combines multiple tasks related to team updates.
- **Coupling:** High It interacts with multiple modules for team data, creating significant dependencies but ensuring comprehensive functionality.

**Update Team Name**

- **Cohesion:** Transform Functional This module updates team names in the database. It focuses on a single, well-defined task.
- **Coupling:** Moderate It depends on team data from other modules, creating some dependencies but maintaining a clear purpose.

**Get Team Detail**

- **Cohesion:** Input Functional This module retrieves team details from the database. It performs a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Run Query for Team Table**

- **Cohesion:** Transform Functional This module runs queries for the team table. It focuses on a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Update Squad**

- **Cohesion:** Transform Communication This module updates squad (list of all the players that can possibly are in the team or elected by that team for that league) information in the database. It ensures that updates are communicated to relevant modules.
- **Coupling:** Moderate It depends on squad data from other modules, creating some dependencies but maintaining a clear purpose.

**Create Team**

- **Cohesion:** Input Functional This module creates a new team by collecting and storing team details in the database. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Store Team Info**

- **Cohesion:** Output Functional This module stores team information in the database. It focuses on a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Get Team Name**

- **Cohesion:** Input Functional This module retrieves team names from the database. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Get Team ID**

- **Cohesion:** Input Functional This module retrieves team IDs from the database. It focuses on a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Squad Management**

- **Cohesion:** Composite Communication This module manages squad-related operations, such as selecting players and updating squad details. It combines multiple tasks related to squads.
- **Coupling:** High It interacts with multiple modules for squad data, creating significant dependencies but ensuring comprehensive functionality.

**Select List of Players**

- **Cohesion:** Input Functional This module selects a list of players for a squad. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Get from Player Table**

- **Cohesion:** Input Functional This module retrieves player data from the player table. It focuses on a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Add to Match-Team Table**

- **Cohesion:** Output Functional This module adds data to the match-team table. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Get Captain**

- **Cohesion:** Input Functional This module retrieves the captain's details from the database. It focuses on a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Get WK (Wicketkeeper)**

- **Cohesion:** Input Functional This module retrieves the wicketkeeper's details from the database. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Per Ball Management**

- **Cohesion:** Composite Communication This module manages per-ball data, such as ball-by-ball updates and match progress. It combines multiple tasks related to per-ball data.
- **Coupling:** High It interacts with multiple modules for per-ball data, creating significant dependencies but ensuring comprehensive functionality.

**Create Match**

- **Cohesion:** Input Functional This module creates a new match by collecting and storing match details in the database. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Get Match Date**

- **Cohesion:** Input Functional This module retrieves the match date from the database. It focuses on a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Get Match Time**

- **Cohesion:** Input Functional This module retrieves the match time from the database. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Manage Team 1**

- **Cohesion:** Transform Communication This module manages data for Team 1, such as player details and performance metrics. It ensures smooth communication of team-related data.
- **Coupling:** Moderate It depends on team data from other modules, creating some dependencies but maintaining a clear purpose.

**Manage Team 2**

- **Cohesion:** Transform Communication This module manages data for Team 2, such as player details and performance metrics. It ensures smooth communication of team-related data.
- **Coupling:** Moderate It depends on team data from other modules, creating some dependencies but maintaining a clear purpose.

**Get League Id (if Present)**

- **Cohesion:** Input Functional This module retrieves the league ID if it is present in the database. It focuses on a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Get Match Over**

- **Cohesion:** Input Functional This module retrieves the match over details from the database. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Get Match Type**

- **Cohesion:** Input Functional This module retrieves the match type details (like one-day,t-20,test etc) from the database. It focuses on a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Get Venue**

- **Cohesion:** Input Functional This module retrieves the venue details from the database. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Get Toss Detail**

- **Cohesion:** Input Functional This module retrieves the toss details from the database. It focuses on a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Save to Match Table (Query to add entry)**

- **Cohesion:** Transform Functional This module saves data to the match table by running a query. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Streaming Management**

- **Cohesion:** Composite Communication This module manages live video streaming, including protocols and output. It combines multiple tasks related to streaming.
- **Coupling:** High It interacts with multiple modules for streaming data, creating significant dependencies but ensuring comprehensive functionality.

**Live Video Streaming**

- **Cohesion:** Output Functional This module handles live video streaming and ensures smooth delivery of video content. It focuses on a single, well-defined task.
- **Coupling:** Moderate It depends on streaming protocols and data from other modules, creating some dependencies but maintaining a clear purpose.

**Protocols Management**

- **Cohesion:** Transform Functional This module manages streaming protocols, such as RTMP and DASH. It ensures that the correct protocols are used for streaming.
- **Coupling:** Moderate It depends on protocol data from other modules, creating some dependencies but maintaining a logical flow.

**RTMP Management**

- **Cohesion:** Transform Functional This module manages the RTMP protocol for streaming. It focuses on a single, specific task.
- **Coupling:** Moderate It depends on RTMP data from other modules, creating some dependencies but maintaining a clear purpose.

**DASH Management**

- **Cohesion:** Transform Functional This module manages the DASH protocol for streaming. It performs a single, well-defined task.
- **Coupling:** Moderate It depends on DASH data from other modules, creating some dependencies but maintaining a logical flow.

**Use of ABS**

- **Cohesion:** Transform Functional This module manages the ABS protocol for streaming. It focuses on a single, specific task.
- **Coupling:** Moderate It depends on ABS data from other modules, creating some dependencies but maintaining a clear purpose.

**Output Video**

- **Cohesion:** Output Functional This module outputs video data to the user. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Key Commentary**

- **Cohesion:** Output Functional This module outputs key commentary during live matches. It focuses on a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Live Score and Scoreboard**

- **Cohesion:** Composite Communication This module manages live scores and scoreboards, ensuring real-time updates. It combines multiple tasks related to score management.
- **Coupling:** High It interacts with multiple modules for score data, creating significant dependencies but ensuring comprehensive functionality.

**Get Specific Match Info**

- **Cohesion:** Composite Functional This module retrieves specific match information, such as scores and player details. It combines multiple tasks related to match data.
- **Coupling:** Moderate It depends on match data from other modules, creating some dependencies but maintaining a clear purpose.

**Get League Name (User)**

- **Cohesion:** Input Functional This module retrieves the league name for the user. It focuses on a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Get Match Name (User)**

- **Cohesion:** Input Functional This module retrieves the match name for the user. It performs a single, well-defined task.

- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Filter**

- **Cohesion:** Transform Functional This module filters data based on specific criteria. It focuses on a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Show Live Score + Scoreboard**

- **Cohesion:** Composite Communication This module displays live scores and scoreboards to the user. It combines multiple tasks related to score display.
- **Coupling:** Moderate It depends on score data from other modules, creating some dependencies but maintaining a clear purpose.

**Get Per Ball Detail**

- **Cohesion:** Input Functional This module retrieves per-ball details from the database. It focuses on a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Query on Per Ball Record Table**

- **Cohesion:** Transform Functional This module queries the per-ball record table for specific data. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Get Specific Match Squad Details**

- **Cohesion:** Input Functional This module retrieves specific match squad details from the database. It focuses on a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Query on Match-Team Table**

- **Cohesion:** Transform Functional This module queries the match-team table for specific data. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Chasing Info**

- **Cohesion:** Composite Communication This module manages chasing information, such as target scores and required run rates. It combines multiple tasks related to chasing data.
- **Coupling:** Moderate It depends on match and team data from other modules, creating some dependencies but maintaining a clear purpose.

**EOI Table**

- **Cohesion:** Output Functional This module outputs data to the EOI (End of Innings) table. It focuses on a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Query Per Ball Record Table**

- **Cohesion:** Transform Functional This module queries the per-ball record table for specific data. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**League Stats Management**

- **Cohesion:** Composite Functional This module manages league statistics, such as rankings and player performance. It combines multiple tasks related to league stats.
- **Coupling:** High It interacts with multiple modules for stats data, creating significant dependencies but ensuring comprehensive functionality.

**Get League Info**

- **Cohesion:** Output Functional This module retrieves league information from the database. It focuses on a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Query Fielding DB**

- **Cohesion:** Output Functional This module queries the fielding database for specific data. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Query League DB**

- **Cohesion:** Output Functional This module queries the league database for specific data. It focuses on a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Query Player DB**

- **Cohesion:** Output Functional This module queries the player database for specific data. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Filter League Stats**

- **Cohesion:** Transform Functional This module filters league statistics based on specific criteria. It focuses on a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Show League Stats**

- **Cohesion:** Output Functional This module displays league statistics to the user. It performs a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Manage Ranking**

- **Cohesion:** Transform Functional This module manages player and team rankings. It focuses on a single, well-defined task.
- **Coupling:** Moderate It depends on ranking data from other modules, creating some dependencies but maintaining a clear purpose.

**Player Profile Management**

- **Cohesion:** Composite Functional This module manages player profiles, including viewing and filtering profiles. It combines multiple tasks related to player profiles.
- **Coupling:** High It interacts with multiple modules for profile data, creating significant dependencies but ensuring comprehensive functionality.

**View Player Profile**

- **Cohesion:** Output Communication This module displays player profiles to the user. It focuses on a single, specific task.
- **Coupling:** Moderate It depends on profile data from other modules, creating some dependencies but maintaining a clear purpose.

**Get Player Info user**

- **Cohesion:** Input Functional This module retrieves player information for the user. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Filter profile**

- **Cohesion:** Transform Functional This module filters player profiles based on specific criteria. It focuses on a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Specific Player Profile**

- **Cohesion:** Output Functional This module displays specific player profiles to the user. It performs a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**EOI Management**

- **Cohesion:** Composite Functional This module manages EOI (End of Innings) data, such as fall of wickets and total scores. It combines multiple tasks related to EOI data.
- **Coupling:** High It interacts with multiple modules for EOI data, creating significant dependencies but ensuring comprehensive functionality.

**Get Match Info**

- **Cohesion:** Input Functional This module retrieves match information from the database. It focuses on a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Fall of Wickets**

- **Cohesion:** Output Functional This module displays the fall of wickets during a match. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Total Score Calculation**

- **Cohesion:** Transform Functional This module calculates the total score for a match. It focuses on a single, well-defined task.
- **Coupling:** Moderate It depends on score data from other modules, creating some dependencies but maintaining a clear purpose.

**Updation to EOI Table**

- **Cohesion:** Output Functional This module updates the EOI table with match results. It performs a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**End of Match Management**

- **Cohesion:** Composite Functional This module manages end-of-match operations, such as scoreboard preparation and result updates. It combines multiple tasks related to match conclusion.
- **Coupling:** High It interacts with multiple modules for match data, creating significant dependencies but ensuring comprehensive functionality.

**Score-Board Preparation**

- **Cohesion:** Output Functional This module prepares the scoreboard for display at the end of a match. It focuses on a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Other Match Management**

- **Cohesion:** Composite Communication This module manages other match-related operations, such as player of the match and result updates. It combines multiple tasks related to match management.
- **Coupling:** High It interacts with multiple modules for match data, creating significant dependencies but ensuring comprehensive functionality.

**Get Player of the Match**

- **Cohesion:** Input Functional This module retrieves the player of the match from the database. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Updation of Result**

- **Cohesion:** Output Functional This module updates the match results in the database. It focuses on a single, specific task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, ensuring minimal dependencies.

**Point Table Management**

- **Cohesion:** Output Functional This module manages the point table for leagues. It performs a single, well-defined task.
- **Coupling:** Loosely coupled It has minimal interaction with other modules, making it independent and easy to maintain.

**Deletion of Per Ball Detail**

- **Cohesion:** Transform Temporal This module deletes per-ball details from the database at the appropriate time. It ensures that deletions are executed when necessary.
- **Coupling:** Moderate It depends on per-ball data from other modules, creating some dependencies but maintaining a logical flow.

# Composition of Final Structured Chart

| Module Type | Count |
|---|---|
| Input | 25 |
| Output | 15 |
| Coordinate | 2 |
| Transform | 20 |
| Composite | 10 |

Table 2: Count of Module Types

# Most Complex/Error Prone Modules

| Category | Module | Description | Why Complex/Error-Prone |
|---|---|---|---|
| **Input** | Login (Google Auth) | Handles user authentication using Google Auth. | • External API dependency<br>• Security concerns<br>• Error handling<br>• User experience impact |
| **Transformation** | Manage Bracket | Processes and manages bracket data sequentially. | • Sequential logic<br>• Data integrity<br>• Error handling<br>• Performance bottlenecks |
| **Output** | Live Video Streaming | Handles live video streaming and outputs video data in real-time. | • Real-time processing<br>• Protocol management<br>• Error handling<br>• Scalability challenges |

Table 3: Most Complex/Error-Prone Modules in Each Category

# Top Modules in Terms of Fan-In and Fan-Out

## Top 3 Modules in Terms of Fan-In

1. **Organiser Management**

- **Reason for High Fan-In:** This module has highest fan-in that is 7 as shown in module diagram (assuming Create  Manage match combined). It directly controls the manager bracket, league creation and management, live streaming, match creation and match management. This module deals with all input provided by the organizer.

2. **Main**
    - **Reason for High Fan-In:** This is the center of coordinating module that controls large size module and have have Fan-in count of 4. This module ensures proper functioning between modules like User Display Management, Organizer Management, Player management, and authentication

.

3. **Live video streaming**
    - **Reason for High Fan-In:** This modules plays important role in live video streaming that controls the 4 module i.e. input video (module to manage input at organiser end) , protocol management, ABS (Adaptive Bitrate streaming) use module, Output video module.

.

## Top 3 Modules in Terms of Fan-Out

1. **Manage Match**
    - **Reason for High Fan-Out:** This module has highest fan-Out that is controlled by 10 module as shown (As "save to match table" is fan-in module for this + getting "match id" include dependency on one more module) . It depends on many small input modules and interacts with DB to create any match. In return it gives match id of to the organiser. Some modules that it depends upon : Valid match Org verification ,managing of teams (team 1 and team 2) , getting of all the required inputs
2. **Create Match**
    - **Reason for High Fan-Out:** This module has 2nd highest fan-Out that is controlled by 9 module as shown (As "save to match table" is fan-in module for this) . It depends on many small input modules and interacts with DB to manage any match after the creation. In return it modifies the update list to db. Some modules that it depends upon : managing of teams (team 1 and team 2) , updation of required inputs
3. **Manage League**
    - **Reason for High Fan-Out:** This module has 3rd highest fan-Out that is controlled by 7 module as shown (As "store league info" is fan-in module for this but getting of league id from "Valid Match Org Verification" introduce on more) . It depends on many input modules and interacts with DB to manage any match after the league creation . In return it modifies the update list to db. Some modules that it depends upon : Getting Updated team info and , other updation of required inputs.

## Detailed Design Specification

These are the list of functional signature of almost module, probable input and the type of output it will return

```
1
2  def main_coordinate(system_data: dict, DBConnection) -> None
3
4  def login_google_auth(user_credentials: dict, DBConnection) -> bool
5
6  def signup(user_details: dict, DBConnection) -> bool
7
8  def logout(user_session: dict) -> None
9
10 def manage_organiser(organiser_data: dict, DBConnection) -> bool
11
12 def manage_bracket(bracket_data: dict, DBConnection) -> bool
13
14 def get_organiser_id(organiser_name: str, DBConnection) -> int
15
16 def get_league_id(league_name: str, DBConnection) -> int
17
18 def valid_bracket_use(bracket_id: int, league_id: int, DBConnection) -> bool
19
20 def manage_pool(pool_data: dict, DBConnection) -> bool
21
22 def create_new_pool(pool_details: dict, DBConnection) -> bool
23
24 def get_old_pool_info(pool_id: int, DBConnection) -> dict
25
26 def update_league_db(league_data: dict, DBConnection) -> bool
```

```
def get_new_pool_info(pool_id: int, DBConnection) -> dict

def manage_match(match_data: dict, DBConnection) -> bool

def create_match(match_details: dict, DBConnection) -> bool

def manage_match_data(match_id: int, match_updates: dict, DBConnection) -> bool

def manage_league(league_data: dict, DBConnection) -> bool

def valid_match_org(match_id: int, organiser_id: int, DBConnection) -> bool

def manage_league_data(league_id: int, league_updates: dict, DBConnection) -> bool

def delete_league(league_id: int, DBConnection) -> bool

def update_league_dates(league_id: int, new_dates: dict, DBConnection) -> bool
def update_match_type_detail(match_id: int, match_type: str, DBConnection) -> bool

def update_pool_info(pool_id: int, pool_updates: dict, DBConnection) -> bool

def update_team_info(team_id: int, team_updates: dict, DBConnection) -> bool

def update_league_info(league_id: int, league_updates: dict, DBConnection) -> bool

def extract_field_from_query(query: str, field_name: str, DBConnection) -> str

def run_query_store_db(query: str, DBConnection) -> bool

def store_in_league_table(league_data: dict, DBConnection) -> bool

def store_in_team_info_table(team_data: dict, DBConnection) -> bool

def manage_team(team_data: dict, DBConnection) -> bool

def update_team(team_id: int, team_updates: dict, DBConnection) -> bool

def update_team_name(team_id: int, new_name: str, DBConnection) -> bool

def get_team_detail(team_id: int, DBConnection) -> dict

def run_query_team_table(query: str, DBConnection) -> dict

def update_squad(team_id: int, squad_updates: dict, DBConnection) -> bool

def create_team(team_details: dict, DBConnection) -> bool

def store_team_info(team_data: dict, DBConnection) -> bool

def get_team_name(team_id: int, DBConnection) -> str

def get_team_id(team_name: str, DBConnection) -> int

def manage_squad(squad_data: dict, DBConnection) -> bool

def select_list_of_players(team_id: int, DBConnection) -> list

def get_from_player_table(player_id: int, DBConnection) -> dict

def add_to_match_team_table(match_id: int, team_id: int, DBConnection) -> bool
def get_captain(team_id: int, DBConnection) -> dict

def get_wk(team_id: int, DBConnection) -> dict

def manage_per_ball(ball_data: dict, DBConnection) -> bool

def create_match(match_details: dict, DBConnection) -> bool

def get_match_date(match_id: int, DBConnection) -> str

def get_match_time(match_id: int, DBConnection) -> str

def manage_team_1(team_id: int, team_updates: dict, DBConnection) -> bool

def manage_team_2(team_id: int, team_updates: dict, DBConnection) -> bool

def get_league_id_if_present(match_id: int, DBConnection) -> int

def get_match_over(match_id: int, DBConnection) -> int
```

```
def get_match_type(match_id: int, DBConnection) -> str

def get_venue(match_id: int, DBConnection) -> str

def get_toss_detail(match_id: int, DBConnection) -> dict

def save_to_match_table(match_data: dict, DBConnection) -> bool

def manage_streaming(streaming_data: dict, DBConnection) -> bool

def live_video_streaming(stream_url: str, DBConnection) -> bool

def manage_protocols(protocol_data: dict, DBConnection) -> bool

def manage_rtmp(rtmp_data: dict, DBConnection) -> bool

def manage_dash(dash_data: dict, DBConnection) -> bool

def use_abs(abs_data: dict, DBConnection) -> bool

def output_video(video_data: dict, DBConnection) -> bool

def key_commentary(commentary_data: dict, DBConnection) -> bool

def manage_live_score(score_data: dict, DBConnection) -> bool

def get_specific_match_info(match_id: int, DBConnection) -> dict

def get_league_name_user(league_id: int, DBConnection) -> str

def get_match_name_user(match_id: int, DBConnection) -> str

def filter_data(data: list, filter_criteria: dict) -> list

def show_live_score_scoreboard(match_id: int, DBConnection) -> dict

def get_per_ball_detail(match_id: int, ball_number: int, DBConnection) -> dict
def query_per_ball_record_table(query: str, DBConnection) -> dict

def get_specific_match_squad_details(match_id: int, DBConnection) -> dict

def query_match_team_table(query: str, DBConnection) -> dict

def manage_chasing_info(match_id: int, DBConnection) -> dict

def update_eoi_table(eoi_data: dict, DBConnection) -> bool

def query_per_ball_record_table(query: str, DBConnection) -> dict

def manage_league_stats(league_id: int, DBConnection) -> dict

def get_league_info(league_id: int, DBConnection) -> dict

def query_fielding_db(query: str, DBConnection) -> dict

def query_league_db(query: str, DBConnection) -> dict

def query_player_db(query: str, DBConnection) -> dict

def filter_league_stats(stats_data: list, filter_criteria: dict) -> list

def show_league_stats(league_id: int, DBConnection) -> dict

def manage_ranking(ranking_data: dict, DBConnection) -> bool

def manage_player_profile(player_id: int, DBConnection) -> dict

def view_player_profile(player_id: int, DBConnection) -> dict

def get_player_info_user(player_id: int, DBConnection) -> dict

def filter_profile(profile_data: list, filter_criteria: dict) -> list

def specific_player_profile(player_id: int, DBConnection) -> dict

def manage_eoi(eoi_data: dict, DBConnection) -> bool

def get_match_info(match_id: int, DBConnection) -> dict

def fall_of_wickets(match_id: int, DBConnection) -> dict

```

```
def total_score_calculation(match_id: int, DBConnection) -> int

def update_eoi_table(eoi_data: dict, DBConnection) -> bool

def manage_end_of_match(match_id: int, DBConnection) -> bool

def prepare_scoreboard(match_id: int, DBConnection) -> dict

def manage_other_match(match_id: int, DBConnection) -> bool

def get_player_of_the_match(match_id: int, DBConnection) -> dict

def update_result(match_id: int, result_data: dict, DBConnection) -> bool

def manage_point_table(league_id: int, DBConnection) -> dict

def delete_per_ball_detail(match_id: int, DBConnection) -> bool
```